GTP Illumination Module Reference Manual

Generated by Doxygen 1.5.1-p1

Wed Mar 28 11:48:46 2007

Contents

Chapter 1

GTP Illumination Module Namespace Index

1.1 GTP Illumination Module Namespace List

Here is a list of all documented namespaces with brief descriptions:

Chapter 2

GTP Illumination Module Hierarchical Index

2.1 GTP Illumination Module Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 3

GTP Illumination Module Class Index

3.1 GTP Illumination Module Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 4

GTP Illumination Module Namespace Documentation

4.1 CausticCasterParsers Namespace Reference

Technique Parsers.

4.1.1 Detailed Description

Technique Parsers.

4.2 ConvolvedCubemapParsers Namespace Reference

Technique Parsers.

4.2.1 Detailed Description

Technique Parsers.

4.3 CubemapParsers Namespace Reference

Technique Parsers.

4.3.1 Detailed Description

Technique Parsers.

4.4 DepthShadowReceiverParsers Namespace Reference

Technique parsers.

4.4.1 Detailed Description

Technique parsers.

4.5 HPSParsers Namespace Reference

Technique Parsers.

Functions

- void parseUseDistCalc (String &params, RenderTechniqueFactory *factory)

4.5.1 Detailed Description

Technique Parsers.

4.5.2 Function Documentation

4.5.2.1 void HPSParsers::parseUseDistCalc (String & *params*, RenderTechniqueFactory * *factory*)

4.6 IllumVolumeParsers Namespace Reference

Technique Parsers.

Functions

- void parseUseDistCalc (String &params, RenderTechniqueFactory *factory)

4.6.1 Detailed Description

Technique Parsers.

4.6.2 Function Documentation

4.6.2.1 void IllumVolumeParsers::parseUseDistCalc (String & *params*, RenderTechniqueFactory * *factory*)

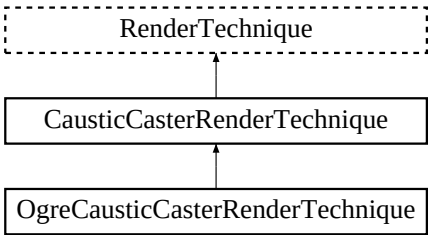
Chapter 5

GTP Illumination Module Class Documentation

5.1 CausticCasterRenderTechnique Class Reference

Base abstract class of a caustic caster.

Inheritance diagram for CausticCasterRenderTechnique::



Public Member Functions

- CausticCasterRenderTechnique (unsigned long startFrame, unsigned long photonMapUpdateInterval, unsigned int photonMapResolution, unsigned int causticCubeMapResolution, bool updateAllFace, bool useDistance, ElementaryRenderable *parentRenderable, TechniqueGroup *parentTechniqueGroup)

  Constructor.

- void runChanged (RenderingRunType runType, RenderingRun *run)

  Called after one of he shared runs changes.

- virtual void runUpdated (RenderingRunType runType, RenderingRun *run)

  Called after one of he shared runs updates.

- virtual void update (unsigned long frameNum)

Updates the resources in the given frame.

- virtual class OgreRenderTechnique * asOgreRenderTechnique ()

  Conversion to OgreRenderTechnique.

- ElementaryRenderable * getParentRenderable ()

  Retrieves the renderable this technique operates on.

Protected Member Functions

- virtual void photonMapRunChanged (RenderingRun *run)=0

  Called if the changed run is a PhotonMapRenderingRun.

- virtual void causticCubeMapRunChanged (RenderingRun *run)=0

  Called if the changed run is a CausticCubeMapRenderingRun.

- virtual void distanceCubeMapRunChanged (RenderingRun *run)=0

  Called if the changed run is a DistanceCubeMapRenderingRun.

- virtual void distanceCubeMapRunUpdated (RenderingRun *run)=0

  Called if the updated run is a DistanceCubeMapRenderingRun.

- virtual RenderingRun * createPhotonMapRun ()=0

  Creates a PhotonMapRenderingRun.

- virtual RenderingRun * createCausticCubeMapRun ()=0

  Creates a CausticCubeMapRenderingRun.

- virtual RenderingRun * createDistanceCubeMapRun ()=0

  Creates a DistanceCubeMapRenderingRun.

Protected Attributes

- bool updateAllFace

  defines if all cubemap faces should be updated in a frame or only one face per frame

- bool useDistance

  tells if a distance cubemap impostor should be used in photon hit calculation (recommended)

- unsigned long photonMapUpdateInterval

  photonmap update frequency

- unsigned int photonMapResolution

  photonmap resolution

- unsigned int causticCubeMapResolution

  caustic cubemap resolution

- unsigned long startFrame

  offset in frame number used during update

- ElementaryRenderable * parentRenderable

  The renderable this technique operates on.

- TechniqueGroup * parentTechniqueGroup

  The TechniqueGroup this RenderedTechnique is attached to.

- SharedRuns * sharedRuns

  The SharedRuns this RenderedTechnique is attached to.

5.1.1 Detailed Description

Base abstract class of a caustic caster.

This technique defines that the given object needs a caustic photon map and a caustic cubemap. The caustic cube map will be used by caustic receivers.

5.1.2 Constructor & Destructor Documentation

5.1.2.1 CausticCasterRenderTechnique::CausticCasterRenderTechnique (unsigned long *startFrame*, unsigned long *photonMapUpdateInterval*, unsigned int *photonMapResolution*, unsigned int *causticCubeMapResolution*, bool *updateAllFace*, bool *useDistance*, ElementaryRenderable * *parentRenderable*, TechniqueGroup * *parentTechniqueGroup*)

Constructor.

Parameters:

startFrame adds an offset to the current frame number to help evenly distribute updates between frames

photonMapUpdateInterval photon map and caustic cubemap update frequency

photonMapResolution photon map resolution

causticCubeMapResolution caustic cubemap resolution

updateAllFace defines if all cubemap faces should be updated in a frame or only one face per frame

useDistance tells if a distance cubemap impostor should be used in photon hit calculation (recommended)

parentRenderable the object to operate on

parentTechniqueGroup the TechniqueGroup this RenderedTechnique is attached to

5.1.3 Member Function Documentation

5.1.3.1 void CausticCasterRenderTechnique::runChanged (RenderingRunType *runType*, RenderingRun * *run*) [virtual]

Called after one of he shared runs changes.

Parameters:

runType enum describing the type of the changed run

run pointer to the changed RenderingRun

Reimplemented from RenderTechnique.

5.1.3.2 void CausticCasterRenderTechnique::runUpdated (RenderingRunType *runType*, RenderingRun * *run*) [virtual]

Called after one of he shared runs updates.

Parameters:

runType enum describing the type of the updated run

run pointer to the updated RenderingRun

Reimplemented from RenderTechnique.

5.1.3.3 virtual void CausticCasterRenderTechnique::photonMapRunChanged (RenderingRun * *run*) [protected, pure virtual]

Called if the changed run is a PhotonMapRenderingRun.

Parameters:

run pointer to the changed PhotonMapRenderingRun

Implemented in OgreCausticCasterRenderTechnique.

5.1.3.4 virtual void CausticCasterRenderTechnique::causticCubeMapRunChanged (RenderingRun * *run*) [protected, pure virtual]

Called if the changed run is a CausticCubeMapRenderingRun.

Parameters:

run pointer to the changed CausticCubeMapRenderingRun

Implemented in OgreCausticCasterRenderTechnique.

5.1.3.5 virtual void CausticCasterRenderTechnique::distanceCubeMapRunChanged (RenderingRun * *run*) [protected, pure virtual]

Called if the changed run is a DistanceCubeMapRenderingRun.

Parameters:

run pointer to the changed DistanceCubeMapRenderingRun

Implemented in OgreCausticCasterRenderTechnique.

5.1.3.6 virtual void CausticCasterRenderTechnique::distanceCubeMapRunUpdated (RenderingRun * *run*) [protected, pure virtual]

Called if the updated run is a DistanceCubeMapRenderingRun.

Parameters:

run pointer to the updated DistanceCubeMapRenderingRun

Implemented in OgreCausticCasterRenderTechnique.

5.1.3.7 virtual RenderingRun* CausticCasterRenderTechnique::createPhotonMapRun () [protected, pure virtual]

Creates a PhotonMapRenderingRun.

Returns:

the new PhotonMapRenderingRun instance.

Implemented in OgreCausticCasterRenderTechnique.

5.1.3.8 virtual RenderingRun* CausticCasterRenderTechnique::createCausticCubeMapRun () [protected, pure virtual]

Creates a CausticCubeMapRenderingRun.

Returns:

the new CausticCubeMapRenderingRun instance.

Implemented in OgreCausticCasterRenderTechnique.

5.1.3.9 virtual RenderingRun* CausticCasterRenderTechnique::createDistanceCubeMapRun () [protected, pure virtual]

Creates a DistanceCubeMapRenderingRun.

Returns:

the new DistanceCubeMapRenderingRun instance.

Implemented in OgreCausticCasterRenderTechnique.

5.1.3.10 virtual void RenderTechnique::update (unsigned long *frameNum*) [inline, virtual, inherited]

Updates the resources in the given frame.

A RenderTechnique is usually need some resources from several runs, so these runs will be updated.

Parameters:

frameNum the actual framenumber

Reimplemented in ColorCubeMapRenderTechnique, ConvolvedCubeMapRenderTechnique, DistanceCubeMapRenderTechnique, HierarchicalParticleSystemTechnique, IllumVolumeRenderTechnique, OgreCausticReceiverRenderTechnique, OgreColorCubeMapRenderTechnique, OgreConvolvedCubeMapRenderTechnique, OgreDepthShadowReceiverRenderTechnique, OgreDistanceCubeMapRenderTechnique, OgreFireRenderTechnique, OgrePathMapRenderTechnique, and OgreSBBRenderTechnique.

5.1.3.11 virtual class OgreRenderTechnique* RenderTechnique::asOgreRenderTechnique () [inline, virtual, inherited]

Conversion to OgreRenderTechnique.

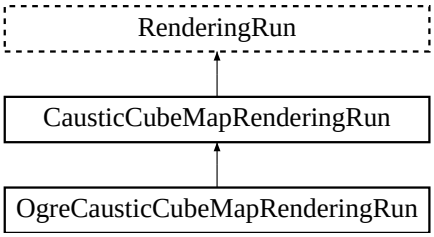
This function is needed because of virtual inheritance.

Reimplemented in OgreRenderTechnique.

5.2 CausticCubeMapRenderingRun Class Reference

Base abstract class that defines a rendering process of a caustic cubemap.

Inheritance diagram for CausticCubeMapRenderingRun::



Public Member Functions

- CausticCubeMapRenderingRun (unsigned long startFrame, unsigned long updateInterval, unsigned int resolution, bool updateAllFace)

  Constructor.

- virtual void photonMapChanged (RenderingRun *run)=0

  Called if the changed run is a PhotonMapRenderingRun.

- bool update (unsigned long frameNum)

  Calls updateFrame() if the run needs update according to its starting frame and update interval and has not been allready updated in this frame.

- virtual class OgreRenderingRun * asOgreRenderingRun ()

  Conversion to OgreRenderRun.

- virtual bool canJoin (RenderingRun *run)

  Returns true if two runs can be joined.

Protected Member Functions

- virtual void createCausticCubeMap ()=0

  Creates a cubemap texture used for the caustic-cubemap.

- virtual void updateCubeFace (int facenum)=0

  Updates one face of the cubemap.

- virtual bool faceNeedsUpdate (int facenum)=0

  Checks if a cubemap face needs to be updated.

- virtual void updateFrame (unsigned long frameNum)

This function does the actual update in a frame.

- virtual bool needUpdate (unsigned long frameNum)

  Returns if this run needs update.

Protected Attributes

- bool updateAllFace

  defines if all cubemap faces should be updated in a frame or only one face per frame

- unsigned char currentFace

  the number of the face to be updated

- unsigned int resolution

  the resolution of the cubemap texture that was created by this run

- unsigned long lastupdated

  The number of the last frame this run was updated.

- unsigned long startFrame

  The number of the frame this run should be updated first.

- unsigned long updateInterval

  Refresh frequency in frames.

5.2.1 Detailed Description

Base abstract class that defines a rendering process of a caustic cubemap.

A caustic cubemap stores caustic light spots caused by a caustic emitter object.

5.2.2 Constructor & Destructor Documentation

5.2.2.1 CausticCubeMapRenderingRun::CausticCubeMapRenderingRun (unsigned long *startFrame*, unsigned long *updateInterval*, unsigned int *resolution*, bool *updateAllFace*)

Constructor.

Parameters:

startFrame adds an offset to the current frame number to help evenly distribute updates between frames

updateInterval photon map update frequency

resolution photon map resolution

updateAllFace defines if all cubemap faces should be updated in a frame or only one face per frame

5.2.3 Member Function Documentation

5.2.3.1 virtual void CausticCubeMapRenderingRun::photonMapChanged (RenderingRun * *run*) [pure virtual]

Called if the changed run is a PhotonMapRenderingRun.

Parameters:

run pointer to the changed PhotonMapRenderingRun

Implemented in OgreCausticCubeMapRenderingRun.

5.2.3.2 virtual void CausticCubeMapRenderingRun::updateCubeFace (int *facenum*) [inline, protected, pure virtual]

Updates one face of the cubemap.

Parameters:

facenum the number of the face to be updated

Implemented in OgreCausticCubeMapRenderingRun.

5.2.3.3 virtual bool CausticCubeMapRenderingRun::faceNeedsUpdate (int *facenum*) [protected, pure virtual]

Checks if a cubemap face needs to be updated.

If the object we are updating the cubemap for is far from the camera, or too small, or the given cubemapface does not have significant effect on the rendering the face can be skipped.

Parameters:

facenum the number of the face to be checked

Implemented in OgreCausticCubeMapRenderingRun.

5.2.3.4 void CausticCubeMapRenderingRun::updateFrame (unsigned long *frameNum*) [protected, virtual]

This function does the actual update in a frame.

Parameters:

frameNum current frame number

Reimplemented from RenderingRun.

5.2.3.5 virtual class OgreRenderingRun* RenderingRun::asOgreRenderingRun () [inline, virtual, inherited]

Conversion to OgreRenderRun.

This function is needed because of virtual inheritance.

Reimplemented in OgreRenderingRun, and OgreRenderingRun.

5.2.3.6 virtual bool RenderingRun::canJoin (RenderingRun * *run*) [inline, virtual, inherited]

Returns true if two runs can be joined.

In some cases special requirements should stand to join two runs (even if they have the same type). Eg.: two caustic cube map generation technique should only be joined if they use the same material when rendering the caustic cubemap.

Reimplemented in OgreCausticCubeMapRenderingRun, OgreChildPSystemRenderingRun, and OgrePhotonMapRenderingRun.

5.2.3.7 virtual bool RenderingRun::needUpdate (unsigned long *frameNum*) [inline, protected, virtual, inherited]

Returns if this run needs update.

This tipically depends on the upate interval and the starting frame number.

Parameters:

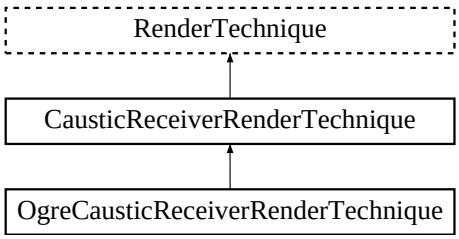
frameNum current frame number

Reimplemented in OgreFocusingMapRenderingRun, and OgrePMEntryPointMapRenderingRun.

5.3 CausticReceiverRenderTechnique Class Reference

Base abstract class of rendering a caustic receiver object.

Inheritance diagram for CausticReceiverRenderTechnique::



Public Member Functions

- CausticReceiverRenderTechnique (ElementaryRenderable *parentRenderable, TechniqueGroup *parentTechniqueGroup)

  Constructor.

- virtual void update (unsigned long frameNum)

  Updates the resources in the given frame.

- virtual void runChanged (RenderingRunType runType, RenderingRun *run)

  Called after one of he shared runs changes.

- virtual void runUpdated (RenderingRunType runType, RenderingRun *run)

  Called after one of he shared runs updates.

- virtual class OgreRenderTechnique * asOgreRenderTechnique ()

  Conversion to OgreRenderTechnique.

- ElementaryRenderable * getParentRenderable ()

  Retrieves the renderable this technique operates on.

Protected Attributes

- ElementaryRenderable * parentRenderable

  The renderable this technique operates on.

- TechniqueGroup * parentTechniqueGroup

  The TechniqueGroup this RenderedTechnique is attached to.

- SharedRuns * sharedRuns

  The SharedRuns this RenderedTechnique is attached to.

5.3.1 Detailed Description

Base abstract class of rendering a caustic receiver object.

5.3.2 Constructor & Destructor Documentation

5.3.2.1 CausticReceiverRenderTechnique::CausticReceiverRenderTechnique (ElementaryRenderable * *parentRenderable*, TechniqueGroup * *parentTechniqueGroup*)

Constructor.

Parameters:

parentRenderable the object to operate on

parentTechniqueGroup the TechniqueGroup this RenderedTechnique is attached to

5.3.3 Member Function Documentation

5.3.3.1 virtual void RenderTechnique::update (unsigned long *frameNum*) [inline, virtual, inherited]

Updates the resources in the given frame.

A RenderTechnique is usually need some resources from several runs, so these runs will be updated.

Parameters:

frameNum the actual framenumber

Reimplemented in ColorCubeMapRenderTechnique, ConvolvedCubeMapRenderTechnique, DistanceCubeMapRenderTechnique, HierarchicalParticleSystemTechnique, IllumVolumeRenderTechnique, OgreCausticReceiverRenderTechnique, OgreColorCubeMapRenderTechnique, OgreConvolvedCubeMapRenderTechnique, OgreDepthShadowReceiverRenderTechnique, OgreDistanceCubeMapRenderTechnique, OgreFireRenderTechnique, OgrePathMapRenderTechnique, and OgreSBBRenderTechnique.

5.3.3.2 virtual void RenderTechnique::runChanged (RenderingRunType *runType*, RenderingRun * ***run***) [inline, virtual, inherited]

Called after one of he shared runs changes.

Parameters:

runType enum describing the type of the changed run

run pointer to the changed RenderingRun

Reimplemented in CausticCasterRenderTechnique, ColorCubeMapRenderTechnique, ConvolvedCubeMapRenderTechnique, CubeMapRenderTechnique, DistanceCubeMapRenderTechnique, HierarchicalParticleSystemTechnique, and IllumVolumeRenderTechnique.

5.3.3.3 virtual void RenderTechnique::runUpdated (RenderingRunType *runType*, RenderingRun * *run*) [inline, virtual, inherited]

Called after one of he shared runs updates.

Parameters:

runType enum describing the type of the updated run

run pointer to the updated RenderingRun

Reimplemented in CausticCasterRenderTechnique, ColorCubeMapRenderTechnique, CubeMapRenderTechnique, DistanceCubeMapRenderTechnique, HierarchicalParticleSystemTechnique, and IllumVolumeRenderTechnique.

5.3.3.4 virtual class OgreRenderTechnique* RenderTechnique::asOgreRenderTechnique () [inline, virtual, inherited]

Conversion to OgreRenderTechnique.

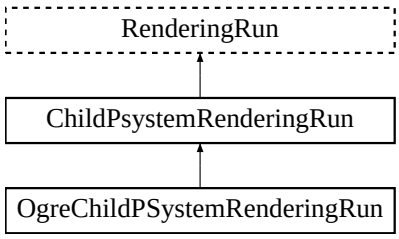
This function is needed because of virtual inheritance.

Reimplemented in OgreRenderTechnique.

5.4 ChildPsystemRenderingRun Class Reference

Base abstract class that defines a rendering process of a particle system impostor image.

Inheritance diagram for ChildPsystemRenderingRun::



Public Member Functions

- ChildPsystemRenderingRun (unsigned int resolution, bool perspectiveRendering, unsigned long startFrame, unsigned long updateInterval)

  Constructor.

- bool update (unsigned long frameNum)

  Calls updateFrame() if the run needs update according to its starting frame and update interval and has not been allready updated in this frame.

- virtual class OgreRenderingRun * asOgreRenderingRun ()

  Conversion to OgreRenderRun.

- virtual bool canJoin (RenderingRun *run)

  Returns true if two runs can be joined.

Protected Member Functions

- virtual void updateFrame (unsigned long frameNum)=0

  This function does the actual update in a frame.

- virtual void createImpostorTexture ()=0

  Creates an impostor texture that can be used as a rendertarget during impostor rendering.

- virtual bool needUpdate (unsigned long frameNum)

  Returns if this run needs update.

Protected Attributes

- unsigned int resolution

  the resolution of the impostor image

- bool perspectiveRendering

  sets if the impostor should be rendered with a perspective projection or orthogonal

- unsigned long lastupdated

  The number of the last frame this run was updated.

- unsigned long startFrame

  The number of the frame this run should be updated first.

- unsigned long updateInterval

  Refresh frequency in frames.

5.4.1 Detailed Description

Base abstract class that defines a rendering process of a particle system impostor image.

This impostor can be used as a texture for ither particle systems. This rendering method is called hiearar-chical particle system.

5.4.2 Constructor & Destructor Documentation

5.4.2.1 ChildPsystemRenderingRun::ChildPsystemRenderingRun (unsigned int *resolution*, bool *perspectiveRendering*, unsigned long *startFrame*, unsigned long *updateInterval*) [inline]

Constructor.

Parameters:

resolution the resolution of the impostor image

perspectiveRendering sets if the impostor should be rendered with a perspective projection or orthog-onal

startFrame adds an offset to the current frame number to help evenly distribute updates between frames

updateInterval update frequency

5.4.3 Member Function Documentation

5.4.3.1 virtual void ChildPsystemRenderingRun::updateFrame (unsigned long *frameNum*) [protected, pure virtual]

This function does the actual update in a frame.

Parameters:

frameNum current frame number

Reimplemented from RenderingRun.

Implemented in OgreChildPSystemRenderingRun.

5.4.3.2 virtual class OgreRenderingRun* RenderingRun::asOgreRenderingRun () [inline, virtual, inherited]

Conversion to OgreRenderRun.

This function is needed because of virtual inheritance.

Reimplemented in OgreRenderingRun, and OgreRenderingRun.

5.4.3.3 virtual bool RenderingRun::canJoin (RenderingRun * *run*) [inline, virtual, inherited]

Returns true if two runs can be joined.

In some cases special requirements should stand to join two runs (even if they have the same type). Eg.: two caustic cube map generation technique should only be joined if they use the same material when rendering the caustic cubemap.

Reimplemented in OgreCausticCubeMapRenderingRun, OgreChildPSystemRenderingRun, and OgrePhotonMapRenderingRun.

5.4.3.4 virtual bool RenderingRun::needUpdate (unsigned long *frameNum*) [inline, protected, virtual, inherited]

Returns if this run needs update.

This tipically depends on the upate interval and the starting frame number.

Parameters:

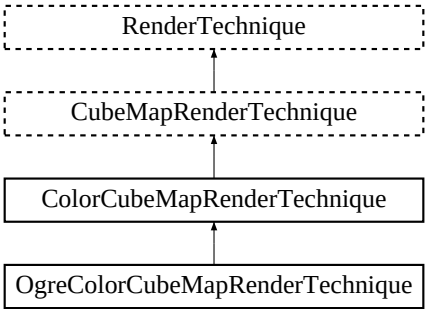
frameNum current frame number

Reimplemented in OgreFocusingMapRenderingRun, and OgrePMEntryPointMapRenderingRun.

5.5 ColorCubeMapRenderTechnique Class Reference

Base abstract class of rendering a color cube map.

Inheritance diagram for ColorCubeMapRenderTechnique::



Public Member Functions

- ColorCubeMapRenderTechnique (unsigned long startFrame, unsigned long cubeMapUpdateInterval, unsigned int cubeMapResolution, bool useDistCalc, bool useFaceAngleCalc, float distTolerance, float angleTolerance, bool updateAllFace, bool renderSelf, bool renderEnvironment, int layer, ElementaryRenderable *parentRenderable, TechniqueGroup *parentTechniqueGroup)

  Constructor.

- virtual void update (unsigned long frameNum)

  Updates the resources in the given frame.

- void runChanged (RenderingRunType runType, RenderingRun *run)

  Called after one of he shared runs changes.

- void runUpdated (RenderingRunType runType, RenderingRun *run)

  Called after one of he shared runs updates.

- virtual class OgreRenderTechnique * asOgreRenderTechnique ()

  Conversion to OgreRenderTechnique.

- ElementaryRenderable * getParentRenderable ()

  Retrieves the renderable this technique operates on.

Protected Member Functions

- virtual void colorCubeMapRunChanged (RenderingRun *run)=0

  Called if the color cubemap rendering run object changes.

- virtual void colorCubeMapRunUpdated (RenderingRun *run)=0

  Called if the color cubemap rendering run object is updated.

- virtual RenderingRun * createCubeMapRun ()=0

  Creates a CubeMapRenderingRun.

- virtual void cubeMapRunChanged (RenderingRun *run)=0

  Called if the changed run is a CubeMapRenderingRun.

- virtual void cubeMapRunUpdated (RenderingRun *run)=0

  Called if the updated run is a CubeMapRenderingRun.

Protected Attributes

- bool useDistCalc

  a flag to skip cube face update if object is far away or too small.

- bool useFaceAngleCalc

  a flag to skip cube face update the face is neglible.

- float distTolerance

  A value used in face skip test.

- float angleTolerance

  A value used in face skip test.

- bool updateAllFace

  defines if all cubemap faces should be updated in a frame or only one face per frame

- unsigned long cubeMapUpdateInterval

  color-cubemap update frequency

- unsigned int cubeMapResolution

  color-cubemap resolution

- unsigned long startFrame

  offset in frame number used during update

- bool renderSelf

  Sets if the object should be rendered to the cube map.

- bool renderEnvironment

  Sets if the environment should be rendered to the cube map.

- int layer

  The layer of the cubemap.

- RenderingRunType cubemapLayer

The exact run type of this run (according to the actual layer).

- ElementaryRenderable * parentRenderable

  The renderable this technique operates on.

- TechniqueGroup * parentTechniqueGroup

  The TechniqueGroup this RenderedTechnique is attached to.

- SharedRuns * sharedRuns

  The SharedRuns this RenderedTechnique is attached to.

5.5.1 Detailed Description

Base abstract class of rendering a color cube map.

This technique defines that the final rendering of an object needs a cubmap of the colors of the surrounding environment.

5.5.2 Constructor & Destructor Documentation

5.5.2.1 ColorCubeMapRenderTechnique::ColorCubeMapRenderTechnique (unsigned long *startFrame*, unsigned long *cubeMapUpdateInterval*, unsigned int *cubeMapResolution*, bool *useDistCalc*, bool *useFaceAngleCalc*, float *distTolerance*, float *angleTolerance*, bool *updateAllFace*, bool *renderSelf*, bool *renderEnvironment*, int *layer*, ElementaryRenderable * *parentRenderable*, TechniqueGroup * *parentTechniqueGroup*)

Constructor.

Parameters:

startFrame adds an offset to the current frame number to help evenly distribute updates between frames

cubeMapUpdateInterval update frequency

cubeMapResolution color cubemap resolution

useDistCalc flag to skip cube face update if object is far away

useFaceAngleCalc flag to skip cube face update if face is neglible

distTolerance distance tolerance used in face skip

angleTolerance angle tolerance used in face skip

updateAllFace defines if all cubemap faces should be updated in a frame or only one face per frame

renderSelf sets if the object should be rendered to the cube map

renderEnvironment sets if the environment should be rendered to the cube map

layer the layer of this cubemap

parentRenderable the object to operate on

parentTechniqueGroup the TechniqueGroup this RenderedTechnique is attached to

5.5.3 Member Function Documentation

5.5.3.1 void ColorCubeMapRenderTechnique::update (unsigned long *frameNum*) [virtual]

Updates the resources in the given frame.

Parameters:

frameNum the actual framenumber

Reimplemented from RenderTechnique.

Reimplemented in OgreColorCubeMapRenderTechnique.

5.5.3.2 void ColorCubeMapRenderTechnique::runChanged (RenderingRunType *runType*, RenderingRun * *run*) [virtual]

Called after one of he shared runs changes.

Parameters:

runType enum describing the type of the changed run

run pointer to the changed RenderingRun

Reimplemented from CubeMapRenderTechnique.

5.5.3.3 void ColorCubeMapRenderTechnique::runUpdated (RenderingRunType *runType*, RenderingRun * *run*) [virtual]

Called after one of he shared runs updates.

Parameters:

runType enum describing the type of the updated run

run pointer to the updated RenderingRun

Reimplemented from CubeMapRenderTechnique.

5.5.3.4 virtual void ColorCubeMapRenderTechnique::colorCubeMapRunChanged (RenderingRun * *run*) [protected, pure virtual]

Called if the color cubemap rendering run object changes.

Parameters:

run pointer to the new rendering run object

Implemented in OgreColorCubeMapRenderTechnique.

5.5.3.5 virtual void ColorCubeMapRenderTechnique::colorCubeMapRunUpdated (RenderingRun * *run*) [protected, pure virtual]

Called if the color cubemap rendering run object is updated.

Parameters:

run pointer to the rendering run object

Implemented in OgreColorCubeMapRenderTechnique.

5.5.3.6 virtual RenderingRun* CubeMapRenderTechnique::createCubeMapRun () [protected, pure virtual, inherited]

Creates a CubeMapRenderingRun.

Returns:

the new CubeMapRenderingRun instance.

Implemented in OgreCubeMapRenderTechnique.

5.5.3.7 virtual void CubeMapRenderTechnique::cubeMapRunChanged (RenderingRun * *run*) [protected, pure virtual, inherited]

Called if the changed run is a CubeMapRenderingRun.

Parameters:

run pointer to the new CubeMapRenderingRun

Implemented in OgreCubeMapRenderTechnique.

5.5.3.8 virtual void CubeMapRenderTechnique::cubeMapRunUpdated (RenderingRun * *run*) [protected, pure virtual, inherited]

Called if the updated run is a CubeMapRenderingRun.

Parameters:

run pointer to the updated CubeMapRenderingRun

Implemented in OgreCubeMapRenderTechnique.

5.5.3.9 virtual class OgreRenderTechnique* RenderTechnique::asOgreRenderTechnique () [inline, virtual, inherited]

Conversion to OgreRenderTechnique.

This function is needed because of virtual inheritance.

Reimplemented in OgreRenderTechnique.

5.5.4 Member Data Documentation

5.5.4.1 bool CubeMapRenderTechnique::useDistCalc [protected, inherited]

a flag to skip cube face update if object is far away or too small.

See also:

distTolerance

5.5.4.2 bool CubeMapRenderTechnique::useFaceAngleCalc [protected, inherited]

a flag to skip cube face update the face is neglible.

See also:

angleTolerance

5.5.4.3 float CubeMapRenderTechnique::distTolerance [protected, inherited]

A value used in face skip test.

The higher this value gets the more precise, but slower the method will be.

5.5.4.4 float CubeMapRenderTechnique::angleTolerance [protected, inherited]

A value used in face skip test.

The higher this value gets the more precise, but slower the method will be.

5.5.4.5 int CubeMapRenderTechnique::layer [protected, inherited]

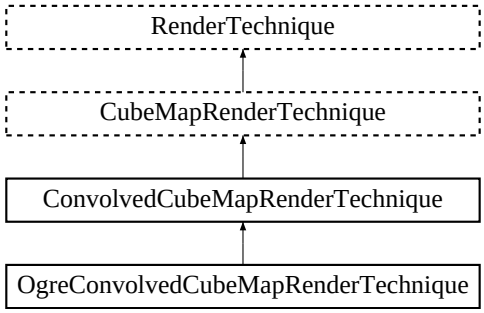
The layer of the cubemap.

This extension was created to achieve effect such as multiple reflections and refractions that require more complete sampling of the scene. With this extension we can render the environment into several cubemap layers eg. with depth peeling.

5.6 ConvolvedCubeMapRenderTechnique Class Reference

Base abstract class of rendering a color cube map which will be reduced.

Inheritance diagram for ConvolvedCubeMapRenderTechnique::



Public Member Functions

- ConvolvedCubeMapRenderTechnique (unsigned long startFrame, unsigned long cubeMapUpdateInterval, unsigned int cubeMapResolution, unsigned int reducedCubeMapResolution, bool useDistCalc, bool useFaceAngleCalc, float distTolerance, float angleTolerance, bool updateAllFace, bool renderSelf, bool renderEnvironment, ElementaryRenderable *parentRenderable, TechniqueGroup *parentTechniqueGroup)

  Constructor.

- virtual void update (unsigned long frameNum)

  Updates the resources in the given frame.

- void runChanged (RenderingRunType runType, RenderingRun *run)

  Called after one of he shared runs changes.

- void runUpdated (RenderingRunType runType, RenderingRun *run)

  Called after one of he shared runs updates.

- virtual class OgreRenderTechnique * asOgreRenderTechnique ()

  Conversion to OgreRenderTechnique.

- ElementaryRenderable * getParentRenderable ()

  Retrieves the renderable this technique operates on.

Protected Member Functions

- virtual void reducedCubeMapRunChanged (RenderingRun *run)=0

  Called if the changed run is a ReducedCubeMapRenderingRun.

- virtual void colorCubeMapRunChanged (RenderingRun *run)=0

  Called if the changed run is a CubeMapRenderingRun (containing colors of the environment).

- virtual RenderingRun * createReducedCubeMapRun ()=0

  Creates a ReducedCubeMapRenderingRun.

- virtual RenderingRun * createCubeMapRun ()=0

  Creates a CubeMapRenderingRun.

- virtual void cubeMapRunChanged (RenderingRun *run)=0

  Called if the changed run is a CubeMapRenderingRun.

- virtual void cubeMapRunUpdated (RenderingRun *run)=0

  Called if the updated run is a CubeMapRenderingRun.

Protected Attributes

- unsigned int reducedCubeMapResolution

  The resolution of the downsampled cubemap created by this run.

- bool useDistCalc

  a flag to skip cube face update if object is far away or too small.

- bool useFaceAngleCalc

  a flag to skip cube face update the face is neglible.

- float distTolerance

  A value used in face skip test.

- float angleTolerance

  A value used in face skip test.

- bool updateAllFace

  defines if all cubemap faces should be updated in a frame or only one face per frame

- unsigned long cubeMapUpdateInterval

  color-cubemap update frequency

- unsigned int cubeMapResolution

  color-cubemap resolution

- unsigned long startFrame

  offset in frame number used during update

- bool renderSelf

  Sets if the object should be rendered to the cube map.

- bool renderEnvironment

Sets if the environment should be rendered to the cube map.

- int layer

  The layer of the cubemap.

- RenderingRunType cubemapLayer

  The exact run type of this run (according to the actual layer).

- ElementaryRenderable * parentRenderable

  The renderable this technique operates on.

- TechniqueGroup * parentTechniqueGroup

  The TechniqueGroup this RenderedTechnique is attached to.

- SharedRuns * sharedRuns

  The SharedRuns this RenderedTechnique is attached to.

5.6.1 Detailed Description

Base abstract class of rendering a color cube map which will be reduced.

This technique defines that the final rendering of an object needs a reduced sized cubmap of the colors of the surrounding environment. This reduced sized cubemap is created with averaging the original cubemap. This reduced cubemap can easily be convolved in the final shading to acheve special effects like diffuse reflections.

5.6.2 Constructor & Destructor Documentation

5.6.2.1 ConvolvedCubeMapRenderTechnique::ConvolvedCubeMapRenderTechnique (unsigned long *startFrame*, unsigned long *cubeMapUpdateInterval*, unsigned int *cubeMapResolution*, unsigned int *reducedCubeMapResolution*, bool *useDistCalc*, bool *useFaceAngleCalc*, float *distTolerance*, float *angleTolerance*, bool *updateAllFace*, bool *renderSelf*, bool *renderEnvironment*, ElementaryRenderable * *parentRenderable*, TechniqueGroup * *parentTechniqueGroup*)

Constructor.

Parameters:

startFrame adds an offset to the current frame number to help evenly distribute updates between frames

cubeMapUpdateInterval update frequency

cubeMapResolution color cubemap resolution

reducedCubeMapResolution the resolution of the reduced cube map

useDistCalc flag to skip cube face update if object is far away

useFaceAngleCalc flag to skip cube face update if face is neglible

distTolerance distance tolerance used in face skip

angleTolerance angle tolerance used in face skip

updateAllFace defines if all cubemap faces should be updated in a frame or only one face per frame

renderSelf sets if the object should be rendered to the cube map

renderEnvironment sets if the environment should be rendered to the cube map

parentRenderable the object to operate on

parentTechniqueGroup the TechniqueGroup this RenderedTechnique is attached to

5.6.3 Member Function Documentation

5.6.3.1 void ConvolvedCubeMapRenderTechnique::update (unsigned long *frameNum*) [virtual]

Updates the resources in the given frame.

Parameters:

frameNum the actual framenumber

Reimplemented from RenderTechnique.

Reimplemented in OgreConvolvedCubeMapRenderTechnique.

5.6.3.2 void ConvolvedCubeMapRenderTechnique::runChanged (RenderingRunType *runType*, RenderingRun * *run*) [virtual]

Called after one of he shared runs changes.

Parameters:

runType enum describing the type of the changed run

run pointer to the changed RenderingRun

Reimplemented from CubeMapRenderTechnique.

5.6.3.3 virtual void ConvolvedCubeMapRenderTechnique::reducedCubeMapRunChanged (RenderingRun * *run*) [protected, pure virtual]

Called if the changed run is a ReducedCubeMapRenderingRun.

Parameters:

run pointer to the changed ReducedCubeMapRenderingRun

Implemented in OgreConvolvedCubeMapRenderTechnique.

5.6.3.4 virtual void ConvolvedCubeMapRenderTechnique::colorCubeMapRunChanged (RenderingRun * *run*) [protected, pure virtual]

Called if the changed run is a CubeMapRenderingRun (containing colors of the environment).

Parameters:

run pointer to the changed CubeMapRenderingRun

Implemented in OgreConvolvedCubeMapRenderTechnique.

5.6.3.5 virtual RenderingRun* ConvolvedCubeMapRenderTechnique::createReducedCubeMapRun () [protected, pure virtual]

Creates a ReducedCubeMapRenderingRun.

Returns:

the new ReducedCubeMapRenderingRun instance.

Implemented in OgreConvolvedCubeMapRenderTechnique.

5.6.3.6 void CubeMapRenderTechnique::runUpdated (RenderingRunType *runType*, RenderingRun * *run*) [virtual, inherited]

Called after one of he shared runs updates.

Parameters:

runType enum describing the type of the updated run

run pointer to the updated RenderingRun

Reimplemented from RenderTechnique.

Reimplemented in ColorCubeMapRenderTechnique, and DistanceCubeMapRenderTechnique.

5.6.3.7 virtual RenderingRun* CubeMapRenderTechnique::createCubeMapRun () [protected, pure virtual, inherited]

Creates a CubeMapRenderingRun.

Returns:

the new CubeMapRenderingRun instance.

Implemented in OgreCubeMapRenderTechnique.

5.6.3.8 virtual void CubeMapRenderTechnique::cubeMapRunChanged (RenderingRun * *run*) [protected, pure virtual, inherited]

Called if the changed run is a CubeMapRenderingRun.

Parameters:

run pointer to the new CubeMapRenderingRun

Implemented in OgreCubeMapRenderTechnique.

5.6.3.9 virtual void CubeMapRenderTechnique::cubeMapRunUpdated (RenderingRun * *run*) [protected, pure virtual, inherited]

Called if the updated run is a CubeMapRenderingRun.

Parameters:

run pointer to the updated CubeMapRenderingRun

Implemented in OgreCubeMapRenderTechnique.

5.6.3.10 virtual class OgreRenderTechnique* RenderTechnique::asOgreRenderTechnique () [inline, virtual, inherited]

Conversion to OgreRenderTechnique.

This function is needed because of virtual inheritance.

Reimplemented in OgreRenderTechnique.

5.6.4 Member Data Documentation

5.6.4.1 bool CubeMapRenderTechnique::useDistCalc [protected, inherited]

a flag to skip cube face update if object is far away or too small.

See also:

distTolerance

5.6.4.2 bool CubeMapRenderTechnique::useFaceAngleCalc [protected, inherited]

a flag to skip cube face update the face is neglible.

See also:

angleTolerance

5.6.4.3 float CubeMapRenderTechnique::distTolerance [protected, inherited]

A value used in face skip test.

The higher this value gets the more precise, but slower the method will be.

5.6.4.4 float CubeMapRenderTechnique::angleTolerance [protected, inherited]

A value used in face skip test.

The higher this value gets the more precise, but slower the method will be.

5.6.4.5 int CubeMapRenderTechnique::layer [protected, inherited]

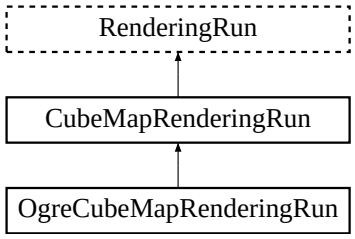
The layer of the cubemap.

This extension was created to achieve effect such as multiple reflections and refractions that require more complete sampling of the scene. With this extension we can render the environment into several cubemap layers eg. with depth peeling.

5.7 CubeMapRenderingRun Class Reference

Base abstract class that defines a rendering process of a cubemap.

Inheritance diagram for CubeMapRenderingRun::



Public Member Functions

- CubeMapRenderingRun (unsigned long startFrame, unsigned long updateInterval, unsigned int resolution, bool useDistCalc, bool useFaceAngleCalc, float distTolerance, float angleTolerance, bool updateAllFace, bool renderSelf, bool renderEnvironment)

  Constructor.

- bool update (unsigned long frameNum)

  Calls updateFrame() if the run needs update according to its starting frame and update interval and has not been allready updated in this frame.

- virtual class OgreRenderingRun * asOgreRenderingRun ()

  Conversion to OgreRenderRun.

- virtual bool canJoin (RenderingRun *run)

  Returns true if two runs can be joined.

Protected Member Functions

- virtual void createCubeMap ()=0

  Creates a cubemap texture used for the color-cubemap.

- virtual void updateCubeFace (int facenum)=0

  Updates one face of the cubemap.

- virtual bool faceNeedsUpdate (int facenum)=0

  Checks if a cubemap face needs to be updated.

- virtual void updateFrame (unsigned long frameNum)

  This function does the actual update in a frame.

- virtual bool needUpdate (unsigned long frameNum)

  Returns if this run needs update.

Protected Attributes

- bool updateAllFace

  defines if all cubemap faces should be updated in a frame or only one face per frame

- unsigned char currentFace

  the number of the face to be updated

- unsigned int resolution

  the resolution of the cubemap texture that was created by this run

- bool useDistCalc

  a flag to skip cube face update if object is far away or too small.

- bool useFaceAngleCalc

  a flag to skip cube face update the face is neglible.

- float distTolerance

  A value used in face skip test.

- float angleTolerance

  A value used in face skip test.

- bool renderSelf

  sets if the object should be rendered to the cube map

- bool renderEnvironment

  sets if the environment should be rendered to the cube map

- unsigned long lastupdated

  The number of the last frame this run was updated.

- unsigned long startFrame

  The number of the frame this run should be updated first.

- unsigned long updateInterval

  Refresh frequency in frames.

5.7.1 Detailed Description

Base abstract class that defines a rendering process of a cubemap.

5.7.2 Constructor & Destructor Documentation

5.7.2.1 CubeMapRenderingRun::CubeMapRenderingRun (unsigned long *startFrame*, unsigned long *updateInterval*, unsigned int *resolution*, bool *useDistCalc*, bool *useFaceAngleCalc*, float *distTolerance*, float *angleTolerance*, bool *updateAllFace*, bool *renderSelf*, bool *renderEnvironment*)

Constructor.

Parameters:

startFrame adds an offset to the current frame number to help evenly distribute updates between frames

updateInterval update frequency

resolution color cubemap resolution

useDistCalc flag to skip cube face update if object is far away

useFaceAngleCalc flag to skip cube face update if face is neglible

distTolerance distance tolerance used in face skip

angleTolerance angle tolerance used in face skip

updateAllFace defines if all cubemap faces should be updated in a frame or only one face per frame

renderSelf sets if the object should be rendered to the cube map

renderEnvironment sets if the environment should be rendered to the cube map

5.7.3 Member Function Documentation

5.7.3.1 virtual void CubeMapRenderingRun::updateCubeFace (int *facenum*) [inline, protected, pure virtual]

Updates one face of the cubemap.

Parameters:

facenum the number of the face to be updated

Implemented in OgreCubeMapRenderingRun.

5.7.3.2 virtual bool CubeMapRenderingRun::faceNeedsUpdate (int *facenum*) [protected, pure virtual]

Checks if a cubemap face needs to be updated.

If the object we are updating the cubemap for is far from the camera, or too small, or the given cubemapface does not have significant effect on the rendering the face can be skipped.

Parameters:

facenum the number of the face to be checked

Implemented in OgreCubeMapRenderingRun.

5.7.3.3 void CubeMapRenderingRun::updateFrame (unsigned long *frameNum*) [protected, virtual]

This function does the actual update in a frame.

Parameters:

frameNum current frame number

Reimplemented from RenderingRun.

5.7.3.4 virtual class OgreRenderingRun* RenderingRun::asOgreRenderingRun () [inline, virtual, inherited]

Conversion to OgreRenderRun.

This function is needed because of virtual inheritance.

Reimplemented in OgreRenderingRun, and OgreRenderingRun.

5.7.3.5 virtual bool RenderingRun::canJoin (RenderingRun * *run*) [inline, virtual, inherited]

Returns true if two runs can be joined.

In some cases special requirements should stand to join two runs (even if they have the same type). Eg.: two caustic cube map generation technique should only be joined if they use the same material when rendering the caustic cubemap.

Reimplemented in OgreCausticCubeMapRenderingRun, OgreChildPSystemRenderingRun, and OgrePhotonMapRenderingRun.

5.7.3.6 virtual bool RenderingRun::needUpdate (unsigned long *frameNum*) [inline, protected, virtual, inherited]

Returns if this run needs update.

This tipically depends on the upate interval and the starting frame number.

Parameters:

frameNum current frame number

Reimplemented in OgreFocusingMapRenderingRun, and OgrePMEntryPointMapRenderingRun.

5.7.4 Member Data Documentation

5.7.4.1 bool CubeMapRenderingRun::useDistCalc [protected]

a flag to skip cube face update if object is far away or too small.

See also:

distTolerance

5.7.4.2 bool CubeMapRenderingRun::useFaceAngleCalc [protected]

a flag to skip cube face update the face is neglible.

See also:

angleTolerance

5.7.4.3 float CubeMapRenderingRun::distTolerance [protected]

A value used in face skip test.

The higher this value gets the more precise, but slower the method will be.

5.7.4.4 float CubeMapRenderingRun::angleTolerance [protected]

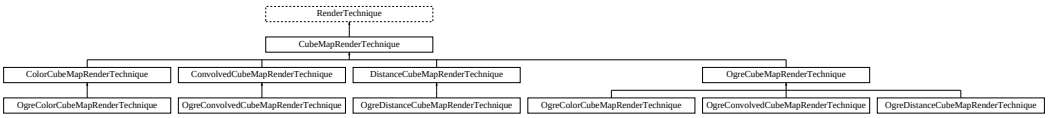
A value used in face skip test.

The higher this value gets the more precise, but slower the method will be.

5.8 CubeMapRenderTechnique Class Reference

Base abstract class of rendering a cube map.

Inheritance diagram for CubeMapRenderTechnique::



Public Member Functions

- CubeMapRenderTechnique (unsigned long startFrame, unsigned long cubeMapUpdateInterval, unsigned int cubeMapResolution, bool useDistCalc, bool useFaceAngleCalc, float distTolerance, float angleTolerance, bool updateAllFace, bool renderSelf, bool renderEnvironment, int layer, ElementaryRenderable *parentRenderable, TechniqueGroup *parentTechniqueGroup)

  Constructor.

- void runChanged (RenderingRunType runType, RenderingRun *run)

  Called after one of he shared runs changes.

- void runUpdated (RenderingRunType runType, RenderingRun *run)

  Called after one of he shared runs updates.

- virtual void update (unsigned long frameNum)

  Updates the resources in the given frame.

- virtual class OgreRenderTechnique * asOgreRenderTechnique ()

  Conversion to OgreRenderTechnique.

- ElementaryRenderable * getParentRenderable ()

  Retrieves the renderable this technique operates on.

Protected Member Functions

- virtual RenderingRun * createCubeMapRun ()=0

  Creates a CubeMapRenderingRun.

- virtual void cubeMapRunChanged (RenderingRun *run)=0

  Called if the changed run is a CubeMapRenderingRun.

- virtual void cubeMapRunUpdated (RenderingRun *run)=0

  Called if the updated run is a CubeMapRenderingRun.

Protected Attributes

- bool useDistCalc

  a flag to skip cube face update if object is far away or too small.

- bool useFaceAngleCalc

  a flag to skip cube face update the face is neglible.

- float distTolerance

  A value used in face skip test.

- float angleTolerance

  A value used in face skip test.

- bool updateAllFace

  defines if all cubemap faces should be updated in a frame or only one face per frame

- unsigned long cubeMapUpdateInterval

  color-cubemap update frequency

- unsigned int cubeMapResolution

  color-cubemap resolution

- unsigned long startFrame

  offset in frame number used during update

- bool renderSelf

  Sets if the object should be rendered to the cube map.

- bool renderEnvironment

  Sets if the environment should be rendered to the cube map.

- int layer

  The layer of the cubemap.

- RenderingRunType cubemapLayer

  The exact run type of this run (according to the actual layer).

- ElementaryRenderable * parentRenderable

  The renderable this technique operates on.

- TechniqueGroup * parentTechniqueGroup

  The TechniqueGroup this RenderedTechnique is attached to.

- SharedRuns * sharedRuns

  The SharedRuns this RenderedTechnique is attached to.

5.8.1 Detailed Description

Base abstract class of rendering a cube map.

This technique defines that the final rendering of an object needs a cubmap of the surrounding environment and/or the object itself.

5.8.2 Constructor & Destructor Documentation

5.8.2.1 CubeMapRenderTechnique::CubeMapRenderTechnique (unsigned long *startFrame*, unsigned long *cubeMapUpdateInterval*, unsigned int *cubeMapResolution*, bool *useDistCalc*, bool *useFaceAngleCalc*, float *distTolerance*, float *angleTolerance*, bool *updateAllFace*, bool *renderSelf*, bool *renderEnvironment*, int *layer*, ElementaryRenderable * *parentRenderable*, TechniqueGroup * *parentTechniqueGroup*)

Constructor.

Parameters:

startFrame adds an offset to the current frame number to help evenly distribute updates between frames

cubeMapUpdateInterval update frequency

cubeMapResolution color cubemap resolution

useDistCalc flag to skip cube face update if object is far away

useFaceAngleCalc flag to skip cube face update if face is neglible

distTolerance distance tolerance used in face skip

angleTolerance angle tolerance used in face skip

updateAllFace defines if all cubemap faces should be updated in a frame or only one face per frame

renderSelf sets if the object should be rendered to the cube map

renderEnvironment sets if the environment should be rendered to the cube map

layer the layer of this cubemap

parentRenderable the object to operate on

parentTechniqueGroup the TechniqueGroup this RenderedTechnique is attached to

5.8.3 Member Function Documentation

5.8.3.1 void CubeMapRenderTechnique::runChanged (RenderingRunType *runType*, RenderingRun * *run*) [virtual]

Called after one of he shared runs changes.

Parameters:

runType enum describing the type of the changed run

run pointer to the changed RenderingRun

Reimplemented from RenderTechnique.

Reimplemented in ColorCubeMapRenderTechnique, ConvolvedCubeMapRenderTechnique, and DistanceCubeMapRenderTechnique.

5.8.3.2 void CubeMapRenderTechnique::runUpdated (RenderingRunType *runType*, RenderingRun * *run*) [virtual]

Called after one of he shared runs updates.

Parameters:

runType enum describing the type of the updated run

run pointer to the updated RenderingRun

Reimplemented from RenderTechnique.

Reimplemented in ColorCubeMapRenderTechnique, and DistanceCubeMapRenderTechnique.

5.8.3.3 virtual RenderingRun* CubeMapRenderTechnique::createCubeMapRun () [protected, pure virtual]

Creates a CubeMapRenderingRun.

Returns:

the new CubeMapRenderingRun instance.

Implemented in OgreCubeMapRenderTechnique.

5.8.3.4 virtual void CubeMapRenderTechnique::cubeMapRunChanged (RenderingRun * *run*) [protected, pure virtual]

Called if the changed run is a CubeMapRenderingRun.

Parameters:

run pointer to the new CubeMapRenderingRun

Implemented in OgreCubeMapRenderTechnique.

5.8.3.5 virtual void CubeMapRenderTechnique::cubeMapRunUpdated (RenderingRun * *run*) [protected, pure virtual]

Called if the updated run is a CubeMapRenderingRun.

Parameters:

run pointer to the updated CubeMapRenderingRun

Implemented in OgreCubeMapRenderTechnique.

5.8.3.6 virtual void RenderTechnique::update (unsigned long *frameNum*) [inline, virtual, inherited]

Updates the resources in the given frame.

A RenderTechnique is usually need some resources from several runs, so these runs will be updated.

Parameters:

frameNum the actual framenumber

Reimplemented in ColorCubeMapRenderTechnique, ConvolvedCubeMapRenderTechnique, DistanceCubeMapRenderTechnique, HierarchicalParticleSystemTechnique, IllumVolumeRenderTechnique, OgreCausticReceiverRenderTechnique, OgreColorCubeMapRenderTechnique, OgreConvolvedCubeMapRenderTechnique, OgreDepthShadowReceiverRenderTechnique, OgreDistanceCubeMapRenderTechnique, OgreFireRenderTechnique, OgrePathMapRenderTechnique, and OgreSBBRenderTechnique.

5.8.3.7 virtual class OgreRenderTechnique* RenderTechnique::asOgreRenderTechnique () [inline, virtual, inherited]

Conversion to OgreRenderTechnique.

This function is needed because of virtual inheritance.

Reimplemented in OgreRenderTechnique.

5.8.4 Member Data Documentation

5.8.4.1 bool CubeMapRenderTechnique::useDistCalc [protected]

a flag to skip cube face update if object is far away or too small.

See also:

distTolerance

5.8.4.2 bool CubeMapRenderTechnique::useFaceAngleCalc [protected]

a flag to skip cube face update the face is neglible.

See also:

angleTolerance

5.8.4.3 float CubeMapRenderTechnique::distTolerance [protected]

A value used in face skip test.

The higher this value gets the more precise, but slower the method will be.

5.8.4.4 float CubeMapRenderTechnique::angleTolerance [protected]

A value used in face skip test.

The higher this value gets the more precise, but slower the method will be.

5.8.4.5 int CubeMapRenderTechnique::layer [protected]

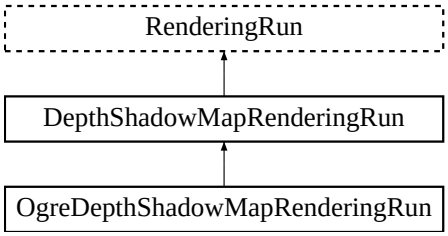
The layer of the cubemap.

This extension was created to achieve effect such as multiple reflections and refractions that require more complete sampling of the scene. With this extension we can render the environment into several cubemap layers eg. with depth peeling.

5.9 DepthShadowMapRenderingRun Class Reference

Base abstract class that defines a rendering process of a shadow map.

Inheritance diagram for DepthShadowMapRenderingRun::



Public Member Functions

- DepthShadowMapRenderingRun (unsigned int resolutionX, unsigned int resolutionY)

  Constructor.

- bool update (unsigned long frameNum)

  Calls updateFrame() if the run needs update according to its starting frame and update interval and has not been allready updated in this frame.

- virtual class OgreRenderingRun * asOgreRenderingRun ()

  Conversion to OgreRenderRun.

- virtual bool canJoin (RenderingRun *run)

  Returns true if two runs can be joined.

Protected Member Functions

- virtual void createDepthMap ()=0

  Creates the depth map texture (2D or CUBE according to light type).

- virtual void updateDepthCubeFace (int facenum)=0

  Updates one face of the depth cubemap (used only in case of point lights).

- virtual void updateDepthMap ()=0

  Updates the depth map (in case of directional and spot lights).

- virtual void updateFrame (unsigned long frameNum)=0

  This function does the actual update in a frame.

- virtual bool needUpdate (unsigned long frameNum)

  Returns if this run needs update.

Protected Attributes

- unsigned int resolutionX
  width of the depth map texture
- unsigned int resolutionY
  height of the depth map texture
- unsigned long lastupdated
  The number of the last frame this run was updated.
- unsigned long startFrame
  The number of the frame this run should be updated first.
- unsigned long updateInterval
  Refresh frequency in frames.

5.9.1 Detailed Description

Base abstract class that defines a rendering process of a shadow map.

A shadow map stores depth values from the lightsource.

5.9.2 Constructor & Destructor Documentation

5.9.2.1 DepthShadowMapRenderingRun::DepthShadowMapRenderingRun (unsigned int *resolutionX*, unsigned int *resolutionY*) [inline]

Constructor.

Parameters:

resolutionX width of the depth map texture
resolutionY height of the depth map texture

5.9.3 Member Function Documentation

5.9.3.1 virtual void DepthShadowMapRenderingRun::updateDepthCubeFace (int *facenum*) [inline, protected, pure virtual]

Updates one face of the depth cubemap (used only in case of point lights).

Parameters:

facenum the number of the face to be updated

Implemented in OgreDepthShadowMapRenderingRun.

5.9.3.2 virtual void DepthShadowMapRenderingRun::updateFrame (unsigned long *frameNum*) [protected, pure virtual]

This function does the actual update in a frame.

Parameters:

frameNum current frame number

Reimplemented from RenderingRun.

Implemented in OgreDepthShadowMapRenderingRun.

5.9.3.3 virtual class OgreRenderingRun* RenderingRun::asOgreRenderingRun () [inline, virtual, inherited]

Conversion to OgreRenderRun.

This function is needed because of virtual inheritance.

Reimplemented in OgreRenderingRun, and OgreRenderingRun.

5.9.3.4 virtual bool RenderingRun::canJoin (RenderingRun * *run*) [inline, virtual, inherited]

Returns true if two runs can be joined.

In some cases special requirements should stand to join two runs (even if they have the same type). Eg.: two caustic cube map generation technique should only be joined if they use the same material when rendering the caustic cubemap.

Reimplemented in OgreCausticCubeMapRenderingRun, OgreChildPSystemRenderingRun, and OgrePhotonMapRenderingRun.

5.9.3.5 virtual bool RenderingRun::needUpdate (unsigned long *frameNum*) [inline, protected, virtual, inherited]

Returns if this run needs update.

This tipically depends on the upate interval and the starting frame number.

Parameters:

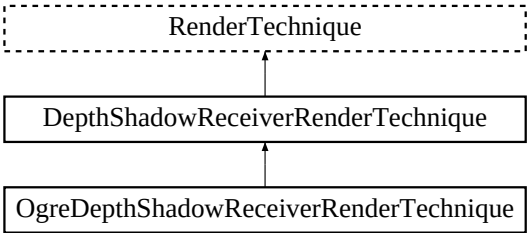
frameNum current frame number

Reimplemented in OgreFocusingMapRenderingRun, and OgrePMEntryPointMapRenderingRun.

5.10 DepthShadowReceiverRenderTechnique Class Reference

Base abstract class of rendering an object that recieves shadows with depth map shadow technique.

Inheritance diagram for DepthShadowReceiverRenderTechnique::



Public Member Functions

- DepthShadowReceiverRenderTechnique (ElementaryRenderable *parentRenderable, TechniqueGroup *parentTechniqueGroup)
  Constructor.
- virtual void update (unsigned long frameNum)
  Updates the resources in the given frame.
- virtual void runChanged (RenderingRunType runType, RenderingRun *run)
  Called after one of he shared runs changes.
- virtual void runUpdated (RenderingRunType runType, RenderingRun *run)
  Called after one of he shared runs updates.
- virtual class OgreRenderTechnique * asOgreRenderTechnique ()
  Conversion to OgreRenderTechnique.
- ElementaryRenderable * getParentRenderable ()
  Retrieves the renderable this technique operates on.

Protected Attributes

- ElementaryRenderable * parentRenderable
  The renderable this technique operates on.
- TechniqueGroup * parentTechniqueGroup
  The TechniqueGroup this RenderedTechnique is attached to.
- SharedRuns * sharedRuns
  The SharedRuns this RenderedTechnique is attached to.

5.10.1 Detailed Description

Base abstract class of rendering an object that recieves shadows with depth map shadow technique.

5.10.2 Constructor & Destructor Documentation

5.10.2.1 DepthShadowReceiverRenderTechnique::DepthShadowReceiverRenderTechnique (ElementaryRenderable * *parentRenderable*, TechniqueGroup * *parentTechniqueGroup*)

Constructor.

Parameters:

parentRenderable the object to operate on

parentTechniqueGroup the TechniqueGroup this RenderedTechnique is attached to

5.10.3 Member Function Documentation

5.10.3.1 virtual void RenderTechnique::update (unsigned long *frameNum*) [inline, virtual, inherited]

Updates the resources in the given frame.

A RenderTechnique is usually need some resources from several runs, so these runs will be updated.

Parameters:

frameNum the actual framenumber

Reimplemented in ColorCubeMapRenderTechnique, ConvolvedCubeMapRenderTechnique, DistanceCubeMapRenderTechnique, HierarchicalParticleSystemTechnique, IllumVolumeRenderTechnique, OgreCausticReceiverRenderTechnique, OgreColorCubeMapRenderTechnique, OgreConvolvedCubeMapRenderTechnique, OgreDepthShadowReceiverRenderTechnique, OgreDistanceCubeMapRenderTechnique, OgreFireRenderTechnique, OgrePathMapRenderTechnique, and OgreSBBRenderTechnique.

5.10.3.2 virtual void RenderTechnique::runChanged (RenderingRunType *runType*, RenderingRun * *run*) [inline, virtual, inherited]

Called after one of he shared runs changes.

Parameters:

runType enum describing the type of the changed run

run pointer to the changed RenderingRun

Reimplemented in CausticCasterRenderTechnique, ColorCubeMapRenderTechnique, ConvolvedCubeMapRenderTechnique, CubeMapRenderTechnique, DistanceCubeMapRenderTechnique, HierarchicalParticleSystemTechnique, and IllumVolumeRenderTechnique.

5.10.3.3 virtual void RenderTechnique::runUpdated (RenderingRunType *runType*, RenderingRun * *run*) [inline, virtual, inherited]

Called after one of he shared runs updates.

Parameters:

runType enum describing the type of the updated run

run pointer to the updated RenderingRun

Reimplemented in CausticCasterRenderTechnique, ColorCubeMapRenderTechnique, CubeMapRenderTechnique, DistanceCubeMapRenderTechnique, HierarchicalParticleSystemTechnique, and IllumVolumeRenderTechnique.

5.10.3.4 virtual class OgreRenderTechnique* RenderTechnique::asOgreRenderTechnique () [inline, virtual, inherited]

Conversion to OgreRenderTechnique.

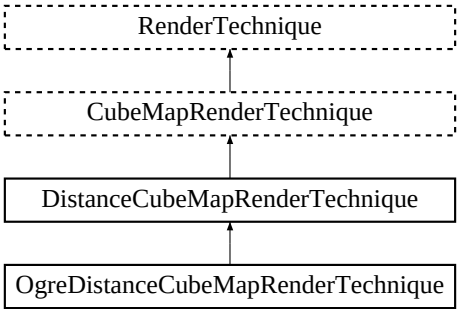
This function is needed because of virtual inheritance.

Reimplemented in OgreRenderTechnique.

5.11 DistanceCubeMapRenderTechnique Class Reference

Base abstract class of rendering a distance cube map.

Inheritance diagram for DistanceCubeMapRenderTechnique::



Public Member Functions

- DistanceCubeMapRenderTechnique (unsigned long startFrame, unsigned long cubeMapUpdateInterval, unsigned int cubeMapResolution, bool useDistCalc, bool useFaceAngleCalc, float distTolerance, float angleTolerance, bool updateAllFace, bool renderSelf, bool renderEnvironment, int layer, ElementaryRenderable *parentRenderable, TechniqueGroup *parentTechniqueGroup)

  Constructor.

- virtual void update (unsigned long frameNum)

  Updates the resources in the given frame.

- virtual void runUpdated (RenderingRunType runType, RenderingRun *run)

  Called after one of he shared runs updates.

- void runChanged (RenderingRunType runType, RenderingRun *run)

  Called after one of he shared runs changes.

- virtual class OgreRenderTechnique * asOgreRenderTechnique ()

  Conversion to OgreRenderTechnique.

- ElementaryRenderable * getParentRenderable ()

  Retrieves the renderable this technique operates on.

Protected Member Functions

- virtual void distanceCubeMapRunChanged (RenderingRun *run)=0

  Called if the changed run is a CubeMapRenderingRun (containing distances of the current layer).

- virtual void distanceCubeMapRunUpdated (RenderingRun *run)=0

  Called if the changed run is a CubeMapRenderingRun (containing distances of the current layer).

- virtual RenderingRun * createCubeMapRun ()=0

  Creates a CubeMapRenderingRun.

- virtual void cubeMapRunChanged (RenderingRun *run)=0

  Called if the changed run is a CubeMapRenderingRun.

- virtual void cubeMapRunUpdated (RenderingRun *run)=0

  Called if the updated run is a CubeMapRenderingRun.

Protected Attributes

- bool useDistCalc

  a flag to skip cube face update if object is far away or too small.

- bool useFaceAngleCalc

  a flag to skip cube face update the face is neglible.

- float distTolerance

  A value used in face skip test.

- float angleTolerance

  A value used in face skip test.

- bool updateAllFace

  defines if all cubemap faces should be updated in a frame or only one face per frame

- unsigned long cubeMapUpdateInterval

  color-cubemap update frequency

- unsigned int cubeMapResolution

  color-cubemap resolution

- unsigned long startFrame

  offset in frame number used during update

- bool renderSelf

  Sets if the object should be rendered to the cube map.

- bool renderEnvironment

  Sets if the environment should be rendered to the cube map.

- int layer

  The layer of the cubemap.

- RenderingRunType cubemapLayer

The exact run type of this run (according to the actual layer).

- ElementaryRenderable * parentRenderable

  The renderable this technique operates on.

- TechniqueGroup * parentTechniqueGroup

  The TechniqueGroup this RenderedTechnique is attached to.

- SharedRuns * sharedRuns

  The SharedRuns this RenderedTechnique is attached to.

5.11.1 Detailed Description

Base abstract class of rendering a distance cube map.

This technique defines that the final rendering of an object needs a cubemap of the distance of the surrounding environment from the cubemap center.

5.11.2 Constructor & Destructor Documentation

5.11.2.1 DistanceCubeMapRenderTechnique::DistanceCubeMapRenderTechnique (unsigned long *startFrame*, unsigned long *cubeMapUpdateInterval*, unsigned int *cubeMapResolution*, bool *useDistCalc*, bool *useFaceAngleCalc*, float *distTolerance*, float *angleTolerance*, bool *updateAllFace*, bool *renderSelf*, bool *renderEnvironment*, int *layer*, ElementaryRenderable * *parentRenderable*, TechniqueGroup * *parentTechniqueGroup*)

Constructor.

Parameters:

startFrame adds an offset to the current frame number to help evenly distribute updates between frames

cubeMapUpdateInterval update frequency

cubeMapResolution distance cubemap resolution

useDistCalc flag to skip cube face update if object is far away

useFaceAngleCalc flag to skip cube face update if face is neglible

distTolerance distance tolerance used in face skip

angleTolerance angle tolerance used in face skip

updateAllFace defines if all cubemap faces should be updated in a frame or only one face per frame

renderSelf sets if the object should be rendered to the cube map

renderEnvironment sets if the environment should be rendered to the cube map

layer the layer of this cubemap

parentRenderable the object to operate on

parentTechniqueGroup the TechniqueGroup this RenderedTechnique is attached to

5.11.3 Member Function Documentation

5.11.3.1 void DistanceCubeMapRenderTechnique::update (unsigned long *frameNum*) [virtual]

Updates the resources in the given frame.

Parameters:

frameNum the actual framenumber

Reimplemented from RenderTechnique.

Reimplemented in OgreDistanceCubeMapRenderTechnique.

5.11.3.2 void DistanceCubeMapRenderTechnique::runUpdated (RenderingRunType *runType*, RenderingRun * *run*) [virtual]

Called after one of he shared runs updates.

Parameters:

runType enum describing the type of the updated run

run pointer to the updated RenderingRun

Reimplemented from CubeMapRenderTechnique.

5.11.3.3 void DistanceCubeMapRenderTechnique::runChanged (RenderingRunType *runType*, RenderingRun * *run*) [virtual]

Called after one of he shared runs changes.

Parameters:

runType enum describing the type of the changed run

run pointer to the changed RenderingRun

Reimplemented from CubeMapRenderTechnique.

5.11.3.4 virtual void DistanceCubeMapRenderTechnique::distanceCubeMapRunChanged (RenderingRun * *run*) [protected, pure virtual]

Called if the changed run is a CubeMapRenderingRun (containing distances of the current layer).

Parameters:

run pointer to the changed CubeMapRenderingRun

Implemented in OgreDistanceCubeMapRenderTechnique.

5.11.3.5 virtual void DistanceCubeMapRenderTechnique::distanceCubeMapRunUpdated (RenderingRun * *run*) [protected, pure virtual]

Called if the changed run is a CubeMapRenderingRun (containing distances of the current layer).

Parameters:

run pointer to the changed CubeMapRenderingRun

Implemented in OgreDistanceCubeMapRenderTechnique.

5.11.3.6 virtual RenderingRun* CubeMapRenderTechnique::createCubeMapRun () [protected, pure virtual, inherited]

Creates a CubeMapRenderingRun.

Returns:

the new CubeMapRenderingRun instance.

Implemented in OgreCubeMapRenderTechnique.

5.11.3.7 virtual void CubeMapRenderTechnique::cubeMapRunChanged (RenderingRun * *run*) [protected, pure virtual, inherited]

Called if the changed run is a CubeMapRenderingRun.

Parameters:

run pointer to the new CubeMapRenderingRun

Implemented in OgreCubeMapRenderTechnique.

5.11.3.8 virtual void CubeMapRenderTechnique::cubeMapRunUpdated (RenderingRun * *run*) [protected, pure virtual, inherited]

Called if the updated run is a CubeMapRenderingRun.

Parameters:

run pointer to the updated CubeMapRenderingRun

Implemented in OgreCubeMapRenderTechnique.

5.11.3.9 virtual class OgreRenderTechnique* RenderTechnique::asOgreRenderTechnique () [inline, virtual, inherited]

Conversion to OgreRenderTechnique.

This function is needed because of virtual inheritance.

Reimplemented in OgreRenderTechnique.

5.11.4 Member Data Documentation

5.11.4.1 bool CubeMapRenderTechnique::useDistCalc [protected, inherited]

a flag to skip cube face update if object is far away or too small.

See also:

distTolerance

5.11.4.2 bool CubeMapRenderTechnique::useFaceAngleCalc [protected, inherited]

a flag to skip cube face update the face is neglible.

See also:

angleTolerance

5.11.4.3 float CubeMapRenderTechnique::distTolerance [protected, inherited]

A value used in face skip test.

The higher this value gets the more precise, but slower the method will be.

5.11.4.4 float CubeMapRenderTechnique::angleTolerance [protected, inherited]

A value used in face skip test.

The higher this value gets the more precise, but slower the method will be.

5.11.4.5 int CubeMapRenderTechnique::layer [protected, inherited]

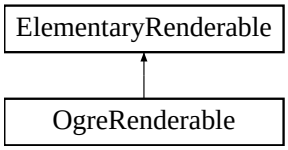
The layer of the cubemap.

This extension was created to achieve effect such as multiple reflections and refractions that require more complete sampling of the scene. With this extension we can render the environment into several cubemap layers eg. with depth peeling.

5.12 ElementaryRenderable Class Reference

Base abstract class for an elementary renderable.

Inheritance diagram for ElementaryRenderable::



Public Member Functions

- virtual void setVisible (bool visible)=0

  Shows or hides the renderable.

- virtual void setRenderGroup (unsigned char groupID)=0

  Sets the rendering group this renderable belongs to.

- virtual bool isVisible ()=0

  Retrieves if the renderable is hided or shown.

- virtual void updateBounds ()=0

  Updates bounding volumes.

5.12.1 Detailed Description

Base abstract class for an elementary renderable.

5.12.2 Member Function Documentation

5.12.2.1 virtual void ElementaryRenderable::setVisible (bool *visible*) [pure virtual]

Shows or hides the renderable.

Parameters:

visible visibility

Implemented in OgreRenderable.

5.12.2.2 virtual void ElementaryRenderable::setRenderGroup (unsigned char *groupID*) [pure virtual]

Sets the rendering group this renderable belongs to.

Rendering groups are to distinguish groups of rendering types (eg.: caustic casters, caustic receivers, shadow casters) if needed. Each group has a unique ID.

Parameters:

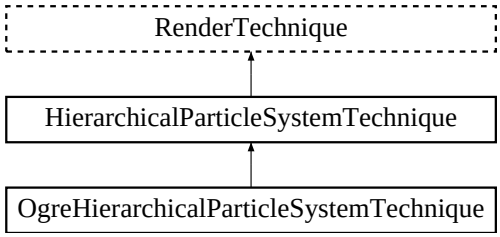
groupID the ID of the group to use

Implemented in OgreRenderable.

5.13 HierarchicalParticleSystemTechnique Class Reference

Base abstract class of rendering a hierarchical particle system.

Inheritance diagram for HierarchicalParticleSystemTechnique::



Public Member Functions

- HierarchicalParticleSystemTechnique (unsigned long startFrame, unsigned long impostorUpdateInterval, unsigned int impostorResolution, bool useDistCalc, bool perspectiveRendering, ElementaryRenderable *parentRenderable, TechniqueGroup *parentTechniqueGroup)

  Constructor.

- void update (unsigned long frameNum)

  Updates the resources in the given frame.

- void runChanged (RenderingRunType runType, RenderingRun *run)

  Called after one of he shared runs changes.

- void runUpdated (RenderingRunType runType, RenderingRun *run)

  Called after one of he shared runs updates.

- virtual class OgreRenderTechnique * asOgreRenderTechnique ()

  Conversion to OgreRenderTechnique.

- ElementaryRenderable * getParentRenderable ()

  Retrieves the renderable this technique operates on.

Protected Member Functions

- virtual RenderingRun * createChildPSysRenderingRun ()=0

  Creates the ChildParticleSystemRenderingRun needed by this technique.

- virtual void impostorChanged (RenderingRun *run)=0

  Called if the impostor rendering run changed.

- virtual void impostorUpdated (RenderingRun *run)=0

  Called if the impostor rendering run is updated.

Protected Attributes

- unsigned long impostorUpdateInterval

  update frequency of the impostor texture (image of the smaller system)

- unsigned int impostorResolution

  resolution of the impostor texture

- unsigned long startFrame

  offset in frame number used during update

- bool useDistCalc

  flag to skip impostor update if object is far away (//not used)

- bool perspectiveRendering

  sets if the impostor should be rendered with a perspective projection or orthogonal

- ElementaryRenderable * parentRenderable

  The renderable this technique operates on.

- TechniqueGroup * parentTechniqueGroup

  The TechniqueGroup this RenderedTechnique is attached to.

- SharedRuns * sharedRuns

  The SharedRuns this RenderedTechnique is attached to.

5.13.1 Detailed Description

Base abstract class of rendering a hierarchical particle system.

A hierarchical particle system is a particle system made out of a smaller particle system. It renders an image of the smaller particle system and multiplies this image to achieve a bigger particle system. This way fewer computation is needed to simulate large number of particles, while the trick is usually unnoticable.

5.13.2 Constructor & Destructor Documentation

5.13.2.1 HierarchicalParticleSystemTechnique::HierarchicalParticleSystemTechnique (unsigned long *startFrame*, unsigned long *impostorUpdateInterval*, unsigned int *impostorResolution*, bool *useDistCalc*, bool *perspectiveRendering*, ElementaryRenderable * *parentRenderable*, TechniqueGroup * *parentTechniqueGroup*)

Constructor.

Parameters:

startFrame adds an offset to the current frame number to help evenly distribute updates between frames

impostorUpdateInterval update frequency of the impostor texture (image of the smaller system)

impostorResolution resolution of the impostor texture

useDistCalc flag to skip impostor update if object is far away (//not used)

perspectiveRendering sets if the impostor should be rendered with a perspective projection or orthogonal

parentRenderable the object to operate on

parentTechniqueGroup the TechniqueGroup this RenderedTechnique is attached to

5.13.3 Member Function Documentation

5.13.3.1 void HierarchicalParticleSystemTechnique::update (unsigned long *frameNum*) [virtual]

Updates the resources in the given frame.

A RenderTechnique is usually need some resources from several runs, so these runs will be updated.

Parameters:

frameNum the actual framenumber

Reimplemented from RenderTechnique.

5.13.3.2 void HierarchicalParticleSystemTechnique::runChanged (RenderingRunType *runType*, RenderingRun * *run*) [virtual]

Called after one of he shared runs changes.

Parameters:

runType enum describing the type of the changed run

run pointer to the changed RenderingRun

Reimplemented from RenderTechnique.

5.13.3.3 void HierarchicalParticleSystemTechnique::runUpdated (RenderingRunType *runType*, RenderingRun * *run*) [virtual]

Called after one of he shared runs updates.

Parameters:

runType enum describing the type of the updated run

run pointer to the updated RenderingRun

Reimplemented from RenderTechnique.

5.13.3.4 virtual RenderingRun* HierarchicalParticleSystemTechnique::createChildPSysRenderingRun () [protected, pure virtual]

Creates the ChildParticleSystemRenderingRun needed by this technique.

Returns:

pointer to the ChildParticleSystemRenderingRun created instance

Implemented in OgreHierarchicalParticleSystemTechnique.

5.13.3.5 virtual void HierarchicalParticleSystemTechnique::impostorChanged (RenderingRun * *run*) [protected, pure virtual]

Called if the impostor rendering run changed.

Parameters:

run pointer to the new ChildParticleSystemRenderingRun instance to use

Implemented in OgreHierarchicalParticleSystemTechnique.

5.13.3.6 virtual void HierarchicalParticleSystemTechnique::impostorUpdated (RenderingRun * *run*) [protected, pure virtual]

Called if the impostor rendering run is updated.

Parameters:

run pointer to the updated ChildParticleSystemRenderingRu.

Implemented in OgreHierarchicalParticleSystemTechnique.

5.13.3.7 virtual class OgreRenderTechnique* RenderTechnique::asOgreRenderTechnique () [inline, virtual, inherited]

Conversion to OgreRenderTechnique.

This function is needed because of virtual inheritance.

Reimplemented in OgreRenderTechnique.

5.14 IlluminationManager Class Reference

Base abstract class of the illumination manager.

Public Member Functions

- virtual void update (unsigned long frameNumber)=0
  The function to be called to render one frame.
- virtual void sharedRunSplit (SharedRuns *old, SharedRuns *new1, SharedRuns *new2)
  The function to be called when a shared run is splitted.
- virtual void sharedRunJoin (SharedRuns *old1, SharedRuns *old2, SharedRuns *newsr)
  The function to be called when two shared runs are joined.
- virtual void joinSharedRuns ()
  Joins shared runs if needed.
- virtual void addSharedRuns (SharedRuns *runs)
  Register a shared run object.

5.14.1 Detailed Description

Base abstract class of the illumination manager.

The illumination manager is responsible for refreshing rendering techniques connected to visible renderables. It also has the resposibility to manage shared runs, to join and split them if needed.

5.14.2 Member Function Documentation

5.14.2.1 virtual void IlluminationManager::update (unsigned long *frameNumber*) [pure virtual]

The function to be called to render one frame.

This is the main refreshing function. It seasrches for visible objects, manages shared runs and updates render techniques. It should be called after all animations are done and before rendering to the frame buffer.

Parameters:

frameNumber current framenumber

5.14.2.2 virtual void IlluminationManager::sharedRunSplit (SharedRuns * *old*, SharedRuns * *new1*, SharedRuns * *new2*) [inline, virtual]

The function to be called when a shared run is splitted.

Parameters:

old pointer to the SharedRuns instance that is split

new1 pointer to one of the SharedRuns instance that remain after split

new2 pointer to the other SharedRuns instance that remain after split

5.14.2.3 virtual void IlluminationManager::sharedRunJoin (SharedRuns * *old1*, SharedRuns * *old2*, SharedRuns * *newsr*) [inline, virtual]

The function to be called when two shared runs are joined.

Parameters:

old1 pointer to one of the SharedRuns instance that are joined

old2 pointer to the other SharedRuns instance that are joined

newsr pointer to the resulting parent SharedRuns instance

5.14.2.4 virtual void IlluminationManager::joinSharedRuns () [inline, virtual]

Joins shared runs if needed.

Searches the registered shared run roots and join them if necessary (they are close enough).

5.14.2.5 virtual void IlluminationManager::addSharedRuns (SharedRuns * *runs*) [inline, virtual]

Register a shared run object.

Only called when new techniques are created.

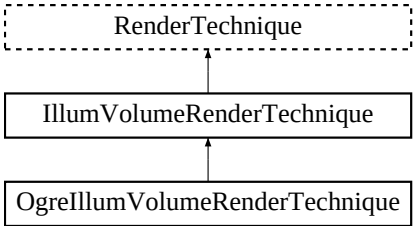
Parameters:

runs pointer to the SharedRuns instance to add

5.15 IllumVolumeRenderTechnique Class Reference

Base abstract class of rendering a light volume of a particle system.

Inheritance diagram for IllumVolumeRenderTechnique::



Public Member Functions

- IllumVolumeRenderTechnique (unsigned long startFrame, unsigned long illumVolumeUpdateInterval, unsigned int illumTextureResolution, unsigned int textureDepth, bool useDistCalc, bool useHierarchicalImpostor, ElementaryRenderable *parentRenderable, TechniqueGroup *parentTechniqueGroup)

  Constructor.

- void update (unsigned long frameNum)

  Updates the resources in the given frame.

- void runChanged (RenderingRunType runType, RenderingRun *run)

  Called after one of he shared runs changes.

- void runUpdated (RenderingRunType runType, RenderingRun *run)

  Called after one of he shared runs updates.

- virtual class OgreRenderTechnique * asOgreRenderTechnique ()

  Conversion to OgreRenderTechnique.

- ElementaryRenderable * getParentRenderable ()

  Retrieves the renderable this technique operates on.

Protected Member Functions

- virtual RenderingRun * createLightVolumeRenderingRun ()=0

  creates a light volume rendering run needed by this technique

- virtual void lightVolumeChanged (RenderingRun *run)=0

  Called if the LightVolumeRenderingRun is chaged.

- virtual void lightVolumeUpdated (RenderingRun *run)=0

  Called if the LightVolumeRenderingRun is updated.

- virtual void hierarchicalImpostorUpdated (RenderingRun *run)=0

  Called if the ChildParticleSystemRenderingRun is chaged.

Protected Attributes

- unsigned long illumVolumeUpdateInterval

  the update frequency of the light volume

- unsigned int illumTextureResolution

  the resolution of the light volume texture

- unsigned int textureDepth

  the number of layers to use (should be set to 1)

- unsigned long startFrame

  offset in frame number used during update

- bool useDistCalc

  flag to skip updates if the shaded particle system is far away (not used)

- bool useHierarchicalImpostor

  set this flag to true if the particle system is a hierarchical particle system

- ElementaryRenderable * parentRenderable

  The renderable this technique operates on.

- TechniqueGroup * parentTechniqueGroup

  The TechniqueGroup this RenderedTechnique is attached to.

- SharedRuns * sharedRuns

  The SharedRuns this RenderedTechnique is attached to.

5.15.1 Detailed Description

Base abstract class of rendering a light volume of a particle system.

Light volumes are used when self shadowing of particle systems should be simulated. Each layer of the volume represents the amount of transmitted light. The current implementation uses four grayscale layers, and places these layers to the four channel of the light volume texture.

5.15.2 Constructor & Destructor Documentation

5.15.2.1 IllumVolumeRenderTechnique::IllumVolumeRenderTechnique (unsigned long *startFrame*, unsigned long *illumVolumeUpdateInterval*, unsigned int *illumTextureResolution*, unsigned int *textureDepth*, bool *useDistCalc*, bool *useHierarchicalImpostor*, ElementaryRenderable * *parentRenderable*, TechniqueGroup * *parentTechniqueGroup*)

Constructor.

Parameters:

startFrame adds an offset to the current frame number to help evenly distribute updates between frames

illumVolumeUpdateInterval the update frequency of the light volume

illumTextureResolution the resolution of the light volume texture

textureDepth the number of layers to use (should be set to 1)

useDistCalc flag to skip updates if the shaded particle system is far away (not used)

useHierarchicalImpostor set this flag to true if the particle system is a hierarchical particle system

parentRenderable the object to operate on

parentTechniqueGroup the TechniqueGroup this RenderedTechnique is attached to

5.15.3 Member Function Documentation

5.15.3.1 void IllumVolumeRenderTechnique::update (unsigned long *frameNum*) [virtual]

Updates the resources in the given frame.

A RenderTechnique is usually need some resources from several runs, so these runs will be updated.

Parameters:

frameNum the actual framenumber

Reimplemented from RenderTechnique.

5.15.3.2 void IllumVolumeRenderTechnique::runChanged (RenderingRunType *runType*, RenderingRun * *run*) [virtual]

Called after one of he shared runs changes.

Parameters:

runType enum describing the type of the changed run

run pointer to the changed RenderingRun

Reimplemented from RenderTechnique.

5.15.3.3 void IllumVolumeRenderTechnique::runUpdated (RenderingRunType *runType*, RenderingRun * *run*) [virtual]

Called after one of he shared runs updates.

Parameters:

runType enum describing the type of the updated run

run pointer to the updated RenderingRun

Reimplemented from RenderTechnique.

5.15.3.4 virtual RenderingRun* IllumVolumeRenderTechnique::createLightVolumeRenderingRun () [protected, pure virtual]

creates a light volume rendering run needed by this technique

Returns:

pointer to the created LightVolumeRenderingRun instance

Implemented in OgreIllumVolumeRenderTechnique.

5.15.3.5 virtual void IllumVolumeRenderTechnique::lightVolumeChanged (RenderingRun * *run*) [protected, pure virtual]

Called if the LightVolumeRenderingRun is chaged.

Parameters:

pointer to the new LightVolumeRenderingRun instance to use

Implemented in OgreIllumVolumeRenderTechnique.

5.15.3.6 virtual void IllumVolumeRenderTechnique::lightVolumeUpdated (RenderingRun * *run*) [protected, pure virtual]

Called if the LightVolumeRenderingRun is updated.

Parameters:

pointer to the updated LightVolumeRenderingRun instance

Implemented in OgreIllumVolumeRenderTechnique.

5.15.3.7 virtual void IllumVolumeRenderTechnique::hierarchicalImpostorUpdated (RenderingRun * *run*) [protected, pure virtual]

Called if the ChildParticleSystemRenderingRun is chaged.

Only called if this particle system is a hierarchical particle system.

Parameters:

pointer to the new ChildParticleSystemRenderingRun instance to use

Implemented in OgreIllumVolumeRenderTechnique.

5.15.3.8 virtual class OgreRenderTechnique* RenderTechnique::asOgreRenderTechnique ()

`[inline, virtual, inherited]`

Conversion to OgreRenderTechnique.

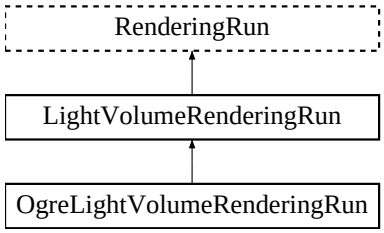
This function is needed because of virtual inheritance.

Reimplemented in OgreRenderTechnique.

5.16 LightVolumeRenderingRun Class Reference

Base abstract class that defines a rendering process of a light volume texture.

Inheritance diagram for LightVolumeRenderingRun::



Public Member Functions

- LightVolumeRenderingRun (unsigned long startFrame, unsigned long updateInterval, unsigned int resolution, unsigned int textureDepth)

  Constructor.

- bool update (unsigned long frameNum)

  Calls updateFrame() if the run needs update according to its starting frame and update interval and has not been allready updated in this frame.

- virtual class OgreRenderingRun * asOgreRenderingRun ()

  Conversion to OgreRenderRun.

- virtual bool canJoin (RenderingRun *run)

  Returns true if two runs can be joined.

Protected Member Functions

- virtual void updateFrame (unsigned long frameNum)=0

  This function does the actual update in a frame.

- virtual void createLightVolumeMap ()=0

  Creates a light volume map.

- virtual bool needUpdate (unsigned long frameNum)

  Returns if this run needs update.

Protected Attributes

- unsigned int resolution

  the resolution of the light volume map

- unsigned int textureDepth

  number of layers (should be 1)

- unsigned long lastupdated

  The number of the last frame this run was updated.

- unsigned long startFrame

  The number of the frame this run should be updated first.

- unsigned long updateInterval

  Refresh frequency in frames.

5.16.1 Detailed Description

Base abstract class that defines a rendering process of a light volume texture.

Light volumes are used when self shadowing of particle systems should be simulated. Each layer of the volume represents the amount of transmitted light. The current implementation uses four grayscale layers, and places these layers to the four channel of the light volume texture.

5.16.2 Constructor & Destructor Documentation

5.16.2.1 LightVolumeRenderingRun::LightVolumeRenderingRun (unsigned long *startFrame*, unsigned long *updateInterval*, unsigned int *resolution*, unsigned int *textureDepth*) [inline]

Constructor.

Parameters:

startFrame adds an offset to the current frame number to help evenly distribute updates between frames

updateInterval update frequency

resolution the resolution of the light volume texture

textureDepth the number of layers (should be set to 1)

5.16.3 Member Function Documentation

5.16.3.1 virtual void LightVolumeRenderingRun::updateFrame (unsigned long *frameNum*) [protected, pure virtual]

This function does the actual update in a frame.

Parameters:

frameNum current frame number

Reimplemented from RenderingRun.

Implemented in OgreLightVolumeRenderingRun.

5.16.3.2 virtual class OgreRenderingRun* RenderingRun::asOgreRenderingRun () [inline, virtual, inherited]

Conversion to OgreRenderRun.

This function is needed because of virtual inheritance.

Reimplemented in OgreRenderingRun, and OgreRenderingRun.

5.16.3.3 virtual bool RenderingRun::canJoin (RenderingRun * *run*) [inline, virtual, inherited]

Returns true if two runs can be joined.

In some cases special requirements should stand to join two runs (even if they have the same type). Eg.: two caustic cube map generation technique should only be joined if they use the same material when rendering the caustic cubemap.

Reimplemented in OgreCausticCubeMapRenderingRun, OgreChildPSystemRenderingRun, and OgrePhotonMapRenderingRun.

5.16.3.4 virtual bool RenderingRun::needUpdate (unsigned long *frameNum*) [inline, protected, virtual, inherited]

Returns if this run needs update.

This tipically depends on the upate interval and the starting frame number.

Parameters:

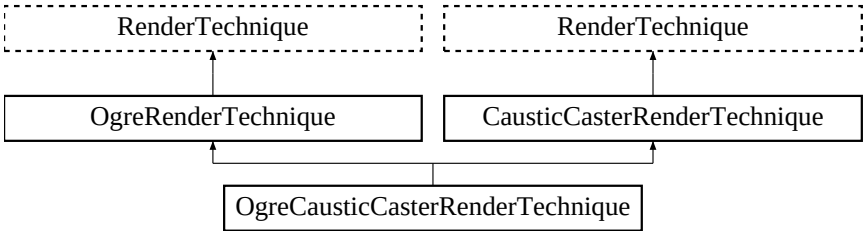
frameNum current frame number

Reimplemented in OgreFocusingMapRenderingRun, and OgrePMEntryPointMapRenderingRun.

5.17 OgreCausticCasterRenderTechnique Class Reference

CausticCasterRenderTechnique used in an OGRE environment.

Inheritance diagram for OgreCausticCasterRenderTechnique::



Public Member Functions

- OgreCausticCasterRenderTechnique (unsigned long startFrame, unsigned long photonMapUpdateInterval, unsigned int photonMapResolution, unsigned int causticCubeMapResolution, String photonMapMaterialName, String causticMapMaterialName, unsigned char photonMapTexID, bool updateAllFace, bool useDistance, float attenuation, bool useTriangles, bool blurCauCubeMap, Pass *pass, OgreRenderable *parentRenderable, OgreTechniqueGroup *parentTechniqueGroup)

  Constructor.

- ~OgreCausticCasterRenderTechnique ()

  Destructor.

- const String & getCausticCubeMapName ()

  Returns the name of the created caustic cubemap.

- float getAttenuation ()
- virtual OgreRenderTechnique * asOgreRenderTechnique ()

  Conversion to OgreRenderTechnique.

- virtual void update (unsigned long frameNum)

  Updates the resources in the given frame.

- virtual void runChanged (RenderingRunType runType, RenderingRun *run)

  Called after one of he shared runs changes.

- virtual void runUpdated (RenderingRunType runType, RenderingRun *run)

  Called after one of he shared runs updates.

- ElementaryRenderable * getParentRenderable ()

  Retrieves the renderable this technique operates on.

- void runChanged (RenderingRunType runType, RenderingRun *run)

Called after one of he shared runs changes.

- virtual void runUpdated (RenderingRunType runType, RenderingRun *run)

  Called after one of he shared runs updates.

Protected Member Functions

- void photonMapRunChanged (RenderingRun *run)

  Called if the changed run is a PhotonMapRenderingRun.

- void causticCubeMapRunChanged (RenderingRun *run)

  Called if the changed run is a CausticCubeMapRenderingRun.

- void distanceCubeMapRunChanged (RenderingRun *run)

  Called if the changed run is a DistanceCubeMapRenderingRun.

- void distanceCubeMapRunUpdated (RenderingRun *run)

  Called if the updated run is a DistanceCubeMapRenderingRun.

- RenderingRun * createPhotonMapRun ()

  Creates a PhotonMapRenderingRun.

- RenderingRun * createCausticCubeMapRun ()

  Creates a CausticCubeMapRenderingRun.

- RenderingRun * createDistanceCubeMapRun ()

  Creates a DistanceCubeMapRenderingRun.

Protected Attributes

- String photonMapMaterialName

  name of the created photon hit map texture

- String causticMapMaterialName

  name of the created caustic cubemap texture

- unsigned char photonMapTexID

  the texture unit state id of the caustic map generation material where the photonhit map should be bound to.

- float attenuation

  attenuation distance of the caustic

- bool useTriangles

  sets if triangles should be rendered into the caustic cubemap instead of sprites

- bool blurCauCubeMap

  sets if the caustic cubemap should be blurred (recommended if rendering caustic triangles)

- OgreRenderable * parentOgreRenderable

  a OgreRenderable pointer to the renderable this technique operates on.

- OgreTechniqueGroup * parentOgreTechniqueGroup

  a OgreTechniqueGroup pointer to the TechniqueGroup this technique is attached to.

- Pass * pass

  a pointer to the pass this technique operates on.

- ElementaryRenderable * parentRenderable

  The renderable this technique operates on.

- TechniqueGroup * parentTechniqueGroup

  The TechniqueGroup this RenderedTechnique is attached to.

- SharedRuns * sharedRuns

  The SharedRuns this RenderedTechnique is attached to.

- bool updateAllFace

  defines if all cubemap faces should be updated in a frame or only one face per frame

- bool useDistance

  tells if a distance cubemap impostor should be used in photon hit calculation (recommended)

- unsigned long photonMapUpdateInterval

  photonmap update frequency

- unsigned int photonMapResolution

  photonmap resolution

- unsigned int causticCubeMapResolution

  caustic cubemap resolution

- unsigned long startFrame

  offset in frame number used during update

5.17.1 Detailed Description

CausticCasterRenderTechnique used in an OGRE environment.

5.17.2 Constructor & Destructor Documentation

5.17.2.1 OgreCausticCasterRenderTechnique::OgreCausticCasterRenderTechnique (unsigned long *startFrame*, unsigned long *photonMapUpdateInterval*, unsigned int *photonMapResolution*, unsigned int *causticCubeMapResolution*, String *photonMapMaterialName*, String *causticMapMaterialName*, unsigned char *photonMapTexID*, bool *updateAllFace*, bool *useDistance*, float *attenuation*, bool *useTriangles*, bool *blurCauCubeMap*, Pass * *pass*, OgreRenderable * *parentRenderable*, OgreTechniqueGroup * *parentTechniqueGroup*)

Constructor.

Parameters:

startFrame adds an offset to the current frame number to help evenly distribute updates between frames

photonMapUpdateInterval photon map and caustic cubemap update frequency

photonMapResolution photon map resolution

causticCubeMapResolution caustic cubemap resolution

photonMapMaterialName the name of the material should be used when rendering the photon hit map

causticMapMaterialName the name of the material that should be used when rendering the caustic cubemap

photonMapTexID the texture unit state id of the caustic map generation material where the photonhit map should be bound to

updateAllFace defines if all cubemap faces should be updated in a frame or only one face per frame

useDistance tells if a distance cubemap impostor should be used in photon hit calculation (recommended)

attenuation attenuation distance of the caustic

useTriangles sets if triangles should be rendered into the caustic cubemap instead of sprites

blurCauCubeMap sets if the caustic cubemap should be blurred (recommended if rendering caustic triangles)

pass the pass to operate on

parentRenderable the object to operate on

parentTechniqueGroup the TechniqueGroup this RenderedTechnique is attached to

5.17.3 Member Function Documentation

5.17.3.1 const String & OgreCausticCasterRenderTechnique::getCausticCubeMapName ()

Returns the name of the created caustic cubemap.

Returns:

name of the caustic cubemap texture

5.17.3.2 float OgreCausticCasterRenderTechnique::getAttenuation () [inline]

See also:

attenuation

5.17.3.3 void OgreCausticCasterRenderTechnique::photonMapRunChanged (RenderingRun * *run*) [protected, virtual]

Called if the changed run is a PhotonMapRenderingRun.

Parameters:

run pointer to the changed PhotonMapRenderingRun

Implements CausticCasterRenderTechnique.

5.17.3.4 void OgreCausticCasterRenderTechnique::causticCubeMapRunChanged (RenderingRun * *run*) [protected, virtual]

Called if the changed run is a CausticCubeMapRenderingRun.

Parameters:

run pointer to the changed CausticCubeMapRenderingRun

Implements CausticCasterRenderTechnique.

5.17.3.5 void OgreCausticCasterRenderTechnique::distanceCubeMapRunChanged (RenderingRun * *run*) [protected, virtual]

Called if the changed run is a DistanceCubeMapRenderingRun.

Parameters:

run pointer to the changed DistanceCubeMapRenderingRun

Implements CausticCasterRenderTechnique.

5.17.3.6 void OgreCausticCasterRenderTechnique::distanceCubeMapRunUpdated (RenderingRun * *run*) [protected, virtual]

Called if the updated run is a DistanceCubeMapRenderingRun.

Parameters:

run pointer to the updated DistanceCubeMapRenderingRun

Implements CausticCasterRenderTechnique.

5.17.3.7 RenderingRun * OgreCausticCasterRenderTechnique::createPhotonMapRun () [protected, virtual]

Creates a PhotonMapRenderingRun.

Returns:

the new PhotonMapRenderingRun instance.

Implements CausticCasterRenderTechnique.

5.17.3.8 RenderingRun * OgreCausticCasterRenderTechnique::createCausticCubeMapRun () [protected, virtual]

Creates a CausticCubeMapRenderingRun.

Returns:

the new CausticCubeMapRenderingRun instance.

Implements CausticCasterRenderTechnique.

5.17.3.9 RenderingRun * OgreCausticCasterRenderTechnique::createDistanceCubeMapRun () [protected, virtual]

Creates a DistanceCubeMapRenderingRun.

Returns:

the new DistanceCubeMapRenderingRun instance.

Implements CausticCasterRenderTechnique.

5.17.3.10 virtual OgreRenderTechnique* OgreRenderTechnique::asOgreRenderTechnique () [inline, virtual, inherited]

Conversion to OgreRenderTechnique.

This function is needed because of virtual inheritance.

Reimplemented from RenderTechnique.

5.17.3.11 virtual void RenderTechnique::update (unsigned long *frameNum*) [inline, virtual, inherited]

Updates the resources in the given frame.

A RenderTechnique is usually need some resources from several runs, so these runs will be updated.

Parameters:

frameNum the actual framenumber

Reimplemented in ColorCubeMapRenderTechnique, ConvolvedCubeMapRenderTechnique, DistanceCubeMapRenderTechnique, HierarchicalParticleSystemTechnique, IllumVolumeRenderTechnique, OgreCausticReceiverRenderTechnique, OgreColorCubeMapRenderTechnique, OgreConvolvedCubeMapRenderTechnique, OgreDepthShadowReceiverRenderTechnique, OgreDistanceCubeMapRenderTechnique, OgreFireRenderTechnique, OgrePathMapRenderTechnique, and OgreSBBRenderTechnique.

5.17.3.12 virtual void RenderTechnique::runChanged (RenderingRunType *runType*, RenderingRun * *run*) [inline, virtual, inherited]

Called after one of he shared runs changes.

Parameters:

runType enum describing the type of the changed run

run pointer to the changed RenderingRun

Reimplemented in CausticCasterRenderTechnique, ColorCubeMapRenderTechnique, ConvolvedCubeMapRenderTechnique, CubeMapRenderTechnique, DistanceCubeMapRenderTechnique, HierarchicalParticleSystemTechnique, and IllumVolumeRenderTechnique.

5.17.3.13 virtual void RenderTechnique::runUpdated (RenderingRunType *runType*, RenderingRun * *run*) [inline, virtual, inherited]

Called after one of he shared runs updates.

Parameters:

runType enum describing the type of the updated run

run pointer to the updated RenderingRun

Reimplemented in CausticCasterRenderTechnique, ColorCubeMapRenderTechnique, CubeMapRenderTechnique, DistanceCubeMapRenderTechnique, HierarchicalParticleSystemTechnique, and IllumVolumeRenderTechnique.

5.17.3.14 void CausticCasterRenderTechnique::runChanged (RenderingRunType *runType*, RenderingRun * *run*) [virtual, inherited]

Called after one of he shared runs changes.

Parameters:

runType enum describing the type of the changed run

run pointer to the changed RenderingRun

Reimplemented from RenderTechnique.

5.17.3.15 void CausticCasterRenderTechnique::runUpdated (RenderingRunType *runType*, RenderingRun * *run*) [virtual, inherited]

Called after one of he shared runs updates.

Parameters:

runType enum describing the type of the updated run

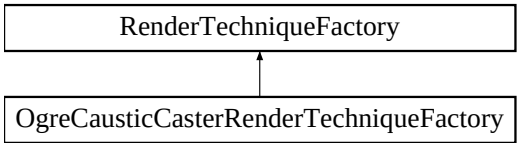
run pointer to the updated RenderingRun

Reimplemented from RenderTechnique.

5.18 OgreCausticCasterRenderTechniqueFactory Class Reference

RenderTechniqueFactory to create OgreCausticCasterRenderTechnique instances.

Inheritance diagram for OgreCausticCasterRenderTechniqueFactory::



Public Member Functions

- OgreCausticCasterRenderTechniqueFactory ()

  Technique factory.

- OgreRenderTechnique * createInstance (IllumTechniqueParams *params, Pass *pass, OgreRenderable *parentRenderable, OgreTechniqueGroup *parentTechniqueGroup)

  Creates a RenderTechnique of the factory type.

- bool isType (String type)

  Returns if this factory can create a RenderTechnique of the given type.

- virtual void parseParams (IllumTechniqueParams *params)

  parses parameters from the material file.

Protected Types

- typedef void(*) ILLUM_ATTRIBUTE_PARSER (String &params, RenderTechniqueFactory *factory)

  function for parsing RenderTechnique attributes

- typedef std::map< String, ILLUM_ATTRIBUTE_PARSER > AttribParserList

  Keyword-mapped attribute parsers.

Protected Attributes

- AttribParserList attributeParsers

  map of parser functions

- String typeName

  factoryname

5.18.1 Detailed Description

RenderTechniqueFactory to create OgreCausticCasterRenderTechnique instances.

5.18.2 Member Typedef Documentation

5.18.2.1 typedef void(∗) RenderTechniqueFactory::ILLUM_ATTRIBUTE_PARSER(String &params, RenderTechniqueFactory ∗factory) [protected, inherited]

function for parsing RenderTechnique attributes

Parameters:

params attribute value stored in a String

5.18.3 Member Function Documentation

5.18.3.1 OgreRenderTechnique ∗ OgreCausticCasterRenderTechniqueFactory::createInstance (IllumTechniqueParams ∗ *params*, Pass ∗ *pass*, OgreRenderable ∗ *parentRenderable*, OgreTechniqueGroup ∗ *parentTechniqueGroup*) [virtual]

Creates a RenderTechnique of the factory type.

Parameters:

params containes constructor parameters as NameValuePairList

pass the Pass to use in RenderTechnique constructor

pass the parentRenderable to pass to RenderTechnique constructor

pass the parentTechniqueGroup to pass to RenderTechnique constructor

Implements RenderTechniqueFactory.

5.18.3.2 bool RenderTechniqueFactory::isType (String *type*) [inline, inherited]

Returns if this factory can create a RenderTechnique of the given type.

Parameters:

type RenderTechnique type

5.18.3.3 void RenderTechniqueFactory::parseParams (IllumTechniqueParams * *params*) [virtual, inherited]

parses parameters from the material file.

The parsed parameters will be passed to the new RenderTechnique's constructor.

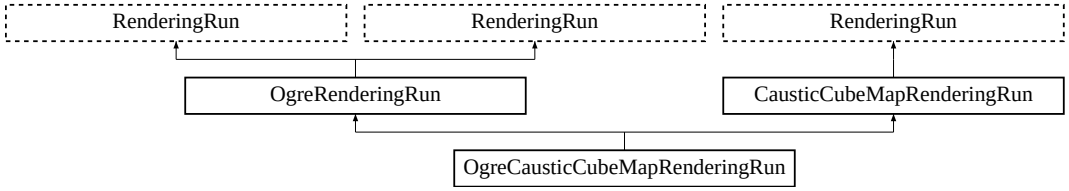
Parameters:

params pointer to the IllumTechniqueParams structure that was read from the material script and containes the parameters to be parsed.

5.19 OgreCausticCubeMapRenderingRun Class Reference

CausticCubeMapRenderingRun used in an OGRE environment.

Inheritance diagram for OgreCausticCubeMapRenderingRun::



Public Member Functions

- OgreCausticCubeMapRenderingRun (OgreSharedRuns *sharedRuns, String name, unsigned long startFrame, unsigned long updateInterval, unsigned int resolution, String materialName, unsigned char photonMapTexId, bool updateAllFace, float attenuation, bool useTriangles, bool blurMap)

  Constructor.

- const String & getCausticCubeMapTextureName ()

  returns the name of the resulting caustic cubemap texture

- void photonMapChanged (RenderingRun *run)

  Called if the changed run is a PhotonMapRenderingRun.

- float getAttenuation ()
- bool canJoin (RenderingRun *run)

  Returns true if two runs can be joined.

- void setBlurMap (bool blur)
- OgreRenderingRun * asOgreRenderingRun ()

  Conversion to OgreRenderRun.

- OgreRenderingRun * asOgreRenderingRun ()

  Conversion to OgreRenderRun.

- bool update (unsigned long frameNum)

  Calls updateFrame() if the run needs update according to its starting frame and update interval and has not been allready updated in this frame.

Protected Member Functions

- void createCausticCubeMap ()

  Creates a cubemap texture used for the caustic-cubemap.

- void updateCubeFace (int facenum)

  Updates one face of the cubemap.

- bool faceNeedsUpdate (int facenum)

  Checks if a cubemap face needs to be updated.

- Vector3 getCubeMapFaceDirection (unsigned char faceId)

  Returns a direction for a cubemap face id.

- Vector3 getCubeMapFaceDirection (unsigned char faceId)

  Returns a direction for a cubemap face id.

- Texture * createCubeRenderTexture (String name, const Vector3 position, unsigned int resolution=512, PixelFormat format=PF_FLOAT16_RGBA, int numMips=0, ColourValue clearColor=ColourValue::Black)

  Creates a cubemap texture.

- Texture * createCubeRenderTexture (String name, const Vector3 position, unsigned int resolution=512, PixelFormat format=PF_FLOAT16_RGBA, int numMips=0, ColourValue clearColor=ColourValue::Black)

  Creates a cubemap texture.

- void setMaterialForRenderables (String &materialName, RenderQueue *rq)

  Sets the given material for each Renderable in a RenderQueue.

- void setMaterialForRenderables (String &materialName, RenderQueue *rq, bool solidsonly=false)

  Sets the given material for each Renderable in a RenderQueue.

- void setMaterialForVisibles (String &materialName, Camera *cam, bool shadowcastersonly=false)

  Sets the given material for each Renderable visible from a given camera.

- void setMaterialForVisibles (String &materialName, Camera *cam, bool shadowcastersonly=false, bool solidsonly=false)

  Sets the given material for each Renderable visible from a given camera.

- void restoreMaterials ()

  Restores previously stored materials.

- void restoreMaterials ()

  Restores previously stored materials.

- void renderFullscreenQuad (String materialName, RenderTarget *target)

  Renderes a full screen quad on a given RendderTarget with a given material.

- void renderFullscreenQuad (String materialName, RenderTarget *target)

  Renderes a full screen quad on a given RendderTarget with a given material.

- void renderPixelSprites (String &materialName, RenderTarget *rt, int width, int height)

Renderes sprites to pixels of the screen on a given RendderTarget with a given material.

- void renderPixelSprites (String &materialName, RenderTarget *rt, int width, int height)
  Renderes sprites to pixels of the screen on a given RendderTarget with a given material.
- void renderFullscreenGrid (String &materialName, RenderTarget *rt, int width, int height)
  Renders a grid onto the screen.
- virtual bool needUpdate (unsigned long frameNum)
  Returns if this run needs update.
- virtual void updateFrame (unsigned long frameNum)
  This function does the actual update in a frame.
- virtual void updateFrame (unsigned long frameNum)
  This function does the actual update in a frame.

Protected Attributes

- unsigned char photonMapTexId
  the texture unit state id of the caustic map generation material where the photonhit map should be bound to
- OgreSharedRuns * sharedRuns
  a pointer to the OgreSharedRuns this run belongs to
- String name
  the name of the cubemap texture that was created by this run
- Texture * causticCubemapTexture
  a pointer to the cubemap texture that was created by this run
- Texture * bluredCausticCubemapTexture
  a pointer to the blurred cubemap texture that was created by this run
- String materialName
  the name of the material that should be used when rendering the caustic cubemap
- float attenuation
  attenuation distance of the caustic
- bool useTriangles
  sets if triangles should be rendered into the caustic cubemap instead of sprites
- bool blurMap
  sets if the caustic cubemap should be blurred (recommended if rendering caustic triangles)
- std::map< Renderable *, String > visibleObjects
  map of Renderables which will be rendered with a given material

- std::map< Renderable *, String > visibleObjects

  map of Renderables which will be rendered with a given material

- SpriteSet * pixelSprites

  SpriteSet used in pixel sprite rendering.

- BillboardSet * pixelSprites

  SpriteSet used in pixel sprite rendering.

- String spriteSetName

  unique name of the SpriteSet used in pixel sprite rendering

- Entity * fullscreenGrid

  Entity used in fullscreen grid rendering.

- unsigned long lastupdated

  The number of the last frame this run was updated.

- unsigned long startFrame

  The number of the frame this run should be updated first.

- unsigned long updateInterval

  Refresh frequency in frames.

- bool updateAllFace

  defines if all cubemap faces should be updated in a frame or only one face per frame

- unsigned char currentFace

  the number of the face to be updated

- unsigned int resolution

  the resolution of the cubemap texture that was created by this run

Static Protected Attributes

- static MovablePlane * fullScreenQuad

  fulls screen quad plane used in full screen quad rendering

- static MovablePlane * fullScreenQuad

  fulls screen quad plane used in full screen quad rendering

- static Entity * fullScreenQuadEntity

  fulls screen quad Entity used in full screen quad rendering

- static Entity * fullScreenQuadEntity

  fulls screen quad Entity used in full screen quad rendering

- static SceneNode * fullScreenQuadNode

fulls screen quad SceneNode used in full screen quad rendering

- static SceneNode * fullScreenQuadNode

  fulls screen quad SceneNode used in full screen quad rendering

5.19.1 Detailed Description

CausticCubeMapRenderingRun used in an OGRE environment.

5.19.2 Constructor & Destructor Documentation

5.19.2.1 OgreCausticCubeMapRenderingRun::OgreCausticCubeMapRenderingRun (OgreSharedRuns * *sharedRuns*, String *name*, unsigned long *startFrame*, unsigned long *updateInterval*, unsigned int *resolution*, String *materialName*, unsigned char *photonMapTexId*, bool *updateAllFace*, float *attenuation*, bool *useTriangles*, bool *blurMap*)

Constructor.

Parameters:

sharedRuns a pointer to the OgreSharedRuns this run belongs to

name the name of the cubemap texture to be created

startFrame adds an offset to the current frame number to help evenly distribute updates between frames

updateInterval update frequency

resolution cubemap resolution

materialName the name of the material that should be used when rendering the caustic cubemap

photonMapTexId the texture unit state id of the caustic map generation material where the photonhit map should be bound to

updateAllFace defines if all cubemap faces should be updated in a frame or only one face per frame

attenuation attenuation distance of the caustic

useTriangles sets if triangles should be rendered into the caustic cubemap instead of sprites

blurMap sets if the caustic cubemap should be blurred (recommended if rendering caustic triangles)

5.19.3 Member Function Documentation

5.19.3.1 void OgreCausticCubeMapRenderingRun::photonMapChanged (RenderingRun * *run*) [virtual]

Called if the changed run is a PhotonMapRenderingRun.

Parameters:

run pointer to the changed PhotonMapRenderingRun

Implements CausticCubeMapRenderingRun.

5.19.3.2 float OgreCausticCubeMapRenderingRun::getAttenuation () [inline]

See also:

attenuation

5.19.3.3 bool OgreCausticCubeMapRenderingRun::canJoin (RenderingRun * *run*) [inline, virtual]

Returns true if two runs can be joined.

In some cases special requirements should stand to join two runs (even if they have the same type). Eg.: two caustic cube map generation technique should only be joined if they use the same material when rendering the caustic cubemap.

Reimplemented from RenderingRun.

5.19.3.4 void OgreCausticCubeMapRenderingRun::setBlurMap (bool *blur*) [inline]

See also:

blurMap

5.19.3.5 void OgreCausticCubeMapRenderingRun::updateCubeFace (int *facenum*) [inline, protected, virtual]

Updates one face of the cubemap.

Parameters:

facenum the number of the face to be updated

Implements CausticCubeMapRenderingRun.

5.19.3.6 bool OgreCausticCubeMapRenderingRun::faceNeedsUpdate (int *facenum*) [protected, virtual]

Checks if a cubemap face needs to be updated.

If the object we are updating the cubemap for is far from the camera, or too small, or the given cubemapface does not have significant effect on the rendering the face can be skipped.

Parameters:

facenum the number of the face to be checked

Implements CausticCubeMapRenderingRun.

5.19.3.7 OgreRenderingRun* OgreRenderingRun::asOgreRenderingRun () [inline, virtual, inherited]

Conversion to OgreRenderRun.

This function is needed because of virtual inheritance.

Reimplemented from RenderingRun.

5.19.3.8 OgreRenderingRun* OgreRenderingRun::asOgreRenderingRun () [inline, virtual, inherited]

Conversion to OgreRenderRun.

This function is needed because of virtual inheritance.

Reimplemented from RenderingRun.

5.19.3.9 Vector3 OgreRenderingRun::getCubeMapFaceDirection (unsigned char *faceId*) [protected, inherited]

Returns a direction for a cubemap face id.

This is a helper function to retrieve the normal direction of a given cubemap face.

Parameters:

faceId the number of the face

5.19.3.10 Vector3 OgreRenderingRun::getCubeMapFaceDirection (unsigned char *faceId*) [protected, inherited]

Returns a direction for a cubemap face id.

This is a helper function to retrieve the normal direction of a given cubemap face.

Parameters:

faceId the number of the face

5.19.3.11 Texture * OgreRenderingRun::createCubeRenderTexture (String *name*, const Vector3 *position*, unsigned int *resolution* = 512, PixelFormat *format* = PF_FLOAT16_RGBA, int *numMips* = 0, ColourValue *clearColor* = ColourValue::Black) [protected, inherited]

Creates a cubemap texture.

This is a helper function to easily create a cubemap texture and automaticly attach viewports to each face so it can be used as a rendertarget.

Parameters:

name the name of the texture to be created

position the initial position of the cubemap

resolution the resolution of one cubemapface

format the pixel format of the cubemap

numMips the number of mipmap levels

clearColor initial color

5.19.3.12 Texture* OgreRenderingRun::createCubeRenderTexture (String *name*, const Vector3 *position*, unsigned int *resolution* = 512, PixelFormat *format* = PF_FLOAT16_RGBA, int *numMips* = 0, ColourValue *clearColor* = ColourValue::Black) [protected, inherited]

Creates a cubemap texture.

This is a helper function to easily create a cubemap texture and automaticly attach viewports to each face so it can be used as a rendertarget.

Parameters:

name the name of the texture to be created

position the initial position of the cubemap

resolution the resolution of one cubemapface

format the pixel format of the cubemap

numMips the number of mipmap levels

clearColor initial color

5.19.3.13 void OgreRenderingRun::setMaterialForRenderables (String & *materialName*, RenderQueue * *rq*) [protected, inherited]

Sets the given material for each Renderable in a RenderQueue.

This is a helper function to set a material to each element of a previously filled Renderque. The orginal material of the Renderables are stored so they can be restored later. The function also tells the current SceneManager not to search for visible objects, as we are going to use the given RenderQueue during the next rendering.

Parameters:

materialName the name of the material to set for the Renderables

rq pointer to the filled Renderqueue instance to set material for

5.19.3.14 void OgreRenderingRun::setMaterialForRenderables (String & *materialName*, RenderQueue * *rq*, bool *solidsonly* = false) [protected, inherited]

Sets the given material for each Renderable in a RenderQueue.

This is a helper function to set a material to each element of a previously filled Renderque. The orginal material of the Renderables are stored so they can be restored later. The function also tells the current SceneManager not to search for visible objects, as we are going to use the given RenderQueue during the next rendering.

Parameters:

materialName the name of the material to set for the Renderables

rq pointer to the filled Renderqueue instance to set material for

5.19.3.15 void OgreRenderingRun::setMaterialForVisibles (String & *materialName*, Camera * *cam*, bool *shadowcastersonly* = false) [protected, inherited]

Sets the given material for each Renderable visible from a given camera.

This helper function is similar to setMaterialForRenderables but it is also responsible for filling the RenderQueue. First the RenderQueue of the current SceneManager fill be filled with the visible objects seen from the given camera. Then the required material will be set for each element of the RenderQueue. The orginal material of the Renderables are stored so they can be restored later. The function also tells the current SceneManager not to search for visible objects, as we are going to use the filled RenderQueue during the next rendering.

Parameters:

materialName the name of the material to set for the Renderables

cam pointer to the camera from which visible objects should be searched

shadowcastersonly flag to search for only shadow casters

5.19.3.16 void OgreRenderingRun::setMaterialForVisibles (String & *materialName*, Camera * *cam*, bool *shadowcastersonly* = false, bool *solidsonly* = false) [protected, inherited]

Sets the given material for each Renderable visible from a given camera.

This helper function is similar to setMaterialForRenderables but it is also responsible for filling the RenderQueue. First the RenderQueue of the current SceneManager fill be filled with the visible objects seen from the given camera. Then the required material will be set for each element of the RenderQueue. The orginal material of the Renderables are stored so they can be restored later. The function also tells the current SceneManager not to search for visible objects, as we are going to use the filled RenderQueue during the next rendering.

Parameters:

materialName the name of the material to set for the Renderables

cam pointer to the camera from which visible objects should be searched

shadowcastersonly flag to search for only shadow casters

5.19.3.17 void OgreRenderingRun::restoreMaterials () [protected, inherited]

Restores previously stored materials.

This helper function is typically used after a setMaterialForRenderables or setMaterialForVisibles call and a rendering process to restore the original material settings. The function also tells the current SceneManager to search for visible objects, as this is the default behaviour of SceneManager.

5.19.3.18 void OgreRenderingRun::restoreMaterials () [protected, inherited]

Restores previously stored materials.

This helper function is typically used after a setMaterialForRenderables or setMaterialForVisibles call and a rendering process to restore the original material settings. The function also tells the current SceneManager to search for visible objects, as this is the default behaviour of SceneManager.

5.19.3.19 void OgreRenderingRun::renderFullscreenQuad (String *materialName*, RenderTarget * *target*) [protected, inherited]

Renderes a full screen quad on a given RendderTarget with a given material.

Parameters:

materialName the name of the material bind to the quad

target the RenderTarget the quad should be rendered on

5.19.3.20 void OgreRenderingRun::renderFullscreenQuad (String *materialName*, RenderTarget * *target*) [protected, inherited]

Renderes a full screen quad on a given RendderTarget with a given material.

Parameters:

materialName the name of the material bind to the quad

target the RenderTarget the quad should be rendered on

5.19.3.21 void OgreRenderingRun::renderPixelSprites (String & *materialName*, RenderTarget * *rt*, int *width*, int *height*) [protected, inherited]

Renderes sprites to pixels of the screen on a given RendderTarget with a given material.

Pixel sprites are pixel sized quads, placed on each pixel of the RenderTarget. The number of sprites not necessary corresponds to the resolution of the rendertarget. The pixel quads will evenly fill the rendertarget's area with sizes corresponding to the given resolution. We can render fewer or more pixel quads than the number of pixels the rendertarget has (eg.: in case of caustic cubemap generation).

Parameters:

materialName the name of the material bind to the pixel sprites

rt the RenderTarget the quads should be rendered on

width the desired resolution width of the sprites

height the desired resolution height of the sprites

5.19.3.22 void OgreRenderingRun::renderPixelSprites (String & *materialName*, RenderTarget * *rt*, int *width*, int *height*) [protected, inherited]

Renderes sprites to pixels of the screen on a given RendderTarget with a given material.

Pixel sprites are pixel sized quads, placed on each pixel of the RenderTarget. The number of sprites not necessary corresponds to the resolution of the rendertarget. The pixel quads will evenly fill the rendertarget's area with sizes corresponding to the given resolution. We can render fewer or more pixel quads than the number of pixels the rendertarget has (eg.: in case of caustic cubemap generation).

Parameters:

materialName the name of the material bind to the pixel sprites

rt the RenderTarget the quads should be rendered on

width the desired resolution width of the sprites

height the desired resolution height of the sprites

5.19.3.23 void OgreRenderingRun::renderFullscreenGrid (String & *materialName*, RenderTarget * *rt*, int *width*, int *height*) [protected, inherited]

Renders a grid onto the screen.

Parameters:

rt the RenderTarget the grid should be rendered on

width the desired horizontal resolution of the grid

height the desired vertical resolution of the grid

5.19.3.24 virtual bool RenderingRun::needUpdate (unsigned long *frameNum*) [inline, protected, virtual, inherited]

Returns if this run needs update.

This tipically depends on the upate interval and the starting frame number.

Parameters:

frameNum current frame number

Reimplemented in OgreFocusingMapRenderingRun, and OgrePMEntryPointMapRenderingRun.

5.19.3.25 virtual void RenderingRun::updateFrame (unsigned long *frameNum*) [inline, protected, virtual, inherited]

This function does the actual update in a frame.

Parameters:

frameNum current frame number

Reimplemented in CausticCubeMapRenderingRun, ChildPsystemRenderingRun, CubeMapRenderingRun, DepthShadowMapRenderingRun, LightVolumeRenderingRun, PhaseTextureRenderingRun, PhotonMapRenderingRun, ReducedCubeMapRenderingRun, SceneCameraDepthRenderingRun, OgreChildPSystemRenderingRun, OgreDepthShadowMapRenderingRun, OgreFocusingMapRenderingRun, OgreLightVolumeRenderingRun, OgrePhaseTextureRenderingRun, OgrePhotonMapRenderingRun, OgrePMEntryPointMapRenderingRun, OgrePMWeightComputeRenderingRun, and OgreSceneCameraDepthRenderingRun.

5.19.3.26 void CausticCubeMapRenderingRun::updateFrame (unsigned long *frameNum*) [protected, virtual, inherited]

This function does the actual update in a frame.

Parameters:

frameNum current frame number

Reimplemented from RenderingRun.

5.19.4 Member Data Documentation

5.19.4.1 std::map<Renderable*, String> OgreRenderingRun::visibleObjects [protected, inherited]

map of Renderables which will be rendered with a given material

The String stores the original material name that will be restored after rendering.

See also:

setMaterialForVisibles
setMaterialForRenderables
restoreMaterials

5.19.4.2 std::map<Renderable*, String> OgreRenderingRun::visibleObjects [protected, inherited]

map of Renderables which will be rendered with a given material

The String stores the original material name that will be restored after rendering.

See also:

setMaterialForVisibles
setMaterialForRenderables
restoreMaterials

5.19.4.3 MovablePlane * OgreRenderingRun::fullScreenQuad [static, protected, inherited]

fulls screen quad plane used in full screen quad rendering

See also:

renderFullscreenQuad

5.19.4.4 MovablePlane* OgreRenderingRun::fullScreenQuad [static, protected, inherited]

fulls screen quad plane used in full screen quad rendering

See also:

renderFullscreenQuad

5.19.4.5 Entity * OgreRenderingRun::fullScreenQuadEntity [static, protected, inherited]

fulls screen quad Entity used in full screen quad rendering

See also:

renderFullscreenQuad

5.19.4.6 Entity* OgreRenderingRun::fullScreenQuadEntity [static, protected, inherited]

fulls screen quad Entity used in full screen quad rendering

See also:

renderFullscreenQuad

5.19.4.7 SceneNode * OgreRenderingRun::fullScreenQuadNode [static, protected, inherited]

fulls screen quad SceneNode used in full screen quad rendering

See also:

renderFullscreenQuad

5.19.4.8 SceneNode* OgreRenderingRun::fullScreenQuadNode [static, protected, inherited]

fulls screen quad SceneNode used in full screen quad rendering

See also:

renderFullscreenQuad

5.19.4.9 SpriteSet* OgreRenderingRun::pixelSprites [protected, inherited]

SpriteSet used in pixel sprite rendering.

See also:

renderPixelSprites

5.19.4.10 BillboardSet* OgreRenderingRun::pixelSprites [protected, inherited]

SpriteSet used in pixel sprite rendering.

See also:

renderPixelSprites

5.19.4.11 String OgreRenderingRun::spriteSetName [protected, inherited]

unique name of the SpriteSet used in pixel sprite rendering

See also:

renderPixelSprites

5.19.4.12 Entity* OgreRenderingRun::fullscreenGrid [protected, inherited]

Entity used in fullscreen grid rendering.

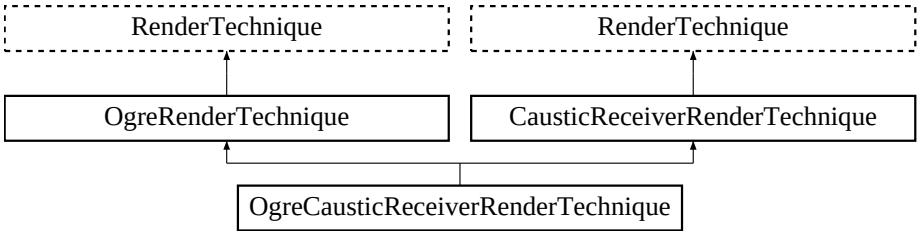
See also:

renderPixelGrid

5.20 OgreCausticReceiverRenderTechnique Class Reference

CausticReceiverRenderTechnique used in an OGRE environment.

Inheritance diagram for OgreCausticReceiverRenderTechnique::



Public Member Functions

- OgreCausticReceiverRenderTechnique (int maxcasters, String causticVertexProgram, String causticFragmentProgram, Pass *pass, OgreRenderable *parentRenderable, OgreTechniqueGroup *parentTechniqueGroup)

  Constructor.

- ~OgreCausticReceiverRenderTechnique ()

  Destructor.

- virtual void update (unsigned long frameNum)

  Updates the resources in the given frame.

- virtual OgreRenderTechnique * asOgreRenderTechnique ()

  Conversion to OgreRenderTechnique.

- virtual void runChanged (RenderingRunType runType, RenderingRun *run)

  Called after one of he shared runs changes.

- virtual void runUpdated (RenderingRunType runType, RenderingRun *run)

  Called after one of he shared runs updates.

- ElementaryRenderable * getParentRenderable ()

  Retrieves the renderable this technique operates on.

Protected Attributes

- int maxcasters

  the maximum number of caustic casters from which this receiver can recieve caustic light

- String causticVertexProgram

the vertex program to be used in the caustic gathering passes

- String causticFragmentProgram

  the fragment program to be used in the caustic gathering passes

- std::vector< Pass * > passes
- std::vector< OgreSharedRuns * > causticCasters

  the nearest caustic casters found during update

- OgreRenderable * parentOgreRenderable

  a OgreRenderable pointer to the renderable this technique operates on.

- OgreTechniqueGroup * parentOgreTechniqueGroup

  a OgreTechniqueGroup pointer to the TechniqueGroup this technique is attached to.

- Pass * pass

  a pointer to the pass this technique operates on.

- ElementaryRenderable * parentRenderable

  The renderable this technique operates on.

- TechniqueGroup * parentTechniqueGroup

  The TechniqueGroup this RenderedTechnique is attached to.

- SharedRuns * sharedRuns

  The SharedRuns this RenderedTechnique is attached to.

5.20.1 Detailed Description

CausticReceiverRenderTechnique used in an OGRE environment.

This technique defines that the object will recieve caustic lighting from caustic caster objects. The caustic light spots will be calculated by the caustic caster's RenderingRuns. These runs will only be updated if caustic redievers are visible, so it is the receiver technique's responsibility to update them.

Each caustic caster's light contribution will be added in separate passes. Each pass will add some light to the shaded image, so these passes should be the last passes. In the constructor the given Pass* parameter will be the pass after which the caustic lighting passes will be added by the technique.

5.20.2 Constructor & Destructor Documentation

5.20.2.1 OgreCausticReceiverRenderTechnique::OgreCausticReceiverRenderTechnique (int *maxcasters*, String *causticVertexProgram*, String *causticFragmentProgram*, Pass * *pass*, OgreRenderable * *parentRenderable*, OgreTechniqueGroup * *parentTechniqueGroup*)

Constructor.

Parameters:

maxcasters the maximum number of caustic casters from which this receiver can recieve caustic light

causticVertexProgram the vertex program to be used in the caustic gathering passes

causticFragmentProgram the fragment program to be used in the caustic gathering passes. It should have one pass and the caustic cubemap of a caster will be bound to the first sampler unit.

pass the pass after which caustic gathering passes should be added

parentRenderable the object to operate on

parentTechniqueGroup the TechniqueGroup this RenderedTechnique is attached to

5.20.3 Member Function Documentation

5.20.3.1 void OgreCausticReceiverRenderTechnique::update (unsigned long *frameNum*) [virtual]

Updates the resources in the given frame.

A RenderTechnique is usually need some resources from several runs, so these runs will be updated.

Parameters:

frameNum the actual framenumber

Reimplemented from RenderTechnique.

5.20.3.2 virtual OgreRenderTechnique* OgreRenderTechnique::asOgreRenderTechnique () [inline, virtual, inherited]

Conversion to OgreRenderTechnique.

This function is needed because of virtual inheritance.

Reimplemented from RenderTechnique.

5.20.3.3 virtual void RenderTechnique::runChanged (RenderingRunType *runType*, RenderingRun * *run*) [inline, virtual, inherited]

Called after one of he shared runs changes.

Parameters:

runType enum describing the type of the changed run

run pointer to the changed RenderingRun

Reimplemented in CausticCasterRenderTechnique, ColorCubeMapRenderTechnique, ConvolvedCubeMapRenderTechnique, CubeMapRenderTechnique, DistanceCubeMapRenderTechnique, HierarchicalParticleSystemTechnique, and IllumVolumeRenderTechnique.

5.20.3.4 virtual void RenderTechnique::runUpdated (RenderingRunType *runType*, RenderingRun * *run*) [inline, virtual, inherited]

Called after one of he shared runs updates.

Parameters:

runType enum describing the type of the updated run

run pointer to the updated RenderingRun

Reimplemented in CausticCasterRenderTechnique, ColorCubeMapRenderTechnique, CubeMapRenderTechnique, DistanceCubeMapRenderTechnique, HierarchicalParticleSystemTechnique, and IllumVolumeRenderTechnique.

5.20.4 Member Data Documentation

5.20.4.1 String OgreCausticReceiverRenderTechnique::causticFragmentProgram [protected]

the fragment program to be used in the caustic gathering passes

It should have one pass and the caustic cubemap of a caster will be bound to the first sampler unit.

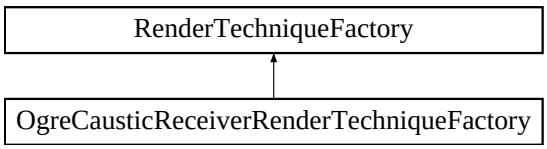
5.20.4.2 std::vector<Pass*> OgreCausticReceiverRenderTechnique::passes [protected]

new passes created by this technique

5.21 OgreCausticReceiverRenderTechniqueFactory Class Reference

RenderTechniqueFactory to create OgreCausticReceiverRenderTechnique instances.

Inheritance diagram for OgreCausticReceiverRenderTechniqueFactory::



Public Member Functions

- OgreRenderTechnique * createInstance (IllumTechniqueParams *params, Pass *pass, OgreRenderable *parentRenderable, OgreTechniqueGroup *parentTechniqueGroup)

  Creates a RenderTechnique of the factory type.

- bool isType (String type)

  Returns if this factory can create a RenderTechnique of the given type.

- virtual void parseParams (IllumTechniqueParams *params)

  parses parameters from the material file.

Protected Types

- typedef void(*) ILLUM_ATTRIBUTE_PARSER (String &params, RenderTechniqueFactory *factory)

  function for parsing RenderTechnique attributes

- typedef std::map< String, ILLUM_ATTRIBUTE_PARSER > AttribParserList

  Keyword-mapped attribute parsers.

Protected Attributes

- AttribParserList attributeParsers

  map of parser functions

- String typeName

  factoryname

5.21.1 Detailed Description

RenderTechniqueFactory to create OgreCausticReceiverRenderTechnique instances.

5.21.2 Member Typedef Documentation

5.21.2.1 typedef void(∗) RenderTechniqueFactory::ILLUM_ATTRIBUTE_PARSER(String &params, RenderTechniqueFactory ∗factory) [protected, inherited]

function for parsing RenderTechnique attributes

Parameters:

params attribute value stored in a String

5.21.3 Member Function Documentation

5.21.3.1 OgreRenderTechnique ∗ OgreCausticReceiverRenderTechniqueFactory::createInstance (IllumTechniqueParams ∗ *params*, Pass ∗ *pass*, OgreRenderable ∗ *parentRenderable*, OgreTechniqueGroup ∗ *parentTechniqueGroup*) [virtual]

Creates a RenderTechnique of the factory type.

Parameters:

params containes constructor parameters as NameValuePairList

pass the Pass to use in RenderTechnique constructor

pass the parentRenderable to pass to RenderTechnique constructor

pass the parentTechniqueGroup to pass to RenderTechnique constructor

Implements RenderTechniqueFactory.

5.21.3.2 bool RenderTechniqueFactory::isType (String *type*) [inline, inherited]

Returns if this factory can create a RenderTechnique of the given type.

Parameters:

type RenderTechnique type

5.21.3.3 void RenderTechniqueFactory::parseParams (IllumTechniqueParams * *params*) [virtual, inherited]

parses parameters from the material file.

The parsed parameters will be passed to the new RenderTechnique's constructor.

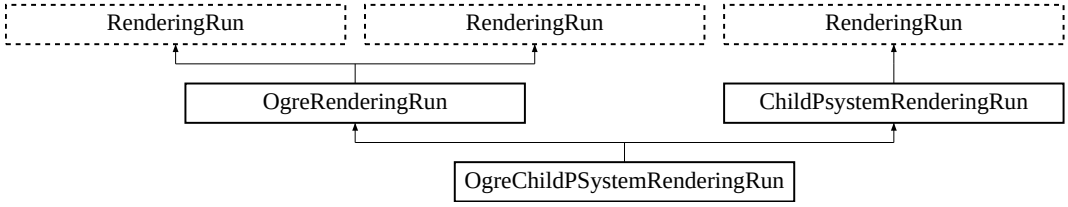
Parameters:

params pointer to the IllumTechniqueParams structure that was read from the material script and containes the parameters to be parsed.

5.22 OgreChildPSystemRenderingRun Class Reference

ChildPsystemRenderingRun used in an OGRE environment.

Inheritance diagram for OgreChildPSystemRenderingRun::



Public Member Functions

- OgreChildPSystemRenderingRun (OgreSharedRuns *sharedRuns, String name, unsigned long startFrame, unsigned long updateInterval, unsigned int resolution, bool perspectiveRendering, String childParticleSystemName, String particleScriptName, bool useOwnMaterial, String materialName)

  Constructor.

- String getImpostorTextureName ()

  returns the name of the resulting photon hit map

- bool canJoin (RenderingRun *run)

  Returns true if two runs can be joined.

- Real getSmallSysRadius ()

  Returns the radius of the small particle system.

- OgreRenderingRun * asOgreRenderingRun ()

  Conversion to OgreRenderRun.

- OgreRenderingRun * asOgreRenderingRun ()

  Conversion to OgreRenderRun.

- bool update (unsigned long frameNum)

  Calls updateFrame() if the run needs update according to its starting frame and update interval and has not been allready updated in this frame.

Protected Member Functions

- void updateFrame (unsigned long frameNum)

  This function does the actual update in a frame.

- void createImpostorTexture ()

  Creates an impostor texture that can be used as a rendertarget during impostor rendering.

- Vector3 getCubeMapFaceDirection (unsigned char faceId)

  Returns a direction for a cubemap face id.

- Vector3 getCubeMapFaceDirection (unsigned char faceId)

  Returns a direction for a cubemap face id.

- Texture * createCubeRenderTexture (String name, const Vector3 position, unsigned int resolution=512, PixelFormat format=PF_FLOAT16_RGBA, int numMips=0, ColourValue clearColor=ColourValue::Black)

  Creates a cubemap texture.

- Texture * createCubeRenderTexture (String name, const Vector3 position, unsigned int resolution=512, PixelFormat format=PF_FLOAT16_RGBA, int numMips=0, ColourValue clearColor=ColourValue::Black)

  Creates a cubemap texture.

- void setMaterialForRenderables (String &materialName, RenderQueue *rq)

  Sets the given material for each Renderable in a RenderQueue.

- void setMaterialForRenderables (String &materialName, RenderQueue *rq, bool solidsonly=false)

  Sets the given material for each Renderable in a RenderQueue.

- void setMaterialForVisibles (String &materialName, Camera *cam, bool shadowcastersonly=false)

  Sets the given material for each Renderable visible from a given camera.

- void setMaterialForVisibles (String &materialName, Camera *cam, bool shadowcastersonly=false, bool solidsonly=false)

  Sets the given material for each Renderable visible from a given camera.

- void restoreMaterials ()

  Restores previously stored materials.

- void restoreMaterials ()

  Restores previously stored materials.

- void renderFullscreenQuad (String materialName, RenderTarget *target)

  Renderes a full screen quad on a given RendderTarget with a given material.

- void renderFullscreenQuad (String materialName, RenderTarget *target)

  Renderes a full screen quad on a given RendderTarget with a given material.

- void renderPixelSprites (String &materialName, RenderTarget *rt, int width, int height)

  Renderes sprites to pixels of the screen on a given RendderTarget with a given material.

- void renderPixelSprites (String &materialName, RenderTarget *rt, int width, int height)

  Renderes sprites to pixels of the screen on a given RendderTarget with a given material.

- void renderFullscreenGrid (String &materialName, RenderTarget *rt, int width, int height)

  Renders a grid onto the screen.

- virtual bool needUpdate (unsigned long frameNum)

  Returns if this run needs update.

Protected Attributes

- bool perspectiveRendering

  Sets if the impostor should be rendered with a perspective projection or orthogonal.

- bool useOwnMaterial

  Use the material that was defined in the particle script.

- String materialName

  Use this specific material while rendering the impostor.

- String childParticleSystemName

  The name of the small particle system.

- String particleScriptName

  The name of the small particle system script.

- Real sysRad

  Radius of the small particle system.

- Camera * impostorCamera

  Camera used while rendering the impostor image.

- OgreSharedRuns * sharedRuns

  A pointer to the OgreSharedRuns this run belongs to.

- String name

  The name of the imostor texture that was created by this run.

- Texture * impostorTexture

  A pointer to the impostor texture that was created by this run.

- std::map< Renderable *, String > visibleObjects

  map of Renderables which will be rendered with a given material

- std::map< Renderable *, String > visibleObjects

  map of Renderables which will be rendered with a given material

- SpriteSet * pixelSprites

  SpriteSet used in pixel sprite rendering.

- BillboardSet * pixelSprites

  SpriteSet used in pixel sprite rendering.

- String spriteSetName

  unique name of the SpriteSet used in pixel sprite rendering

- Entity * fullscreenGrid

  Entity used in fullscreen grid rendering.

- unsigned long lastupdated

  The number of the last frame this run was updated.

- unsigned long startFrame

  The number of the frame this run should be updated first.

- unsigned long updateInterval

  Refresh frequency in frames.

- unsigned int resolution

  the resolution of the impostor image

Static Protected Attributes

- static MovablePlane * fullScreenQuad

  fulls screen quad plane used in full screen quad rendering

- static MovablePlane * fullScreenQuad

  fulls screen quad plane used in full screen quad rendering

- static Entity * fullScreenQuadEntity

  fulls screen quad Entity used in full screen quad rendering

- static Entity * fullScreenQuadEntity

  fulls screen quad Entity used in full screen quad rendering

- static SceneNode * fullScreenQuadNode

  fulls screen quad SceneNode used in full screen quad rendering

- static SceneNode * fullScreenQuadNode

  fulls screen quad SceneNode used in full screen quad rendering

5.22.1 Detailed Description

ChildPsystemRenderingRun used in an OGRE environment.

5.22.2 Constructor & Destructor Documentation

5.22.2.1 OgreChildPSystemRenderingRun::OgreChildPSystemRenderingRun (OgreSharedRuns * *sharedRuns*, String *name*, unsigned long *startFrame*, unsigned long *updateInterval*, unsigned int *resolution*, bool *perspectiveRendering*, String *childParticleSystemName*, String *particleScriptName*, bool *useOwnMaterial*, String *materialName*)

Constructor.

Parameters:

sharedRuns a pointer to the OgreSharedRuns this run belongs to

name the name of the texture to be created

startFrame adds an offset to the current frame number to help evenly distribute updates between frames

updateInterval update frequency

resolution resolution of the impostor texture

perspectiveRendering sets if the impostor should be rendered with a perspective projection or orthogonal

childPSysScriptName the name of the particle system script

useOwnMaterial use the material that was defined in the particle script

materialName use this specific material while rendering the impostor

5.22.3 Member Function Documentation

5.22.3.1 bool OgreChildPSystemRenderingRun::canJoin (RenderingRun * *run*) [inline, virtual]

Returns true if two runs can be joined.

In some cases special requirements should stand to join two runs (even if they have the same type). Eg.: two caustic cube map generation technique should only be joined if they use the same material when rendering the caustic cubemap.

Reimplemented from RenderingRun.

5.22.3.2 void OgreChildPSystemRenderingRun::updateFrame (unsigned long *frameNum*) [protected, virtual]

This function does the actual update in a frame.

Parameters:

frameNum current frame number

Implements ChildPsystemRenderingRun.

5.22.3.3 OgreRenderingRun* OgreRenderingRun::asOgreRenderingRun () [inline, virtual, inherited]

Conversion to OgreRenderRun.

This function is needed because of virtual inheritance.

Reimplemented from RenderingRun.

5.22.3.4 OgreRenderingRun* OgreRenderingRun::asOgreRenderingRun () [inline, virtual, inherited]

Conversion to OgreRenderRun.

This function is needed because of virtual inheritance.

Reimplemented from RenderingRun.

5.22.3.5 Vector3 OgreRenderingRun::getCubeMapFaceDirection (unsigned char *faceId*) [protected, inherited]

Returns a direction for a cubemap face id.

This is a helper function to retrieve the normal direction of a given cubemap face.

Parameters:

faceId the number of the face

5.22.3.6 Vector3 OgreRenderingRun::getCubeMapFaceDirection (unsigned char *faceId*) [protected, inherited]

Returns a direction for a cubemap face id.

This is a helper function to retrieve the normal direction of a given cubemap face.

Parameters:

faceId the number of the face

5.22.3.7 Texture * OgreRenderingRun::createCubeRenderTexture (String *name*, const Vector3 *position*, unsigned int *resolution* = 512, PixelFormat *format* = PF_FLOAT16_RGBA, int *numMips* = 0, ColourValue *clearColor* = ColourValue::Black) [protected, inherited]

Creates a cubemap texture.

This is a helper function to easily create a cubemap texture and automaticly attach viewports to each face so it can be used as a rendertarget.

Parameters:

name the name of the texture to be created

position the initial position of the cubemap

resolution the resolution of one cubemapface

format the pixel format of the cubemap

numMips the number of mipmap levels

clearColor initial color

5.22.3.8 Texture* OgreRenderingRun::createCubeRenderTexture (String *name*, const Vector3 *position*, unsigned int *resolution* = `512`, PixelFormat *format* = `PF_FLOAT16_RGBA`, int *numMips* = `0`, ColourValue *clearColor* = `ColourValue::Black`) `[protected, inherited]`

Creates a cubemap texture.

This is a helper function to easily create a cubemap texture and automaticly attach viewports to each face so it can be used as a rendertarget.

Parameters:

name the name of the texture to be created

position the initial position of the cubemap

resolution the resolution of one cubemapface

format the pixel format of the cubemap

numMips the number of mipmap levels

clearColor initial color

5.22.3.9 void OgreRenderingRun::setMaterialForRenderables (String & *materialName*, RenderQueue * *rq*) `[protected, inherited]`

Sets the given material for each Renderable in a RenderQueue.

This is a helper function to set a material to each element of a previously filled Renderque. The orginal material of the Renderables are stored so they can be restored later. The function also tells the current SceneManager not to search for visible objects, as we are going to use the given RenderQueue during the next rendering.

Parameters:

materialName the name of the material to set for the Renderables

rq pointer to the filled Renderqueue instance to set material for

5.22.3.10 void OgreRenderingRun::setMaterialForRenderables (String & *materialName*, RenderQueue * *rq*, bool *solidsonly* = false) [protected, inherited]

Sets the given material for each Renderable in a RenderQueue.

This is a helper function to set a material to each element of a previously filled Renderque. The orginal material of the Renderables are stored so they can be restored later. The function also tells the current SceneManager not to search for visible objects, as we are going to use the given RenderQueue during the next rendering.

Parameters:

materialName the name of the material to set for the Renderables

rq pointer to the filled Renderqueue instance to set material for

5.22.3.11 void OgreRenderingRun::setMaterialForVisibles (String & *materialName*, Camera * *cam*, bool *shadowcastersonly* = false) [protected, inherited]

Sets the given material for each Renderable visible from a given camera.

This helper function is similar to setMaterialForRenderables but it is also responsible for filling the RenderQueue. First the RenderQueue of the current SceneManager fill be filled with the visible objects seen from the given camera. Then the required material will be set for each element of the RenderQueue. The orginal material of the Renderables are stored so they can be restored later. The function also tells the current SceneManager not to search for visible objects, as we are going to use the filled RenderQueue during the next rendering.

Parameters:

materialName the name of the material to set for the Renderables

cam pointer to the camera from which visible objects should be searched

shadowcastersonly flag to search for only shadow casters

5.22.3.12 void OgreRenderingRun::setMaterialForVisibles (String & *materialName*, Camera * *cam*, bool *shadowcastersonly* = false, bool *solidsonly* = false) [protected, inherited]

Sets the given material for each Renderable visible from a given camera.

This helper function is similar to setMaterialForRenderables but it is also responsible for filling the RenderQueue. First the RenderQueue of the current SceneManager fill be filled with the visible objects seen from the given camera. Then the required material will be set for each element of the RenderQueue. The orginal material of the Renderables are stored so they can be restored later. The function also tells the current SceneManager not to search for visible objects, as we are going to use the filled RenderQueue during the next rendering.

Parameters:

materialName the name of the material to set for the Renderables

cam pointer to the camera from which visible objects should be searched

shadowcastersonly flag to search for only shadow casters

5.22.3.13 void OgreRenderingRun::restoreMaterials () [protected, inherited]

Restores previously stored materials.

This helper function is typically used after a setMaterialForRenderables or setMaterialForVisibles call and a rendering process to restore the original material settings. The function also tells the current SceneManager to search for visible objects, as this is the default behaviour of SceneManager.

5.22.3.14 void OgreRenderingRun::restoreMaterials () [protected, inherited]

Restores previously stored materials.

This helper function is typically used after a setMaterialForRenderables or setMaterialForVisibles call and a rendering process to restore the original material settings. The function also tells the current SceneManager to search for visible objects, as this is the default behaviour of SceneManager.

5.22.3.15 void OgreRenderingRun::renderFullscreenQuad (String *materialName*, RenderTarget * *target*) [protected, inherited]

Renderes a full screen quad on a given RendderTarget with a given material.

Parameters:

materialName the name of the material bind to the quad

target the RenderTarget the quad should be rendered on

5.22.3.16 void OgreRenderingRun::renderFullscreenQuad (String *materialName*, RenderTarget * *target*) [protected, inherited]

Renderes a full screen quad on a given RendderTarget with a given material.

Parameters:

materialName the name of the material bind to the quad

target the RenderTarget the quad should be rendered on

5.22.3.17 void OgreRenderingRun::renderPixelSprites (String & *materialName*, RenderTarget * *rt*, int *width*, int *height*) [protected, inherited]

Renderes sprites to pixels of the screen on a given RendderTarget with a given material.

Pixel sprites are pixel sized quads, placed on each pixel of the RenderTarget. The number of sprites not necessary corresponds to the resolution of the rendertarget. The pixel quads will evenly fill the rendertarget's area with sizes corresponding to the given resolution. We can render fewer or more pixel quads than the number of pixels the rendertarget has (eg.: in case of caustic cubemap generation).

Parameters:

materialName the name of the material bind to the pixel sprites

rt the RenderTarget the quads should be rendered on

width the desired resolution width of the sprites

height the desired resolution height of the sprites

5.22.3.18 void OgreRenderingRun::renderPixelSprites (String & *materialName*, RenderTarget * *rt*, int *width*, int *height*) [protected, inherited]

Renderes sprites to pixels of the screen on a given RendderTarget with a given material.

Pixel sprites are pixel sized quads, placed on each pixel of the RenderTarget. The number of sprites not necessary corresponds to the resolution of the rendertarget. The pixel quads will evenly fill the rendertarget's area with sizes corresponding to the given resolution. We can render fewer or more pixel quads than the number of pixels the rendertarget has (eg.: in case of caustic cubemap generation).

Parameters:

materialName the name of the material bind to the pixel sprites

rt the RenderTarget the quads should be rendered on

width the desired resolution width of the sprites

height the desired resolution height of the sprites

5.22.3.19 void OgreRenderingRun::renderFullscreenGrid (String & *materialName*, RenderTarget * *rt*, int *width*, int *height*) [protected, inherited]

Renders a grid onto the screen.

Parameters:

rt the RenderTarget the grid should be rendered on

width the desired horizontal resolution of the grid

height the desired vertical resolution of the grid

5.22.3.20 virtual bool RenderingRun::needUpdate (unsigned long *frameNum*) [inline, protected, virtual, inherited]

Returns if this run needs update.

This tipically depends on the upate interval and the starting frame number.

Parameters:

frameNum current frame number

Reimplemented in OgreFocusingMapRenderingRun, and OgrePMEntryPointMapRenderingRun.

5.22.4 Member Data Documentation

5.22.4.1 std::map<Renderable*, String> OgreRenderingRun::visibleObjects [protected, inherited]

map of Renderables which will be rendered with a given material

The String stores the original material name that will be restored after rendering.

See also:

setMaterialForVisibles
setMaterialForRenderables
restoreMaterials

5.22.4.2 std::map<Renderable*, String> OgreRenderingRun::visibleObjects [protected, inherited]

map of Renderables which will be rendered with a given material

The String stores the original material name that will be restored after rendering.

See also:

setMaterialForVisibles
setMaterialForRenderables
restoreMaterials

5.22.4.3 MovablePlane * OgreRenderingRun::fullScreenQuad [static, protected, inherited]

fulls screen quad plane used in full screen quad rendering

See also:

renderFullscreenQuad

5.22.4.4 MovablePlane* OgreRenderingRun::fullScreenQuad [static, protected, inherited]

fulls screen quad plane used in full screen quad rendering

See also:

renderFullscreenQuad

5.22.4.5 Entity * OgreRenderingRun::fullScreenQuadEntity [static, protected, inherited]

fulls screen quad Entity used in full screen quad rendering

See also:

renderFullscreenQuad

5.22.4.6 Entity* OgreRenderingRun::fullScreenQuadEntity [static, protected, inherited]

fulls screen quad Entity used in full screen quad rendering

See also:

renderFullscreenQuad

5.22.4.7 SceneNode * OgreRenderingRun::fullScreenQuadNode [static, protected, inherited]

fulls screen quad SceneNode used in full screen quad rendering

See also:

renderFullscreenQuad

5.22.4.8 SceneNode* OgreRenderingRun::fullScreenQuadNode [static, protected, inherited]

fulls screen quad SceneNode used in full screen quad rendering

See also:

renderFullscreenQuad

5.22.4.9 SpriteSet* OgreRenderingRun::pixelSprites [protected, inherited]

SpriteSet used in pixel sprite rendering.

See also:

renderPixelSprites

5.22.4.10 BillboardSet* OgreRenderingRun::pixelSprites [protected, inherited]

SpriteSet used in pixel sprite rendering.

See also:

renderPixelSprites

5.22.4.11 String OgreRenderingRun::spriteSetName [protected, inherited]

unique name of the SpriteSet used in pixel sprite rendering

See also:

renderPixelSprites

5.22.4.12 Entity* OgreRenderingRun::fullscreenGrid [protected, inherited]

Entity used in fullscreen grid rendering.

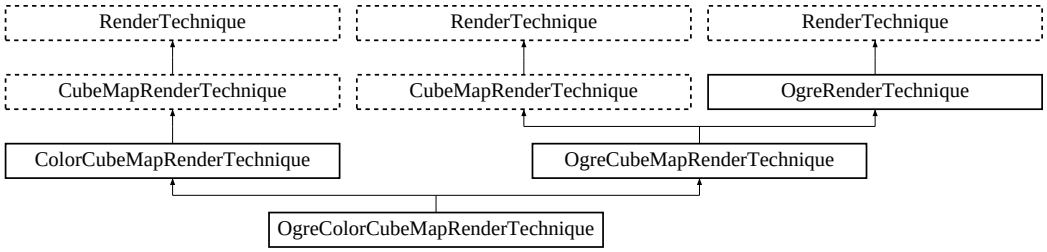
See also:

renderPixelGrid

5.23 OgreColorCubeMapRenderTechnique Class Reference

ColorCubeMapRenderTechnique used in an Ogre environment.

Inheritance diagram for OgreColorCubeMapRenderTechnique::



Public Member Functions

- OgreColorCubeMapRenderTechnique (unsigned long startFrame, unsigned long cubeMapUpdateInterval, unsigned int cubeMapResolution, unsigned char texID, bool useDistCalc, bool useFaceAngleCalc, float distTolerance, float angleTolerance, bool updateAllFace, bool renderSelf, bool renderEnvironment, String selfMaterial, String environmentMaterial, int layer, bool getMinMax, bool attachToTexUnit, String minVariableName, String maxVariableName, Pass *pass, OgreRenderable *parentRenderable, OgreTechniqueGroup *parentTechniqueGroup)

  Constructor.

- ~OgreColorCubeMapRenderTechnique ()

  Destructor.

- void update (unsigned long frameNum)

  Updates the resources in the given frame.

- void runChanged (RenderingRunType runType, RenderingRun *run)

  Called after one of he shared runs changes.

- void runUpdated (RenderingRunType runType, RenderingRun *run)

  Called after one of he shared runs updates.

- virtual class OgreRenderTechnique * asOgreRenderTechnique ()

  Conversion to OgreRenderTechnique.

- ElementaryRenderable * getParentRenderable ()

  Retrieves the renderable this technique operates on.

- virtual OgreRenderTechnique * asOgreRenderTechnique ()

  Conversion to OgreRenderTechnique.

Protected Member Functions

- void colorCubeMapRunChanged (RenderingRun *run)
  Called if the color cubemap rendering run object changes.
- void colorCubeMapRunUpdated (RenderingRun *run)
  Called if the color cubemap rendering run object is updated.
- virtual RenderingRun * createCubeMapRun ()=0
  Creates a CubeMapRenderingRun.
- virtual void cubeMapRunChanged (RenderingRun *run)=0
  Called if the changed run is a CubeMapRenderingRun.
- virtual void cubeMapRunUpdated (RenderingRun *run)=0
  Called if the updated run is a CubeMapRenderingRun.
- RenderingRun * createCubeMapRun ()
  Creates a CubeMapRenderingRun.
- void cubeMapRunChanged (RenderingRun *run)
  Called if the changed run is a CubeMapRenderingRun.
- void cubeMapRunUpdated (RenderingRun *run)
  Called if the updated run is a CubeMapRenderingRun.

Protected Attributes

- bool useDistCalc
  a flag to skip cube face update if object is far away or too small.
- bool useFaceAngleCalc
  a flag to skip cube face update the face is neglible.
- float distTolerance
  A value used in face skip test.
- float angleTolerance
  A value used in face skip test.
- bool updateAllFace
  defines if all cubemap faces should be updated in a frame or only one face per frame
- unsigned long cubeMapUpdateInterval
  color-cubemap update frequency
- unsigned int cubeMapResolution
  color-cubemap resolution

- unsigned long startFrame

  offset in frame number used during update

- bool renderSelf

  Sets if the object should be rendered to the cube map.

- bool renderEnvironment

  Sets if the environment should be rendered to the cube map.

- int layer

  The layer of the cubemap.

- RenderingRunType cubemapLayer

  The exact run type of this run (according to the actual layer).

- ElementaryRenderable * parentRenderable

  The renderable this technique operates on.

- TechniqueGroup * parentTechniqueGroup

  The TechniqueGroup this RenderedTechnique is attached to.

- SharedRuns * sharedRuns

  The SharedRuns this RenderedTechnique is attached to.

- unsigned char texID

  the id of the texture unit state the resulting cubemap should be bound to

- String selfMaterial

  the material that should be set for the object while rendering the cubemap

- String environmentMaterial

  the material that should be set for the environment while rendering the cubemap

- bool getMinMax

  sets if the minimum and maximum values of the cubemap should be computed

- bool attachToTexUnit

  sets if this cubemap should be attach to a texture unit of the pass

- String minVariableName

  sets the name of the gpu shader program parameter to which the minimum value should be bound to

- String maxVariableName

  sets the name of the gpu shader program parameter to which the maximum value should be bound to

- OgreRenderable * parentOgreRenderable

  a OgreRenderable pointer to the renderable this technique operates on.

- OgreTechniqueGroup * parentOgreTechniqueGroup

  a OgreTechniqueGroup pointer to the TechniqueGroup this technique is attached to.

- Pass * pass
  a pointer to the pass this technique operates on.

5.23.1 Detailed Description

ColorCubeMapRenderTechnique used in an Ogre environment.

5.23.2 Constructor & Destructor Documentation

5.23.2.1 OgreColorCubeMapRenderTechnique::OgreColorCubeMapRenderTechnique (unsigned long *startFrame*, unsigned long *cubeMapUpdateInterval*, unsigned int *cubeMapResolution*, unsigned char *texID*, bool *useDistCalc*, bool *useFaceAngleCalc*, float *distTolerance*, float *angleTolerance*, bool *updateAllFace*, bool *renderSelf*, bool *renderEnvironment*, String *selfMaterial*, String *environmentMaterial*, int *layer*, bool *getMinMax*, bool *attachToTexUnit*, String *minVariableName*, String *maxVariableName*, Pass * *pass*, OgreRenderable * *parentRenderable*, OgreTechniqueGroup * *parentTechniqueGroup*)

Constructor.

Parameters:

startFrame adds an offset to the current frame number to help evenly distribute updates between frames

cubeMapUpdateInterval update frequency

cubeMapResolution color cubemap resolution

texID the id of the texture unit state the resulting cubemap should be bound to

useDistCalc flag to skip cube face update if object is far away

useFaceAngleCalc flag to skip cube face update if face is neglible

distTolerance distance tolerance used in face skip

angleTolerance angle tolerance used in face skip

updateAllFace defines if all cubemap faces should be updated in a frame or only one face per frame

renderSelf sets if the object should be rendered to the cube map

renderEnvironment sets if the environment should be rendered to the cube map

selfMaterial the material that should be set for the object while rendering the cubemap

environmentMaterial the material that should be set for the environment while rendering the cubemap

layer the layer of this cubemap

getMinMax sets if the minimum and maximum values of the cubemap should be computed

attachToTexUnit sets if this cubemap should be attach to a texture unit of the pass

minVariableName sets the name of the gpu shader program parameter to which the minimum value should be bound to

maxVariableName sets the name of the gpu shader program parameter to which the maximum value should be bound to

pass the pass to operate on

parentRenderable the object to operate on

parentTechniqueGroup the TechniqueGroup this RenderedTechnique is attached to

5.23.3 Member Function Documentation

5.23.3.1 void OgreColorCubeMapRenderTechnique::update (unsigned long *frameNum*) [virtual]

Updates the resources in the given frame.

Parameters:

frameNum the actual framenumber

Reimplemented from ColorCubeMapRenderTechnique.

5.23.3.2 void OgreColorCubeMapRenderTechnique::colorCubeMapRunChanged (RenderingRun * ***run***) [protected, virtual]

Called if the color cubemap rendering run object changes.

Parameters:

run pointer to the new rendering run object

Implements ColorCubeMapRenderTechnique.

5.23.3.3 void OgreColorCubeMapRenderTechnique::colorCubeMapRunUpdated (RenderingRun * ***run***) [protected, virtual]

Called if the color cubemap rendering run object is updated.

Parameters:

run pointer to the rendering run object

Implements ColorCubeMapRenderTechnique.

5.23.3.4 void ColorCubeMapRenderTechnique::runChanged (RenderingRunType *runType*, RenderingRun * ***run***) [virtual, inherited]

Called after one of he shared runs changes.

Parameters:

runType enum describing the type of the changed run

run pointer to the changed RenderingRun

Reimplemented from CubeMapRenderTechnique.

5.23.3.5 void ColorCubeMapRenderTechnique::runUpdated (RenderingRunType *runType*, RenderingRun * *run*) [virtual, inherited]

Called after one of he shared runs updates.

Parameters:

runType enum describing the type of the updated run

run pointer to the updated RenderingRun

Reimplemented from CubeMapRenderTechnique.

5.23.3.6 virtual RenderingRun* CubeMapRenderTechnique::createCubeMapRun () [protected, pure virtual, inherited]

Creates a CubeMapRenderingRun.

Returns:

the new CubeMapRenderingRun instance.

Implemented in OgreCubeMapRenderTechnique.

5.23.3.7 virtual void CubeMapRenderTechnique::cubeMapRunChanged (RenderingRun * *run*) [protected, pure virtual, inherited]

Called if the changed run is a CubeMapRenderingRun.

Parameters:

run pointer to the new CubeMapRenderingRun

Implemented in OgreCubeMapRenderTechnique.

5.23.3.8 virtual void CubeMapRenderTechnique::cubeMapRunUpdated (RenderingRun * *run*) [protected, pure virtual, inherited]

Called if the updated run is a CubeMapRenderingRun.

Parameters:

run pointer to the updated CubeMapRenderingRun

Implemented in OgreCubeMapRenderTechnique.

5.23.3.9 virtual class OgreRenderTechnique* RenderTechnique::asOgreRenderTechnique () [inline, virtual, inherited]

Conversion to OgreRenderTechnique.

This function is needed because of virtual inheritance.

Reimplemented in OgreRenderTechnique.

5.23.3.10 RenderingRun * OgreCubeMapRenderTechnique::createCubeMapRun ()
[protected, virtual, inherited]

Creates a CubeMapRenderingRun.

Returns:

the new CubeMapRenderingRun instance.

Implements CubeMapRenderTechnique.

5.23.3.11 void OgreCubeMapRenderTechnique::cubeMapRunChanged (RenderingRun * *run*)
[protected, virtual, inherited]

Called if the changed run is a CubeMapRenderingRun.

Parameters:

run pointer to the new CubeMapRenderingRun

Implements CubeMapRenderTechnique.

5.23.3.12 void OgreCubeMapRenderTechnique::cubeMapRunUpdated (RenderingRun * *run*)
[protected, virtual, inherited]

Called if the updated run is a CubeMapRenderingRun.

Parameters:

run pointer to the updated CubeMapRenderingRun

Implements CubeMapRenderTechnique.

5.23.3.13 virtual OgreRenderTechnique* OgreRenderTechnique::asOgreRenderTechnique ()
[inline, virtual, inherited]

Conversion to OgreRenderTechnique.

This function is needed because of virtual inheritance.

Reimplemented from RenderTechnique.

5.23.4 Member Data Documentation

5.23.4.1 bool CubeMapRenderTechnique::useDistCalc [protected, inherited]

a flag to skip cube face update if object is far away or too small.

See also:

distTolerance

5.23.4.2 bool CubeMapRenderTechnique::useFaceAngleCalc [protected, inherited]

a flag to skip cube face update the face is neglible.

See also:

angleTolerance

5.23.4.3 float CubeMapRenderTechnique::distTolerance [protected, inherited]

A value used in face skip test.

The higher this value gets the more precise, but slower the method will be.

5.23.4.4 float CubeMapRenderTechnique::angleTolerance [protected, inherited]

A value used in face skip test.

The higher this value gets the more precise, but slower the method will be.

5.23.4.5 int CubeMapRenderTechnique::layer [protected, inherited]

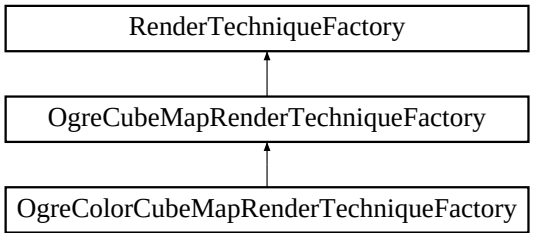
The layer of the cubemap.

This extension was created to achieve effect such as multiple reflections and refractions that require more complete sampling of the scene. With this extension we can render the environment into several cubemap layers eg. with depth peeling.

5.24 OgreColorCubeMapRenderTechniqueFactory Class Reference

RenderTechniqueFactory to create OgreColorCubeMapRenderTechnique instances.

Inheritance diagram for OgreColorCubeMapRenderTechniqueFactory::



Public Member Functions

- OgreColorCubeMapRenderTechniqueFactory ()

  Technique factory.

- OgreRenderTechnique * createInstance (IllumTechniqueParams *params, Pass *pass, OgreRenderable *parentRenderable, OgreTechniqueGroup *parentTechniqueGroup)

  Creates a RenderTechnique of the factory type.

- bool isType (String type)

  Returns if this factory can create a RenderTechnique of the given type.

- virtual void parseParams (IllumTechniqueParams *params)

  parses parameters from the material file.

Protected Types

- typedef void(*) ILLUM_ATTRIBUTE_PARSER (String &params, RenderTechniqueFactory *factory)

  function for parsing RenderTechnique attributes

- typedef std::map< String, ILLUM_ATTRIBUTE_PARSER > AttribParserList

  Keyword-mapped attribute parsers.

Protected Attributes

- AttribParserList attributeParsers

  map of parser functions

- String typeName

factoryname

5.24.1 Detailed Description

RenderTechniqueFactory to create OgreColorCubeMapRenderTechnique instances.

5.24.2 Member Typedef Documentation

5.24.2.1 typedef void(∗) RenderTechniqueFactory::ILLUM_ATTRIBUTE_PARSER(String &params, RenderTechniqueFactory ∗factory) [protected, inherited]

function for parsing RenderTechnique attributes

Parameters:

params attribute value stored in a String

5.24.3 Member Function Documentation

5.24.3.1 OgreRenderTechnique ∗ OgreColorCubeMapRenderTechniqueFactory::createInstance (IllumTechniqueParams ∗ *params*, Pass ∗ *pass*, OgreRenderable ∗ *parentRenderable*, OgreTechniqueGroup ∗ *parentTechniqueGroup*) [virtual]

Creates a RenderTechnique of the factory type.

Parameters:

params containes constructor parameters as NameValuePairList

pass the Pass to use in RenderTechnique constructor

pass the parentRenderable to pass to RenderTechnique constructor

pass the parentTechniqueGroup to pass to RenderTechnique constructor

Reimplemented from OgreCubeMapRenderTechniqueFactory.

5.24.3.2 bool RenderTechniqueFactory::isType (String *type*) [inline, inherited]

Returns if this factory can create a RenderTechnique of the given type.

Parameters:

type RenderTechnique type

5.24.3.3 void RenderTechniqueFactory::parseParams (IllumTechniqueParams * *params*) [virtual, inherited]

parses parameters from the material file.

The parsed parameters will be passed to the new RenderTechnique's constructor.

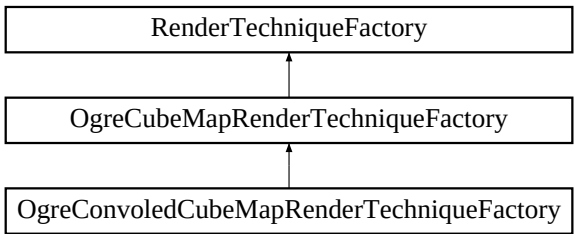
Parameters:

params pointer to the IllumTechniqueParams structure that was read from the material script and containes the parameters to be parsed.

5.25 OgreConvoledCubeMapRenderTechniqueFactory Class Reference

RenderTechniqueFactory to create OgreConvoledCubeMapRenderTechnique instances.

Inheritance diagram for OgreConvoledCubeMapRenderTechniqueFactory::



Public Member Functions

- OgreConvoledCubeMapRenderTechniqueFactory ()

  Technique factory.

- OgreRenderTechnique * createInstance (IllumTechniqueParams *params, Pass *pass, OgreRenderable *parentRenderable, OgreTechniqueGroup *parentTechniqueGroup)

  Creates a RenderTechnique of the factory type.

- bool isType (String type)

  Returns if this factory can create a RenderTechnique of the given type.

- virtual void parseParams (IllumTechniqueParams *params)

  parses parameters from the material file.

Protected Types

- typedef void(*) ILLUM_ATTRIBUTE_PARSER (String &params, RenderTechniqueFactory *factory)

  function for parsing RenderTechnique attributes

- typedef std::map< String, ILLUM_ATTRIBUTE_PARSER > AttribParserList

  Keyword-mapped attribute parsers.

Protected Attributes

- AttribParserList attributeParsers

  map of parser functions

- String typeName
  factoryname

5.25.1 Detailed Description

RenderTechniqueFactory to create OgreConvoledCubeMapRenderTechnique instances.

5.25.2 Member Typedef Documentation

5.25.2.1 typedef void(∗) RenderTechniqueFactory::ILLUM_ATTRIBUTE_PARSER(String &params, RenderTechniqueFactory ∗factory) [protected, inherited]

function for parsing RenderTechnique attributes

Parameters:

params attribute value stored in a String

5.25.3 Member Function Documentation

5.25.3.1 OgreRenderTechnique ∗ OgreConvoledCubeMapRenderTechniqueFactory::createInstance (IllumTechniqueParams ∗ *params*, Pass ∗ *pass*, OgreRenderable ∗ *parentRenderable*, OgreTechniqueGroup ∗ *parentTechniqueGroup*) [virtual]

Creates a RenderTechnique of the factory type.

Parameters:

params containes constructor parameters as NameValuePairList

pass the Pass to use in RenderTechnique constructor

pass the parentRenderable to pass to RenderTechnique constructor

pass the parentTechniqueGroup to pass to RenderTechnique constructor

Reimplemented from OgreCubeMapRenderTechniqueFactory.

5.25.3.2 bool RenderTechniqueFactory::isType (String *type*) [inline, inherited]

Returns if this factory can create a RenderTechnique of the given type.

Parameters:

type RenderTechnique type

5.25.3.3 void RenderTechniqueFactory::parseParams (IllumTechniqueParams * *params*) [virtual, inherited]

parses parameters from the material file.

The parsed parameters will be passed to the new RenderTechnique's constructor.

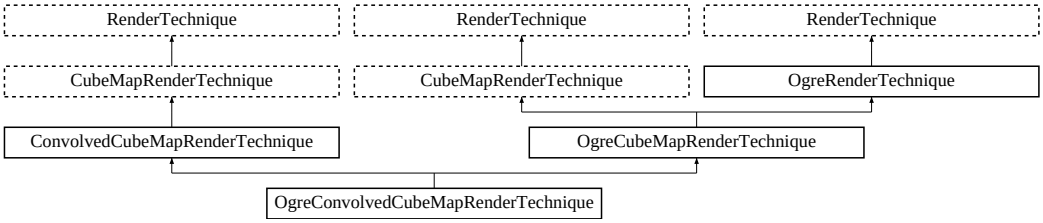
Parameters:

params pointer to the IllumTechniqueParams structure that was read from the material script and containes the parameters to be parsed.

5.26 OgreConvolvedCubeMapRenderTechnique Class Reference

ConvolvedCubeMapRenderTechnique used in an Ogre environment.

Inheritance diagram for OgreConvolvedCubeMapRenderTechnique::



Public Member Functions

- OgreConvolvedCubeMapRenderTechnique (unsigned long startFrame, unsigned long cubeMapUpdateInterval, unsigned int cubeMapResolution, unsigned int reducedCubeMapResolution, unsigned char texID, bool useDistCalc, bool useFaceAngleCalc, float distTolerance, float angleTolerance, bool updateAllFace, bool renderSelf, bool renderEnvironment, String selfMaterial, String environmentMaterial, bool getMinMax, bool attachToTexUnit, String minVariableName, String maxVariableName, Pass *pass, OgreRenderable *parentRenderable, OgreTechniqueGroup *parentTechniqueGroup)

  Constructor.

- ~OgreConvolvedCubeMapRenderTechnique ()

  Destructor.

- void update (unsigned long frameNum)

  Updates the resources in the given frame.

- void runChanged (RenderingRunType runType, RenderingRun *run)

  Called after one of he shared runs changes.

- void runUpdated (RenderingRunType runType, RenderingRun *run)

  Called after one of he shared runs updates.

- virtual class OgreRenderTechnique * asOgreRenderTechnique ()

  Conversion to OgreRenderTechnique.

- ElementaryRenderable * getParentRenderable ()

  Retrieves the renderable this technique operates on.

- virtual OgreRenderTechnique * asOgreRenderTechnique ()

  Conversion to OgreRenderTechnique.

Protected Member Functions

- void reducedCubeMapRunChanged (RenderingRun *run)

  Called if the changed run is a ReducedCubeMapRenderingRun.

- void colorCubeMapRunChanged (RenderingRun *run)

  Called if the changed run is a CubeMapRenderingRun (containing colors of the environment).

- RenderingRun * createReducedCubeMapRun ()

  Creates a ReducedCubeMapRenderingRun.

- virtual RenderingRun * createCubeMapRun ()=0

  Creates a CubeMapRenderingRun.

- virtual void cubeMapRunChanged (RenderingRun *run)=0

  Called if the changed run is a CubeMapRenderingRun.

- virtual void cubeMapRunUpdated (RenderingRun *run)=0

  Called if the updated run is a CubeMapRenderingRun.

- RenderingRun * createCubeMapRun ()

  Creates a CubeMapRenderingRun.

- void cubeMapRunChanged (RenderingRun *run)

  Called if the changed run is a CubeMapRenderingRun.

- void cubeMapRunUpdated (RenderingRun *run)

  Called if the updated run is a CubeMapRenderingRun.

Protected Attributes

- unsigned int reducedCubeMapResolution

  The resolution of the downsampled cubemap created by this run.

- bool useDistCalc

  a flag to skip cube face update if object is far away or too small.

- bool useFaceAngleCalc

  a flag to skip cube face update the face is neglible.

- float distTolerance

  A value used in face skip test.

- float angleTolerance

  A value used in face skip test.

- bool updateAllFace

  defines if all cubemap faces should be updated in a frame or only one face per frame

- unsigned long cubeMapUpdateInterval
  color-cubemap update frequency
- unsigned int cubeMapResolution
  color-cubemap resolution
- unsigned long startFrame
  offset in frame number used during update
- bool renderSelf
  Sets if the object should be rendered to the cube map.
- bool renderEnvironment
  Sets if the environment should be rendered to the cube map.
- int layer
  The layer of the cubemap.
- RenderingRunType cubemapLayer
  The exact run type of this run (according to the actual layer).
- ElementaryRenderable * parentRenderable
  The renderable this technique operates on.
- TechniqueGroup * parentTechniqueGroup
  The TechniqueGroup this RenderedTechnique is attached to.
- SharedRuns * sharedRuns
  The SharedRuns this RenderedTechnique is attached to.
- unsigned char texID
  the id of the texture unit state the resulting cubemap should be bound to
- String selfMaterial
  the material that should be set for the object while rendering the cubemap
- String environmentMaterial
  the material that should be set for the environment while rendering the cubemap
- bool getMinMax
  sets if the minimum and maximum values of the cubemap should be computed
- bool attachToTexUnit
  sets if this cubemap should be attach to a texture unit of the pass
- String minVariableName
  sets the name of the gpu shader program parameter to which the minimum value should be bound to
- String maxVariableName
  sets the name of the gpu shader program parameter to which the maximum value should be bound to

- OgreRenderable * parentOgreRenderable

  a OgreRenderable pointer to the renderable this technique operates on.

- OgreTechniqueGroup * parentOgreTechniqueGroup

  a OgreTechniqueGroup pointer to the TechniqueGroup this technique is attached to.

- Pass * pass

  a pointer to the pass this technique operates on.

5.26.1 Detailed Description

ConvolvedCubeMapRenderTechnique used in an Ogre environment.

5.26.2 Constructor & Destructor Documentation

5.26.2.1 OgreConvolvedCubeMapRenderTechnique::OgreConvolvedCubeMapRenderTechnique (unsigned long *startFrame*, unsigned long *cubeMapUpdateInterval*, unsigned int *cubeMapResolution*, unsigned int *reducedCubeMapResolution*, unsigned char *texID*, bool *useDistCalc*, bool *useFaceAngleCalc*, float *distTolerance*, float *angleTolerance*, bool *updateAllFace*, bool *renderSelf*, bool *renderEnvironment*, String *selfMaterial*, String *environmentMaterial*, bool *getMinMax*, bool *attachToTexUnit*, String *minVariableName*, String *maxVariableName*, Pass * *pass*, OgreRenderable * *parentRenderable*, OgreTechniqueGroup * *parentTechniqueGroup*)

Constructor.

Parameters:

startFrame adds an offset to the current frame number to help evenly distribute updates between frames

cubeMapUpdateInterval update frequency

cubeMapResolution color cubemap resolution

reducedCubeMapResolution reduced sized color cubemap resolution

reducedTexID the id of the texture unit state the resulting cubemap should be bound to

useDistCalc flag to skip cube face update if object is far away

useFaceAngleCalc flag to skip cube face update if face is neglible

distTolerance distance tolerance used in face skip

angleTolerance angle tolerance used in face skip

updateAllFace defines if all cubemap faces should be updated in a frame or only one face per frame

renderSelf sets if the object should be rendered to the cube map

renderEnvironment sets if the environment should be rendered to the cube map

selfMaterial the material that should be set for the object while rendering the cubemap

environmentMaterial the material that should be set for the environment while rendering the cubemap

getMinMax sets if the minimum and maximum values of the cubemap should be computed

attachToTexUnit sets if this cubemap should be attach to a texture unit of the pass

minVariableName sets the name of the gpu shader program parameter to which the minimum value should be bound to

maxVariableName sets the name of the gpu shader program parameter to which the maximum value should be bound to

pass the pass to operate on

parentRenderable the object to operate on

parentTechniqueGroup the TechniqueGroup this RenderedTechnique is attached to

5.26.3 Member Function Documentation

5.26.3.1 void OgreConvolvedCubeMapRenderTechnique::update (unsigned long *frameNum*) [virtual]

Updates the resources in the given frame.

Parameters:

frameNum the actual framenumber

Reimplemented from ConvolvedCubeMapRenderTechnique.

5.26.3.2 void OgreConvolvedCubeMapRenderTechnique::reducedCubeMapRunChanged (**RenderingRun** * ***run***) [protected, virtual]

Called if the changed run is a ReducedCubeMapRenderingRun.

Parameters:

run pointer to the changed ReducedCubeMapRenderingRun

Implements ConvolvedCubeMapRenderTechnique.

5.26.3.3 void OgreConvolvedCubeMapRenderTechnique::colorCubeMapRunChanged (**RenderingRun** * ***run***) [protected, virtual]

Called if the changed run is a CubeMapRenderingRun (containing colors of the environment).

Parameters:

run pointer to the changed CubeMapRenderingRun

Implements ConvolvedCubeMapRenderTechnique.

5.26.3.4 RenderingRun * OgreConvolvedCubeMapRenderTechnique::createReducedCubeMapRun () [protected, virtual]

Creates a ReducedCubeMapRenderingRun.

Returns:

the new ReducedCubeMapRenderingRun instance.

Implements ConvolvedCubeMapRenderTechnique.

5.26.3.5 void ConvolvedCubeMapRenderTechnique::runChanged (RenderingRunType *runType*, RenderingRun * *run*) [virtual, inherited]

Called after one of he shared runs changes.

Parameters:

runType enum describing the type of the changed run

run pointer to the changed RenderingRun

Reimplemented from CubeMapRenderTechnique.

5.26.3.6 void CubeMapRenderTechnique::runUpdated (RenderingRunType *runType*, RenderingRun * *run*) [virtual, inherited]

Called after one of he shared runs updates.

Parameters:

runType enum describing the type of the updated run

run pointer to the updated RenderingRun

Reimplemented from RenderTechnique.

Reimplemented in ColorCubeMapRenderTechnique, and DistanceCubeMapRenderTechnique.

5.26.3.7 virtual RenderingRun* CubeMapRenderTechnique::createCubeMapRun () [protected, pure virtual, inherited]

Creates a CubeMapRenderingRun.

Returns:

the new CubeMapRenderingRun instance.

Implemented in OgreCubeMapRenderTechnique.

5.26.3.8 virtual void CubeMapRenderTechnique::cubeMapRunChanged (RenderingRun * *run*)

`[protected, pure virtual, inherited]`

Called if the changed run is a CubeMapRenderingRun.

Parameters:

run pointer to the new CubeMapRenderingRun

Implemented in OgreCubeMapRenderTechnique.

5.26.3.9 virtual void CubeMapRenderTechnique::cubeMapRunUpdated (RenderingRun * *run*)

`[protected, pure virtual, inherited]`

Called if the updated run is a CubeMapRenderingRun.

Parameters:

run pointer to the updated CubeMapRenderingRun

Implemented in OgreCubeMapRenderTechnique.

5.26.3.10 virtual class OgreRenderTechnique* RenderTechnique::asOgreRenderTechnique ()

`[inline, virtual, inherited]`

Conversion to OgreRenderTechnique.

This function is needed because of virtual inheritance.

Reimplemented in OgreRenderTechnique.

5.26.3.11 RenderingRun * OgreCubeMapRenderTechnique::createCubeMapRun ()

`[protected, virtual, inherited]`

Creates a CubeMapRenderingRun.

Returns:

the new CubeMapRenderingRun instance.

Implements CubeMapRenderTechnique.

5.26.3.12 void OgreCubeMapRenderTechnique::cubeMapRunChanged (RenderingRun * *run*)

`[protected, virtual, inherited]`

Called if the changed run is a CubeMapRenderingRun.

Parameters:

run pointer to the new CubeMapRenderingRun

Implements CubeMapRenderTechnique.

5.26.3.13 void OgreCubeMapRenderTechnique::cubeMapRunUpdated (RenderingRun * *run*) [protected, virtual, inherited]

Called if the updated run is a CubeMapRenderingRun.

Parameters:

run pointer to the updated CubeMapRenderingRun

Implements CubeMapRenderTechnique.

5.26.3.14 virtual OgreRenderTechnique* OgreRenderTechnique::asOgreRenderTechnique () [inline, virtual, inherited]

Conversion to OgreRenderTechnique.

This function is needed because of virtual inheritance.

Reimplemented from RenderTechnique.

5.26.4 Member Data Documentation

5.26.4.1 bool CubeMapRenderTechnique::useDistCalc [protected, inherited]

a flag to skip cube face update if object is far away or too small.

See also:

distTolerance

5.26.4.2 bool CubeMapRenderTechnique::useFaceAngleCalc [protected, inherited]

a flag to skip cube face update the face is neglible.

See also:

angleTolerance

5.26.4.3 float CubeMapRenderTechnique::distTolerance [protected, inherited]

A value used in face skip test.

The higher this value gets the more precise, but slower the method will be.

5.26.4.4 float CubeMapRenderTechnique::angleTolerance [protected, inherited]

A value used in face skip test.

The higher this value gets the more precise, but slower the method will be.

5.26.4.5 int CubeMapRenderTechnique::layer [protected, inherited]

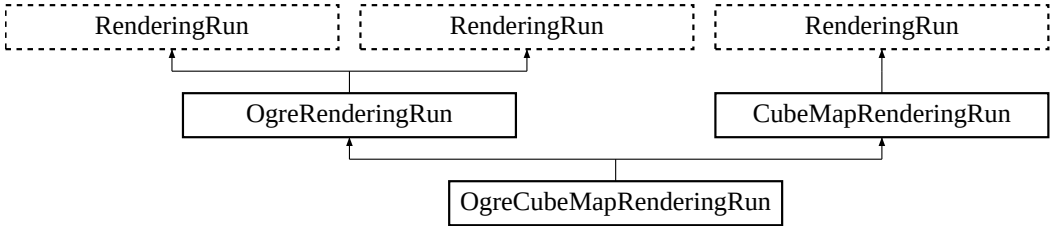
The layer of the cubemap.

This extension was created to achieve effect such as multiple reflections and refractions that require more complete sampling of the scene. With this extension we can render the environment into several cubemap layers eg. with depth peeling.

5.27 OgreCubeMapRenderingRun Class Reference

CubeMapRenderingRun used in an OGRE environment.

Inheritance diagram for OgreCubeMapRenderingRun::



Public Member Functions

- OgreCubeMapRenderingRun (OgreSharedRuns *sharedRuns, String name, unsigned long startFrame, unsigned long updateInterval, unsigned int resolution, bool useDistCalc, bool useFaceAngleCalc, float distTolerance, float angleTolerance, bool updateAllFace, bool renderSelf, bool renderEnvironment, String selfMaterial, String environmentMaterial, bool getMinMax, RenderingRunType cubemapRunType)

  Constructor.

- String getCubeMapTextureName ()

  returns the name of the resulting color cubemap texture

- OgreRenderingRun * asOgreRenderingRun ()

  Conversion to OgreRenderRun.

- OgreRenderingRun * asOgreRenderingRun ()

  Conversion to OgreRenderRun.

- bool update (unsigned long frameNum)

  Calls updateFrame() if the run needs update according to its starting frame and update interval and has not been allready updated in this frame.

- virtual bool canJoin (RenderingRun *run)

  Returns true if two runs can be joined.

Protected Member Functions

- void createCubeMap ()

  Creates a cubemap texture used for the color-cubemap.

- void updateCubeFace (int facenum)

Updates one face of the cubemap.

- bool faceNeedsUpdate (int facenum)

  Checks if a cubemap face needs to be updated.

- Vector3 getCubeMapFaceDirection (unsigned char faceId)

  Returns a direction for a cubemap face id.

- Vector3 getCubeMapFaceDirection (unsigned char faceId)

  Returns a direction for a cubemap face id.

- Texture * createCubeRenderTexture (String name, const Vector3 position, unsigned int resolution=512, PixelFormat format=PF_FLOAT16_RGBA, int numMips=0, ColourValue clearColor=ColourValue::Black)

  Creates a cubemap texture.

- Texture * createCubeRenderTexture (String name, const Vector3 position, unsigned int resolution=512, PixelFormat format=PF_FLOAT16_RGBA, int numMips=0, ColourValue clearColor=ColourValue::Black)

  Creates a cubemap texture.

- void setMaterialForRenderables (String &materialName, RenderQueue *rq)

  Sets the given material for each Renderable in a RenderQueue.

- void setMaterialForRenderables (String &materialName, RenderQueue *rq, bool solidsonly=false)

  Sets the given material for each Renderable in a RenderQueue.

- void setMaterialForVisibles (String &materialName, Camera *cam, bool shadowcastersonly=false)

  Sets the given material for each Renderable visible from a given camera.

- void setMaterialForVisibles (String &materialName, Camera *cam, bool shadowcastersonly=false, bool solidsonly=false)

  Sets the given material for each Renderable visible from a given camera.

- void restoreMaterials ()

  Restores previously stored materials.

- void restoreMaterials ()

  Restores previously stored materials.

- void renderFullscreenQuad (String materialName, RenderTarget *target)

  Renderes a full screen quad on a given RendderTarget with a given material.

- void renderFullscreenQuad (String materialName, RenderTarget *target)

  Renderes a full screen quad on a given RendderTarget with a given material.

- void renderPixelSprites (String &materialName, RenderTarget *rt, int width, int height)

  Renderes sprites to pixels of the screen on a given RendderTarget with a given material.

- void renderPixelSprites (String &materialName, RenderTarget *rt, int width, int height)
  Renderes sprites to pixels of the screen on a given RendderTarget with a given material.
- void renderFullscreenGrid (String &materialName, RenderTarget *rt, int width, int height)
  Renders a grid onto the screen.
- virtual bool needUpdate (unsigned long frameNum)
  Returns if this run needs update.
- virtual void updateFrame (unsigned long frameNum)
  This function does the actual update in a frame.
- virtual void updateFrame (unsigned long frameNum)
  This function does the actual update in a frame.

Protected Attributes

- OgreSharedRuns * sharedRuns
  A pointer to the OgreSharedRuns this run belongs to.
- String name
  The name of the cubemap texture that was created by this run.
- Texture * cubemapTexture
  A pointer to the cubemap texture that was created by this run.
- std::map< Renderable *, String > visibleObjects
  map of Renderables which will be rendered with a given material
- std::map< Renderable *, String > visibleObjects
  map of Renderables which will be rendered with a given material
- SpriteSet * pixelSprites
  SpriteSet used in pixel sprite rendering.
- BillboardSet * pixelSprites
  SpriteSet used in pixel sprite rendering.
- String spriteSetName
  unique name of the SpriteSet used in pixel sprite rendering
- Entity * fullscreenGrid
  Entity used in fullscreen grid rendering.
- unsigned long lastupdated
  The number of the last frame this run was updated.
- unsigned long startFrame

The number of the frame this run should be updated first.

- unsigned long updateInterval

  Refresh frequency in frames.

- bool updateAllFace

  defines if all cubemap faces should be updated in a frame or only one face per frame

- unsigned char currentFace

  the number of the face to be updated

- unsigned int resolution

  the resolution of the cubemap texture that was created by this run

- bool useDistCalc

  a flag to skip cube face update if object is far away or too small.

- bool useFaceAngleCalc

  a flag to skip cube face update the face is neglible.

- float distTolerance

  A value used in face skip test.

- float angleTolerance

  A value used in face skip test.

- bool renderSelf

  sets if the object should be rendered to the cube map

- bool renderEnvironment

  sets if the environment should be rendered to the cube map

Static Protected Attributes

- static MovablePlane * fullScreenQuad

  fulls screen quad plane used in full screen quad rendering

- static MovablePlane * fullScreenQuad

  fulls screen quad plane used in full screen quad rendering

- static Entity * fullScreenQuadEntity

  fulls screen quad Entity used in full screen quad rendering

- static Entity * fullScreenQuadEntity

  fulls screen quad Entity used in full screen quad rendering

- static SceneNode * fullScreenQuadNode

  fulls screen quad SceneNode used in full screen quad rendering

- static SceneNode * fullScreenQuadNode

  fulls screen quad SceneNode used in full screen quad rendering

5.27.1 Detailed Description

CubeMapRenderingRun used in an OGRE environment.

5.27.2 Constructor & Destructor Documentation

5.27.2.1 OgreCubeMapRenderingRun::OgreCubeMapRenderingRun (OgreSharedRuns * *sharedRuns*, String *name*, unsigned long *startFrame*, unsigned long *updateInterval*, unsigned int *resolution*, bool *useDistCalc*, bool *useFaceAngleCalc*, float *distTolerance*, float *angleTolerance*, bool *updateAllFace*, bool *renderSelf*, bool *renderEnvironment*, String *selfMaterial*, String *environmentMaterial*, bool *getMinMax*, RenderingRunType *cubemapRunType*)

Constructor.

Parameters:

sharedRuns a pointer to the OgreSharedRuns this run belongs to

name the name of the cubemap texture to be created

startFrame adds an offset to the current frame number to help evenly distribute updates between frames

updateInterval update frequency

resolution cubemap resolution

useDistCalc flag to skip cube face update if object is far away

useFaceAngleCalc flag to skip cube face update if face is neglible

distTolerance distance tolerance used in face skip

angleTolerance angle tolerance used in face skip

updateAllFace defines if all cubemap faces should be updated in a frame or only one face per frame

renderSelf sets if the object should be rendered to the cube map

renderEnvironment sets if the environment should be rendered to the cube map

selfMaterial the material that should be set for the object while rendering the cubemap

environmentMaterial the material that should be set for the environment while rendering the cubemap

getMinMax sets if the minimum and maximum values of the cubemap should be computed

5.27.3 Member Function Documentation

5.27.3.1 void OgreCubeMapRenderingRun::updateCubeFace (int *facenum*) [inline, protected, virtual]

Updates one face of the cubemap.

Parameters:

facenum the number of the face to be updated

Implements CubeMapRenderingRun.

5.27.3.2 bool OgreCubeMapRenderingRun::faceNeedsUpdate (int *facenum*) [protected, virtual]

Checks if a cubemap face needs to be updated.

If the object we are updating the cubemap for is far from the camera, or too small, or the given cubemapface does not have significant effect on the rendering the face can be skipped.

Parameters:

facenum the number of the face to be checked

Implements CubeMapRenderingRun.

5.27.3.3 OgreRenderingRun* OgreRenderingRun::asOgreRenderingRun () [inline, virtual, inherited]

Conversion to OgreRenderRun.

This function is needed because of virtual inheritance.

Reimplemented from RenderingRun.

5.27.3.4 OgreRenderingRun* OgreRenderingRun::asOgreRenderingRun () [inline, virtual, inherited]

Conversion to OgreRenderRun.

This function is needed because of virtual inheritance.

Reimplemented from RenderingRun.

5.27.3.5 Vector3 OgreRenderingRun::getCubeMapFaceDirection (unsigned char *faceId*) [protected, inherited]

Returns a direction for a cubemap face id.

This is a helper function to retrieve the normal direction of a given cubemap face.

Parameters:

faceId the number of the face

5.27.3.6 Vector3 OgreRenderingRun::getCubeMapFaceDirection (unsigned char *faceId*) [protected, inherited]

Returns a direction for a cubemap face id.

This is a helper function to retrieve the normal direction of a given cubemap face.

Parameters:

faceId the number of the face

5.27.3.7 Texture * OgreRenderingRun::createCubeRenderTexture (String *name*, const Vector3 *position*, unsigned int *resolution* = 512, PixelFormat *format* = PF_FLOAT16_RGBA, int *numMips* = 0, ColourValue *clearColor* = ColourValue::Black) [protected, inherited]

Creates a cubemap texture.

This is a helper function to easily create a cubemap texture and automaticly attach viewports to each face so it can be used as a rendertarget.

Parameters:

name the name of the texture to be created
position the initial position of the cubemap
resolution the resolution of one cubemapface
format the pixel format of the cubemap
numMips the number of mipmap levels
clearColor initial color

5.27.3.8 Texture* OgreRenderingRun::createCubeRenderTexture (String *name*, const Vector3 *position*, unsigned int *resolution* = 512, PixelFormat *format* = PF_FLOAT16_RGBA, int *numMips* = 0, ColourValue *clearColor* = ColourValue::Black) [protected, inherited]

Creates a cubemap texture.

This is a helper function to easily create a cubemap texture and automaticly attach viewports to each face so it can be used as a rendertarget.

Parameters:

name the name of the texture to be created
position the initial position of the cubemap
resolution the resolution of one cubemapface
format the pixel format of the cubemap
numMips the number of mipmap levels
clearColor initial color

5.27.3.9 void OgreRenderingRun::setMaterialForRenderables (String & *materialName*, RenderQueue * *rq*) [protected, inherited]

Sets the given material for each Renderable in a RenderQueue.

This is a helper function to set a material to each element of a previously filled Renderque. The orginal material of the Renderables are stored so they can be restored later. The function also tells the current SceneManager not to search for visible objects, as we are going to use the given RenderQueue during the next rendering.

Parameters:

materialName the name of the material to set for the Renderables

rq pointer to the filled Renderqueue instance to set material for

5.27.3.10 void OgreRenderingRun::setMaterialForRenderables (String & *materialName*, RenderQueue * *rq*, bool *solidsonly* = false) [protected, inherited]

Sets the given material for each Renderable in a RenderQueue.

This is a helper function to set a material to each element of a previously filled Renderque. The orginal material of the Renderables are stored so they can be restored later. The function also tells the current SceneManager not to search for visible objects, as we are going to use the given RenderQueue during the next rendering.

Parameters:

materialName the name of the material to set for the Renderables

rq pointer to the filled Renderqueue instance to set material for

5.27.3.11 void OgreRenderingRun::setMaterialForVisibles (String & *materialName*, Camera * *cam*, bool *shadowcastersonly* = false) [protected, inherited]

Sets the given material for each Renderable visible from a given camera.

This helper function is similar to setMaterialForRenderables but it is also responsible for filling the RenderQueue. First the RenderQueue of the current SceneManager fill be filled with the visible objects seen from the given camera. Then the required material will be set for each element of the RenderQueue. The orginal material of the Renderables are stored so they can be restored later. The function also tells the current SceneManager not to search for visible objects, as we are going to use the filled RenderQueue during the next rendering.

Parameters:

materialName the name of the material to set for the Renderables

cam pointer to the camera from which visible objects should be searched

shadowcastersonly flag to search for only shadow casters

5.27.3.12 void OgreRenderingRun::setMaterialForVisibles (String & *materialName*, Camera * *cam*, bool *shadowcastersonly* = false, bool *solidsonly* = false) [protected, inherited]

Sets the given material for each Renderable visible from a given camera.

This helper function is similar to setMaterialForRenderables but it is also responsible for filling the RenderQueue. First the RenderQueue of the current SceneManager fill be filled with the visible objects seen from the given camera. Then the required material will be set for each element of the RenderQueue. The orginal material of the Renderables are stored so they can be restored later. The function also tells the current SceneManager not to search for visible objects, as we are going to use the filled RenderQueue during the next rendering.

Parameters:

materialName the name of the material to set for the Renderables

cam pointer to the camera from which visible objects should be searched

shadowcastersonly flag to search for only shadow casters

5.27.3.13 void OgreRenderingRun::restoreMaterials () [protected, inherited]

Restores previously stored materials.

This helper function is typically used after a setMaterialForRenderables or setMaterialForVisibles call and a rendering process to restore the original material settings. The function also tells the current SceneManager to search for visible objects, as this is the default behaviour of SceneManager.

5.27.3.14 void OgreRenderingRun::restoreMaterials () [protected, inherited]

Restores previously stored materials.

This helper function is typically used after a setMaterialForRenderables or setMaterialForVisibles call and a rendering process to restore the original material settings. The function also tells the current SceneManager to search for visible objects, as this is the default behaviour of SceneManager.

5.27.3.15 void OgreRenderingRun::renderFullscreenQuad (String *materialName*, RenderTarget * *target*) [protected, inherited]

Renderes a full screen quad on a given RendderTarget with a given material.

Parameters:

materialName the name of the material bind to the quad

target the RenderTarget the quad should be rendered on

5.27.3.16 void OgreRenderingRun::renderFullscreenQuad (String *materialName*, RenderTarget * *target*) [protected, inherited]

Renderes a full screen quad on a given RendderTarget with a given material.

Parameters:

materialName the name of the material bind to the quad

target the RenderTarget the quad should be rendered on

5.27.3.17 void OgreRenderingRun::renderPixelSprites (String & *materialName*, RenderTarget * *rt*, int *width*, int *height*) [protected, inherited]

Renderes sprites to pixels of the screen on a given RendderTarget with a given material.

Pixel sprites are pixel sized quads, placed on each pixel of the RenderTarget. The number of sprites not necessary corresponds to the resolution of the rendertarget. The pixel quads will evenly fill the rendertarget's area with sizes corresponding to the given resolution. We can render fewer or more pixel quads than the number of pixels the rendertarget has (eg.: in case of caustic cubemap generation).

Parameters:

materialName the name of the material bind to the pixel sprites

rt the RenderTarget the quads should be rendered on

width the desired resolution width of the sprites

height the desired resolution height of the sprites

5.27.3.18 void OgreRenderingRun::renderPixelSprites (String & *materialName*, RenderTarget * *rt*, int *width*, int *height*) [protected, inherited]

Renderes sprites to pixels of the screen on a given RendderTarget with a given material.

Pixel sprites are pixel sized quads, placed on each pixel of the RenderTarget. The number of sprites not necessary corresponds to the resolution of the rendertarget. The pixel quads will evenly fill the rendertarget's area with sizes corresponding to the given resolution. We can render fewer or more pixel quads than the number of pixels the rendertarget has (eg.: in case of caustic cubemap generation).

Parameters:

materialName the name of the material bind to the pixel sprites

rt the RenderTarget the quads should be rendered on

width the desired resolution width of the sprites

height the desired resolution height of the sprites

5.27.3.19 void OgreRenderingRun::renderFullscreenGrid (String & *materialName*, RenderTarget * *rt*, int *width*, int *height*) [protected, inherited]

Renders a grid onto the screen.

Parameters:

rt the RenderTarget the grid should be rendered on

width the desired horizontal resolution of the grid

height the desired vertical resolution of the grid

5.27.3.20 virtual bool RenderingRun::canJoin (RenderingRun * *run*) [inline, virtual, inherited]

Returns true if two runs can be joined.

In some cases special requirements should stand to join two runs (even if they have the same type). Eg.: two caustic cube map generation technique should only be joined if they use the same material when rendering the caustic cubemap.

Reimplemented in OgreCausticCubeMapRenderingRun, OgreChildPSystemRenderingRun, and OgrePhotonMapRenderingRun.

5.27.3.21 virtual bool RenderingRun::needUpdate (unsigned long *frameNum*) [inline, protected, virtual, inherited]

Returns if this run needs update.

This tipically depends on the upate interval and the starting frame number.

Parameters:

frameNum current frame number

Reimplemented in OgreFocusingMapRenderingRun, and OgrePMEntryPointMapRenderingRun.

5.27.3.22 virtual void RenderingRun::updateFrame (unsigned long *frameNum*) [inline, protected, virtual, inherited]

This function does the actual update in a frame.

Parameters:

frameNum current frame number

Reimplemented in CausticCubeMapRenderingRun, ChildPsystemRenderingRun, CubeMapRenderingRun, DepthShadowMapRenderingRun, LightVolumeRenderingRun, PhaseTextureRenderingRun, PhotonMapRenderingRun, ReducedCubeMapRenderingRun, SceneCameraDepthRenderingRun, OgreChildPSystemRenderingRun, OgreDepthShadowMapRenderingRun, OgreFocusingMapRenderingRun, OgreLightVolumeRenderingRun, OgrePhaseTextureRenderingRun, OgrePhotonMapRenderingRun, OgrePMEntryPointMapRenderingRun, OgrePMWeightComputeRenderingRun, and OgreSceneCameraDepthRenderingRun.

5.27.3.23 void CubeMapRenderingRun::updateFrame (unsigned long *frameNum*) [protected, virtual, inherited]

This function does the actual update in a frame.

Parameters:

frameNum current frame number

Reimplemented from RenderingRun.

5.27.4 Member Data Documentation

5.27.4.1 std::map<Renderable*, String> OgreRenderingRun::visibleObjects [protected, inherited]

map of Renderables which will be rendered with a given material

The String stores the original material name that will be restored after rendering.

See also:

setMaterialForVisibles
setMaterialForRenderables
restoreMaterials

5.27.4.2 std::map<Renderable*, String> OgreRenderingRun::visibleObjects [protected, inherited]

map of Renderables which will be rendered with a given material

The String stores the original material name that will be restored after rendering.

See also:

setMaterialForVisibles
setMaterialForRenderables
restoreMaterials

5.27.4.3 MovablePlane * OgreRenderingRun::fullScreenQuad [static, protected, inherited]

fulls screen quad plane used in full screen quad rendering

See also:

renderFullscreenQuad

5.27.4.4 MovablePlane* OgreRenderingRun::fullScreenQuad [static, protected, inherited]

fulls screen quad plane used in full screen quad rendering

See also:

renderFullscreenQuad

5.27.4.5 Entity * OgreRenderingRun::fullScreenQuadEntity [static, protected, inherited]

fulls screen quad Entity used in full screen quad rendering

See also:

renderFullscreenQuad

5.27.4.6 Entity* OgreRenderingRun::fullScreenQuadEntity [static, protected, inherited]

fulls screen quad Entity used in full screen quad rendering

See also:

renderFullscreenQuad

5.27.4.7 SceneNode * OgreRenderingRun::fullScreenQuadNode [static, protected, inherited]

fulls screen quad SceneNode used in full screen quad rendering

See also:

renderFullscreenQuad

5.27.4.8 SceneNode* OgreRenderingRun::fullScreenQuadNode [static, protected, inherited]

fulls screen quad SceneNode used in full screen quad rendering

See also:

renderFullscreenQuad

5.27.4.9 SpriteSet* OgreRenderingRun::pixelSprites [protected, inherited]

SpriteSet used in pixel sprite rendering.

See also:

renderPixelSprites

5.27.4.10 BillboardSet* OgreRenderingRun::pixelSprites [protected, inherited]

SpriteSet used in pixel sprite rendering.

See also:

renderPixelSprites

5.27.4.11 String OgreRenderingRun::spriteSetName [protected, inherited]

unique name of the SpriteSet used in pixel sprite rendering

See also:

renderPixelSprites

5.27.4.12 Entity* OgreRenderingRun::fullscreenGrid [protected, inherited]

Entity used in fullscreen grid rendering.

See also:

renderPixelGrid

5.27.4.13 bool CubeMapRenderingRun::useDistCalc [protected, inherited]

a flag to skip cube face update if object is far away or too small.

See also:

distTolerance

5.27.4.14 bool CubeMapRenderingRun::useFaceAngleCalc [protected, inherited]

a flag to skip cube face update the face is neglible.

See also:

angleTolerance

5.27.4.15 float CubeMapRenderingRun::distTolerance [protected, inherited]

A value used in face skip test.

The higher this value gets the more precise, but slower the method will be.

5.27.4.16 float CubeMapRenderingRun::angleTolerance [protected, inherited]

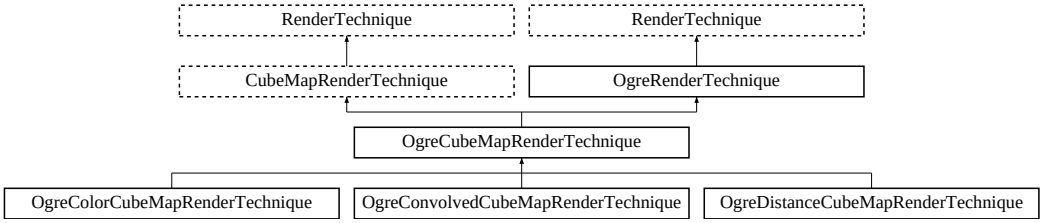
A value used in face skip test.

The higher this value gets the more precise, but slower the method will be.

5.28 OgreCubeMapRenderTechnique Class Reference

CubeMapRenderTechnique used in an Ogre environment.

Inheritance diagram for OgreCubeMapRenderTechnique::



Public Member Functions

- OgreCubeMapRenderTechnique (unsigned long startFrame, unsigned long cubeMapUpdateInterval, unsigned int cubeMapResolution, unsigned char texID, bool useDistCalc, bool useFaceAngleCalc, float distTolerance, float angleTolerance, bool updateAllFace, bool renderSelf, bool renderEnvironment, String selfMaterial, String environmentMaterial, int layer, bool getMinMax, bool attachToTexUnit, String minVariableName, String maxVariableName, Pass *pass, OgreRenderable *parentRenderable, OgreTechniqueGroup *parentTechniqueGroup, bool create-CubeRun=false)

  Constructor.

- ~OgreCubeMapRenderTechnique ()

  Destructor.

- void runChanged (RenderingRunType runType, RenderingRun *run)

  Called after one of he shared runs changes.

- void runUpdated (RenderingRunType runType, RenderingRun *run)

  Called after one of he shared runs updates.

- virtual void update (unsigned long frameNum)

  Updates the resources in the given frame.

- virtual class OgreRenderTechnique * asOgreRenderTechnique ()

  Conversion to OgreRenderTechnique.

- ElementaryRenderable * getParentRenderable ()

  Retrieves the renderable this technique operates on.

- virtual OgreRenderTechnique * asOgreRenderTechnique ()

  Conversion to OgreRenderTechnique.

Protected Member Functions

- RenderingRun * createCubeMapRun ()

  Creates a CubeMapRenderingRun.

- void cubeMapRunChanged (RenderingRun *run)

  Called if the changed run is a CubeMapRenderingRun.

- void cubeMapRunUpdated (RenderingRun *run)

  Called if the updated run is a CubeMapRenderingRun.

Protected Attributes

- unsigned char texID

  the id of the texture unit state the resulting cubemap should be bound to

- String selfMaterial

  the material that should be set for the object while rendering the cubemap

- String environmentMaterial

  the material that should be set for the environment while rendering the cubemap

- bool getMinMax

  sets if the minimum and maximum values of the cubemap should be computed

- bool attachToTexUnit

  sets if this cubemap should be attach to a texture unit of the pass

- String minVariableName

  sets the name of the gpu shader program parameter to which the minimum value should be bound to

- String maxVariableName

  sets the name of the gpu shader program parameter to which the maximum value should be bound to

- bool useDistCalc

  a flag to skip cube face update if object is far away or too small.

- bool useFaceAngleCalc

  a flag to skip cube face update the face is neglible.

- float distTolerance

  A value used in face skip test.

- float angleTolerance

  A value used in face skip test.

- bool updateAllFace

  defines if all cubemap faces should be updated in a frame or only one face per frame

- unsigned long cubeMapUpdateInterval

  color-cubemap update frequency

- unsigned int cubeMapResolution

  color-cubemap resolution

- unsigned long startFrame

  offset in frame number used during update

- bool renderSelf

  Sets if the object should be rendered to the cube map.

- bool renderEnvironment

  Sets if the environment should be rendered to the cube map.

- int layer

  The layer of the cubemap.

- RenderingRunType cubemapLayer

  The exact run type of this run (according to the actual layer).

- ElementaryRenderable * parentRenderable

  The renderable this technique operates on.

- TechniqueGroup * parentTechniqueGroup

  The TechniqueGroup this RenderedTechnique is attached to.

- SharedRuns * sharedRuns

  The SharedRuns this RenderedTechnique is attached to.

- OgreRenderable * parentOgreRenderable

  a OgreRenderable pointer to the renderable this technique operates on.

- OgreTechniqueGroup * parentOgreTechniqueGroup

  a OgreTechniqueGroup pointer to the TechniqueGroup this technique is attached to.

- Pass * pass

  a pointer to the pass this technique operates on.

5.28.1 Detailed Description

CubeMapRenderTechnique used in an Ogre environment.

5.28.2 Constructor & Destructor Documentation

5.28.2.1 OgreCubeMapRenderTechnique::OgreCubeMapRenderTechnique (unsigned long *startFrame*, unsigned long *cubeMapUpdateInterval*, unsigned int *cubeMapResolution*, unsigned char *texID*, bool *useDistCalc*, bool *useFaceAngleCalc*, float *distTolerance*, float *angleTolerance*, bool *updateAllFace*, bool *renderSelf*, bool *renderEnvironment*, String *selfMaterial*, String *environmentMaterial*, int *layer*, bool *getMinMax*, bool *attachToTexUnit*, String *minVariableName*, String *maxVariableName*, Pass * *pass*, OgreRenderable * *parentRenderable*, OgreTechniqueGroup * *parentTechniqueGroup*, bool *createCubeRun* = `false`)

Constructor.

Parameters:

startFrame adds an offset to the current frame number to help evenly distribute updates between frames

cubeMapUpdateInterval update frequency

cubeMapResolution color cubemap resolution

texID the id of the texture unit state the resulting cubemap should be bound to

useDistCalc flag to skip cube face update if object is far away

useFaceAngleCalc flag to skip cube face update if face is neglible

distTolerance distance tolerance used in face skip

angleTolerance angle tolerance used in face skip

updateAllFace defines if all cubemap faces should be updated in a frame or only one face per frame

renderSelf sets if the object should be rendered to the cube map

renderEnvironment sets if the environment should be rendered to the cube map

selfMaterial the material that should be set for the object while rendering the cubemap

environmentMaterial the material that should be set for the environment while rendering the cubemap

layer the layer of this cubemap

getMinMax sets if the minimum and maximum values of the cubemap should be computed

attachToTexUnit sets if this cubemap should be attach to a texture unit of the pass

minVariableName sets the name of the gpu shader program parameter to which the minimum value should be bound to

maxVariableName sets the name of the gpu shader program parameter to which the maximum value should be bound to

pass the pass to operate on

parentRenderable the object to operate on

parentTechniqueGroup the TechniqueGroup this RenderedTechnique is attached to

5.28.3 Member Function Documentation

5.28.3.1 RenderingRun * OgreCubeMapRenderTechnique::createCubeMapRun () `[protected, virtual]`

Creates a CubeMapRenderingRun.

Returns:

the new CubeMapRenderingRun instance.

Implements CubeMapRenderTechnique.

5.28.3.2 void OgreCubeMapRenderTechnique::cubeMapRunChanged (RenderingRun * *run*) [protected, virtual]

Called if the changed run is a CubeMapRenderingRun.

Parameters:

run pointer to the new CubeMapRenderingRun

Implements CubeMapRenderTechnique.

5.28.3.3 void OgreCubeMapRenderTechnique::cubeMapRunUpdated (RenderingRun * *run*) [protected, virtual]

Called if the updated run is a CubeMapRenderingRun.

Parameters:

run pointer to the updated CubeMapRenderingRun

Implements CubeMapRenderTechnique.

5.28.3.4 void CubeMapRenderTechnique::runChanged (RenderingRunType *runType*, RenderingRun * *run*) [virtual, inherited]

Called after one of he shared runs changes.

Parameters:

runType enum describing the type of the changed run

run pointer to the changed RenderingRun

Reimplemented from RenderTechnique.

Reimplemented in ColorCubeMapRenderTechnique, ConvolvedCubeMapRenderTechnique, and DistanceCubeMapRenderTechnique.

5.28.3.5 void CubeMapRenderTechnique::runUpdated (RenderingRunType *runType*, RenderingRun * *run*) [virtual, inherited]

Called after one of he shared runs updates.

Parameters:

runType enum describing the type of the updated run

run pointer to the updated RenderingRun

Reimplemented from RenderTechnique.

Reimplemented in ColorCubeMapRenderTechnique, and DistanceCubeMapRenderTechnique.

5.28.3.6 virtual void RenderTechnique::update (unsigned long *frameNum*) [inline, virtual, inherited]

Updates the resources in the given frame.

A RenderTechnique is usually need some resources from several runs, so these runs will be updated.

Parameters:

frameNum the actual framenumber

Reimplemented in ColorCubeMapRenderTechnique, ConvolvedCubeMapRenderTechnique, DistanceCubeMapRenderTechnique, HierarchicalParticleSystemTechnique, IllumVolumeRenderTechnique, OgreCausticReceiverRenderTechnique, OgreColorCubeMapRenderTechnique, OgreConvolvedCubeMapRenderTechnique, OgreDepthShadowReceiverRenderTechnique, OgreDistanceCubeMapRenderTechnique, OgreFireRenderTechnique, OgrePathMapRenderTechnique, and OgreSBBRenderTechnique.

5.28.3.7 virtual class OgreRenderTechnique* RenderTechnique::asOgreRenderTechnique () [inline, virtual, inherited]

Conversion to OgreRenderTechnique.

This function is needed because of virtual inheritance.

Reimplemented in OgreRenderTechnique.

5.28.3.8 virtual OgreRenderTechnique* OgreRenderTechnique::asOgreRenderTechnique () [inline, virtual, inherited]

Conversion to OgreRenderTechnique.

This function is needed because of virtual inheritance.

Reimplemented from RenderTechnique.

5.28.4 Member Data Documentation

5.28.4.1 bool CubeMapRenderTechnique::useDistCalc [protected, inherited]

a flag to skip cube face update if object is far away or too small.

See also:

distTolerance

5.28.4.2 bool CubeMapRenderTechnique::useFaceAngleCalc [protected, inherited]

a flag to skip cube face update the face is neglible.

See also:

angleTolerance

5.28.4.3 float CubeMapRenderTechnique::distTolerance [protected, inherited]

A value used in face skip test.

The higher this value gets the more precise, but slower the method will be.

5.28.4.4 float CubeMapRenderTechnique::angleTolerance [protected, inherited]

A value used in face skip test.

The higher this value gets the more precise, but slower the method will be.

5.28.4.5 int CubeMapRenderTechnique::layer [protected, inherited]

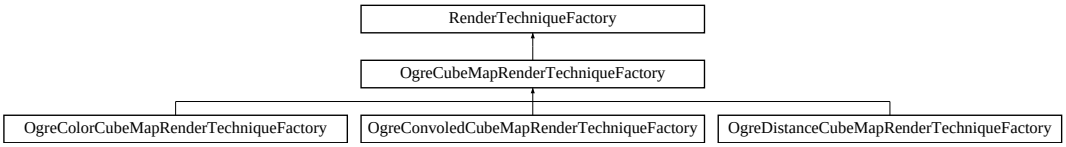
The layer of the cubemap.

This extension was created to achieve effect such as multiple reflections and refractions that require more complete sampling of the scene. With this extension we can render the environment into several cubemap layers eg. with depth peeling.

5.29 OgreCubeMapRenderTechniqueFactory Class Reference

RenderTechniqueFactory to create OgreCubeMapRenderTechnique instances.

Inheritance diagram for OgreCubeMapRenderTechniqueFactory::



Public Member Functions

- OgreCubeMapRenderTechniqueFactory ()

  Technique factory.

- OgreRenderTechnique * createInstance (IllumTechniqueParams *params, Pass *pass, OgreRenderable *parentRenderable, OgreTechniqueGroup *parentTechniqueGroup)

  Creates a RenderTechnique of the factory type.

- bool isType (String type)

  Returns if this factory can create a RenderTechnique of the given type.

- virtual void parseParams (IllumTechniqueParams *params)

  parses parameters from the material file.

Protected Types

- typedef void(*) ILLUM_ATTRIBUTE_PARSER (String &params, RenderTechniqueFactory *factory)

  function for parsing RenderTechnique attributes

- typedef std::map< String, ILLUM_ATTRIBUTE_PARSER > AttribParserList

  Keyword-mapped attribute parsers.

Protected Attributes

- AttribParserList attributeParsers

  map of parser functions

- String typeName

  factoryname

5.29.1 Detailed Description

RenderTechniqueFactory to create OgreCubeMapRenderTechnique instances.

5.29.2 Member Typedef Documentation

5.29.2.1 typedef void(*) RenderTechniqueFactory::ILLUM_ATTRIBUTE_PARSER(String &params, RenderTechniqueFactory *factory) [protected, inherited]

function for parsing RenderTechnique attributes

Parameters:

params attribute value stored in a String

5.29.3 Member Function Documentation

5.29.3.1 OgreRenderTechnique * OgreCubeMapRenderTechniqueFactory::createInstance (IllumTechniqueParams * *params*, Pass * *pass*, OgreRenderable * *parentRenderable*, OgreTechniqueGroup * *parentTechniqueGroup*) [virtual]

Creates a RenderTechnique of the factory type.

Parameters:

params containes constructor parameters as NameValuePairList

pass the Pass to use in RenderTechnique constructor

pass the parentRenderable to pass to RenderTechnique constructor

pass the parentTechniqueGroup to pass to RenderTechnique constructor

Implements RenderTechniqueFactory.

Reimplemented in OgreColorCubeMapRenderTechniqueFactory, OgreConvoledCubeMapRenderTechniqueFactory, and OgreDistanceCubeMapRenderTechniqueFactory.

5.29.3.2 bool RenderTechniqueFactory::isType (String *type*) [inline, inherited]

Returns if this factory can create a RenderTechnique of the given type.

Parameters:

type RenderTechnique type

5.29.3.3 void RenderTechniqueFactory::parseParams (IllumTechniqueParams * *params*) [virtual, inherited]

parses parameters from the material file.

The parsed parameters will be passed to the new RenderTechnique's constructor.

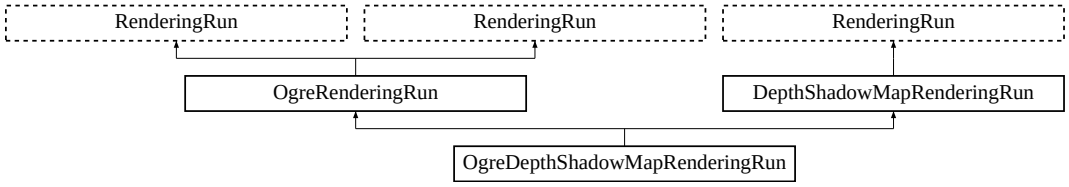
Parameters:

params pointer to the IllumTechniqueParams structure that was read from the material script and containes the parameters to be parsed.

5.30 OgreDepthShadowMapRenderingRun Class Reference

DepthShadowMapRenderingRun used in an OGRE environment.

Inheritance diagram for OgreDepthShadowMapRenderingRun::



Public Member Functions

- OgreDepthShadowMapRenderingRun (OgreSharedRuns *sharedRuns, String name, Light *light, unsigned int resolutionX, unsigned int resolutionY, String materialName)

  Constructor.

- const String & getDepthMapTextureName ()

  returns the depth shadow map texture created by this run

- void refreshLight (unsigned long frameNum)

  Refreshes light camera matrices, called in each update.

- Matrix4 getLightViewMatrix ()

  Retuns the view matrix of the camera from which the depth shadow map was created.

- Matrix4 getLightViewProjMatrix ()

  Retuns the concatenation of the view and projection matrices of the camera from which the depth shadow map was created.

- Real getLightFarPlane ()

  Retuns the far plane of the light projection.

- OgreRenderingRun * asOgreRenderingRun ()

  Conversion to OgreRenderRun.

- OgreRenderingRun * asOgreRenderingRun ()

  Conversion to OgreRenderRun.

- bool update (unsigned long frameNum)

  Calls updateFrame() if the run needs update according to its starting frame and update interval and has not been allready updated in this frame.

- virtual bool canJoin (RenderingRun *run)

  Returns true if two runs can be joined.

Protected Member Functions

- void updateFrame (unsigned long frameNum)

  This function does the actual update in a frame.

- void createDepthMap ()

  Creates the depth map texture (2D or CUBE according to light type).

- void updateDepthCubeFace (int facenum)

  Updates one face of the depth cubemap (used only in case of point lights).

- void updateDepthMap ()

  Updates the depth map (in case of directional and spot lights).

- Vector3 getCubeMapFaceDirection (unsigned char faceId)

  Returns a direction for a cubemap face id.

- Vector3 getCubeMapFaceDirection (unsigned char faceId)

  Returns a direction for a cubemap face id.

- Texture * createCubeRenderTexture (String name, const Vector3 position, unsigned int resolution=512, PixelFormat format=PF_FLOAT16_RGBA, int numMips=0, ColourValue clearColor=ColourValue::Black)

  Creates a cubemap texture.

- Texture * createCubeRenderTexture (String name, const Vector3 position, unsigned int resolution=512, PixelFormat format=PF_FLOAT16_RGBA, int numMips=0, ColourValue clearColor=ColourValue::Black)

  Creates a cubemap texture.

- void setMaterialForRenderables (String &materialName, RenderQueue *rq)

  Sets the given material for each Renderable in a RenderQueue.

- void setMaterialForRenderables (String &materialName, RenderQueue *rq, bool solidsonly=false)

  Sets the given material for each Renderable in a RenderQueue.

- void setMaterialForVisibles (String &materialName, Camera *cam, bool shadowcastersonly=false)

  Sets the given material for each Renderable visible from a given camera.

- void setMaterialForVisibles (String &materialName, Camera *cam, bool shadowcastersonly=false, bool solidsonly=false)

  Sets the given material for each Renderable visible from a given camera.

- void restoreMaterials ()

  Restores previously stored materials.

- void restoreMaterials ()

  Restores previously stored materials.

- void renderFullscreenQuad (String materialName, RenderTarget *target)

Renderes a full screen quad on a given RendderTarget with a given material.

- void renderFullscreenQuad (String materialName, RenderTarget *target)

  Renderes a full screen quad on a given RendderTarget with a given material.

- void renderPixelSprites (String &materialName, RenderTarget *rt, int width, int height)

  Renderes sprites to pixels of the screen on a given RendderTarget with a given material.

- void renderPixelSprites (String &materialName, RenderTarget *rt, int width, int height)

  Renderes sprites to pixels of the screen on a given RendderTarget with a given material.

- void renderFullscreenGrid (String &materialName, RenderTarget *rt, int width, int height)

  Renders a grid onto the screen.

- virtual bool needUpdate (unsigned long frameNum)

  Returns if this run needs update.

Protected Attributes

- Light * light

  The light source this depth shadow map belongs to.

- Camera * depthMapCamera

  Pointer to the camera of the lightsource.

- String materialName

  The name of the material to be used when rendering the depth shadow map.

- OgreSharedRuns * sharedRuns

  A pointer to the OgreSharedRuns this run belongs to.

- String name

  The name of the depth shadow map texture that was created by this run.

- String blurredname

  The name of the blurred depth shadow map texture that was created by this run.

- Texture * depthMapTexture

  A pointer to the depth shadow texture that was created by this run.

- Texture * blurredDepthMapTexture

  A pointer to the blurred depth shadow texture that was created by this run.

- std::map< Renderable *, String > visibleObjects

  map of Renderables which will be rendered with a given material

- std::map< Renderable *, String > visibleObjects

  map of Renderables which will be rendered with a given material

- SpriteSet * pixelSprites

  SpriteSet used in pixel sprite rendering.

- BillboardSet * pixelSprites

  SpriteSet used in pixel sprite rendering.

- String spriteSetName

  unique name of the SpriteSet used in pixel sprite rendering

- Entity * fullscreenGrid

  Entity used in fullscreen grid rendering.

- unsigned long lastupdated

  The number of the last frame this run was updated.

- unsigned long startFrame

  The number of the frame this run should be updated first.

- unsigned long updateInterval

  Refresh frequency in frames.

- unsigned int resolutionX

  width of the depth map texture

- unsigned int resolutionY

  height of the depth map texture

Static Protected Attributes

- static MovablePlane * fullScreenQuad

  fulls screen quad plane used in full screen quad rendering

- static MovablePlane * fullScreenQuad

  fulls screen quad plane used in full screen quad rendering

- static Entity * fullScreenQuadEntity

  fulls screen quad Entity used in full screen quad rendering

- static Entity * fullScreenQuadEntity

  fulls screen quad Entity used in full screen quad rendering

- static SceneNode * fullScreenQuadNode

  fulls screen quad SceneNode used in full screen quad rendering

- static SceneNode * fullScreenQuadNode

  fulls screen quad SceneNode used in full screen quad rendering

5.30.1 Detailed Description

DepthShadowMapRenderingRun used in an OGRE environment.

5.30.2 Constructor & Destructor Documentation

5.30.2.1 OgreDepthShadowMapRenderingRun::OgreDepthShadowMapRenderingRun (OgreSharedRuns * *sharedRuns*, String *name*, Light * *light*, unsigned int *resolutionX*, unsigned int *resolutionY*, String *materialName*)

Constructor.

Parameters:

sharedRuns a pointer to the OgreSharedRuns this run belongs to

name the name of the depth map texture to be created

light the light source this depth shadow map belongs to

resolutionX the resolution width of the depth shadow map

resolutionY the resolution height of the depth shadow map

materialName the name of the material to be used when rendering the depth shadow map

5.30.3 Member Function Documentation

5.30.3.1 void OgreDepthShadowMapRenderingRun::refreshLight (unsigned long *frameNum*)

Refreshes light camera matrices, called in each update.

n = 100000;

n = 100000;

5.30.3.2 void OgreDepthShadowMapRenderingRun::updateFrame (unsigned long *frameNum*) [protected, virtual]

This function does the actual update in a frame.

Parameters:

frameNum current frame number

Implements DepthShadowMapRenderingRun.

5.30.3.3 void OgreDepthShadowMapRenderingRun::updateDepthCubeFace (int *facenum*) [protected, virtual]

Updates one face of the depth cubemap (used only in case of point lights).

Parameters:

facenum the number of the face to be updated

Implements DepthShadowMapRenderingRun.

5.30.3.4 OgreRenderingRun* OgreRenderingRun::asOgreRenderingRun () [inline, virtual, inherited]

Conversion to OgreRenderRun.

This function is needed because of virtual inheritance.

Reimplemented from RenderingRun.

5.30.3.5 OgreRenderingRun* OgreRenderingRun::asOgreRenderingRun () [inline, virtual, inherited]

Conversion to OgreRenderRun.

This function is needed because of virtual inheritance.

Reimplemented from RenderingRun.

5.30.3.6 Vector3 OgreRenderingRun::getCubeMapFaceDirection (unsigned char *faceId*) [protected, inherited]

Returns a direction for a cubemap face id.

This is a helper function to retrieve the normal direction of a given cubemap face.

Parameters:

faceId the number of the face

5.30.3.7 Vector3 OgreRenderingRun::getCubeMapFaceDirection (unsigned char *faceId*) [protected, inherited]

Returns a direction for a cubemap face id.

This is a helper function to retrieve the normal direction of a given cubemap face.

Parameters:

faceId the number of the face

5.30.3.8 Texture * OgreRenderingRun::createCubeRenderTexture (String *name*, const Vector3 *position*, unsigned int *resolution* = 512, PixelFormat *format* = PF_FLOAT16_RGBA, int *numMips* = 0, ColourValue *clearColor* = ColourValue::Black) [protected, inherited]

Creates a cubemap texture.

This is a helper function to easily create a cubemap texture and automaticly attach viewports to each face so it can be used as a rendertarget.

Parameters:

name the name of the texture to be created

position the initial position of the cubemap

resolution the resolution of one cubemapface

format the pixel format of the cubemap

numMips the number of mipmap levels

clearColor initial color

5.30.3.9 Texture* OgreRenderingRun::createCubeRenderTexture (String *name*, const Vector3 *position*, unsigned int *resolution* = 512, PixelFormat *format* = PF_FLOAT16_RGBA, int *numMips* = 0, ColourValue *clearColor* = ColourValue::Black) [protected, inherited]

Creates a cubemap texture.

This is a helper function to easily create a cubemap texture and automaticly attach viewports to each face so it can be used as a rendertarget.

Parameters:

name the name of the texture to be created

position the initial position of the cubemap

resolution the resolution of one cubemapface

format the pixel format of the cubemap

numMips the number of mipmap levels

clearColor initial color

5.30.3.10 void OgreRenderingRun::setMaterialForRenderables (String & *materialName*, RenderQueue * *rq*) [protected, inherited]

Sets the given material for each Renderable in a RenderQueue.

This is a helper function to set a material to each element of a previously filled Renderque. The orginal material of the Renderables are stored so they can be restored later. The function also tells the current SceneManager not to search for visible objects, as we are going to use the given RenderQueue during the next rendering.

Parameters:

materialName the name of the material to set for the Renderables

rq pointer to the filled Renderqueue instance to set material for

5.30.3.11 void OgreRenderingRun::setMaterialForRenderables (String & *materialName*, RenderQueue * *rq*, bool *solidsonly* = `false`) [protected, inherited]

Sets the given material for each Renderable in a RenderQueue.

This is a helper function to set a material to each element of a previously filled Renderque. The orginal material of the Renderables are stored so they can be restored later. The function also tells the current SceneManager not to search for visible objects, as we are going to use the given RenderQueue during the next rendering.

Parameters:

materialName the name of the material to set for the Renderables

rq pointer to the filled Renderqueue instance to set material for

5.30.3.12 void OgreRenderingRun::setMaterialForVisibles (String & *materialName*, Camera * *cam*, bool *shadowcastersonly* = `false`) [protected, inherited]

Sets the given material for each Renderable visible from a given camera.

This helper function is similar to setMaterialForRenderables but it is also responsible for filling the RenderQueue. First the RenderQueue of the current SceneManager fill be filled with the visible objects seen from the given camera. Then the required material will be set for each element of the RenderQueue. The orginal material of the Renderables are stored so they can be restored later. The function also tells the current SceneManager not to search for visible objects, as we are going to use the filled RenderQueue during the next rendering.

Parameters:

materialName the name of the material to set for the Renderables

cam pointer to the camera from which visible objects should be searched

shadowcastersonly flag to search for only shadow casters

5.30.3.13 void OgreRenderingRun::setMaterialForVisibles (String & *materialName*, Camera * *cam*, bool *shadowcastersonly* = `false`, bool *solidsonly* = `false`) [protected, inherited]

Sets the given material for each Renderable visible from a given camera.

This helper function is similar to setMaterialForRenderables but it is also responsible for filling the RenderQueue. First the RenderQueue of the current SceneManager fill be filled with the visible objects seen from the given camera. Then the required material will be set for each element of the RenderQueue. The orginal

material of the Renderables are stored so they can be restored later. The function also tells the current SceneManager not to search for visible objects, as we are going to use the filled RenderQueue during the next rendering.

Parameters:

materialName the name of the material to set for the Renderables

cam pointer to the camera from which visible objects should be searched

shadowcastersonly flag to search for only shadow casters

5.30.3.14 void OgreRenderingRun::restoreMaterials () [protected, inherited]

Restores previously stored materials.

This helper function is typically used after a setMaterialForRenderables or setMaterialForVisibles call and a rendering process to restore the original material settings. The function also tells the current SceneManager to search for visible objects, as this is the default behaviour of SceneManager.

5.30.3.15 void OgreRenderingRun::restoreMaterials () [protected, inherited]

Restores previously stored materials.

This helper function is typically used after a setMaterialForRenderables or setMaterialForVisibles call and a rendering process to restore the original material settings. The function also tells the current SceneManager to search for visible objects, as this is the default behaviour of SceneManager.

5.30.3.16 void OgreRenderingRun::renderFullscreenQuad (String *materialName*, RenderTarget * *target*) [protected, inherited]

Renderes a full screen quad on a given RendderTarget with a given material.

Parameters:

materialName the name of the material bind to the quad

target the RenderTarget the quad should be rendered on

5.30.3.17 void OgreRenderingRun::renderFullscreenQuad (String *materialName*, RenderTarget * *target*) [protected, inherited]

Renderes a full screen quad on a given RendderTarget with a given material.

Parameters:

materialName the name of the material bind to the quad

target the RenderTarget the quad should be rendered on

5.30.3.18 void OgreRenderingRun::renderPixelSprites (String & *materialName*, RenderTarget * *rt*, int *width*, int *height*) [protected, inherited]

Renderes sprites to pixels of the screen on a given RendderTarget with a given material.

Pixel sprites are pixel sized quads, placed on each pixel of the RenderTarget. The number of sprites not necessary corresponds to the resolution of the rendertarget. The pixel quads will evenly fill the rendertarget's area with sizes corresponding to the given resolution. We can render fewer or more pixel quads than the number of pixels the rendertarget has (eg.: in case of caustic cubemap generation).

Parameters:

materialName the name of the material bind to the pixel sprites

rt the RenderTarget the quads should be rendered on

width the desired resolution width of the sprites

height the desired resolution height of the sprites

5.30.3.19 void OgreRenderingRun::renderPixelSprites (String & *materialName*, RenderTarget * *rt*, int *width*, int *height*) [protected, inherited]

Renderes sprites to pixels of the screen on a given RendderTarget with a given material.

Pixel sprites are pixel sized quads, placed on each pixel of the RenderTarget. The number of sprites not necessary corresponds to the resolution of the rendertarget. The pixel quads will evenly fill the rendertarget's area with sizes corresponding to the given resolution. We can render fewer or more pixel quads than the number of pixels the rendertarget has (eg.: in case of caustic cubemap generation).

Parameters:

materialName the name of the material bind to the pixel sprites

rt the RenderTarget the quads should be rendered on

width the desired resolution width of the sprites

height the desired resolution height of the sprites

5.30.3.20 void OgreRenderingRun::renderFullscreenGrid (String & *materialName*, RenderTarget * *rt*, int *width*, int *height*) [protected, inherited]

Renders a grid onto the screen.

Parameters:

rt the RenderTarget the grid should be rendered on

width the desired horizontal resolution of the grid

height the desired vertical resolution of the grid

5.30.3.21 virtual bool RenderingRun::canJoin (RenderingRun * *run*) [inline, virtual, inherited]

Returns true if two runs can be joined.

In some cases special requirements should stand to join two runs (even if they have the same type). Eg.: two caustic cube map generation technique should only be joined if they use the same material when rendering the caustic cubemap.

Reimplemented in OgreCausticCubeMapRenderingRun, OgreChildPSystemRenderingRun, and OgrePhotonMapRenderingRun.

5.30.3.22 virtual bool RenderingRun::needUpdate (unsigned long *frameNum*) [inline, protected, virtual, inherited]

Returns if this run needs update.

This tipically depends on the upate interval and the starting frame number.

Parameters:

frameNum current frame number

Reimplemented in OgreFocusingMapRenderingRun, and OgrePMEntryPointMapRenderingRun.

5.30.4 Member Data Documentation

5.30.4.1 std::map<Renderable*, String> OgreRenderingRun::visibleObjects [protected, inherited]

map of Renderables which will be rendered with a given material

The String stores the original material name that will be restored after rendering.

See also:

setMaterialForVisibles
setMaterialForRenderables
restoreMaterials

5.30.4.2 std::map<Renderable*, String> OgreRenderingRun::visibleObjects [protected, inherited]

map of Renderables which will be rendered with a given material

The String stores the original material name that will be restored after rendering.

See also:

setMaterialForVisibles

setMaterialForRenderables
restoreMaterials

5.30.4.3 MovablePlane * OgreRenderingRun::fullScreenQuad [static, protected, inherited]

fulls screen quad plane used in full screen quad rendering

See also:

renderFullscreenQuad

5.30.4.4 MovablePlane* OgreRenderingRun::fullScreenQuad [static, protected, inherited]

fulls screen quad plane used in full screen quad rendering

See also:

renderFullscreenQuad

5.30.4.5 Entity * OgreRenderingRun::fullScreenQuadEntity [static, protected, inherited]

fulls screen quad Entity used in full screen quad rendering

See also:

renderFullscreenQuad

5.30.4.6 Entity* OgreRenderingRun::fullScreenQuadEntity [static, protected, inherited]

fulls screen quad Entity used in full screen quad rendering

See also:

renderFullscreenQuad

5.30.4.7 SceneNode * OgreRenderingRun::fullScreenQuadNode [static, protected, inherited]

fulls screen quad SceneNode used in full screen quad rendering

See also:

renderFullscreenQuad

5.30.4.8 SceneNode* OgreRenderingRun::fullScreenQuadNode [static, protected, inherited]

fulls screen quad SceneNode used in full screen quad rendering

See also:

renderFullscreenQuad

5.30.4.9 SpriteSet* OgreRenderingRun::pixelSprites [protected, inherited]

SpriteSet used in pixel sprite rendering.

See also:

renderPixelSprites

5.30.4.10 BillboardSet* OgreRenderingRun::pixelSprites [protected, inherited]

SpriteSet used in pixel sprite rendering.

See also:

renderPixelSprites

5.30.4.11 String OgreRenderingRun::spriteSetName [protected, inherited]

unique name of the SpriteSet used in pixel sprite rendering

See also:

renderPixelSprites

5.30.4.12 Entity* OgreRenderingRun::fullscreenGrid [protected, inherited]

Entity used in fullscreen grid rendering.

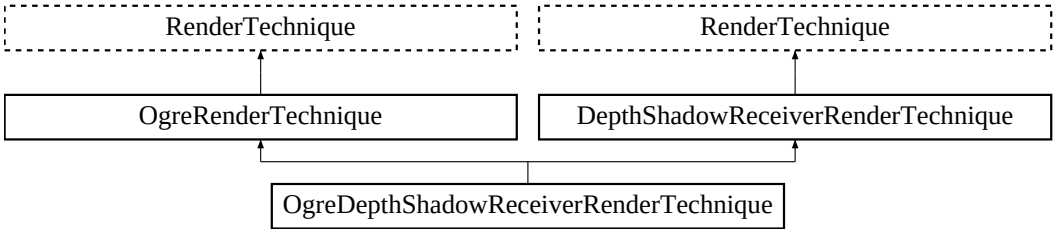
See also:

renderPixelGrid

5.31 OgreDepthShadowReceiverRenderTechnique Class Reference

DepthShadowReceiverRenderTechnique used in an OGRE environment.

Inheritance diagram for OgreDepthShadowReceiverRenderTechnique::



Public Member Functions

- OgreDepthShadowReceiverRenderTechnique (int maxlights, String shadowVertexProgram, String shadowFragmentProgram, String WorldViewProjParamName, String WorldParamName, bool setLightViewMatrix, bool setLightViewProjMatrix, bool setLightProjFarPlane, String lightViewProjParamName, String lightViewParamName, String lightFarPlaneParamName, Pass *pass, OgreRenderable *parentRenderable, OgreTechniqueGroup *parentTechniqueGroup)

  Constructor.

- ~OgreDepthShadowReceiverRenderTechnique ()

  Destructor.

- virtual void update (unsigned long frameNum)

  Updates the resources in the given frame.

- virtual OgreRenderTechnique * asOgreRenderTechnique ()

  Conversion to OgreRenderTechnique.

- virtual void runChanged (RenderingRunType runType, RenderingRun *run)

  Called after one of he shared runs changes.

- virtual void runUpdated (RenderingRunType runType, RenderingRun *run)

  Called after one of he shared runs updates.

- ElementaryRenderable * getParentRenderable ()

  Retrieves the renderable this technique operates on.

Protected Attributes

- int maxlights

  the maximum number of light sources to recieve shadow from

- String shadowVertexProgram

  the vertex program to be used in the shadowing passes

- String shadowFragmentProgram

  the fragment program to be used in the shadowing passes

- std::vector< Pass * > passes
- bool setLightViewMatrix

  bound light space view matrix to a gpu program parameter

- bool setLightViewProjMatrix

  bound light space view-projection matrix to a gpu program parameter

- bool setLightProjFarPlane

  bound light space projection far plane to a gpu program parameter

- String lightViewProjParamName

  the name of the gpu program parameter the light space view-projection matrix should be bound to

- String lightViewParamName

  the name of the gpu program parameter the light space view matrix should be bound to

- String lightFarPlaneParamName

  the name of the gpu program parameter the light space projection far plane should be bound to

- String WorldViewProjParamName

  the name of the gpu program parameter the world-view-projection matrix should be bound to

- String WorldParamName

  the name of the gpu program parameter the world matrix should be bound to

- OgreRenderable * parentOgreRenderable

  a OgreRenderable pointer to the renderable this technique operates on.

- OgreTechniqueGroup * parentOgreTechniqueGroup

  a OgreTechniqueGroup pointer to the TechniqueGroup this technique is attached to.

- Pass * pass

  a pointer to the pass this technique operates on.

- ElementaryRenderable * parentRenderable

  The renderable this technique operates on.

- TechniqueGroup * parentTechniqueGroup

  The TechniqueGroup this RenderedTechnique is attached to.

- SharedRuns * sharedRuns

  The SharedRuns this RenderedTechnique is attached to.

5.31.1 Detailed Description

DepthShadowReceiverRenderTechnique used in an OGRE environment.

This technique defines that the object will recieve shadows with the help of depth shadow maps. Each lightsource can have a depth map assigned to it. These are going to be refreshed only if shadow receivers are visible. It is the shadow receiver technique's resposibility to refresh them.

The shadows from each light are calculated in separate passes. Each pass will modulate the shaded image, so thes should be the last passes (but before caustic passes). The given Pass* parameter n the constructor defines the pass after which new shadow recieving passes will be added by the technique.

5.31.2 Constructor & Destructor Documentation

5.31.2.1 OgreDepthShadowReceiverRenderTechnique::OgreDepthShadowReceiverRenderTechnique (int *maxlights*, String *shadowVertexProgram*, String *shadowFragmentProgram*, String *WorldViewProjParamName*, String *WorldParamName*, bool *setLightViewMatrix*, bool *setLightViewProjMatrix*, bool *setLightProjFarPlane*, String *lightViewProjParamName*, String *lightViewParamName*, String *lightFarPlaneParamName*, Pass * *pass*, OgreRenderable * *parentRenderable*, OgreTechniqueGroup * *parentTechniqueGroup*)

Constructor.

Parameters:

maxlights the maximum number of light sources to recieve shadow from

shadowVertexProgram the vertex program to be used in the shadowing passes

shadowFragmentProgram the fragment program to be used in the shadowing passes It should have one pass and the depth map of a light will be bound to the first sampler unit.

WorldViewProjParamName the name of the gpu program parameter the world-view-projection matrix should be bound to

WorldParamName the name of the gpu program parameter the world matrix should be bound to

setLightViewMatrix bound light space view matrix to a gpu program parameter

setLightViewProjMatrix bound light space view-projection matrix to a gpu program parameter

setLightProjFarPlane bound light space projection far plane to a gpu program parameter

lightViewProjParamName the name of the gpu program parameter the light space view-projection matrix should be bound to

lightViewParamName the name of the gpu program parameter the light space view matrix should be bound to

lightFarPlaneParamName the name of the gpu program parameter the light space projection far plane should be bound to

pass the pass after which shadowing passes should be added

parentRenderable the object to operate on

parentTechniqueGroup the TechniqueGroup this RenderedTechnique is attached to

5.31.3 Member Function Documentation

5.31.3.1 void OgreDepthShadowReceiverRenderTechnique::update (unsigned long *frameNum*) [virtual]

Updates the resources in the given frame.

A RenderTechnique is usually need some resources from several runs, so these runs will be updated.

Parameters:

frameNum the actual framenumber

Reimplemented from RenderTechnique.

5.31.3.2 virtual OgreRenderTechnique* OgreRenderTechnique::asOgreRenderTechnique () [inline, virtual, inherited]

Conversion to OgreRenderTechnique.

This function is needed because of virtual inheritance.

Reimplemented from RenderTechnique.

5.31.3.3 virtual void RenderTechnique::runChanged (RenderingRunType *runType*, RenderingRun * *run*) [inline, virtual, inherited]

Called after one of he shared runs changes.

Parameters:

runType enum describing the type of the changed run

run pointer to the changed RenderingRun

Reimplemented in CausticCasterRenderTechnique, ColorCubeMapRenderTechnique, ConvolvedCubeMapRenderTechnique, CubeMapRenderTechnique, DistanceCubeMapRenderTechnique, HierarchicalParticleSystemTechnique, and IllumVolumeRenderTechnique.

5.31.3.4 virtual void RenderTechnique::runUpdated (RenderingRunType *runType*, RenderingRun * *run*) [inline, virtual, inherited]

Called after one of he shared runs updates.

Parameters:

runType enum describing the type of the updated run

run pointer to the updated RenderingRun

Reimplemented in CausticCasterRenderTechnique, ColorCubeMapRenderTechnique, CubeMapRenderTechnique, DistanceCubeMapRenderTechnique, HierarchicalParticleSystemTechnique, and IllumVolumeRenderTechnique.

5.31.4 Member Data Documentation

5.31.4.1 int OgreDepthShadowReceiverRenderTechnique::maxlights [protected]

the maximum number of light sources to recieve shadow from

During update the nearest light sources will be found and used.

5.31.4.2 String OgreDepthShadowReceiverRenderTechnique::shadowFragmentProgram [protected]

the fragment program to be used in the shadowing passes

It should have one pass and the depth map of a light will be bound to the first sampler unit.

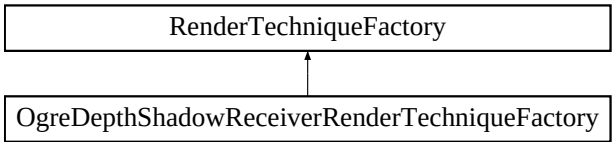
5.31.4.3 std::vector<Pass*> OgreDepthShadowReceiverRenderTechnique::passes [protected]

new passes created by this technique

5.32 OgreDepthShadowReceiverRenderTechniqueFactory Class Reference

RenderTechniqueFactory to create OgreDepthShadowReceiverRenderTechnique instances.

Inheritance diagram for OgreDepthShadowReceiverRenderTechniqueFactory::



Public Member Functions

- OgreRenderTechnique * createInstance (IllumTechniqueParams *params, Pass *pass, OgreRenderable *parentRenderable, OgreTechniqueGroup *parentTechniqueGroup)

  Creates a RenderTechnique of the factory type.

- bool isType (String type)

  Returns if this factory can create a RenderTechnique of the given type.

- virtual void parseParams (IllumTechniqueParams *params)

  parses parameters from the material file.

Protected Types

- typedef void(*) ILLUM_ATTRIBUTE_PARSER (String &params, RenderTechniqueFactory *factory)

  function for parsing RenderTechnique attributes

- typedef std::map< String, ILLUM_ATTRIBUTE_PARSER > AttribParserList

  Keyword-mapped attribute parsers.

Protected Attributes

- AttribParserList attributeParsers

  map of parser functions

- String typeName

  factoryname

5.32.1 Detailed Description

RenderTechniqueFactory to create OgreDepthShadowReceiverRenderTechnique instances.

5.32.2 Member Typedef Documentation

5.32.2.1 typedef void(∗) RenderTechniqueFactory::ILLUM_ATTRIBUTE_PARSER(String &params, RenderTechniqueFactory ∗factory) [protected, inherited]

function for parsing RenderTechnique attributes

Parameters:

params attribute value stored in a String

5.32.3 Member Function Documentation

5.32.3.1 OgreRenderTechnique ∗ OgreDepthShadowReceiverRenderTechniqueFactory::createInstance (IllumTechniqueParams ∗ *params*, Pass ∗ *pass*, OgreRenderable ∗ *parentRenderable*, OgreTechniqueGroup ∗ *parentTechniqueGroup*) [virtual]

Creates a RenderTechnique of the factory type.

Parameters:

params containes constructor parameters as NameValuePairList

pass the Pass to use in RenderTechnique constructor

pass the parentRenderable to pass to RenderTechnique constructor

pass the parentTechniqueGroup to pass to RenderTechnique constructor

Implements RenderTechniqueFactory.

5.32.3.2 bool RenderTechniqueFactory::isType (String *type*) [inline, inherited]

Returns if this factory can create a RenderTechnique of the given type.

Parameters:

type RenderTechnique type

5.32.3.3 void RenderTechniqueFactory::parseParams (IllumTechniqueParams * *params*) [virtual, inherited]

parses parameters from the material file.

The parsed parameters will be passed to the new RenderTechnique's constructor.

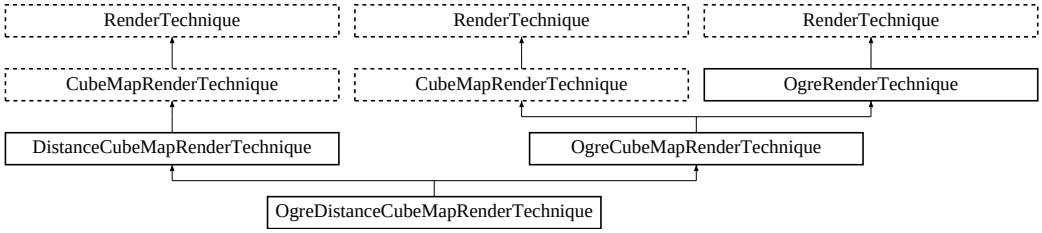
Parameters:

params pointer to the IllumTechniqueParams structure that was read from the material script and containes the parameters to be parsed.

5.33 OgreDistanceCubeMapRenderTechnique Class Reference

DistanceCubeMapRenderTechnique used in an Ogre environment.

Inheritance diagram for OgreDistanceCubeMapRenderTechnique::



Public Member Functions

- OgreDistanceCubeMapRenderTechnique (unsigned long startFrame, unsigned long cubeMapUpdateInterval, unsigned int cubeMapResolution, unsigned char texID, bool useDistCalc, bool useFaceAngleCalc, float distTolerance, float angleTolerance, bool updateAllFace, bool renderSelf, bool renderEnvironment, String selfMaterial, String environmentMaterial, int layer, bool getMinMax, bool attachToTexUnit, String minVariableName, String maxVariableName, Pass *pass, OgreRenderable *parentRenderable, OgreTechniqueGroup *parentTechniqueGroup)

  Constructor.

- ~OgreDistanceCubeMapRenderTechnique ()

  Destructor.

- void update (unsigned long frameNum)

  Updates the resources in the given frame.

- virtual void runUpdated (RenderingRunType runType, RenderingRun *run)

  Called after one of he shared runs updates.

- void runChanged (RenderingRunType runType, RenderingRun *run)

  Called after one of he shared runs changes.

- virtual class OgreRenderTechnique * asOgreRenderTechnique ()

  Conversion to OgreRenderTechnique.

- ElementaryRenderable * getParentRenderable ()

  Retrieves the renderable this technique operates on.

- virtual OgreRenderTechnique * asOgreRenderTechnique ()

  Conversion to OgreRenderTechnique.

Protected Member Functions

- void distanceCubeMapRunChanged (RenderingRun *run)

  Called if the changed run is a CubeMapRenderingRun (containing distances of the current layer).

- void distanceCubeMapRunUpdated (RenderingRun *run)

  Called if the changed run is a CubeMapRenderingRun (containing distances of the current layer).

- virtual RenderingRun * createCubeMapRun ()=0

  Creates a CubeMapRenderingRun.

- virtual void cubeMapRunChanged (RenderingRun *run)=0

  Called if the changed run is a CubeMapRenderingRun.

- virtual void cubeMapRunUpdated (RenderingRun *run)=0

  Called if the updated run is a CubeMapRenderingRun.

- RenderingRun * createCubeMapRun ()

  Creates a CubeMapRenderingRun.

- void cubeMapRunChanged (RenderingRun *run)

  Called if the changed run is a CubeMapRenderingRun.

- void cubeMapRunUpdated (RenderingRun *run)

  Called if the updated run is a CubeMapRenderingRun.

Protected Attributes

- bool useDistCalc

  a flag to skip cube face update if object is far away or too small.

- bool useFaceAngleCalc

  a flag to skip cube face update the face is neglible.

- float distTolerance

  A value used in face skip test.

- float angleTolerance

  A value used in face skip test.

- bool updateAllFace

  defines if all cubemap faces should be updated in a frame or only one face per frame

- unsigned long cubeMapUpdateInterval

  color-cubemap update frequency

- unsigned int cubeMapResolution

  color-cubemap resolution

- unsigned long startFrame

  offset in frame number used during update

- bool renderSelf

  Sets if the object should be rendered to the cube map.

- bool renderEnvironment

  Sets if the environment should be rendered to the cube map.

- int layer

  The layer of the cubemap.

- RenderingRunType cubemapLayer

  The exact run type of this run (according to the actual layer).

- ElementaryRenderable * parentRenderable

  The renderable this technique operates on.

- TechniqueGroup * parentTechniqueGroup

  The TechniqueGroup this RenderedTechnique is attached to.

- SharedRuns * sharedRuns

  The SharedRuns this RenderedTechnique is attached to.

- unsigned char texID

  the id of the texture unit state the resulting cubemap should be bound to

- String selfMaterial

  the material that should be set for the object while rendering the cubemap

- String environmentMaterial

  the material that should be set for the environment while rendering the cubemap

- bool getMinMax

  sets if the minimum and maximum values of the cubemap should be computed

- bool attachToTexUnit

  sets if this cubemap should be attach to a texture unit of the pass

- String minVariableName

  sets the name of the gpu shader program parameter to which the minimum value should be bound to

- String maxVariableName

  sets the name of the gpu shader program parameter to which the maximum value should be bound to

- OgreRenderable * parentOgreRenderable

  a OgreRenderable pointer to the renderable this technique operates on.

- OgreTechniqueGroup * parentOgreTechniqueGroup

  a OgreTechniqueGroup pointer to the TechniqueGroup this technique is attached to.

- Pass * pass

  a pointer to the pass this technique operates on.

5.33.1 Detailed Description

DistanceCubeMapRenderTechnique used in an Ogre environment.

5.33.2 Constructor & Destructor Documentation

5.33.2.1 OgreDistanceCubeMapRenderTechnique::OgreDistanceCubeMapRenderTechnique (unsigned long *startFrame*, unsigned long *cubeMapUpdateInterval*, unsigned int *cubeMapResolution*, unsigned char *texID*, bool *useDistCalc*, bool *useFaceAngleCalc*, float *distTolerance*, float *angleTolerance*, bool *updateAllFace*, bool *renderSelf*, bool *renderEnvironment*, String *selfMaterial*, String *environmentMaterial*, int *layer*, bool *getMinMax*, bool *attachToTexUnit*, String *minVariableName*, String *maxVariableName*, Pass * *pass*, OgreRenderable * *parentRenderable*, OgreTechniqueGroup * *parentTechniqueGroup*)

Constructor.

Parameters:

startFrame adds an offset to the current frame number to help evenly distribute updates between frames

cubeMapUpdateInterval update frequency

cubeMapResolution distance cubemap resolution

texID the id of the texture unit state the resulting cubemap should be bound to

useDistCalc flag to skip cube face update if object is far away

useFaceAngleCalc flag to skip cube face update if face is neglible

distTolerance distance tolerance used in face skip

angleTolerance angle tolerance used in face skip

updateAllFace defines if all cubemap faces should be updated in a frame or only one face per frame

renderSelf sets if the object should be rendered to the cube map

renderEnvironment sets if the environment should be rendered to the cube map

selfMaterial the material that should be set for the object while rendering the cubemap

environmentMaterial the material that should be set for the environment while rendering the cubemap

layer the layer of this cubemap

getMinMax sets if the minimum and maximum values of the cubemap should be computed

attachToTexUnit sets if this cubemap should be attach to a texture unit of the pass

minVariableName sets the name of the gpu shader program parameter to which the minimum value should be bound to

maxVariableName sets the name of the gpu shader program parameter to which the maximum value should be bound to

pass the pass to operate on

parentRenderable the object to operate on

parentTechniqueGroup the TechniqueGroup this RenderedTechnique is attached to

5.33.3 Member Function Documentation

5.33.3.1 void OgreDistanceCubeMapRenderTechnique::update (unsigned long *frameNum*) [virtual]

Updates the resources in the given frame.

Parameters:

frameNum the actual framenumber

Reimplemented from DistanceCubeMapRenderTechnique.

5.33.3.2 void OgreDistanceCubeMapRenderTechnique::distanceCubeMapRunChanged (**RenderingRun** * ***run***) [protected, virtual]

Called if the changed run is a CubeMapRenderingRun (containing distances of the current layer).

Parameters:

run pointer to the changed CubeMapRenderingRun

Implements DistanceCubeMapRenderTechnique.

5.33.3.3 void OgreDistanceCubeMapRenderTechnique::distanceCubeMapRunUpdated (**RenderingRun** * ***run***) [protected, virtual]

Called if the changed run is a CubeMapRenderingRun (containing distances of the current layer).

Parameters:

run pointer to the changed CubeMapRenderingRun

Implements DistanceCubeMapRenderTechnique.

5.33.3.4 void DistanceCubeMapRenderTechnique::runUpdated (RenderingRunType *runType*, **RenderingRun** * ***run***) [virtual, inherited]

Called after one of he shared runs updates.

Parameters:

runType enum describing the type of the updated run

run pointer to the updated RenderingRun

Reimplemented from CubeMapRenderTechnique.

5.33.3.5 void DistanceCubeMapRenderTechnique::runChanged (RenderingRunType *runType*, RenderingRun * *run*) [virtual, inherited]

Called after one of he shared runs changes.

Parameters:

runType enum describing the type of the changed run

run pointer to the changed RenderingRun

Reimplemented from CubeMapRenderTechnique.

5.33.3.6 virtual RenderingRun* CubeMapRenderTechnique::createCubeMapRun () [protected, pure virtual, inherited]

Creates a CubeMapRenderingRun.

Returns:

the new CubeMapRenderingRun instance.

Implemented in OgreCubeMapRenderTechnique.

5.33.3.7 virtual void CubeMapRenderTechnique::cubeMapRunChanged (RenderingRun * *run*) [protected, pure virtual, inherited]

Called if the changed run is a CubeMapRenderingRun.

Parameters:

run pointer to the new CubeMapRenderingRun

Implemented in OgreCubeMapRenderTechnique.

5.33.3.8 virtual void CubeMapRenderTechnique::cubeMapRunUpdated (RenderingRun * *run*) [protected, pure virtual, inherited]

Called if the updated run is a CubeMapRenderingRun.

Parameters:

run pointer to the updated CubeMapRenderingRun

Implemented in OgreCubeMapRenderTechnique.

5.33.3.9 virtual class OgreRenderTechnique* RenderTechnique::asOgreRenderTechnique () [inline, virtual, inherited]

Conversion to OgreRenderTechnique.

This function is needed because of virtual inheritance.

Reimplemented in OgreRenderTechnique.

5.33.3.10 RenderingRun * OgreCubeMapRenderTechnique::createCubeMapRun ()
[protected, virtual, inherited]

Creates a CubeMapRenderingRun.

Returns:

the new CubeMapRenderingRun instance.

Implements CubeMapRenderTechnique.

5.33.3.11 void OgreCubeMapRenderTechnique::cubeMapRunChanged (RenderingRun * *run*)
[protected, virtual, inherited]

Called if the changed run is a CubeMapRenderingRun.

Parameters:

run pointer to the new CubeMapRenderingRun

Implements CubeMapRenderTechnique.

5.33.3.12 void OgreCubeMapRenderTechnique::cubeMapRunUpdated (RenderingRun * *run*)
[protected, virtual, inherited]

Called if the updated run is a CubeMapRenderingRun.

Parameters:

run pointer to the updated CubeMapRenderingRun

Implements CubeMapRenderTechnique.

5.33.3.13 virtual OgreRenderTechnique* OgreRenderTechnique::asOgreRenderTechnique ()
[inline, virtual, inherited]

Conversion to OgreRenderTechnique.

This function is needed because of virtual inheritance.

Reimplemented from RenderTechnique.

5.33.4 Member Data Documentation

5.33.4.1 bool CubeMapRenderTechnique::useDistCalc [protected, inherited]

a flag to skip cube face update if object is far away or too small.

See also:

distTolerance

5.33.4.2 bool CubeMapRenderTechnique::useFaceAngleCalc [protected, inherited]

a flag to skip cube face update the face is neglible.

See also:

angleTolerance

5.33.4.3 float CubeMapRenderTechnique::distTolerance [protected, inherited]

A value used in face skip test.

The higher this value gets the more precise, but slower the method will be.

5.33.4.4 float CubeMapRenderTechnique::angleTolerance [protected, inherited]

A value used in face skip test.

The higher this value gets the more precise, but slower the method will be.

5.33.4.5 int CubeMapRenderTechnique::layer [protected, inherited]

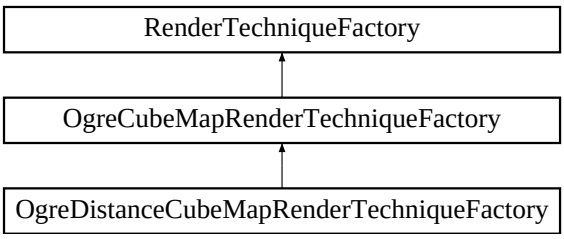
The layer of the cubemap.

This extension was created to achieve effect such as multiple reflections and refractions that require more complete sampling of the scene. With this extension we can render the environment into several cubemap layers eg. with depth peeling.

5.34 OgreDistanceCubeMapRenderTechniqueFactory Class Reference

RenderTechniqueFactory to create OgreDistanceCubeMapRenderTechnique instances.

Inheritance diagram for OgreDistanceCubeMapRenderTechniqueFactory::



Public Member Functions

- OgreDistanceCubeMapRenderTechniqueFactory ()

  Technique factory.

- OgreRenderTechnique * createInstance (IllumTechniqueParams *params, Pass *pass, OgreRenderable *parentRenderable, OgreTechniqueGroup *parentTechniqueGroup)

  Creates a RenderTechnique of the factory type.

- bool isType (String type)

  Returns if this factory can create a RenderTechnique of the given type.

- virtual void parseParams (IllumTechniqueParams *params)

  parses parameters from the material file.

Protected Types

- typedef void(*) ILLUM_ATTRIBUTE_PARSER (String &params, RenderTechniqueFactory *factory)

  function for parsing RenderTechnique attributes

- typedef std::map< String, ILLUM_ATTRIBUTE_PARSER > AttribParserList

  Keyword-mapped attribute parsers.

Protected Attributes

- AttribParserList attributeParsers

  map of parser functions

- String typeName
  factoryname

5.34.1 Detailed Description

RenderTechniqueFactory to create OgreDistanceCubeMapRenderTechnique instances.

5.34.2 Member Typedef Documentation

5.34.2.1 typedef void(∗) RenderTechniqueFactory::ILLUM_ATTRIBUTE_PARSER(String &params, RenderTechniqueFactory ∗factory) [protected, inherited]

function for parsing RenderTechnique attributes

Parameters:

params attribute value stored in a String

5.34.3 Member Function Documentation

5.34.3.1 OgreRenderTechnique ∗ OgreDistanceCubeMapRenderTechniqueFactory::createInstance (IllumTechniqueParams ∗ *params*, Pass ∗ *pass*, OgreRenderable ∗ *parentRenderable*, OgreTechniqueGroup ∗ *parentTechniqueGroup*) [virtual]

Creates a RenderTechnique of the factory type.

Parameters:

params containes constructor parameters as NameValuePairList
pass the Pass to use in RenderTechnique constructor
pass the parentRenderable to pass to RenderTechnique constructor
pass the parentTechniqueGroup to pass to RenderTechnique constructor

Reimplemented from OgreCubeMapRenderTechniqueFactory.

5.34.3.2 bool RenderTechniqueFactory::isType (String *type*) [inline, inherited]

Returns if this factory can create a RenderTechnique of the given type.

Parameters:

type RenderTechnique type

5.34.3.3 void RenderTechniqueFactory::parseParams (IllumTechniqueParams * *params*) [virtual, inherited]

parses parameters from the material file.

The parsed parameters will be passed to the new RenderTechnique's constructor.

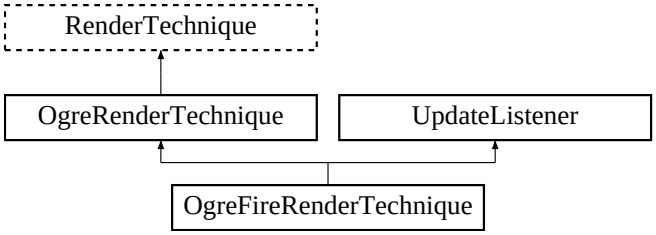
Parameters:

params pointer to the IllumTechniqueParams structure that was read from the material script and containes the parameters to be parsed.

5.35 OgreFireRenderTechnique Class Reference

A special SBBRenderTechnique used in an OGRE environment.

Inheritance diagram for OgreFireRenderTechnique::



Public Member Functions

- OgreFireRenderTechnique (unsigned char depthTexID, Pass *pass, OgreRenderable *parentRenderable, OgreTechniqueGroup *parentTechniqueGroup)

  Constructor.

- virtual void update (unsigned long frameNum)

  Updates the resources in the given frame.

- void preAllUpdates ()

  Called before RenderTechnique updates.

- virtual OgreRenderTechnique * asOgreRenderTechnique ()

  Conversion to OgreRenderTechnique.

- virtual void runChanged (RenderingRunType runType, RenderingRun *run)

  Called after one of he shared runs changes.

- virtual void runUpdated (RenderingRunType runType, RenderingRun *run)

  Called after one of he shared runs updates.

- ElementaryRenderable * getParentRenderable ()

  Retrieves the renderable this technique operates on.

- virtual void postAllUpdates ()

  Called after RenderTechnique updates.

Protected Attributes

- unsigned char depthTexID
- OgreRenderable * parentOgreRenderable

a OgreRenderable pointer to the renderable this technique operates on.

- OgreTechniqueGroup * parentOgreTechniqueGroup

  a OgreTechniqueGroup pointer to the TechniqueGroup this technique is attached to.

- Pass * pass

  a pointer to the pass this technique operates on.

- ElementaryRenderable * parentRenderable

  The renderable this technique operates on.

- TechniqueGroup * parentTechniqueGroup

  The TechniqueGroup this RenderedTechnique is attached to.

- SharedRuns * sharedRuns

  The SharedRuns this RenderedTechnique is attached to.

5.35.1 Detailed Description

A special SBBRenderTechnique used in an OGRE environment.

Instead of rendering to the frame buffer this Technique renders into two render targets at the same time. The firs render target will be used as it were an ordinary backbuffer. The second rendertarget will be used to simulate heat shimmering (offset values will be written to it). The shimmering effect will be achieved with post processing: combining the backbuffer with the first render target using the second render target as uv offset. Just like the spherical billboard techniqeue, this thechnique requires a depth image taken from main camera's viewpoint.

5.35.2 Constructor & Destructor Documentation

5.35.2.1 OgreFireRenderTechnique::OgreFireRenderTechnique (unsigned char *depthTexID*, Pass * *pass*, OgreRenderable * *parentRenderable*, OgreTechniqueGroup * *parentTechniqueGroup*)

Constructor.

Parameters:

depthTexID the id of the texture unit state the resulting scene depth map should be bound to

pass the pass to operate on

parentRenderable the object to operate on

parentTechniqueGroup the TechniqueGroup this RenderedTechnique is attached to

5.35.3 Member Function Documentation

5.35.3.1 void OgreFireRenderTechnique::update (unsigned long *frameNum*) [virtual]

Updates the resources in the given frame.

A RenderTechnique is usually need some resources from several runs, so these runs will be updated.

Parameters:

frameNum the actual framenumber

Reimplemented from RenderTechnique.

5.35.3.2 virtual OgreRenderTechnique* OgreRenderTechnique::asOgreRenderTechnique () [inline, virtual, inherited]

Conversion to OgreRenderTechnique.

This function is needed because of virtual inheritance.

Reimplemented from RenderTechnique.

5.35.3.3 virtual void RenderTechnique::runChanged (RenderingRunType *runType*, RenderingRun * *run*) [inline, virtual, inherited]

Called after one of he shared runs changes.

Parameters:

runType enum describing the type of the changed run

run pointer to the changed RenderingRun

Reimplemented in CausticCasterRenderTechnique, ColorCubeMapRenderTechnique, ConvolvedCubeMapRenderTechnique, CubeMapRenderTechnique, DistanceCubeMapRenderTechnique, HierarchicalParticleSystemTechnique, and IllumVolumeRenderTechnique.

5.35.3.4 virtual void RenderTechnique::runUpdated (RenderingRunType *runType*, RenderingRun * *run*) [inline, virtual, inherited]

Called after one of he shared runs updates.

Parameters:

runType enum describing the type of the updated run

run pointer to the updated RenderingRun

Reimplemented in CausticCasterRenderTechnique, ColorCubeMapRenderTechnique, CubeMapRenderTechnique, DistanceCubeMapRenderTechnique, HierarchicalParticleSystemTechnique, and IllumVolumeRenderTechnique.

5.35.4 Member Data Documentation

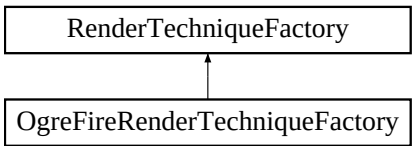
5.35.4.1 unsigned char OgreFireRenderTechnique::depthTexID [protected]

&brief the id of the texture unit state the resulting scene depth map should be bound to

5.36 OgreFireRenderTechniqueFactory Class Reference

RenderTechniqueFactory to create OgreFireRenderTechnique instances.

Inheritance diagram for OgreFireRenderTechniqueFactory::



Public Member Functions

- OgreRenderTechnique * createInstance (IllumTechniqueParams *params, Pass *pass, OgreRenderable *parentRenderable, OgreTechniqueGroup *parentTechniqueGroup)

  Creates a RenderTechnique of the factory type.

- bool isType (String type)

  Returns if this factory can create a RenderTechnique of the given type.

- virtual void parseParams (IllumTechniqueParams *params)

  parses parameters from the material file.

Protected Types

- typedef void(*) ILLUM_ATTRIBUTE_PARSER (String &params, RenderTechniqueFactory *factory)

  function for parsing RenderTechnique attributes

- typedef std::map< String, ILLUM_ATTRIBUTE_PARSER > AttribParserList

  Keyword-mapped attribute parsers.

Protected Attributes

- AttribParserList attributeParsers

  map of parser functions

- String typeName

  factoryname

5.36.1 Detailed Description

RenderTechniqueFactory to create OgreFireRenderTechnique instances.

5.36.2 Member Typedef Documentation

5.36.2.1 typedef void(∗) RenderTechniqueFactory::ILLUM_ATTRIBUTE_PARSER(String &params, RenderTechniqueFactory ∗factory) [protected, inherited]

function for parsing RenderTechnique attributes

Parameters:

params attribute value stored in a String

5.36.3 Member Function Documentation

5.36.3.1 OgreRenderTechnique ∗ OgreFireRenderTechniqueFactory::createInstance (IllumTechniqueParams ∗ *params*, Pass ∗ *pass*, OgreRenderable ∗ *parentRenderable*, OgreTechniqueGroup ∗ *parentTechniqueGroup*) [virtual]

Creates a RenderTechnique of the factory type.

Parameters:

params containes constructor parameters as NameValuePairList

pass the Pass to use in RenderTechnique constructor

pass the parentRenderable to pass to RenderTechnique constructor

pass the parentTechniqueGroup to pass to RenderTechnique constructor

Implements RenderTechniqueFactory.

5.36.3.2 bool RenderTechniqueFactory::isType (String *type*) [inline, inherited]

Returns if this factory can create a RenderTechnique of the given type.

Parameters:

type RenderTechnique type

5.36.3.3 void RenderTechniqueFactory::parseParams (IllumTechniqueParams * *params*) [virtual, inherited]

parses parameters from the material file.

The parsed parameters will be passed to the new RenderTechnique's constructor.

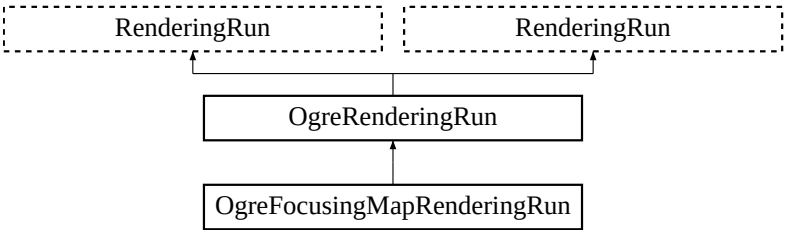
Parameters:

params pointer to the IllumTechniqueParams structure that was read from the material script and containes the parameters to be parsed.

5.37 OgreFocusingMapRenderingRun Class Reference

FocusingMapRenderingRun.

Inheritance diagram for OgreFocusingMapRenderingRun::



Public Member Functions

- OgreFocusingMapRenderingRun (String name, Matrix4 lightMatrix, unsigned int focusingMapSize)

  Constructor.

- String getFocusingTextureName ()

  returns the name of the focusing texture

- void getMinMax (Vector3 &min, Vector3 &max)

  gets the boundig box min-max points

- void setLightMatrix (Matrix4 &m)

  Sets the light matrix that should be focused.

- void setCameraMatrices (const Matrix4 &view, const Matrix4 &projection)

  Sets the player camera matrices.

- OgreRenderingRun * asOgreRenderingRun ()

  Conversion to OgreRenderRun.

- OgreRenderingRun * asOgreRenderingRun ()

  Conversion to OgreRenderRun.

- bool update (unsigned long frameNum)

  Calls updateFrame() if the run needs update according to its starting frame and update interval and has not been allready updated in this frame.

- virtual bool canJoin (RenderingRun *run)

  Returns true if two runs can be joined.

Protected Member Functions

- void updateFrame (unsigned long frameNum)

  This function does the actual update in a frame.

- void createFocusingMap ()

  creates the focusing map texture

- bool needUpdate (unsigned long frameNum)

  Returns if this run needs update.

- Vector3 getCubeMapFaceDirection (unsigned char faceId)

  Returns a direction for a cubemap face id.

- Vector3 getCubeMapFaceDirection (unsigned char faceId)

  Returns a direction for a cubemap face id.

- Texture * createCubeRenderTexture (String name, const Vector3 position, unsigned int resolution=512, PixelFormat format=PF_FLOAT16_RGBA, int numMips=0, ColourValue clearColor=ColourValue::Black)

  Creates a cubemap texture.

- Texture * createCubeRenderTexture (String name, const Vector3 position, unsigned int resolution=512, PixelFormat format=PF_FLOAT16_RGBA, int numMips=0, ColourValue clearColor=ColourValue::Black)

  Creates a cubemap texture.

- void setMaterialForRenderables (String &materialName, RenderQueue *rq)

  Sets the given material for each Renderable in a RenderQueue.

- void setMaterialForRenderables (String &materialName, RenderQueue *rq, bool solidsonly=false)

  Sets the given material for each Renderable in a RenderQueue.

- void setMaterialForVisibles (String &materialName, Camera *cam, bool shadowcastersonly=false)

  Sets the given material for each Renderable visible from a given camera.

- void setMaterialForVisibles (String &materialName, Camera *cam, bool shadowcastersonly=false, bool solidsonly=false)

  Sets the given material for each Renderable visible from a given camera.

- void restoreMaterials ()

  Restores previously stored materials.

- void restoreMaterials ()

  Restores previously stored materials.

- void renderFullscreenQuad (String materialName, RenderTarget *target)

  Renderes a full screen quad on a given RendderTarget with a given material.

- void renderFullscreenQuad (String materialName, RenderTarget *target)

Renderes a full screen quad on a given RendderTarget with a given material.

- void renderPixelSprites (String &materialName, RenderTarget *rt, int width, int height)
  Renderes sprites to pixels of the screen on a given RendderTarget with a given material.
- void renderPixelSprites (String &materialName, RenderTarget *rt, int width, int height)
  Renderes sprites to pixels of the screen on a given RendderTarget with a given material.
- void renderFullscreenGrid (String &materialName, RenderTarget *rt, int width, int height)
  Renders a grid onto the screen.

Protected Attributes

- Matrix4 lightMatrix
  light matrix to be focused
- Camera * camera
  fosusing map camera
- String name
  the name of the focusing texture that was created by this run
- Texture * focusingTexture
  a pointer to the focusing texture that was created by this run
- unsigned int focusingMapSize
  the size of the focusing map
- std::map< Renderable *, String > visibleObjects
  map of Renderables which will be rendered with a given material
- std::map< Renderable *, String > visibleObjects
  map of Renderables which will be rendered with a given material
- SpriteSet * pixelSprites
  SpriteSet used in pixel sprite rendering.
- BillboardSet * pixelSprites
  SpriteSet used in pixel sprite rendering.
- String spriteSetName
  unique name of the SpriteSet used in pixel sprite rendering
- Entity * fullscreenGrid
  Entity used in fullscreen grid rendering.
- unsigned long lastupdated
  The number of the last frame this run was updated.

- unsigned long startFrame

  The number of the frame this run should be updated first.

- unsigned long updateInterval

  Refresh frequency in frames.

Static Protected Attributes

- static MovablePlane * fullScreenQuad

  fulls screen quad plane used in full screen quad rendering

- static MovablePlane * fullScreenQuad

  fulls screen quad plane used in full screen quad rendering

- static Entity * fullScreenQuadEntity

  fulls screen quad Entity used in full screen quad rendering

- static Entity * fullScreenQuadEntity

  fulls screen quad Entity used in full screen quad rendering

- static SceneNode * fullScreenQuadNode

  fulls screen quad SceneNode used in full screen quad rendering

- static SceneNode * fullScreenQuadNode

  fulls screen quad SceneNode used in full screen quad rendering

5.37.1 Detailed Description

FocusingMapRenderingRun.

A run to help focusing light projections for shadow mapping.

5.37.2 Constructor & Destructor Documentation

5.37.2.1 OgreFocusingMapRenderingRun::OgreFocusingMapRenderingRun (String *name*, Matrix4 *lightMatrix*, unsigned int *focusingMapSize*)

Constructor.

Parameters:

name the name of the focusing map texture to be created

lightMatrix the light matrix to be focused

focusingMapSize size of the focusing map

5.37.3 Member Function Documentation

5.37.3.1 void OgreFocusingMapRenderingRun::getMinMax (Vector3 & *min*, Vector3 & *max*)

gets the boundig box min-max points

These values are given in light space and can be used to adjust the light matrices to focuse on important objects only.

5.37.3.2 void OgreFocusingMapRenderingRun::updateFrame (unsigned long *frameNum*) [protected, virtual]

This function does the actual update in a frame.

Parameters:

frameNum current frame number

Reimplemented from RenderingRun.

5.37.3.3 bool OgreFocusingMapRenderingRun::needUpdate (unsigned long *frameNum*) [inline, protected, virtual]

Returns if this run needs update.

This tipically depends on the upate interval and the starting frame number.

Parameters:

frameNum current frame number

Reimplemented from RenderingRun.

5.37.3.4 OgreRenderingRun* OgreRenderingRun::asOgreRenderingRun () [inline, virtual, inherited]

Conversion to OgreRenderRun.

This function is needed because of virtual inheritance.

Reimplemented from RenderingRun.

5.37.3.5 OgreRenderingRun* OgreRenderingRun::asOgreRenderingRun () [inline, virtual, inherited]

Conversion to OgreRenderRun.

This function is needed because of virtual inheritance.

Reimplemented from RenderingRun.

5.37.3.6 Vector3 OgreRenderingRun::getCubeMapFaceDirection (unsigned char *faceId*) [protected, inherited]

Returns a direction for a cubemap face id.

This is a helper function to retrieve the normal direction of a given cubemap face.

Parameters:

faceId the number of the face

5.37.3.7 Vector3 OgreRenderingRun::getCubeMapFaceDirection (unsigned char *faceId*) [protected, inherited]

Returns a direction for a cubemap face id.

This is a helper function to retrieve the normal direction of a given cubemap face.

Parameters:

faceId the number of the face

5.37.3.8 Texture * OgreRenderingRun::createCubeRenderTexture (String *name*, const Vector3 *position*, unsigned int *resolution* = 512, PixelFormat *format* = PF_FLOAT16_RGBA, int *numMips* = 0, ColourValue *clearColor* = ColourValue::Black) [protected, inherited]

Creates a cubemap texture.

This is a helper function to easily create a cubemap texture and automaticly attach viewports to each face so it can be used as a rendertarget.

Parameters:

name the name of the texture to be created

position the initial position of the cubemap

resolution the resolution of one cubemapface

format the pixel format of the cubemap

numMips the number of mipmap levels

clearColor initial color

5.37.3.9 Texture* OgreRenderingRun::createCubeRenderTexture (String *name*, const Vector3 *position*, unsigned int *resolution* = 512, PixelFormat *format* = PF_FLOAT16_RGBA, int *numMips* = 0, ColourValue *clearColor* = ColourValue::Black) [protected, inherited]

Creates a cubemap texture.

This is a helper function to easily create a cubemap texture and automaticly attach viewports to each face so it can be used as a rendertarget.

Parameters:

name the name of the texture to be created

position the initial position of the cubemap

resolution the resolution of one cubemapface

format the pixel format of the cubemap

numMips the number of mipmap levels

clearColor initial color

5.37.3.10 void OgreRenderingRun::setMaterialForRenderables (String & *materialName*, RenderQueue * *rq*) [protected, inherited]

Sets the given material for each Renderable in a RenderQueue.

This is a helper function to set a material to each element of a previously filled Renderque. The orginal material of the Renderables are stored so they can be restored later. The function also tells the current SceneManager not to search for visible objects, as we are going to use the given RenderQueue during the next rendering.

Parameters:

materialName the name of the material to set for the Renderables

rq pointer to the filled Renderqueue instance to set material for

5.37.3.11 void OgreRenderingRun::setMaterialForRenderables (String & *materialName*, RenderQueue * *rq*, bool *solidsonly* = false) [protected, inherited]

Sets the given material for each Renderable in a RenderQueue.

This is a helper function to set a material to each element of a previously filled Renderque. The orginal material of the Renderables are stored so they can be restored later. The function also tells the current SceneManager not to search for visible objects, as we are going to use the given RenderQueue during the next rendering.

Parameters:

materialName the name of the material to set for the Renderables

rq pointer to the filled Renderqueue instance to set material for

5.37.3.12 void OgreRenderingRun::setMaterialForVisibles (String & *materialName*, Camera * *cam*, bool *shadowcastersonly* = false) [protected, inherited]

Sets the given material for each Renderable visible from a given camera.

This helper function is similar to setMaterialForRenderables but it is also responsible for filling the RenderQueue. First the RenderQueue of the current SceneManager fill be filled with the visible objects seen from the given camera. Then the required material will be set for each element of the RenderQueue. The orginal material of the Renderables are stored so they can be restored later. The function also tells the current SceneManager not to search for visible objects, as we are going to use the filled RenderQueue during the next rendering.

Parameters:

materialName the name of the material to set for the Renderables

cam pointer to the camera from which visible objects should be searched

shadowcastersonly flag to search for only shadow casters

5.37.3.13 void OgreRenderingRun::setMaterialForVisibles (String & *materialName*, Camera * *cam*, bool *shadowcastersonly* = false, bool *solidsonly* = false) [protected, inherited]

Sets the given material for each Renderable visible from a given camera.

This helper function is similar to setMaterialForRenderables but it is also responsible for filling the RenderQueue. First the RenderQueue of the current SceneManager fill be filled with the visible objects seen from the given camera. Then the required material will be set for each element of the RenderQueue. The orginal material of the Renderables are stored so they can be restored later. The function also tells the current SceneManager not to search for visible objects, as we are going to use the filled RenderQueue during the next rendering.

Parameters:

materialName the name of the material to set for the Renderables

cam pointer to the camera from which visible objects should be searched

shadowcastersonly flag to search for only shadow casters

5.37.3.14 void OgreRenderingRun::restoreMaterials () [protected, inherited]

Restores previously stored materials.

This helper function is typically used after a setMaterialForRenderables or setMaterialForVisibles call and a rendering process to restore the original material settings. The function also tells the current SceneManager to search for visible objects, as this is the default behaviour of SceneManager.

5.37.3.15 void OgreRenderingRun::restoreMaterials () [protected, inherited]

Restores previously stored materials.

This helper function is typically used after a setMaterialForRenderables or setMaterialForVisibles call and a rendering process to restore the original material settings. The function also tells the current SceneManager to search for visible objects, as this is the default behaviour of SceneManager.

5.37.3.16 void OgreRenderingRun::renderFullscreenQuad (String *materialName*, RenderTarget * *target*) [protected, inherited]

Renderes a full screen quad on a given RendderTarget with a given material.

Parameters:

materialName the name of the material bind to the quad

target the RenderTarget the quad should be rendered on

5.37.3.17 void OgreRenderingRun::renderFullscreenQuad (String *materialName*, RenderTarget * *target*) [protected, inherited]

Renderes a full screen quad on a given RendderTarget with a given material.

Parameters:

materialName the name of the material bind to the quad

target the RenderTarget the quad should be rendered on

5.37.3.18 void OgreRenderingRun::renderPixelSprites (String & *materialName*, RenderTarget * *rt*, int *width*, int *height*) [protected, inherited]

Renderes sprites to pixels of the screen on a given RendderTarget with a given material.

Pixel sprites are pixel sized quads, placed on each pixel of the RenderTarget. The number of sprites not necessary corresponds to the resolution of the rendertarget. The pixel quads will evenly fill the rendertarget's area with sizes corresponding to the given resolution. We can render fewer or more pixel quads than the number of pixels the rendertarget has (eg.: in case of caustic cubemap generation).

Parameters:

materialName the name of the material bind to the pixel sprites

rt the RenderTarget the quads should be rendered on

width the desired resolution width of the sprites

height the desired resolution height of the sprites

5.37.3.19 void OgreRenderingRun::renderPixelSprites (String & *materialName*, RenderTarget * *rt*, int *width*, int *height*) [protected, inherited]

Renderes sprites to pixels of the screen on a given RendderTarget with a given material.

Pixel sprites are pixel sized quads, placed on each pixel of the RenderTarget. The number of sprites not necessary corresponds to the resolution of the rendertarget. The pixel quads will evenly fill the rendertarget's area with sizes corresponding to the given resolution. We can render fewer or more pixel quads than the number of pixels the rendertarget has (eg.: in case of caustic cubemap generation).

Parameters:

materialName the name of the material bind to the pixel sprites

rt the RenderTarget the quads should be rendered on

width the desired resolution width of the sprites

height the desired resolution height of the sprites

5.37.3.20 void OgreRenderingRun::renderFullscreenGrid (String & *materialName*, RenderTarget * *rt*, int *width*, int *height*) [protected, inherited]

Renders a grid onto the screen.

Parameters:

rt the RenderTarget the grid should be rendered on

width the desired horizontal resolution of the grid

height the desired vertical resolution of the grid

5.37.3.21 virtual bool RenderingRun::canJoin (RenderingRun * *run*) [inline, virtual, inherited]

Returns true if two runs can be joined.

In some cases special requirements should stand to join two runs (even if they have the same type). Eg.: two caustic cube map generation technique should only be joined if they use the same material when rendering the caustic cubemap.

Reimplemented in OgreCausticCubeMapRenderingRun, OgreChildPSystemRenderingRun, and OgrePhotonMapRenderingRun.

5.37.4 Member Data Documentation

5.37.4.1 std::map<Renderable*, String> OgreRenderingRun::visibleObjects [protected, inherited]

map of Renderables which will be rendered with a given material

The String stores the original material name that will be restored after rendering.

See also:

setMaterialForVisibles
setMaterialForRenderables
restoreMaterials

5.37.4.2 std::map<Renderable*, String> OgreRenderingRun::visibleObjects [protected, inherited]

map of Renderables which will be rendered with a given material

The String stores the original material name that will be restored after rendering.

See also:

setMaterialForVisibles
setMaterialForRenderables
restoreMaterials

5.37.4.3 MovablePlane * OgreRenderingRun::fullScreenQuad [static, protected, inherited]

fulls screen quad plane used in full screen quad rendering

See also:

renderFullscreenQuad

5.37.4.4 MovablePlane* OgreRenderingRun::fullScreenQuad [static, protected, inherited]

fulls screen quad plane used in full screen quad rendering

See also:

renderFullscreenQuad

5.37.4.5 Entity * OgreRenderingRun::fullScreenQuadEntity [static, protected, inherited]

fulls screen quad Entity used in full screen quad rendering

See also:

renderFullscreenQuad

5.37.4.6 Entity* OgreRenderingRun::fullScreenQuadEntity [static, protected, inherited]

fulls screen quad Entity used in full screen quad rendering

See also:

renderFullscreenQuad

5.37.4.7 SceneNode * OgreRenderingRun::fullScreenQuadNode [static, protected, inherited]

fulls screen quad SceneNode used in full screen quad rendering

See also:

renderFullscreenQuad

5.37.4.8 SceneNode* OgreRenderingRun::fullScreenQuadNode [static, protected, inherited]

fulls screen quad SceneNode used in full screen quad rendering

See also:

renderFullscreenQuad

5.37.4.9 SpriteSet* OgreRenderingRun::pixelSprites [protected, inherited]

SpriteSet used in pixel sprite rendering.

See also:

renderPixelSprites

5.37.4.10 BillboardSet* OgreRenderingRun::pixelSprites [protected, inherited]

SpriteSet used in pixel sprite rendering.

See also:

renderPixelSprites

5.37.4.11 String OgreRenderingRun::spriteSetName [protected, inherited]

unique name of the SpriteSet used in pixel sprite rendering

See also:

renderPixelSprites

5.37.4.12 Entity* OgreRenderingRun::fullscreenGrid [protected, inherited]

Entity used in fullscreen grid rendering.

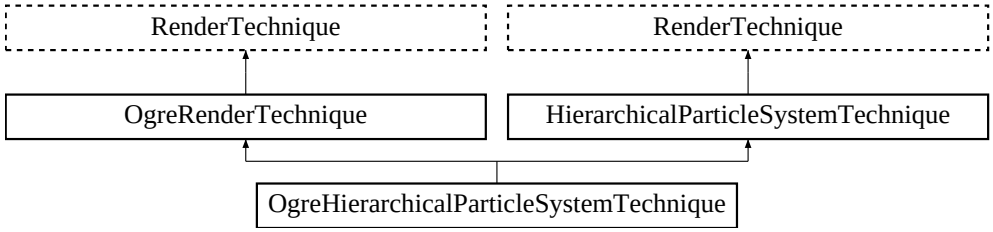
See also:

renderPixelGrid

5.38 OgreHierarchicalParticleSystemTechnique Class Reference

HierarchicalParticleSystemTechnique used in an OGRE environment.

Inheritance diagram for OgreHierarchicalParticleSystemTechnique::



Public Member Functions

- OgreHierarchicalParticleSystemTechnique (unsigned long startFrame, unsigned long impostorUpdateInterval, unsigned int impostorResolution, unsigned char impostorTexID, bool useDistCalc, bool perspectiveRendering, String childPSysScriptName, bool useOwnMaterial, String impostorMaterialName, bool useVParam, String VParamRadius, bool useFParam, String FParamRadius, Pass *pass, OgreRenderable *parentRenderable, OgreTechniqueGroup *parentTechniqueGroup)

  Constructor.

- ~OgreHierarchicalParticleSystemTechnique ()

  Destructor.

- virtual OgreRenderTechnique * asOgreRenderTechnique ()

  Conversion to OgreRenderTechnique.

- virtual void update (unsigned long frameNum)

  Updates the resources in the given frame.

- virtual void runChanged (RenderingRunType runType, RenderingRun *run)

  Called after one of he shared runs changes.

- virtual void runUpdated (RenderingRunType runType, RenderingRun *run)

  Called after one of he shared runs updates.

- ElementaryRenderable * getParentRenderable ()

  Retrieves the renderable this technique operates on.

- void update (unsigned long frameNum)

  Updates the resources in the given frame.

- void runChanged (RenderingRunType runType, RenderingRun *run)

  Called after one of he shared runs changes.

- void runUpdated (RenderingRunType runType, RenderingRun *run)

  Called after one of he shared runs updates.

Protected Member Functions

- RenderingRun * createChildPSysRenderingRun ()

  Creates the ChildParticleSystemRenderingRun needed by this technique.

- virtual void impostorChanged (RenderingRun *run)

  Called if the impostor rendering run changed.

- virtual void impostorUpdated (RenderingRun *run)

  Called if the impostor rendering run is updated.

Protected Attributes

- String impostorMaterialName

  use this specific material for the small particle system

- String childPSysScriptName

  name of the small particle system script

- String childPSysName

  name of the created child particle system

- unsigned char impostorTexID

  the id of the texture unit state the impostor image should be bound to

- bool useOwnMaterial

  use the material for the smaller system that was defined in the particle script

- String VParamRadius

  name of the gpu vertex program parameter the particle radius should be bound to

- String FParamRadius

  name of the gpu fragment program parameter the particle radius should be bound to

- bool useVParam

  bound particle radius to a gpu vertex program parameter

- bool useFParam

  bound particle radius to a gpu fragment program parameter

- OgreRenderable * parentOgreRenderable

  a OgreRenderable pointer to the renderable this technique operates on.

- OgreTechniqueGroup * parentOgreTechniqueGroup

  a OgreTechniqueGroup pointer to the TechniqueGroup this technique is attached to.

- Pass * pass

  a pointer to the pass this technique operates on.

- ElementaryRenderable * parentRenderable

  The renderable this technique operates on.

- TechniqueGroup * parentTechniqueGroup

  The TechniqueGroup this RenderedTechnique is attached to.

- SharedRuns * sharedRuns

  The SharedRuns this RenderedTechnique is attached to.

- unsigned long impostorUpdateInterval

  update frequency of the impostor texture (image of the smaller system)

- unsigned int impostorResolution

  resolution of the impostor texture

- unsigned long startFrame

  offset in frame number used during update

- bool useDistCalc

  flag to skip impostor update if object is far away (//not used)

- bool perspectiveRendering

  sets if the impostor should be rendered with a perspective projection or orthogonal

5.38.1 Detailed Description

HierarchicalParticleSystemTechnique used in an OGRE environment.

5.38.2 Constructor & Destructor Documentation

5.38.2.1 OgreHierarchicalParticleSystemTechnique::OgreHierarchicalParticleSystemTechnique (unsigned long *startFrame*, unsigned long *impostorUpdateInterval*, unsigned int *impostorResolution*, unsigned char *impostorTexID*, bool *useDistCalc*, bool *perspectiveRendering*, String *childPSysScriptName*, bool *useOwnMaterial*, String *impostorMaterialName*, bool *useVParam*, String *VParamRadius*, bool *useFParam*, String *FParamRadius*, Pass * *pass*, OgreRenderable * *parentRenderable*, OgreTechniqueGroup * *parentTechniqueGroup*)

Constructor.

Parameters:

startFrame adds an offset to the current frame number to help evenly distribute updates between frames

impostorUpdateInterval update frequency of the impostor texture (image of the smaller system)

impostorResolution resolution of the impostor texture

impostorTexID the id of the texture unit state the impostor image should be bound to

useDistCalc flag to skip impostor update if object is far away (//not used)

perspectiveRendering sets if the impostor should be rendered with a perspective projection or orthogonal

childPSysScriptName name of the small particle system script

useOwnMaterial use the material for the smaller system that was defined in the particle script

impostorMaterialName use this specific material for the small particle system

useVParam bound particle radius to a gpu vertex program parameter

VParamRadius name of the gpu vertex program parameter the particle radius should be bound to

useFParam bound particle radius to a gpu fragment program parameter

FParamRadius name of the gpu fragment program parameter the particle radius should be bound to

pass the pass to operate on

parentRenderable the object to operate on

parentTechniqueGroup the TechniqueGroup this RenderedTechnique is attached to

5.38.3 Member Function Documentation

5.38.3.1 RenderingRun * OgreHierarchicalParticleSystemTechnique::createChildPSysRenderingRun () [protected, virtual]

Creates the ChildParticleSystemRenderingRun needed by this technique.

Returns:

pointer to the ChildParticleSystemRenderingRun created instance

Implements HierarchicalParticleSystemTechnique.

5.38.3.2 void OgreHierarchicalParticleSystemTechnique::impostorChanged (RenderingRun * *run*) [protected, virtual]

Called if the impostor rendering run changed.

Parameters:

run pointer to the new ChildParticleSystemRenderingRun instance to use

Implements HierarchicalParticleSystemTechnique.

5.38.3.3 void OgreHierarchicalParticleSystemTechnique::impostorUpdated (RenderingRun * *run*) [protected, virtual]

Called if the impostor rendering run is updated.

Parameters:

run pointer to the updated ChildParticleSystemRenderingRu.

Implements HierarchicalParticleSystemTechnique.

5.38.3.4 virtual OgreRenderTechnique* OgreRenderTechnique::asOgreRenderTechnique () [inline, virtual, inherited]

Conversion to OgreRenderTechnique.

This function is needed because of virtual inheritance.

Reimplemented from RenderTechnique.

5.38.3.5 virtual void RenderTechnique::update (unsigned long *frameNum*) [inline, virtual, inherited]

Updates the resources in the given frame.

A RenderTechnique is usually need some resources from several runs, so these runs will be updated.

Parameters:

frameNum the actual framenumber

Reimplemented in ColorCubeMapRenderTechnique, ConvolvedCubeMapRenderTechnique, DistanceCubeMapRenderTechnique, HierarchicalParticleSystemTechnique, IllumVolumeRenderTechnique, OgreCausticReceiverRenderTechnique, OgreColorCubeMapRenderTechnique, OgreConvolvedCubeMapRenderTechnique, OgreDepthShadowReceiverRenderTechnique, OgreDistanceCubeMapRenderTechnique, OgreFireRenderTechnique, OgrePathMapRenderTechnique, and OgreSBBRenderTechnique.

5.38.3.6 virtual void RenderTechnique::runChanged (RenderingRunType *runType*, RenderingRun * *run*) [inline, virtual, inherited]

Called after one of he shared runs changes.

Parameters:

runType enum describing the type of the changed run

run pointer to the changed RenderingRun

Reimplemented in CausticCasterRenderTechnique, ColorCubeMapRenderTechnique, ConvolvedCubeMapRenderTechnique, CubeMapRenderTechnique, DistanceCubeMapRenderTechnique, HierarchicalParticleSystemTechnique, and IllumVolumeRenderTechnique.

5.38.3.7 virtual void RenderTechnique::runUpdated (RenderingRunType *runType*, RenderingRun * *run*) [inline, virtual, inherited]

Called after one of he shared runs updates.

Parameters:

runType enum describing the type of the updated run

run pointer to the updated RenderingRun

Reimplemented in CausticCasterRenderTechnique, ColorCubeMapRenderTechnique, CubeMapRenderTechnique, DistanceCubeMapRenderTechnique, HierarchicalParticleSystemTechnique, and IllumVolumeRenderTechnique.

5.38.3.8 void HierarchicalParticleSystemTechnique::update (unsigned long *frameNum*) [virtual, inherited]

Updates the resources in the given frame.

A RenderTechnique is usually need some resources from several runs, so these runs will be updated.

Parameters:

frameNum the actual framenumber

Reimplemented from RenderTechnique.

5.38.3.9 void HierarchicalParticleSystemTechnique::runChanged (RenderingRunType *runType*, RenderingRun * *run*) [virtual, inherited]

Called after one of he shared runs changes.

Parameters:

runType enum describing the type of the changed run

run pointer to the changed RenderingRun

Reimplemented from RenderTechnique.

5.38.3.10 void HierarchicalParticleSystemTechnique::runUpdated (RenderingRunType *runType*, RenderingRun * *run*) [virtual, inherited]

Called after one of he shared runs updates.

Parameters:

runType enum describing the type of the updated run

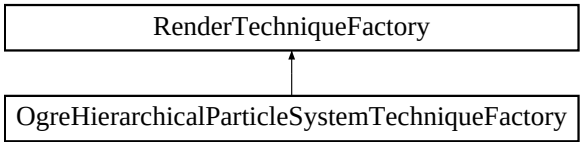
run pointer to the updated RenderingRun

Reimplemented from RenderTechnique.

5.39 OgreHierarchicalParticleSystemTechniqueFactory Class Reference

RenderTechniqueFactory to create OgreHierarchicalParticleSystemTechnique instances.

Inheritance diagram for OgreHierarchicalParticleSystemTechniqueFactory::



Public Member Functions

- OgreHierarchicalParticleSystemTechniqueFactory ()
  Technique factory.
- OgreRenderTechnique * createInstance (IllumTechniqueParams *params, Pass *pass, OgreRenderable *parentRenderable, OgreTechniqueGroup *parentTechniqueGroup)
  Creates a RenderTechnique of the factory type.
- bool isType (String type)
  Returns if this factory can create a RenderTechnique of the given type.
- virtual void parseParams (IllumTechniqueParams *params)
  parses parameters from the material file.

Protected Types

- typedef void(*) ILLUM_ATTRIBUTE_PARSER (String &params, RenderTechniqueFactory *factory)
  function for parsing RenderTechnique attributes
- typedef std::map< String, ILLUM_ATTRIBUTE_PARSER > AttribParserList
  Keyword-mapped attribute parsers.

Protected Attributes

- AttribParserList attributeParsers
  map of parser functions
- String typeName
  factoryname

5.39.1 Detailed Description

RenderTechniqueFactory to create OgreHierarchicalParticleSystemTechnique instances.

5.39.2 Member Typedef Documentation

5.39.2.1 typedef void(∗) RenderTechniqueFactory::ILLUM_ATTRIBUTE_PARSER(String &params, RenderTechniqueFactory ∗factory) [protected, inherited]

function for parsing RenderTechnique attributes

Parameters:

params attribute value stored in a String

5.39.3 Member Function Documentation

5.39.3.1 OgreRenderTechnique ∗ OgreHierarchicalParticleSystemTechniqueFactory::createInstance (IllumTechniqueParams ∗ *params*, Pass ∗ *pass*, OgreRenderable ∗ *parentRenderable*, OgreTechniqueGroup ∗ *parentTechniqueGroup*) [virtual]

Creates a RenderTechnique of the factory type.

Parameters:

params containes constructor parameters as NameValuePairList

pass the Pass to use in RenderTechnique constructor

pass the parentRenderable to pass to RenderTechnique constructor

pass the parentTechniqueGroup to pass to RenderTechnique constructor

Implements RenderTechniqueFactory.

5.39.3.2 bool RenderTechniqueFactory::isType (String *type*) [inline, inherited]

Returns if this factory can create a RenderTechnique of the given type.

Parameters:

type RenderTechnique type

5.39.3.3 void RenderTechniqueFactory::parseParams (IllumTechniqueParams * *params*) [virtual, inherited]

parses parameters from the material file.

The parsed parameters will be passed to the new RenderTechnique's constructor.

Parameters:

params pointer to the IllumTechniqueParams structure that was read from the material script and containes the parameters to be parsed.

5.40 OgreIlluminationManager Class Reference

Implementation of IlluminationManager in an OGRE environment.

Public Member Functions

- void addUpdateListener (UpdateListener *l)

  Registers an UpdateListener instance.

- void addRenderTechniqueFactory (RenderTechniqueFactory *factory)

  registers a RenderTechniqueFactory

- float getMaxJoinRadius ()

  retirieves the maximum bounding sphere radius with two SharedRuns can be joined.

- float getMaxJoinRadius (RenderingRunType type)

  Retirieves the maximum shared bounding sphere radius for a given run type.

- void setMaxJoinRadius (float rad)

  sets the maximum bounding sphere radius with two SharedRuns can be joined for all run type.

- void setMaxJoinRadius (RenderingRunType type, float rad)

  Sets the maximum shared bounding sphere radius for a given run type.

- void setFocusingMapSize (unsigned int size)
- void setPhaseTextureSize (unsigned int size)
- void setShadowMapSize (unsigned int size)
- void update (unsigned long frameNumber, RenderTarget *rt)

  The function to be called to render one frame.

- void initTechniques ()

  searches for RenderTechniques in materials and creates them for all objects.

- void initTechniques (Entity *e)

  searches for RenderTechniques in materials and creates them for an Entity.

- void initTechniques (BillboardSet *bbs, ParticleSystem *sys)

  searches for RenderTechniques in materials and creates them for a Billboardset.

- Camera * getMainCamera ()

  Returns a pointer to the player camera.

- Viewport * getMainViewport ()

  Returns a pointer to the viewport attached to the player camera.

- void setMainCamera (Camera *camera)

Sets the player camera.

- void setMainViewport (Viewport *viewport)

  Sets the viewport attached to the player camera.

- void sharedRunSplit (SharedRuns *old, SharedRuns *new1, SharedRuns *new2)

  The function to be called when a shared run is splitted.

- void sharedRunJoin (SharedRuns *old1, SharedRuns *old2, SharedRuns *newsr)

  The function to be called when two shared runs are joined.

- void joinSharedRuns ()

  Joins shared runs if needed.

- void addSharedRuns (SharedRuns *runs)

  Register a shared run object.

- void getNearestCausticCasters (Vector3 position, std::vector< OgreSharedRuns * > *nearestcasters, unsigned int maxCount)

  Searches for the nearest object groups (SharedRuns) that are caustic casters from a given point.

- void createGlobalRun (RenderingRunType runType)

  Creates a global RenderingRun of the given type.

- RenderingRun * getGlobalRun (RenderingRunType runType)

  Returns the global RendderingRun with the given type.

- void updateGlobalRun (RenderingRunType runType, unsigned long frameNum)

  Updates a global RenderingRun with the given type.

- void createPerLightRun (String lightName, RenderingRunType runType)

  Creates a RenderingRun attached to a lightsource with the given type.

- RenderingRun * getPerLightRun (String lightName, RenderingRunType runType)

  Retuns a RenderingRun attached to a lightsource with the given type.

- void updatePerLightRun (String lightName, RenderingRunType runType, unsigned long frameNum)

  Updates a RenderingRun attached to a lightsource with the given type.

- void savePhaseTextureToFile (String filename)

  Saves the phase texture to the given file.

- bool frameStarted (const FrameEvent &evt)

  Frame listener event handler function.

- bool getUseLISPSM ()
- bool getFocusingShadowMap ()
- bool getBlurShadowMap ()
- void setUseLISPSM (bool use)

- void setFocusingSM (bool use)
- void setBlurShadowMap (bool use)
- void setShadowMapMaterialName (String name)
- void addPathMapClusters (String subEntityName, PathMapClusters clusters)

  Registers a PathMapClusters structure for a given subentity.

- PathMapClusters * getPathMapClusters (String subEntityName)

  Returns the PathMapClusters structure registered for a given subentity.

- void addPathMapEntryPoint (PathMapEntryPoint p)

  Adds a new PathMapEntryPoint cluster to the entrypoint list.

- std::vector< PathMapEntryPoint > & getPathMapEntryPoints ()

  Returns the list of entrypoints.

- void addPathMapClusterLength (unsigned int l)

  Adds a new cluster size.

- unsigned int getPathMapClusterLengthsSize ()

  Gets the number of clusters.

- unsigned int getPathMapClusterLength (unsigned int index)

  Gets the size of the given cluster.

- float getAreaLightRadius ()
- void setAreaLigtRadius (float radius)
- void setFireRenderTargetSize (int size)

  Sets the fire rendertarget - frame buffer resolution ratio.

Static Public Member Functions

- static OgreIlluminationManager & getSingleton ()

  Returns the one and only OgreIlluminationManager instance.

Protected Member Functions

- OgreIlluminationManager ()

  Protected constructor (OgreIlluminationManager is a singleton).

- virtual ~OgreIlluminationManager ()

  Protected destructor.

- void fillVisibleList (RenderQueue *rq)

  Searches for visible renderables with valid TechniqueGroups in a renderqueue.

- void createTechnique (IllumTechniqueParams *params, Pass *pass, OgreRenderable *rend, OgreSharedRuns *sRuns)

creates a specific type of RenderTechnique for a Renderable's pass.

- BillboardSet * findRenderableInParticleSystem (ParticleSystem *system)

  A helper function to find the renderable object attached to a particle system (ONLY BILLBOARDSETS ARE SUPPORTED).

- void preAllUpdates ()

  Fires preAllUpdates for registered UpdateListeners.

- void postAllUpdates ()

  Fires postAllUpdates for registered UpdateListeners.

Protected Attributes

- std::list< RenderTechniqueFactory * > techniqueFactories

  registered RenderTechniqueFactories

- float maxRad

  The maximum bounding sphere radius that groupped objects (see SharedRuns class) can have.

- unsigned int focusingMapSize

  Size of the focusing map.

- unsigned int shadowMapSize

  Size of the shadow maps.

- float areaLightRadius

  Size of area lights for soft shadows.

- bool useLISPSM

  Sets if light space perspective shadow mapping should be used.

- bool blurSM

  Sets if the shadow maps should be blurred.

- bool focusingSM

  Sets if shadow maps should be focused.

- String shadowMapMaterialName

  The material name that should be used during rendering the shadow maps.

- unsigned int phaseTextureSize

  Size of the phase texture.

- std::map< RenderingRunType, float > maxRads

  Stores maximum bounding radius values for each rendering run type.

- std::map< String, PathMapClusters > pathMapClusters

  Stores PathMapClusters structures for each subentity.

- std::vector< PathMapEntryPoint > pathMapEntryPoints

  PathMapEntryPoint list.

- std::vector< unsigned int > pathMapClusterLengths

  Stores cluster size for each path map cluster.

- Camera * mainCamera

  The camera attached to the player.

- Viewport * mainViewport

  The viewport of the player camera.

- VisibleFinderVisitor * visitor

  VisibleFinderVisitor instance.

- std::vector< const Renderable * > visibleObjects

  Vector containing visible renderables with valid TechniqueGroups that must be refreshed.

- std::list< SharedRuns * > sharedRunRoots

  List containing SharedRuns roots.

- OgreSharedRuns globalSharedRuns

  Group of RenderingRuns that are used globaly.

- std::map< String, OgreSharedRuns * > perLightRuns

  Stores groups of RenderingRuns that are attached to individual light sources.

- std::vector< UpdateListener * > updateListeners

  Stores registered UpdateListeners.

Static Protected Attributes

- static OgreIlluminationManager * instance

  The one and only OgreIlluminationManager instance.

5.40.1 Detailed Description

Implementation of IlluminationManager in an OGRE environment.

5.40.2 Member Function Documentation

5.40.2.1 void OgreIlluminationManager::fillVisibleList (RenderQueue * *rq*) [protected]

Searches for visible renderables with valid TechniqueGroups in a renderqueue.

Parameters:

rq pointer to the RenderQueue instance to search in

5.40.2.2 void OgreIlluminationManager::createTechnique (IllumTechniqueParams * *params*, Pass * *pass*, OgreRenderable * *rend*, OgreSharedRuns * *sRuns*) [protected]

creates a specific type of RenderTechnique for a Renderable's pass.

It searches all registered RenderTechniqueFactories.

5.40.2.3 BillboardSet * OgreIlluminationManager::findRenderableInParticleSystem (ParticleSystem * *system*) [protected]

A helper function to find the renderable object attached to a particle system (ONLY BILLBOARDSETS ARE SUPPORTED).

Parameters:

system pointer to the ParticleSystem instance to search in

Returns:

pointer the connected BillboardSet instance

5.40.2.4 void OgreIlluminationManager::preAllUpdates () [protected]

Fires preAllUpdates for registered UpdateListeners.

This is called in each frame before updating the RenderTechniques.

5.40.2.5 void OgreIlluminationManager::postAllUpdates () [protected]

Fires postAllUpdates for registered UpdateListeners.

This is called in each frame after updating the RenderTechniques.

5.40.2.6 void OgreIlluminationManager::addUpdateListener (UpdateListener * *l*) [inline]

Registers an UpdateListener instance.

See also:

UpdateListener

5.40.2.7 float OgreIlluminationManager::getMaxJoinRadius () [inline]

retirieves the maximum bounding sphere radius with two SharedRuns can be joined.

Only valid fi all run types use the same radius. This can be set with calling setMaxJoinRadius().

See also:

setMaxJoinRadius

5.40.2.8 void OgreIlluminationManager::setFocusingMapSize (unsigned int *size*) [inline]

See also:

focusingMapSize

5.40.2.9 void OgreIlluminationManager::setPhaseTextureSize (unsigned int *size*) [inline]

See also:

phaseTextureSize

5.40.2.10 void OgreIlluminationManager::setShadowMapSize (unsigned int *size*) [inline]

See also:

shadowMapSize

5.40.2.11 void OgreIlluminationManager::update (unsigned long *frameNumber*, RenderTarget * *rt*)

The function to be called to render one frame.

This is the main refreshing function. It seasrches for visible objects, manages shared runs, updates render techniques and finally renders the scene to framebuffer.

Parameters:

frameNumber current framenumber

rt the rendertarget window. Needed to find the viewports that need to be refresh.

5.40.2.12 Camera* OgreIlluminationManager::getMainCamera () [inline]

Returns a pointer to the player camera.

Returns:

pointer to the main player camera. Needed by RenderTechnique and RenderingRun classes.

5.40.2.13 Viewport* OgreIlluminationManager::getMainViewport () [inline]

Returns a pointer to the viewport attached to the player camera.

Returns:

pointer to the viewport. Needed by RenderTechnique and RenderingRun classes.

5.40.2.14 void OgreIlluminationManager::setMainCamera (Camera * *camera*) [inline]

Sets the player camera.

Parameters:

camera pointer to the main player camera

5.40.2.15 void OgreIlluminationManager::setMainViewport (Viewport * *viewport*) [inline]

Sets the viewport attached to the player camera.

Parameters:

viewport pointer to the viewport

5.40.2.16 void OgreIlluminationManager::sharedRunSplit (SharedRuns * *old*, SharedRuns * *new1*, SharedRuns * *new2*)

The function to be called when a shared run is splitted.

Parameters:

old pointer to the SharedRuns instance that is split

new1 pointer to one of the SharedRuns instance that remain after split

new2 pointer to the other SharedRuns instance that remain after split

5.40.2.17 void OgreIlluminationManager::sharedRunJoin (SharedRuns * *old1*, SharedRuns * *old2*, SharedRuns * *newsr*)

The function to be called when two shared runs are joined.

Parameters:

old1 pointer to one of the SharedRuns instance that are joined
old2 pointer to the other SharedRuns instance that are joined
newsr pointer to the resulting parent SharedRuns instance

5.40.2.18 void OgreIlluminationManager::joinSharedRuns ()

Joins shared runs if needed.

Searches the registered shared run roots and join them if necessary (they are close enough).

5.40.2.19 void OgreIlluminationManager::addSharedRuns (SharedRuns * *runs*)

Register a shared run object.

Only called when new techniques are created.

Parameters:

runs pointer to the SharedRuns instance to add

5.40.2.20 void OgreIlluminationManager::getNearestCausticCasters (Vector3 *position*, std::vector< OgreSharedRuns * > * *nearestcasters*, unsigned int *maxCount*)

Searches for the nearest object groups (SharedRuns) that are caustic casters from a given point.

Parameters:

position the point to obtain distances from
nearestcasters vector to put the nearest caustic caster SharedRuns to
maxCount the maximum number of nearest casters to search for

5.40.2.21 void OgreIlluminationManager::createGlobalRun (RenderingRunType *runType*)

Creates a global RenderingRun of the given type.

If a RenderingRun with the given type already exist there is nothing to do.

Parameters:

runType type enum of the RenderingRun to create

5.40.2.22 RenderingRun * OgreIlluminationManager::getGlobalRun (RenderingRunType *runType*)

Returns the global RendderingRun with the given type.

Parameters:

runType type enum of the RenderingRun to retrieve

Returns:

pointer to the RenderingRun, NULL if no RenderingRun with the given type exists

5.40.2.23 void OgreIlluminationManager::updateGlobalRun (RenderingRunType *runType*, unsigned long *frameNum*)

Updates a global RenderingRun with the given type.

Parameters:

runType type enum of the RenderingRun to update

frameNum current framenumber

5.40.2.24 void OgreIlluminationManager::createPerLightRun (String *lightName*, RenderingRunType *runType*)

Creates a RenderingRun attached to a lightsource with the given type.

Parameters:

lightName name of the lightsource

runType type enum of the RenderingRun to create

create sharedruns

5.40.2.25 RenderingRun * OgreIlluminationManager::getPerLightRun (String *lightName*, RenderingRunType *runType*)

Retuns a RenderingRun attached to a lightsource with the given type.

Parameters:

lightName name of the lightsource

runType type enum of the RenderingRun to return

Returns:

pointer to the RenderingRun, NULL if no RenderingRun with the given type exists

5.40.2.26 void OgreIlluminationManager::updatePerLightRun (String *lightName*, RenderingRunType *runType*, unsigned long *frameNum*)

Updates a RenderingRun attached to a lightsource with the given type.

Parameters:

lightName name of the lightsource

runType type enum of the RenderingRun to update

frameNum current framenumber

5.40.2.27 bool OgreIlluminationManager::frameStarted (const FrameEvent & *evt*) [inline]

Frame listener event handler function.

Inherited from FrameListener. Called at the beginning of each frame.

5.40.2.28 bool OgreIlluminationManager::getUseLISPSM () [inline]

See also:

useLISPSM

5.40.2.29 bool OgreIlluminationManager::getFocusingShadowMap () [inline]

See also:

focusingSM

5.40.2.30 bool OgreIlluminationManager::getBlurShadowMap () [inline]

See also:

blurSM

5.40.2.31 void OgreIlluminationManager::setUseLISPSM (bool *use*) [inline]

See also:

useLISPSM

5.40.2.32 void OgreIlluminationManager::setFocusingSM (bool *use*) [inline]

See also:

focusingSM

5.40.2.33 void OgreIlluminationManager::setBlurShadowMap (bool *use*) [inline]

See also:

blurSM

5.40.2.34 void OgreIlluminationManager::setShadowMapMaterialName (String *name*) [inline]

See also:

shadowMapMaterialName

5.40.2.35 void OgreIlluminationManager::addPathMapClusters (String *subEntityName*, PathMapClusters *clusters*) [inline]

Registers a PathMapClusters structure for a given subentity.

Parameters:

subEntityName name of te subentity

clusters the PathMapClusters that belongs to the given subentity

5.40.2.36 PathMapClusters* OgreIlluminationManager::getPathMapClusters (String *subEntityName*) [inline]

Returns the PathMapClusters structure registered for a given subentity.

Parameters:

subEntityName name of te subentity

Returns:

pointer to the PathMapClusters structure that belongs to the given subentity

5.40.2.37 unsigned int OgreIlluminationManager::getPathMapClusterLength (unsigned int *index*) [inline]

Gets the size of the given cluster.

Parameters:

index of the cluster

Returns:

the size of the cluster

5.40.2.38 float OgreIlluminationManager::getAreaLightRadius () [inline]

See also:

areaLightRadius

5.40.2.39 void OgreIlluminationManager::setAreaLigtRadius (float *radius*) [inline]

See also:

areaLightRadius

5.40.3 Member Data Documentation

5.40.3.1 float OgreIlluminationManager::maxRad [protected]

The maximum bounding sphere radius that groupped objects (see SharedRuns class) can have.

See also:

canJoin
joinRuns

5.40.3.2 unsigned int OgreIlluminationManager::focusingMapSize [protected]

Size of the focusing map.

This map is used if the shadow maps should be focused.

5.40.3.3 bool OgreIlluminationManager::blurSM [protected]

Sets if the shadow maps should be blurred.

Used in variance shadow mapping.

5.40.3.4 String OgreIlluminationManager::shadowMapMaterialName [protected]

The material name that should be used during rendering the shadow maps.

There are several predefined materials that can be used to render shadow maps:

- GTP/Basic/Depth : writes projected depth values of front facing polygons
- GTP/Basic/DepthCCW : writes projected depth values of back facing polygons
- GTP/Basic/Distance : writes distance values (from eyepoint) of front facing polygons
- GTP/Basic/DistanceCCW : writes distance values (from eyepoint) of back facing polygons
- GTP/Basic/Distance_Normalized : writes normalized distance values (distance values devided by projection farplane - which is set to the attenuation range in case of shadow maps) of front facing polygons
- GTP/Basic/Distance_NormalizedCCW : writes normalized distance values of back facing polygons

The default material is GTP/Basic/Distance_NormalizedCCW. Recommended materials for different light types:

- spot and point lights : GTP/Basic/Distance_NormalizedCCW or GTP/Basic/Distance_Normalized
- directional lights : GTP/Basic/Depth or GTP/Basic/DepthCCW

5.40.3.5 std::map<String, PathMapClusters> OgreIlluminationManager::pathMapClusters [protected]

Stores PathMapClusters structures for each subentity.

The String key is the name of the subentity.

5.40.3.6 class VisibleFinderVisitor* OgreIlluminationManager::visitor [protected]

VisibleFinderVisitor instance.

Used for adding visible renderables with valid TechniqueGroups to the visibleObjects vector.

5.40.3.7 std::list<SharedRuns*> OgreIlluminationManager::sharedRunRoots [protected]

List containing SharedRuns roots.

It is the IlluminationManager's task to find the SharedRuns which can be joined. Only the root SharedRuns needs to be checked.

5.40.3.8 OgreSharedRuns OgreIlluminationManager::globalSharedRuns [protected]

Group of RenderingRuns that are used globaly.

Some RenderingRuns have only one instance per application (for example scene depth map). These resources are shared between all RenderTechniques.

5.40.3.9 std::map<String, OgreSharedRuns*> OgreIlluminationManager::perLightRuns [protected]

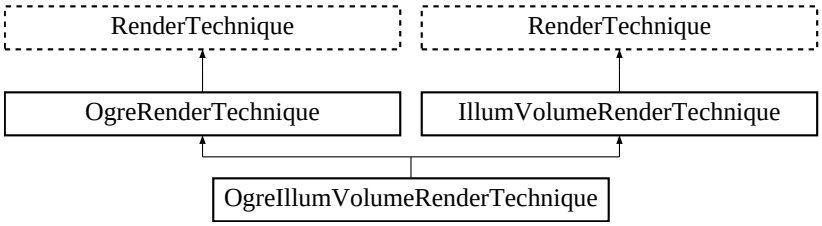
Stores groups of RenderingRuns that are attached to individual light sources.

These resources need separate instances for each lightsource (for example depth shadow maps). They are grouped by the name of the lightsource.

5.41 OgreIllumVolumeRenderTechnique Class Reference

IllumVolumeRenderTechnique used in an OGRE environment.

Inheritance diagram for OgreIllumVolumeRenderTechnique::



Public Member Functions

- OgreIllumVolumeRenderTechnique (unsigned long startFrame, unsigned long illumVolumeUpdateInterval, unsigned int illumTextureResolution, unsigned int textureDepth, unsigned char illumTexID, bool useDistCalc, String materialName, String lightMatrixGPUParamName, bool useHierarchicalImpostor, unsigned char impostorTexID, Pass *pass, OgreRenderable *parentRenderable, OgreTechniqueGroup *parentTechniqueGroup)

  Constructor.

- ~OgreIllumVolumeRenderTechnique ()

  Destructor.

- virtual OgreRenderTechnique * asOgreRenderTechnique ()

  Conversion to OgreRenderTechnique.

- virtual void update (unsigned long frameNum)

  Updates the resources in the given frame.

- virtual void runChanged (RenderingRunType runType, RenderingRun *run)

  Called after one of he shared runs changes.

- virtual void runUpdated (RenderingRunType runType, RenderingRun *run)

  Called after one of he shared runs updates.

- ElementaryRenderable * getParentRenderable ()

  Retrieves the renderable this technique operates on.

- void update (unsigned long frameNum)

  Updates the resources in the given frame.

- void runChanged (RenderingRunType runType, RenderingRun *run)

  Called after one of he shared runs changes.

- void runUpdated (RenderingRunType runType, RenderingRun *run)

  Called after one of he shared runs updates.

Protected Member Functions

- RenderingRun * createLightVolumeRenderingRun ()

  creates a light volume rendering run needed by this technique

- void lightVolumeChanged (RenderingRun *run)

  Called if the LightVolumeRenderingRun is chaged.

- void lightVolumeUpdated (RenderingRun *run)

  Called if the LightVolumeRenderingRun is updated.

- void hierarchicalImpostorUpdated (RenderingRun *run)

  Called if the ChildParticleSystemRenderingRun is chaged.

Protected Attributes

- String materialName

  the name of the material that is used while rendering the light volume

- unsigned char illumTexID

  the id of the texture unit state the resulting illumevolume should be bound to

- String lightMatrixGPUParamName

  the name of the gpu program parameter where the light matrix should be bound to

- unsigned char impostorTexID

  the id of the texture unit state where the impostor image of the smaller system should be bound to

- OgreRenderable * parentOgreRenderable

  a OgreRenderable pointer to the renderable this technique operates on.

- OgreTechniqueGroup * parentOgreTechniqueGroup

  a OgreTechniqueGroup pointer to the TechniqueGroup this technique is attached to.

- Pass * pass

  a pointer to the pass this technique operates on.

- ElementaryRenderable * parentRenderable

  The renderable this technique operates on.

- TechniqueGroup * parentTechniqueGroup

  The TechniqueGroup this RenderedTechnique is attached to.

- SharedRuns * sharedRuns

The SharedRuns this RenderedTechnique is attached to.

- unsigned long illumVolumeUpdateInterval

  the update frequency of the light volume

- unsigned int illumTextureResolution

  the resolution of the light volume texture

- unsigned int textureDepth

  the number of layers to use (should be set to 1)

- unsigned long startFrame

  offset in frame number used during update

- bool useDistCalc

  flag to skip updates if the shaded particle system is far away (not used)

- bool useHierarchicalImpostor

  set this flag to true if the particle system is a hierarchical particle system

5.41.1 Detailed Description

IllumVolumeRenderTechnique used in an OGRE environment.

5.41.2 Constructor & Destructor Documentation

5.41.2.1 OgreIllumVolumeRenderTechnique::OgreIllumVolumeRenderTechnique (unsigned long *startFrame*, unsigned long *illumVolumeUpdateInterval*, unsigned int *illumTextureResolution*, unsigned int *textureDepth*, unsigned char *illumTexID*, bool *useDistCalc*, String *materialName*, String *lightMatrixGPUParamName*, bool *useHierarchicalImpostor*, unsigned char *impostorTexID*, Pass * *pass*, OgreRenderable * *parentRenderable*, OgreTechniqueGroup * *parentTechniqueGroup*)

Constructor.

Parameters:

startFrame adds an offset to the current frame number to help evenly distribute updates between frames

illumVolumeUpdateInterval the update frequency of the light volume

illumTextureResolution the resolution of the light volume texture

textureDepth the number of layers to use (should be set to 1)

illumTexID the id of the texture unit state the resulting illumevolume should be bound to

useDistCalc flag to skip updates if the shaded particle system is far away (not used)

materialName the name of the material that is used while rendering the light volume

lightMatrixGPUParamName the name of the gpu program parameter where the light matrix should be bound to

useHierarchicalImpostor set this flag to true if the particle system is a hierarchical particle system

impostorTexID the id of the texture unit state where the impostor image of the smaller system should be bound to

pass the pass to operate on

parentRenderable the object to operate on

parentTechniqueGroup the TechniqueGroup this RenderedTechnique is attached to

5.41.3 Member Function Documentation

5.41.3.1 RenderingRun * OgreIllumVolumeRenderTechnique::createLightVolumeRenderingRun () [protected, virtual]

creates a light volume rendering run needed by this technique

Returns:

pointer to the created LightVolumeRenderingRun instance

Implements IllumVolumeRenderTechnique.

5.41.3.2 void OgreIllumVolumeRenderTechnique::lightVolumeChanged (RenderingRun * *run*) [protected, virtual]

Called if the LightVolumeRenderingRun is chaged.

Parameters:

pointer to the new LightVolumeRenderingRun instance to use

Implements IllumVolumeRenderTechnique.

5.41.3.3 void OgreIllumVolumeRenderTechnique::lightVolumeUpdated (RenderingRun * *run*) [protected, virtual]

Called if the LightVolumeRenderingRun is updated.

Parameters:

pointer to the updated LightVolumeRenderingRun instance

Implements IllumVolumeRenderTechnique.

5.41.3.4 void OgreIllumVolumeRenderTechnique::hierarchicalImpostorUpdated (RenderingRun * *run*) [protected, virtual]

Called if the ChildParticleSystemRenderingRun is chaged.

Only called if this particle system is a hierarchical particle system.

Parameters:

pointer to the new ChildParticleSystemRenderingRun instance to use

Implements IllumVolumeRenderTechnique.

5.41.3.5 virtual OgreRenderTechnique* OgreRenderTechnique::asOgreRenderTechnique () [inline, virtual, inherited]

Conversion to OgreRenderTechnique.

This function is needed because of virtual inheritance.

Reimplemented from RenderTechnique.

5.41.3.6 virtual void RenderTechnique::update (unsigned long *frameNum*) [inline, virtual, inherited]

Updates the resources in the given frame.

A RenderTechnique is usually need some resources from several runs, so these runs will be updated.

Parameters:

frameNum the actual framenumber

Reimplemented in ColorCubeMapRenderTechnique, ConvolvedCubeMapRenderTechnique, DistanceCubeMapRenderTechnique, HierarchicalParticleSystemTechnique, IllumVolumeRenderTechnique, OgreCausticReceiverRenderTechnique, OgreColorCubeMapRenderTechnique, OgreConvolvedCubeMapRenderTechnique, OgreDepthShadowReceiverRenderTechnique, OgreDistanceCubeMapRenderTechnique, OgreFireRenderTechnique, OgrePathMapRenderTechnique, and OgreSBBRenderTechnique.

5.41.3.7 virtual void RenderTechnique::runChanged (RenderingRunType *runType*, RenderingRun * *run*) [inline, virtual, inherited]

Called after one of he shared runs changes.

Parameters:

runType enum describing the type of the changed run

run pointer to the changed RenderingRun

Reimplemented in CausticCasterRenderTechnique, ColorCubeMapRenderTechnique, ConvolvedCubeMapRenderTechnique, CubeMapRenderTechnique, DistanceCubeMapRenderTechnique, HierarchicalParticleSystemTechnique, and IllumVolumeRenderTechnique.

5.41.3.8 virtual void RenderTechnique::runUpdated (RenderingRunType *runType*, RenderingRun * *run*) [inline, virtual, inherited]

Called after one of he shared runs updates.

Parameters:

runType enum describing the type of the updated run

run pointer to the updated RenderingRun

Reimplemented in CausticCasterRenderTechnique, ColorCubeMapRenderTechnique, CubeMapRenderTechnique, DistanceCubeMapRenderTechnique, HierarchicalParticleSystemTechnique, and IllumVolumeRenderTechnique.

5.41.3.9 void IllumVolumeRenderTechnique::update (unsigned long *frameNum*) [virtual, inherited]

Updates the resources in the given frame.

A RenderTechnique is usually need some resources from several runs, so these runs will be updated.

Parameters:

frameNum the actual framenumber

Reimplemented from RenderTechnique.

5.41.3.10 void IllumVolumeRenderTechnique::runChanged (RenderingRunType *runType*, RenderingRun * *run*) [virtual, inherited]

Called after one of he shared runs changes.

Parameters:

runType enum describing the type of the changed run

run pointer to the changed RenderingRun

Reimplemented from RenderTechnique.

5.41.3.11 void IllumVolumeRenderTechnique::runUpdated (RenderingRunType *runType*, RenderingRun * *run*) [virtual, inherited]

Called after one of he shared runs updates.

Parameters:

runType enum describing the type of the updated run

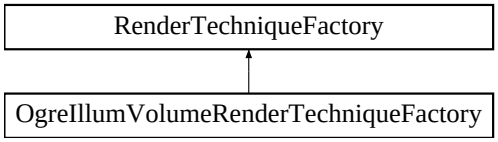
run pointer to the updated RenderingRun

Reimplemented from RenderTechnique.

5.42 OgreIllumVolumeRenderTechniqueFactory Class Reference

RenderTechniqueFactory to create OgreIllumVolumeRenderTechnique instances.

Inheritance diagram for OgreIllumVolumeRenderTechniqueFactory::



Public Member Functions

- OgreIllumVolumeRenderTechniqueFactory ()

  Technique factory.

- OgreRenderTechnique * createInstance (IllumTechniqueParams *params, Pass *pass, OgreRenderable *parentRenderable, OgreTechniqueGroup *parentTechniqueGroup)

  Creates a RenderTechnique of the factory type.

- bool isType (String type)

  Returns if this factory can create a RenderTechnique of the given type.

- virtual void parseParams (IllumTechniqueParams *params)

  parses parameters from the material file.

Protected Types

- typedef void(*) ILLUM_ATTRIBUTE_PARSER (String &params, RenderTechniqueFactory *factory)

  function for parsing RenderTechnique attributes

- typedef std::map< String, ILLUM_ATTRIBUTE_PARSER > AttribParserList

  Keyword-mapped attribute parsers.

Protected Attributes

- AttribParserList attributeParsers

  map of parser functions

- String typeName

  factoryname

5.42.1 Detailed Description

RenderTechniqueFactory to create OgreIllumVolumeRenderTechnique instances.

5.42.2 Member Typedef Documentation

5.42.2.1 typedef void(∗) RenderTechniqueFactory::ILLUM_ATTRIBUTE_PARSER(String &params, RenderTechniqueFactory ∗factory) [protected, inherited]

function for parsing RenderTechnique attributes

Parameters:

params attribute value stored in a String

5.42.3 Member Function Documentation

5.42.3.1 OgreRenderTechnique ∗ OgreIllumVolumeRenderTechniqueFactory::createInstance (IllumTechniqueParams ∗ *params*, Pass ∗ *pass*, OgreRenderable ∗ *parentRenderable*, OgreTechniqueGroup ∗ *parentTechniqueGroup*) [virtual]

Creates a RenderTechnique of the factory type.

Parameters:

params containes constructor parameters as NameValuePairList

pass the Pass to use in RenderTechnique constructor

pass the parentRenderable to pass to RenderTechnique constructor

pass the parentTechniqueGroup to pass to RenderTechnique constructor

Implements RenderTechniqueFactory.

5.42.3.2 bool RenderTechniqueFactory::isType (String *type*) [inline, inherited]

Returns if this factory can create a RenderTechnique of the given type.

Parameters:

type RenderTechnique type

5.42.3.3 void RenderTechniqueFactory::parseParams (IllumTechniqueParams * *params*) [virtual, inherited]

parses parameters from the material file.

The parsed parameters will be passed to the new RenderTechnique's constructor.

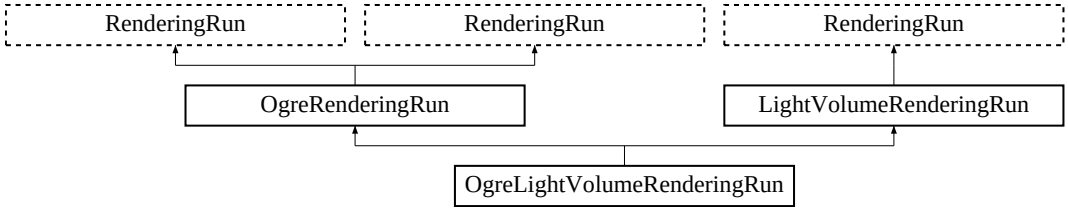
Parameters:

params pointer to the IllumTechniqueParams structure that was read from the material script and containes the parameters to be parsed.

5.43 OgreLightVolumeRenderingRun Class Reference

LightVolumeRenderingRun used in an OGRE environment.

Inheritance diagram for OgreLightVolumeRenderingRun::



Public Member Functions

- OgreLightVolumeRenderingRun (OgreSharedRuns *sharedRuns, String name, unsigned long startFrame, unsigned long updateInterval, unsigned int resolution, unsigned int textureDepth, String materialName)

  Constructor.

- String getLightVolumeTextureName ()

  returns the name of the resulting light volume texture

- void refreshLight ()

  Refreshes light camera matrices, called in each update.

- Matrix4 getLightViewProjectionMatrix ()

  Returns the light matrix used while rendering the light volume.

- OgreRenderingRun * asOgreRenderingRun ()

  Conversion to OgreRenderRun.

- OgreRenderingRun * asOgreRenderingRun ()

  Conversion to OgreRenderRun.

- bool update (unsigned long frameNum)

  Calls updateFrame() if the run needs update according to its starting frame and update interval and has not been allready updated in this frame.

- virtual bool canJoin (RenderingRun *run)

  Returns true if two runs can be joined.

Protected Member Functions

- void updateFrame (unsigned long frameNum)

  This function does the actual update in a frame.

- void createLightVolumeMap ()

  Creates a light volume map.

- Vector3 getCubeMapFaceDirection (unsigned char faceId)

  Returns a direction for a cubemap face id.

- Vector3 getCubeMapFaceDirection (unsigned char faceId)

  Returns a direction for a cubemap face id.

- Texture * createCubeRenderTexture (String name, const Vector3 position, unsigned int resolution=512, PixelFormat format=PF_FLOAT16_RGBA, int numMips=0, ColourValue clearColor=ColourValue::Black)

  Creates a cubemap texture.

- Texture * createCubeRenderTexture (String name, const Vector3 position, unsigned int resolution=512, PixelFormat format=PF_FLOAT16_RGBA, int numMips=0, ColourValue clearColor=ColourValue::Black)

  Creates a cubemap texture.

- void setMaterialForRenderables (String &materialName, RenderQueue *rq)

  Sets the given material for each Renderable in a RenderQueue.

- void setMaterialForRenderables (String &materialName, RenderQueue *rq, bool solidsonly=false)

  Sets the given material for each Renderable in a RenderQueue.

- void setMaterialForVisibles (String &materialName, Camera *cam, bool shadowcastersonly=false)

  Sets the given material for each Renderable visible from a given camera.

- void setMaterialForVisibles (String &materialName, Camera *cam, bool shadowcastersonly=false, bool solidsonly=false)

  Sets the given material for each Renderable visible from a given camera.

- void restoreMaterials ()

  Restores previously stored materials.

- void restoreMaterials ()

  Restores previously stored materials.

- void renderFullscreenQuad (String materialName, RenderTarget *target)

  Renderes a full screen quad on a given RendderTarget with a given material.

- void renderFullscreenQuad (String materialName, RenderTarget *target)

  Renderes a full screen quad on a given RendderTarget with a given material.

- void renderPixelSprites (String &materialName, RenderTarget *rt, int width, int height)

Renderes sprites to pixels of the screen on a given RendderTarget with a given material.

- void renderPixelSprites (String &materialName, RenderTarget *rt, int width, int height)
  Renderes sprites to pixels of the screen on a given RendderTarget with a given material.
- void renderFullscreenGrid (String &materialName, RenderTarget *rt, int width, int height)
  Renders a grid onto the screen.
- virtual bool needUpdate (unsigned long frameNum)
  Returns if this run needs update.

Protected Attributes

- String materialName
  the name of the material should be used when rendering the light volume
- Light * light
  pointer to the nearest light source to the particle system
- Camera * lightVolumeCamera
  pointer to the camera used while rendering the light volume
- OgreSharedRuns * sharedRuns
  a pointer to the OgreSharedRuns this run belongs to
- String name
  the name of the light volume texture that was created by this run
- float systemRadius
  the radius of the particle system
- std::map< Renderable *, String > visibleObjects
  map of Renderables which will be rendered with a given material
- std::map< Renderable *, String > visibleObjects
  map of Renderables which will be rendered with a given material
- SpriteSet * pixelSprites
  SpriteSet used in pixel sprite rendering.
- BillboardSet * pixelSprites
  SpriteSet used in pixel sprite rendering.
- String spriteSetName
  unique name of the SpriteSet used in pixel sprite rendering
- Entity * fullscreenGrid
  Entity used in fullscreen grid rendering.

- unsigned long lastupdated

  The number of the last frame this run was updated.

- unsigned long startFrame

  The number of the frame this run should be updated first.

- unsigned long updateInterval

  Refresh frequency in frames.

- unsigned int resolution

  the resolution of the light volume map

- unsigned int textureDepth

  number of layers (should be 1)

Static Protected Attributes

- static MovablePlane * fullScreenQuad

  fulls screen quad plane used in full screen quad rendering

- static MovablePlane * fullScreenQuad

  fulls screen quad plane used in full screen quad rendering

- static Entity * fullScreenQuadEntity

  fulls screen quad Entity used in full screen quad rendering

- static Entity * fullScreenQuadEntity

  fulls screen quad Entity used in full screen quad rendering

- static SceneNode * fullScreenQuadNode

  fulls screen quad SceneNode used in full screen quad rendering

- static SceneNode * fullScreenQuadNode

  fulls screen quad SceneNode used in full screen quad rendering

5.43.1 Detailed Description

LightVolumeRenderingRun used in an OGRE environment.

5.43.2 Constructor & Destructor Documentation

5.43.2.1 OgreLightVolumeRenderingRun::OgreLightVolumeRenderingRun (OgreSharedRuns * *sharedRuns*, String *name*, unsigned long *startFrame*, unsigned long *updateInterval*, unsigned int *resolution*, unsigned int *textureDepth*, String *materialName*)

Constructor.

Parameters:

sharedRuns a pointer to the OgreSharedRuns this run belongs to

name the name of the light volume texture to be created

startFrame adds an offset to the current frame number to help evenly distribute updates between frames

updateInterval update frequency

resolution light volume texture resolution

materialName the name of the material should be used when rendering the light volume

5.43.3 Member Function Documentation

5.43.3.1 void OgreLightVolumeRenderingRun::updateFrame (unsigned long *frameNum*) [protected, virtual]

This function does the actual update in a frame.

Parameters:

frameNum current frame number

Implements LightVolumeRenderingRun.

5.43.3.2 OgreRenderingRun* OgreRenderingRun::asOgreRenderingRun () [inline, virtual, inherited]

Conversion to OgreRenderRun.

This function is needed because of virtual inheritance.

Reimplemented from RenderingRun.

5.43.3.3 OgreRenderingRun* OgreRenderingRun::asOgreRenderingRun () [inline, virtual, inherited]

Conversion to OgreRenderRun.

This function is needed because of virtual inheritance.

Reimplemented from RenderingRun.

5.43.3.4 Vector3 OgreRenderingRun::getCubeMapFaceDirection (unsigned char *faceId*) [protected, inherited]

Returns a direction for a cubemap face id.

This is a helper function to retrieve the normal direction of a given cubemap face.

Parameters:

faceId the number of the face

5.43.3.5 Vector3 OgreRenderingRun::getCubeMapFaceDirection (unsigned char *faceId*) [protected, inherited]

Returns a direction for a cubemap face id.

This is a helper function to retrieve the normal direction of a given cubemap face.

Parameters:

faceId the number of the face

5.43.3.6 Texture * OgreRenderingRun::createCubeRenderTexture (String *name*, const Vector3 *position*, unsigned int *resolution* = 512, PixelFormat *format* = PF_FLOAT16_RGBA, int *numMips* = 0, ColourValue *clearColor* = ColourValue::Black) [protected, inherited]

Creates a cubemap texture.

This is a helper function to easily create a cubemap texture and automaticly attach viewports to each face so it can be used as a rendertarget.

Parameters:

name the name of the texture to be created

position the initial position of the cubemap

resolution the resolution of one cubemapface

format the pixel format of the cubemap

numMips the number of mipmap levels

clearColor initial color

5.43.3.7 Texture* OgreRenderingRun::createCubeRenderTexture (String *name*, const Vector3 *position*, unsigned int *resolution* = 512, PixelFormat *format* = PF_FLOAT16_RGBA, int *numMips* = 0, ColourValue *clearColor* = ColourValue::Black) [protected, inherited]

Creates a cubemap texture.

This is a helper function to easily create a cubemap texture and automaticly attach viewports to each face so it can be used as a rendertarget.

Parameters:

name the name of the texture to be created

position the initial position of the cubemap

resolution the resolution of one cubemapface

format the pixel format of the cubemap

numMips the number of mipmap levels

clearColor initial color

5.43.3.8 void OgreRenderingRun::setMaterialForRenderables (String & *materialName*, RenderQueue * *rq*) [protected, inherited]

Sets the given material for each Renderable in a RenderQueue.

This is a helper function to set a material to each element of a previously filled Renderque. The orginal material of the Renderables are stored so they can be restored later. The function also tells the current SceneManager not to search for visible objects, as we are going to use the given RenderQueue during the next rendering.

Parameters:

materialName the name of the material to set for the Renderables

rq pointer to the filled Renderqueue instance to set material for

5.43.3.9 void OgreRenderingRun::setMaterialForRenderables (String & *materialName*, RenderQueue * *rq*, bool *solidsonly* = false) [protected, inherited]

Sets the given material for each Renderable in a RenderQueue.

This is a helper function to set a material to each element of a previously filled Renderque. The orginal material of the Renderables are stored so they can be restored later. The function also tells the current SceneManager not to search for visible objects, as we are going to use the given RenderQueue during the next rendering.

Parameters:

materialName the name of the material to set for the Renderables

rq pointer to the filled Renderqueue instance to set material for

5.43.3.10 void OgreRenderingRun::setMaterialForVisibles (String & *materialName*, Camera * *cam*, bool *shadowcastersonly* = false) [protected, inherited]

Sets the given material for each Renderable visible from a given camera.

This helper function is similar to setMaterialForRenderables but it is also responsible for filling the RenderQueue. First the RenderQueue of the current SceneManager fill be filled with the visible objects seen from the given camera. Then the required material will be set for each element of the RenderQueue. The orginal material of the Renderables are stored so they can be restored later. The function also tells the current SceneManager not to search for visible objects, as we are going to use the filled RenderQueue during the next rendering.

Parameters:

materialName the name of the material to set for the Renderables

cam pointer to the camera from which visible objects should be searched

shadowcastersonly flag to search for only shadow casters

5.43.3.11 void OgreRenderingRun::setMaterialForVisibles (String & *materialName*, Camera * *cam*, bool *shadowcastersonly* = false, bool *solidsonly* = false) [protected, inherited]

Sets the given material for each Renderable visible from a given camera.

This helper function is similar to setMaterialForRenderables but it is also responsible for filling the RenderQueue. First the RenderQueue of the current SceneManager fill be filled with the visible objects seen from the given camera. Then the required material will be set for each element of the RenderQueue. The orginal material of the Renderables are stored so they can be restored later. The function also tells the current SceneManager not to search for visible objects, as we are going to use the filled RenderQueue during the next rendering.

Parameters:

materialName the name of the material to set for the Renderables

cam pointer to the camera from which visible objects should be searched

shadowcastersonly flag to search for only shadow casters

5.43.3.12 void OgreRenderingRun::restoreMaterials () [protected, inherited]

Restores previously stored materials.

This helper function is typically used after a setMaterialForRenderables or setMaterialForVisibles call and a rendering process to restore the original material settings. The function also tells the current SceneManager to search for visible objects, as this is the default behaviour of SceneManager.

5.43.3.13 void OgreRenderingRun::restoreMaterials () [protected, inherited]

Restores previously stored materials.

This helper function is typically used after a setMaterialForRenderables or setMaterialForVisibles call and a rendering process to restore the original material settings. The function also tells the current SceneManager to search for visible objects, as this is the default behaviour of SceneManager.

5.43.3.14 void OgreRenderingRun::renderFullscreenQuad (String *materialName*, RenderTarget * *target*) [protected, inherited]

Renderes a full screen quad on a given RendderTarget with a given material.

Parameters:

materialName the name of the material bind to the quad

target the RenderTarget the quad should be rendered on

5.43.3.15 void OgreRenderingRun::renderFullscreenQuad (String *materialName*, RenderTarget * *target*) [protected, inherited]

Renderes a full screen quad on a given RendderTarget with a given material.

Parameters:

materialName the name of the material bind to the quad

target the RenderTarget the quad should be rendered on

5.43.3.16 void OgreRenderingRun::renderPixelSprites (String & *materialName*, RenderTarget * *rt*, int *width*, int *height*) [protected, inherited]

Renderes sprites to pixels of the screen on a given RendderTarget with a given material.

Pixel sprites are pixel sized quads, placed on each pixel of the RenderTarget. The number of sprites not necessary corresponds to the resolution of the rendertarget. The pixel quads will evenly fill the rendertarget's area with sizes corresponding to the given resolution. We can render fewer or more pixel quads than the number of pixels the rendertarget has (eg.: in case of caustic cubemap generation).

Parameters:

materialName the name of the material bind to the pixel sprites

rt the RenderTarget the quads should be rendered on

width the desired resolution width of the sprites

height the desired resolution height of the sprites

5.43.3.17 void OgreRenderingRun::renderPixelSprites (String & *materialName*, RenderTarget * *rt*, int *width*, int *height*) [protected, inherited]

Renderes sprites to pixels of the screen on a given RendderTarget with a given material.

Pixel sprites are pixel sized quads, placed on each pixel of the RenderTarget. The number of sprites not necessary corresponds to the resolution of the rendertarget. The pixel quads will evenly fill the rendertarget's area with sizes corresponding to the given resolution. We can render fewer or more pixel quads than the number of pixels the rendertarget has (eg.: in case of caustic cubemap generation).

Parameters:

materialName the name of the material bind to the pixel sprites

rt the RenderTarget the quads should be rendered on

width the desired resolution width of the sprites

height the desired resolution height of the sprites

5.43.3.18 void OgreRenderingRun::renderFullscreenGrid (String & *materialName*, RenderTarget * *rt*, int *width*, int *height*) [protected, inherited]

Renders a grid onto the screen.

Parameters:

rt the RenderTarget the grid should be rendered on

width the desired horizontal resolution of the grid

height the desired vertical resolution of the grid

5.43.3.19 virtual bool RenderingRun::canJoin (RenderingRun * *run*) [inline, virtual, inherited]

Returns true if two runs can be joined.

In some cases special requirements should stand to join two runs (even if they have the same type). Eg.: two caustic cube map generation technique should only be joined if they use the same material when rendering the caustic cubemap.

Reimplemented in OgreCausticCubeMapRenderingRun, OgreChildPSystemRenderingRun, and OgrePhotonMapRenderingRun.

5.43.3.20 virtual bool RenderingRun::needUpdate (unsigned long *frameNum*) [inline, protected, virtual, inherited]

Returns if this run needs update.

This tipically depends on the upate interval and the starting frame number.

Parameters:

frameNum current frame number

Reimplemented in OgreFocusingMapRenderingRun, and OgrePMEntryPointMapRenderingRun.

5.43.4 Member Data Documentation

5.43.4.1 std::map<Renderable*, String> OgreRenderingRun::visibleObjects [protected, inherited]

map of Renderables which will be rendered with a given material

The String stores the original material name that will be restored after rendering.

See also:

setMaterialForVisibles
setMaterialForRenderables
restoreMaterials

5.43.4.2 std::map<Renderable*, String> OgreRenderingRun::visibleObjects [protected, inherited]

map of Renderables which will be rendered with a given material

The String stores the original material name that will be restored after rendering.

See also:

setMaterialForVisibles
setMaterialForRenderables
restoreMaterials

5.43.4.3 MovablePlane * OgreRenderingRun::fullScreenQuad [static, protected, inherited]

fulls screen quad plane used in full screen quad rendering

See also:

renderFullscreenQuad

5.43.4.4 MovablePlane* OgreRenderingRun::fullScreenQuad [static, protected, inherited]

fulls screen quad plane used in full screen quad rendering

See also:

renderFullscreenQuad

5.43.4.5 Entity * OgreRenderingRun::fullScreenQuadEntity [static, protected, inherited]

fulls screen quad Entity used in full screen quad rendering

See also:

renderFullscreenQuad

5.43.4.6 Entity* OgreRenderingRun::fullScreenQuadEntity [static, protected, inherited]

fulls screen quad Entity used in full screen quad rendering

See also:

renderFullscreenQuad

5.43.4.7 SceneNode * OgreRenderingRun::fullScreenQuadNode [static, protected, inherited]

fulls screen quad SceneNode used in full screen quad rendering

See also:

renderFullscreenQuad

5.43.4.8 SceneNode* OgreRenderingRun::fullScreenQuadNode [static, protected, inherited]

fulls screen quad SceneNode used in full screen quad rendering

See also:

renderFullscreenQuad

5.43.4.9 SpriteSet* OgreRenderingRun::pixelSprites [protected, inherited]

SpriteSet used in pixel sprite rendering.

See also:

renderPixelSprites

5.43.4.10 BillboardSet* OgreRenderingRun::pixelSprites [protected, inherited]

SpriteSet used in pixel sprite rendering.

See also:

renderPixelSprites

5.43.4.11 String OgreRenderingRun::spriteSetName [protected, inherited]

unique name of the SpriteSet used in pixel sprite rendering

See also:

renderPixelSprites

5.43.4.12 Entity* OgreRenderingRun::fullscreenGrid [protected, inherited]

Entity used in fullscreen grid rendering.

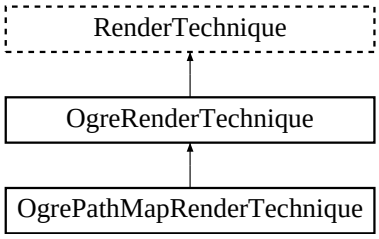
See also:

renderPixelGrid

5.44 OgrePathMapRenderTechnique Class Reference

A technique that defines that the rendering of the object will use the path map technique.

Inheritance diagram for OgrePathMapRenderTechnique::



Public Member Functions

- OgrePathMapRenderTechnique (Pass *pass, OgreRenderable *parentRenderable, OgreTechniqueGroup *parentTechniqueGroup)

  Constructor.

- ~OgrePathMapRenderTechnique ()

  Destructor.

- virtual void update (unsigned long frameNum)

  Updates the resources in the given frame.

- virtual OgreRenderTechnique * asOgreRenderTechnique ()

  Conversion to OgreRenderTechnique.

- virtual void runChanged (RenderingRunType runType, RenderingRun *run)

  Called after one of he shared runs changes.

- virtual void runUpdated (RenderingRunType runType, RenderingRun *run)

  Called after one of he shared runs updates.

- ElementaryRenderable * getParentRenderable ()

  Retrieves the renderable this technique operates on.

Protected Member Functions

- void createWeightIndexTexture ()

  create a weight index lookup map

Protected Attributes

- Pass * pathMapPass
  the new pass created by this technique
- PathMapClusters * clusters
  the PathMapClusters structure that belongs to the subentity renderable
- Texture * weightIndexTexture
  the weight index lookup map created by this technique
- OgreRenderable * parentOgreRenderable
  a OgreRenderable pointer to the renderable this technique operates on.
- OgreTechniqueGroup * parentOgreTechniqueGroup
  a OgreTechniqueGroup pointer to the TechniqueGroup this technique is attached to.
- Pass * pass
  a pointer to the pass this technique operates on.
- ElementaryRenderable * parentRenderable
  The renderable this technique operates on.
- TechniqueGroup * parentTechniqueGroup
  The TechniqueGroup this RenderedTechnique is attached to.
- SharedRuns * sharedRuns
  The SharedRuns this RenderedTechnique is attached to.

5.44.1 Detailed Description

A technique that defines that the rendering of the object will use the path map technique.

This rendering technique can add indirect lighting to the scene.

5.44.2 Constructor & Destructor Documentation

5.44.2.1 OgrePathMapRenderTechnique::OgrePathMapRenderTechnique (Pass * *pass*, OgreRenderable * *parentRenderable*, OgreTechniqueGroup * *parentTechniqueGroup*)

Constructor.

Parameters:

pass the pass after which shadowing passes should be added

parentRenderable the object to operate on

parentTechniqueGroup the TechniqueGroup this RenderedTechnique is attached to

5.44.3 Member Function Documentation

5.44.3.1 void OgrePathMapRenderTechnique::update (unsigned long *frameNum*) [virtual]

Updates the resources in the given frame.

A RenderTechnique is usually need some resources from several runs, so these runs will be updated.

Parameters:

frameNum the actual framenumber

Reimplemented from RenderTechnique.

5.44.3.2 virtual OgreRenderTechnique* OgreRenderTechnique::asOgreRenderTechnique () [inline, virtual, inherited]

Conversion to OgreRenderTechnique.

This function is needed because of virtual inheritance.

Reimplemented from RenderTechnique.

5.44.3.3 virtual void RenderTechnique::runChanged (RenderingRunType *runType*, RenderingRun * *run*) [inline, virtual, inherited]

Called after one of he shared runs changes.

Parameters:

runType enum describing the type of the changed run

run pointer to the changed RenderingRun

Reimplemented in CausticCasterRenderTechnique, ColorCubeMapRenderTechnique, ConvolvedCubeMapRenderTechnique, CubeMapRenderTechnique, DistanceCubeMapRenderTechnique, HierarchicalParticleSystemTechnique, and IllumVolumeRenderTechnique.

5.44.3.4 virtual void RenderTechnique::runUpdated (RenderingRunType *runType*, RenderingRun * *run*) [inline, virtual, inherited]

Called after one of he shared runs updates.

Parameters:

runType enum describing the type of the updated run

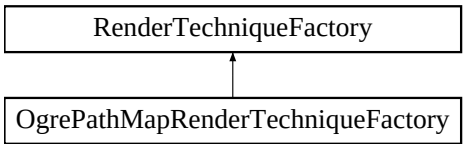
run pointer to the updated RenderingRun

Reimplemented in CausticCasterRenderTechnique, ColorCubeMapRenderTechnique, CubeMapRenderTechnique, DistanceCubeMapRenderTechnique, HierarchicalParticleSystemTechnique, and IllumVolumeRenderTechnique.

5.45 OgrePathMapRenderTechniqueFactory Class Reference

RenderTechniqueFactory to create OgrePathMapRenderTechnique instances.

Inheritance diagram for OgrePathMapRenderTechniqueFactory::



Public Member Functions

- OgreRenderTechnique * createInstance (IllumTechniqueParams *params, Pass *pass, OgreRenderable *parentRenderable, OgreTechniqueGroup *parentTechniqueGroup)

  Creates a RenderTechnique of the factory type.

- bool isType (String type)

  Returns if this factory can create a RenderTechnique of the given type.

- virtual void parseParams (IllumTechniqueParams *params)

  parses parameters from the material file.

Protected Types

- typedef void(*) ILLUM_ATTRIBUTE_PARSER (String &params, RenderTechniqueFactory *factory)

  function for parsing RenderTechnique attributes

- typedef std::map< String, ILLUM_ATTRIBUTE_PARSER > AttribParserList

  Keyword-mapped attribute parsers.

Protected Attributes

- AttribParserList attributeParsers

  map of parser functions

- String typeName

  factoryname

5.45.1 Detailed Description

RenderTechniqueFactory to create OgrePathMapRenderTechnique instances.

5.45.2 Member Typedef Documentation

5.45.2.1 typedef void(∗) RenderTechniqueFactory::ILLUM_ATTRIBUTE_PARSER(String &params, RenderTechniqueFactory ∗factory) [protected, inherited]

function for parsing RenderTechnique attributes

Parameters:

params attribute value stored in a String

5.45.3 Member Function Documentation

5.45.3.1 OgreRenderTechnique ∗ OgrePathMapRenderTechniqueFactory::createInstance (IllumTechniqueParams ∗ *params*, Pass ∗ *pass*, OgreRenderable ∗ *parentRenderable*, OgreTechniqueGroup ∗ *parentTechniqueGroup*) [virtual]

Creates a RenderTechnique of the factory type.

Parameters:

params containes constructor parameters as NameValuePairList

pass the Pass to use in RenderTechnique constructor

pass the parentRenderable to pass to RenderTechnique constructor

pass the parentTechniqueGroup to pass to RenderTechnique constructor

Implements RenderTechniqueFactory.

5.45.3.2 bool RenderTechniqueFactory::isType (String *type*) [inline, inherited]

Returns if this factory can create a RenderTechnique of the given type.

Parameters:

type RenderTechnique type

5.45.3.3 void RenderTechniqueFactory::parseParams (IllumTechniqueParams * *params*) [virtual, inherited]

parses parameters from the material file.

The parsed parameters will be passed to the new RenderTechnique's constructor.

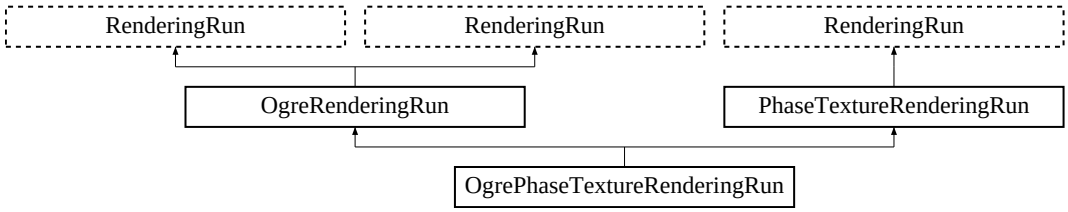
Parameters:

params pointer to the IllumTechniqueParams structure that was read from the material script and containes the parameters to be parsed.

5.46 OgrePhaseTextureRenderingRun Class Reference

PhaseTextureRenderingRun used in an OGRE environment.

Inheritance diagram for OgrePhaseTextureRenderingRun::



Public Member Functions

- OgrePhaseTextureRenderingRun (String name, unsigned int resolutionX, unsigned int resolutionY, String materialName)

  Constructor.

- String getPhaseTextureName ()

  returns the name of thew phase texture created by this run

- OgreRenderingRun * asOgreRenderingRun ()

  Conversion to OgreRenderRun.

- OgreRenderingRun * asOgreRenderingRun ()

  Conversion to OgreRenderRun.

- bool update (unsigned long frameNum)

  Calls updateFrame() if the run needs update according to its starting frame and update interval and has not been allready updated in this frame.

- virtual bool canJoin (RenderingRun *run)

  Returns true if two runs can be joined.

Protected Member Functions

- void updateFrame (unsigned long frameNum)

  This function does the actual update in a frame.

- void createPhaseTexture ()

  Creates the phase texture.

- Vector3 getCubeMapFaceDirection (unsigned char faceId)

  Returns a direction for a cubemap face id.

- Vector3 getCubeMapFaceDirection (unsigned char faceId)

  Returns a direction for a cubemap face id.

- Texture * createCubeRenderTexture (String name, const Vector3 position, unsigned int resolution=512, PixelFormat format=PF_FLOAT16_RGBA, int numMips=0, ColourValue clearColor=ColourValue::Black)

  Creates a cubemap texture.

- Texture * createCubeRenderTexture (String name, const Vector3 position, unsigned int resolution=512, PixelFormat format=PF_FLOAT16_RGBA, int numMips=0, ColourValue clearColor=ColourValue::Black)

  Creates a cubemap texture.

- void setMaterialForRenderables (String &materialName, RenderQueue *rq)

  Sets the given material for each Renderable in a RenderQueue.

- void setMaterialForRenderables (String &materialName, RenderQueue *rq, bool solidsonly=false)

  Sets the given material for each Renderable in a RenderQueue.

- void setMaterialForVisibles (String &materialName, Camera *cam, bool shadowcastersonly=false)

  Sets the given material for each Renderable visible from a given camera.

- void setMaterialForVisibles (String &materialName, Camera *cam, bool shadowcastersonly=false, bool solidsonly=false)

  Sets the given material for each Renderable visible from a given camera.

- void restoreMaterials ()

  Restores previously stored materials.

- void restoreMaterials ()

  Restores previously stored materials.

- void renderFullscreenQuad (String materialName, RenderTarget *target)

  Renderes a full screen quad on a given RendderTarget with a given material.

- void renderFullscreenQuad (String materialName, RenderTarget *target)

  Renderes a full screen quad on a given RendderTarget with a given material.

- void renderPixelSprites (String &materialName, RenderTarget *rt, int width, int height)

  Renderes sprites to pixels of the screen on a given RendderTarget with a given material.

- void renderPixelSprites (String &materialName, RenderTarget *rt, int width, int height)

  Renderes sprites to pixels of the screen on a given RendderTarget with a given material.

- void renderFullscreenGrid (String &materialName, RenderTarget *rt, int width, int height)

  Renders a grid onto the screen.

- virtual bool needUpdate (unsigned long frameNum)

  Returns if this run needs update.

Protected Attributes

- String materialName

  the name of the material to be used when rendering the depth shadow map

- String name

  the name of the phase texture that was created by this run

- Texture * phaseTexture

  a pointer to the phase texture that was created by this run

- Camera * phaseCamera

  a pointer to the camera that was used while rendering the phase texture

- std::map< Renderable *, String > visibleObjects

  map of Renderables which will be rendered with a given material

- std::map< Renderable *, String > visibleObjects

  map of Renderables which will be rendered with a given material

- SpriteSet * pixelSprites

  SpriteSet used in pixel sprite rendering.

- BillboardSet * pixelSprites

  SpriteSet used in pixel sprite rendering.

- String spriteSetName

  unique name of the SpriteSet used in pixel sprite rendering

- Entity * fullscreenGrid

  Entity used in fullscreen grid rendering.

- unsigned long lastupdated

  The number of the last frame this run was updated.

- unsigned long startFrame

  The number of the frame this run should be updated first.

- unsigned long updateInterval

  Refresh frequency in frames.

- unsigned int resolutionX

  width of the texture

- unsigned int resolutionY

  height of the texture

Static Protected Attributes

- static MovablePlane * fullScreenQuad

  fulls screen quad plane used in full screen quad rendering

- static MovablePlane * fullScreenQuad

  fulls screen quad plane used in full screen quad rendering

- static Entity * fullScreenQuadEntity

  fulls screen quad Entity used in full screen quad rendering

- static Entity * fullScreenQuadEntity

  fulls screen quad Entity used in full screen quad rendering

- static SceneNode * fullScreenQuadNode

  fulls screen quad SceneNode used in full screen quad rendering

- static SceneNode * fullScreenQuadNode

  fulls screen quad SceneNode used in full screen quad rendering

5.46.1 Detailed Description

PhaseTextureRenderingRun used in an OGRE environment.

5.46.2 Constructor & Destructor Documentation

5.46.2.1 OgrePhaseTextureRenderingRun::OgrePhaseTextureRenderingRun (String *name*, unsigned int *resolutionX*, unsigned int *resolutionY*, String *materialName*)

Constructor.

Parameters:

name the name of the phase texture to be created

resolutionX the resolution width of the phase texture

resolutionY the resolution height of the phase texture

materialName the name of the material to be used when rendering the phase texture

5.46.3 Member Function Documentation

5.46.3.1 void OgrePhaseTextureRenderingRun::updateFrame (unsigned long *frameNum*) [protected, virtual]

This function does the actual update in a frame.

Parameters:

frameNum current frame number

Implements PhaseTextureRenderingRun.

5.46.3.2 OgreRenderingRun* OgreRenderingRun::asOgreRenderingRun () [inline, virtual, inherited]

Conversion to OgreRenderRun.

This function is needed because of virtual inheritance.

Reimplemented from RenderingRun.

5.46.3.3 OgreRenderingRun* OgreRenderingRun::asOgreRenderingRun () [inline, virtual, inherited]

Conversion to OgreRenderRun.

This function is needed because of virtual inheritance.

Reimplemented from RenderingRun.

5.46.3.4 Vector3 OgreRenderingRun::getCubeMapFaceDirection (unsigned char *faceId*) [protected, inherited]

Returns a direction for a cubemap face id.

This is a helper function to retrieve the normal direction of a given cubemap face.

Parameters:

faceId the number of the face

5.46.3.5 Vector3 OgreRenderingRun::getCubeMapFaceDirection (unsigned char *faceId*) [protected, inherited]

Returns a direction for a cubemap face id.

This is a helper function to retrieve the normal direction of a given cubemap face.

Parameters:

faceId the number of the face

5.46.3.6 Texture * OgreRenderingRun::createCubeRenderTexture (String *name*, const Vector3 *position*, unsigned int *resolution* = 512, PixelFormat *format* = PF_FLOAT16_RGBA, int *numMips* = 0, ColourValue *clearColor* = ColourValue::Black) [protected, inherited]

Creates a cubemap texture.

This is a helper function to easily create a cubemap texture and automaticly attach viewports to each face so it can be used as a rendertarget.

Parameters:

name the name of the texture to be created

position the initial position of the cubemap

resolution the resolution of one cubemapface

format the pixel format of the cubemap

numMips the number of mipmap levels

clearColor initial color

5.46.3.7 Texture* OgreRenderingRun::createCubeRenderTexture (String *name*, const Vector3 *position*, unsigned int *resolution* = 512, PixelFormat *format* = PF_FLOAT16_RGBA, int *numMips* = 0, ColourValue *clearColor* = ColourValue::Black) [protected, inherited]

Creates a cubemap texture.

This is a helper function to easily create a cubemap texture and automaticly attach viewports to each face so it can be used as a rendertarget.

Parameters:

name the name of the texture to be created

position the initial position of the cubemap

resolution the resolution of one cubemapface

format the pixel format of the cubemap

numMips the number of mipmap levels

clearColor initial color

5.46.3.8 void OgreRenderingRun::setMaterialForRenderables (String & *materialName*, RenderQueue * *rq*) [protected, inherited]

Sets the given material for each Renderable in a RenderQueue.

This is a helper function to set a material to each element of a previously filled Renderque. The orginal material of the Renderables are stored so they can be restored later. The function also tells the current SceneManager not to search for visible objects, as we are going to use the given RenderQueue during the next rendering.

Parameters:

materialName the name of the material to set for the Renderables

rq pointer to the filled Renderqueue instance to set material for

5.46.3.9 void OgreRenderingRun::setMaterialForRenderables (String & *materialName*, RenderQueue * *rq*, bool *solidsonly* = false) [protected, inherited]

Sets the given material for each Renderable in a RenderQueue.

This is a helper function to set a material to each element of a previously filled Renderque. The orginal material of the Renderables are stored so they can be restored later. The function also tells the current SceneManager not to search for visible objects, as we are going to use the given RenderQueue during the next rendering.

Parameters:

materialName the name of the material to set for the Renderables

rq pointer to the filled Renderqueue instance to set material for

5.46.3.10 void OgreRenderingRun::setMaterialForVisibles (String & *materialName*, Camera * *cam*, bool *shadowcastersonly* = false) [protected, inherited]

Sets the given material for each Renderable visible from a given camera.

This helper function is similar to setMaterialForRenderables but it is also responsible for filling the RenderQueue. First the RenderQueue of the current SceneManager fill be filled with the visible objects seen from the given camera. Then the required material will be set for each element of the RenderQueue. The orginal material of the Renderables are stored so they can be restored later. The function also tells the current SceneManager not to search for visible objects, as we are going to use the filled RenderQueue during the next rendering.

Parameters:

materialName the name of the material to set for the Renderables

cam pointer to the camera from which visible objects should be searched

shadowcastersonly flag to search for only shadow casters

5.46.3.11 void OgreRenderingRun::setMaterialForVisibles (String & *materialName*, Camera * *cam*, bool *shadowcastersonly* = false, bool *solidsonly* = false) [protected, inherited]

Sets the given material for each Renderable visible from a given camera.

This helper function is similar to setMaterialForRenderables but it is also responsible for filling the RenderQueue. First the RenderQueue of the current SceneManager fill be filled with the visible objects seen from the given camera. Then the required material will be set for each element of the RenderQueue. The orginal

material of the Renderables are stored so they can be restored later. The function also tells the current SceneManager not to search for visible objects, as we are going to use the filled RenderQueue during the next rendering.

Parameters:

materialName the name of the material to set for the Renderables

cam pointer to the camera from which visible objects should be searched

shadowcastersonly flag to search for only shadow casters

5.46.3.12 void OgreRenderingRun::restoreMaterials () [protected, inherited]

Restores previously stored materials.

This helper function is typically used after a setMaterialForRenderables or setMaterialForVisibles call and a rendering process to restore the original material settings. The function also tells the current SceneManager to search for visible objects, as this is the default behaviour of SceneManager.

5.46.3.13 void OgreRenderingRun::restoreMaterials () [protected, inherited]

Restores previously stored materials.

This helper function is typically used after a setMaterialForRenderables or setMaterialForVisibles call and a rendering process to restore the original material settings. The function also tells the current SceneManager to search for visible objects, as this is the default behaviour of SceneManager.

5.46.3.14 void OgreRenderingRun::renderFullscreenQuad (String *materialName*, RenderTarget * *target*) [protected, inherited]

Renderes a full screen quad on a given RendderTarget with a given material.

Parameters:

materialName the name of the material bind to the quad

target the RenderTarget the quad should be rendered on

5.46.3.15 void OgreRenderingRun::renderFullscreenQuad (String *materialName*, RenderTarget * *target*) [protected, inherited]

Renderes a full screen quad on a given RendderTarget with a given material.

Parameters:

materialName the name of the material bind to the quad

target the RenderTarget the quad should be rendered on

5.46.3.16 void OgreRenderingRun::renderPixelSprites (String & *materialName*, RenderTarget * *rt*, int *width*, int *height*) [protected, inherited]

Renderes sprites to pixels of the screen on a given RendderTarget with a given material.

Pixel sprites are pixel sized quads, placed on each pixel of the RenderTarget. The number of sprites not necessary corresponds to the resolution of the rendertarget. The pixel quads will evenly fill the rendertarget's area with sizes corresponding to the given resolution. We can render fewer or more pixel quads than the number of pixels the rendertarget has (eg.: in case of caustic cubemap generation).

Parameters:

materialName the name of the material bind to the pixel sprites

rt the RenderTarget the quads should be rendered on

width the desired resolution width of the sprites

height the desired resolution height of the sprites

5.46.3.17 void OgreRenderingRun::renderPixelSprites (String & *materialName*, RenderTarget * *rt*, int *width*, int *height*) [protected, inherited]

Renderes sprites to pixels of the screen on a given RendderTarget with a given material.

Pixel sprites are pixel sized quads, placed on each pixel of the RenderTarget. The number of sprites not necessary corresponds to the resolution of the rendertarget. The pixel quads will evenly fill the rendertarget's area with sizes corresponding to the given resolution. We can render fewer or more pixel quads than the number of pixels the rendertarget has (eg.: in case of caustic cubemap generation).

Parameters:

materialName the name of the material bind to the pixel sprites

rt the RenderTarget the quads should be rendered on

width the desired resolution width of the sprites

height the desired resolution height of the sprites

5.46.3.18 void OgreRenderingRun::renderFullscreenGrid (String & *materialName*, RenderTarget * *rt*, int *width*, int *height*) [protected, inherited]

Renders a grid onto the screen.

Parameters:

rt the RenderTarget the grid should be rendered on

width the desired horizontal resolution of the grid

height the desired vertical resolution of the grid

5.46.3.19 virtual bool RenderingRun::canJoin (RenderingRun * *run*) [inline, virtual, inherited]

Returns true if two runs can be joined.

In some cases special requirements should stand to join two runs (even if they have the same type). Eg.: two caustic cube map generation technique should only be joined if they use the same material when rendering the caustic cubemap.

Reimplemented in OgreCausticCubeMapRenderingRun, OgreChildPSystemRenderingRun, and OgrePhotonMapRenderingRun.

5.46.3.20 virtual bool RenderingRun::needUpdate (unsigned long *frameNum*) [inline, protected, virtual, inherited]

Returns if this run needs update.

This tipically depends on the upate interval and the starting frame number.

Parameters:

frameNum current frame number

Reimplemented in OgreFocusingMapRenderingRun, and OgrePMEntryPointMapRenderingRun.

5.46.4 Member Data Documentation

5.46.4.1 std::map<Renderable*, String> OgreRenderingRun::visibleObjects [protected, inherited]

map of Renderables which will be rendered with a given material

The String stores the original material name that will be restored after rendering.

See also:

setMaterialForVisibles
setMaterialForRenderables
restoreMaterials

5.46.4.2 std::map<Renderable*, String> OgreRenderingRun::visibleObjects [protected, inherited]

map of Renderables which will be rendered with a given material

The String stores the original material name that will be restored after rendering.

See also:

setMaterialForVisibles

setMaterialForRenderables
restoreMaterials

5.46.4.3 MovablePlane * OgreRenderingRun::fullScreenQuad [static, protected, inherited]

fulls screen quad plane used in full screen quad rendering

See also:

renderFullscreenQuad

5.46.4.4 MovablePlane* OgreRenderingRun::fullScreenQuad [static, protected, inherited]

fulls screen quad plane used in full screen quad rendering

See also:

renderFullscreenQuad

5.46.4.5 Entity * OgreRenderingRun::fullScreenQuadEntity [static, protected, inherited]

fulls screen quad Entity used in full screen quad rendering

See also:

renderFullscreenQuad

5.46.4.6 Entity* OgreRenderingRun::fullScreenQuadEntity [static, protected, inherited]

fulls screen quad Entity used in full screen quad rendering

See also:

renderFullscreenQuad

5.46.4.7 SceneNode * OgreRenderingRun::fullScreenQuadNode [static, protected, inherited]

fulls screen quad SceneNode used in full screen quad rendering

See also:

renderFullscreenQuad

5.46.4.8 SceneNode* OgreRenderingRun::fullScreenQuadNode [static, protected, inherited]

fulls screen quad SceneNode used in full screen quad rendering

See also:

renderFullscreenQuad

5.46.4.9 SpriteSet* OgreRenderingRun::pixelSprites [protected, inherited]

SpriteSet used in pixel sprite rendering.

See also:

renderPixelSprites

5.46.4.10 BillboardSet* OgreRenderingRun::pixelSprites [protected, inherited]

SpriteSet used in pixel sprite rendering.

See also:

renderPixelSprites

5.46.4.11 String OgreRenderingRun::spriteSetName [protected, inherited]

unique name of the SpriteSet used in pixel sprite rendering

See also:

renderPixelSprites

5.46.4.12 Entity* OgreRenderingRun::fullscreenGrid [protected, inherited]

Entity used in fullscreen grid rendering.

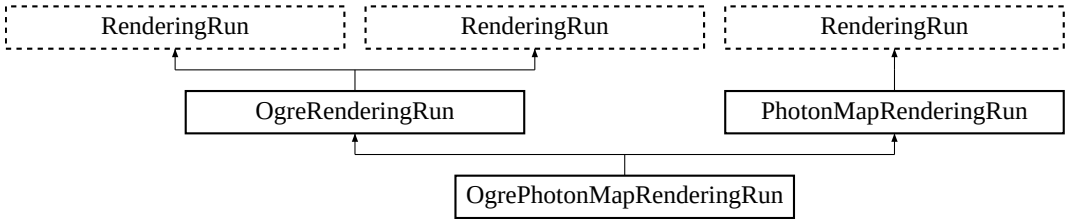
See also:

renderPixelGrid

5.47 OgrePhotonMapRenderingRun Class Reference

PhotonMapRenderingRun used in an OGRE environment.

Inheritance diagram for OgrePhotonMapRenderingRun::



Public Member Functions

- OgrePhotonMapRenderingRun (OgreSharedRuns *sharedRuns, String name, unsigned long startFrame, unsigned long updateInterval, unsigned int resolution, String materialName, bool useDistance)

  Constructor.

- String getPhotonMapTextureName ()

  returns the name of the resulting photon hit map texture

- void refreshLight ()

  Refreshes light camera matrices, called in each update.

- bool canJoin (RenderingRun *run)

  Returns true if two runs can be joined.

- OgreRenderingRun * asOgreRenderingRun ()

  Conversion to OgreRenderRun.

- OgreRenderingRun * asOgreRenderingRun ()

  Conversion to OgreRenderRun.

- bool update (unsigned long frameNum)

  Calls updateFrame() if the run needs update according to its starting frame and update interval and has not been allready updated in this frame.

Protected Member Functions

- void updateFrame (unsigned long frameNum)

  This function does the actual update in a frame.

- void createPhotonMap ()

  Creates a photon hit map.

- Vector3 getCubeMapFaceDirection (unsigned char faceId)

  Returns a direction for a cubemap face id.

- Vector3 getCubeMapFaceDirection (unsigned char faceId)

  Returns a direction for a cubemap face id.

- Texture * createCubeRenderTexture (String name, const Vector3 position, unsigned int resolution=512, PixelFormat format=PF_FLOAT16_RGBA, int numMips=0, ColourValue clearColor=ColourValue::Black)

  Creates a cubemap texture.

- Texture * createCubeRenderTexture (String name, const Vector3 position, unsigned int resolution=512, PixelFormat format=PF_FLOAT16_RGBA, int numMips=0, ColourValue clearColor=ColourValue::Black)

  Creates a cubemap texture.

- void setMaterialForRenderables (String &materialName, RenderQueue *rq)

  Sets the given material for each Renderable in a RenderQueue.

- void setMaterialForRenderables (String &materialName, RenderQueue *rq, bool solidsonly=false)

  Sets the given material for each Renderable in a RenderQueue.

- void setMaterialForVisibles (String &materialName, Camera *cam, bool shadowcastersonly=false)

  Sets the given material for each Renderable visible from a given camera.

- void setMaterialForVisibles (String &materialName, Camera *cam, bool shadowcastersonly=false, bool solidsonly=false)

  Sets the given material for each Renderable visible from a given camera.

- void restoreMaterials ()

  Restores previously stored materials.

- void restoreMaterials ()

  Restores previously stored materials.

- void renderFullscreenQuad (String materialName, RenderTarget *target)

  Renderes a full screen quad on a given RendderTarget with a given material.

- void renderFullscreenQuad (String materialName, RenderTarget *target)

  Renderes a full screen quad on a given RendderTarget with a given material.

- void renderPixelSprites (String &materialName, RenderTarget *rt, int width, int height)

  Renderes sprites to pixels of the screen on a given RendderTarget with a given material.

- void renderPixelSprites (String &materialName, RenderTarget *rt, int width, int height)

  Renderes sprites to pixels of the screen on a given RendderTarget with a given material.

- void renderFullscreenGrid (String &materialName, RenderTarget *rt, int width, int height)

  Renders a grid onto the screen.

- virtual bool needUpdate (unsigned long frameNum)

  Returns if this run needs update.

Protected Attributes

- String materialName

  the name of the material should be used when rendering the photon hit map

- Light * light

  pointer to the nearest light source from the caster object

- Camera * photonMapCamera

  the camera used while rendering the photon hit map

- bool useDistance

  tells if a distance cubemap impostor should be used in photon hit calculation (recommended)

- OgreSharedRuns * sharedRuns

  a pointer to the OgreSharedRuns this run belongs to

- String name

  the name of the photon map texture that was created by this run

- Texture * photonMapTexture

  a pointer to the photon map texture that was created by this run

- unsigned int resolution

  the resolution of the photonmap texture that was created by this run

- std::map< Renderable *, String > visibleObjects

  map of Renderables which will be rendered with a given material

- std::map< Renderable *, String > visibleObjects

  map of Renderables which will be rendered with a given material

- SpriteSet * pixelSprites

  SpriteSet used in pixel sprite rendering.

- BillboardSet * pixelSprites

  SpriteSet used in pixel sprite rendering.

- String spriteSetName

  unique name of the SpriteSet used in pixel sprite rendering

- Entity * fullscreenGrid

  Entity used in fullscreen grid rendering.

- unsigned long lastupdated

  The number of the last frame this run was updated.

- unsigned long startFrame

  The number of the frame this run should be updated first.

- unsigned long updateInterval

  Refresh frequency in frames.

Static Protected Attributes

- static MovablePlane * fullScreenQuad

  fulls screen quad plane used in full screen quad rendering

- static MovablePlane * fullScreenQuad

  fulls screen quad plane used in full screen quad rendering

- static Entity * fullScreenQuadEntity

  fulls screen quad Entity used in full screen quad rendering

- static Entity * fullScreenQuadEntity

  fulls screen quad Entity used in full screen quad rendering

- static SceneNode * fullScreenQuadNode

  fulls screen quad SceneNode used in full screen quad rendering

- static SceneNode * fullScreenQuadNode

  fulls screen quad SceneNode used in full screen quad rendering

5.47.1 Detailed Description

PhotonMapRenderingRun used in an OGRE environment.

5.47.2 Constructor & Destructor Documentation

5.47.2.1 OgrePhotonMapRenderingRun::OgrePhotonMapRenderingRun (OgreSharedRuns * *sharedRuns*, String *name*, unsigned long *startFrame*, unsigned long *updateInterval*, unsigned int *resolution*, String *materialName*, bool *useDistance*)

Constructor.

Parameters:

sharedRuns a pointer to the OgreSharedRuns this run belongs to

name the name of the texture to be created

startFrame adds an offset to the current frame number to help evenly distribute updates between frames

updateInterval update frequency

resolution photon map resolution

materialName the name of the material should be used when rendering the photon hit map

useDistance tells if a distance cubemap impostor should be used in photon hit calculation (recommended)

5.47.3 Member Function Documentation

5.47.3.1 void OgrePhotonMapRenderingRun::refreshLight ()

Refreshes light camera matrices, called in each update.

TODO: search nearest light, set light params

5.47.3.2 bool OgrePhotonMapRenderingRun::canJoin (RenderingRun * *run*) [inline, virtual]

Returns true if two runs can be joined.

In some cases special requirements should stand to join two runs (even if they have the same type). Eg.: two caustic cube map generation technique should only be joined if they use the same material when rendering the caustic cubemap.

Reimplemented from RenderingRun.

5.47.3.3 void OgrePhotonMapRenderingRun::updateFrame (unsigned long *frameNum*) [protected, virtual]

This function does the actual update in a frame.

Parameters:

frameNum current frame number

Implements PhotonMapRenderingRun.

5.47.3.4 OgreRenderingRun* OgreRenderingRun::asOgreRenderingRun () [inline, virtual, inherited]

Conversion to OgreRenderRun.

This function is needed because of virtual inheritance.

Reimplemented from RenderingRun.

5.47.3.5 OgreRenderingRun* OgreRenderingRun::asOgreRenderingRun () [inline, virtual, inherited]

Conversion to OgreRenderRun.

This function is needed because of virtual inheritance.

Reimplemented from RenderingRun.

5.47.3.6 Vector3 OgreRenderingRun::getCubeMapFaceDirection (unsigned char *faceId*) [protected, inherited]

Returns a direction for a cubemap face id.

This is a helper function to retrieve the normal direction of a given cubemap face.

Parameters:

faceId the number of the face

5.47.3.7 Vector3 OgreRenderingRun::getCubeMapFaceDirection (unsigned char *faceId*) [protected, inherited]

Returns a direction for a cubemap face id.

This is a helper function to retrieve the normal direction of a given cubemap face.

Parameters:

faceId the number of the face

5.47.3.8 Texture * OgreRenderingRun::createCubeRenderTexture (String *name*, const Vector3 *position*, unsigned int *resolution* = 512, PixelFormat *format* = PF_FLOAT16_RGBA, int *numMips* = 0, ColourValue *clearColor* = ColourValue::Black) [protected, inherited]

Creates a cubemap texture.

This is a helper function to easily create a cubemap texture and automaticly attach viewports to each face so it can be used as a rendertarget.

Parameters:

name the name of the texture to be created

position the initial position of the cubemap

resolution the resolution of one cubemapface

format the pixel format of the cubemap

numMips the number of mipmap levels

clearColor initial color

5.47.3.9 Texture* OgreRenderingRun::createCubeRenderTexture (String *name*, const Vector3 *position*, unsigned int *resolution* = 512, PixelFormat *format* = PF_FLOAT16_RGBA, int *numMips* = 0, ColourValue *clearColor* = ColourValue::Black) [protected, inherited]

Creates a cubemap texture.

This is a helper function to easily create a cubemap texture and automaticly attach viewports to each face so it can be used as a rendertarget.

Parameters:

name the name of the texture to be created

position the initial position of the cubemap

resolution the resolution of one cubemapface

format the pixel format of the cubemap

numMips the number of mipmap levels

clearColor initial color

5.47.3.10 void OgreRenderingRun::setMaterialForRenderables (String & *materialName*, RenderQueue * *rq*) [protected, inherited]

Sets the given material for each Renderable in a RenderQueue.

This is a helper function to set a material to each element of a previously filled Renderque. The orginal material of the Renderables are stored so they can be restored later. The function also tells the current SceneManager not to search for visible objects, as we are going to use the given RenderQueue during the next rendering.

Parameters:

materialName the name of the material to set for the Renderables

rq pointer to the filled Renderqueue instance to set material for

5.47.3.11 void OgreRenderingRun::setMaterialForRenderables (String & *materialName*, RenderQueue * *rq*, bool *solidsonly* = false) [protected, inherited]

Sets the given material for each Renderable in a RenderQueue.

This is a helper function to set a material to each element of a previously filled Renderque. The orginal material of the Renderables are stored so they can be restored later. The function also tells the current SceneManager not to search for visible objects, as we are going to use the given RenderQueue during the next rendering.

Parameters:

materialName the name of the material to set for the Renderables

rq pointer to the filled Renderqueue instance to set material for

5.47.3.12 void OgreRenderingRun::setMaterialForVisibles (String & *materialName*, Camera * *cam*, bool *shadowcastersonly* = false) [protected, inherited]

Sets the given material for each Renderable visible from a given camera.

This helper function is similar to setMaterialForRenderables but it is also responsible for filling the RenderQueue. First the RenderQueue of the current SceneManager fill be filled with the visible objects seen from the given camera. Then the required material will be set for each element of the RenderQueue. The orginal material of the Renderables are stored so they can be restored later. The function also tells the current SceneManager not to search for visible objects, as we are going to use the filled RenderQueue during the next rendering.

Parameters:

materialName the name of the material to set for the Renderables

cam pointer to the camera from which visible objects should be searched

shadowcastersonly flag to search for only shadow casters

5.47.3.13 void OgreRenderingRun::setMaterialForVisibles (String & *materialName*, Camera * *cam*, bool *shadowcastersonly* = false, bool *solidsonly* = false) [protected, inherited]

Sets the given material for each Renderable visible from a given camera.

This helper function is similar to setMaterialForRenderables but it is also responsible for filling the RenderQueue. First the RenderQueue of the current SceneManager fill be filled with the visible objects seen from the given camera. Then the required material will be set for each element of the RenderQueue. The orginal material of the Renderables are stored so they can be restored later. The function also tells the current SceneManager not to search for visible objects, as we are going to use the filled RenderQueue during the next rendering.

Parameters:

materialName the name of the material to set for the Renderables

cam pointer to the camera from which visible objects should be searched

shadowcastersonly flag to search for only shadow casters

5.47.3.14 void OgreRenderingRun::restoreMaterials () [protected, inherited]

Restores previously stored materials.

This helper function is typically used after a setMaterialForRenderables or setMaterialForVisibles call and a rendering process to restore the original material settings. The function also tells the current SceneManager to search for visible objects, as this is the default behaviour of SceneManager.

5.47.3.15 void OgreRenderingRun::restoreMaterials () [protected, inherited]

Restores previously stored materials.

This helper function is typically used after a setMaterialForRenderables or setMaterialForVisibles call and a rendering process to restore the original material settings. The function also tells the current SceneManager to search for visible objects, as this is the default behaviour of SceneManager.

5.47.3.16 void OgreRenderingRun::renderFullscreenQuad (String *materialName*, RenderTarget * *target*) [protected, inherited]

Renderes a full screen quad on a given RendderTarget with a given material.

Parameters:

materialName the name of the material bind to the quad

target the RenderTarget the quad should be rendered on

5.47.3.17 void OgreRenderingRun::renderFullscreenQuad (String *materialName*, RenderTarget * *target*) [protected, inherited]

Renderes a full screen quad on a given RendderTarget with a given material.

Parameters:

materialName the name of the material bind to the quad

target the RenderTarget the quad should be rendered on

5.47.3.18 void OgreRenderingRun::renderPixelSprites (String & *materialName*, RenderTarget * *rt*, int *width*, int *height*) [protected, inherited]

Renderes sprites to pixels of the screen on a given RendderTarget with a given material.

Pixel sprites are pixel sized quads, placed on each pixel of the RenderTarget. The number of sprites not necessary corresponds to the resolution of the rendertarget. The pixel quads will evenly fill the rendertarget's area with sizes corresponding to the given resolution. We can render fewer or more pixel quads than the number of pixels the rendertarget has (eg.: in case of caustic cubemap generation).

Parameters:

materialName the name of the material bind to the pixel sprites

rt the RenderTarget the quads should be rendered on

width the desired resolution width of the sprites

height the desired resolution height of the sprites

5.47.3.19 void OgreRenderingRun::renderPixelSprites (String & *materialName*, RenderTarget * *rt*, int *width*, int *height*) [protected, inherited]

Renderes sprites to pixels of the screen on a given RendderTarget with a given material.

Pixel sprites are pixel sized quads, placed on each pixel of the RenderTarget. The number of sprites not necessary corresponds to the resolution of the rendertarget. The pixel quads will evenly fill the rendertarget's area with sizes corresponding to the given resolution. We can render fewer or more pixel quads than the number of pixels the rendertarget has (eg.: in case of caustic cubemap generation).

Parameters:

materialName the name of the material bind to the pixel sprites

rt the RenderTarget the quads should be rendered on

width the desired resolution width of the sprites

height the desired resolution height of the sprites

5.47.3.20 void OgreRenderingRun::renderFullscreenGrid (String & *materialName*, RenderTarget * *rt*, int *width*, int *height*) [protected, inherited]

Renders a grid onto the screen.

Parameters:

rt the RenderTarget the grid should be rendered on

width the desired horizontal resolution of the grid

height the desired vertical resolution of the grid

5.47.3.21 virtual bool RenderingRun::needUpdate (unsigned long *frameNum*) [inline, protected, virtual, inherited]

Returns if this run needs update.

This tipically depends on the upate interval and the starting frame number.

Parameters:

frameNum current frame number

Reimplemented in OgreFocusingMapRenderingRun, and OgrePMEntryPointMapRenderingRun.

5.47.4 Member Data Documentation

5.47.4.1 std::map<Renderable*, String> OgreRenderingRun::visibleObjects [protected, inherited]

map of Renderables which will be rendered with a given material

The String stores the original material name that will be restored after rendering.

See also:

setMaterialForVisibles
setMaterialForRenderables
restoreMaterials

5.47.4.2 std::map<Renderable*, String> OgreRenderingRun::visibleObjects [protected, inherited]

map of Renderables which will be rendered with a given material

The String stores the original material name that will be restored after rendering.

See also:

setMaterialForVisibles
setMaterialForRenderables
restoreMaterials

5.47.4.3 MovablePlane * OgreRenderingRun::fullScreenQuad [static, protected, inherited]

fulls screen quad plane used in full screen quad rendering

See also:

renderFullscreenQuad

5.47.4.4 MovablePlane* OgreRenderingRun::fullScreenQuad [static, protected, inherited]

fulls screen quad plane used in full screen quad rendering

See also:

renderFullscreenQuad

5.47.4.5 Entity * OgreRenderingRun::fullScreenQuadEntity [static, protected, inherited]

fulls screen quad Entity used in full screen quad rendering

See also:

renderFullscreenQuad

5.47.4.6 Entity* OgreRenderingRun::fullScreenQuadEntity [static, protected, inherited]

fulls screen quad Entity used in full screen quad rendering

See also:

renderFullscreenQuad

5.47.4.7 SceneNode * OgreRenderingRun::fullScreenQuadNode [static, protected, inherited]

fulls screen quad SceneNode used in full screen quad rendering

See also:

renderFullscreenQuad

5.47.4.8 SceneNode* OgreRenderingRun::fullScreenQuadNode [static, protected, inherited]

fulls screen quad SceneNode used in full screen quad rendering

See also:

renderFullscreenQuad

5.47.4.9 SpriteSet* OgreRenderingRun::pixelSprites [protected, inherited]

SpriteSet used in pixel sprite rendering.

See also:

renderPixelSprites

5.47.4.10 BillboardSet* OgreRenderingRun::pixelSprites [protected, inherited]

SpriteSet used in pixel sprite rendering.

See also:

renderPixelSprites

5.47.4.11 String OgreRenderingRun::spriteSetName [protected, inherited]

unique name of the SpriteSet used in pixel sprite rendering

See also:

renderPixelSprites

5.47.4.12 Entity* OgreRenderingRun::fullscreenGrid [protected, inherited]

Entity used in fullscreen grid rendering.

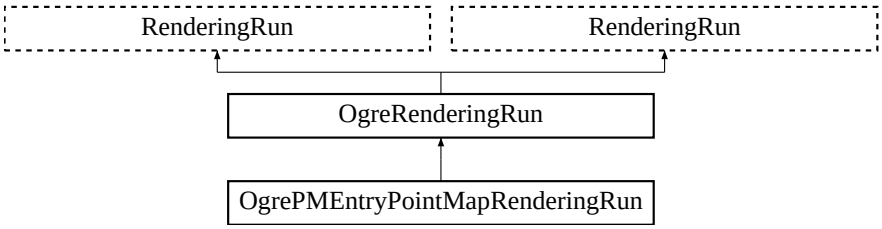
See also:

renderPixelGrid

5.48 OgrePMEntryPointMapRenderingRun Class Reference

Entry point map computing run.

Inheritance diagram for OgrePMEntryPointMapRenderingRun::



Public Member Functions

- OgrePMEntryPointMapRenderingRun (String name)
  Constructor.
- String getEntryPointTextureName ()
  returns the name of the entry point texture
- String getClusterLengthTextureName ()
  returns the name of the cluster length texture.
- OgreRenderingRun * asOgreRenderingRun ()
  Conversion to OgreRenderRun.
- OgreRenderingRun * asOgreRenderingRun ()
  Conversion to OgreRenderRun.
- bool update (unsigned long frameNum)
  Calls updateFrame() if the run needs update according to its starting frame and update interval and has not been allready updated in this frame.
- virtual bool canJoin (RenderingRun *run)
  Returns true if two runs can be joined.

Protected Member Functions

- void updateFrame (unsigned long frameNum)
  This function does the actual update in a frame.
- void createEntryPointMap ()
  Creates and fills the entry point and the cluster length texture.

- bool needUpdate (unsigned long frameNum)

  Returns if this run needs update.

- Vector3 getCubeMapFaceDirection (unsigned char faceId)

  Returns a direction for a cubemap face id.

- Vector3 getCubeMapFaceDirection (unsigned char faceId)

  Returns a direction for a cubemap face id.

- Texture * createCubeRenderTexture (String name, const Vector3 position, unsigned int resolution=512, PixelFormat format=PF_FLOAT16_RGBA, int numMips=0, ColourValue clearColor=ColourValue::Black)

  Creates a cubemap texture.

- Texture * createCubeRenderTexture (String name, const Vector3 position, unsigned int resolution=512, PixelFormat format=PF_FLOAT16_RGBA, int numMips=0, ColourValue clearColor=ColourValue::Black)

  Creates a cubemap texture.

- void setMaterialForRenderables (String &materialName, RenderQueue *rq)

  Sets the given material for each Renderable in a RenderQueue.

- void setMaterialForRenderables (String &materialName, RenderQueue *rq, bool solidsonly=false)

  Sets the given material for each Renderable in a RenderQueue.

- void setMaterialForVisibles (String &materialName, Camera *cam, bool shadowcastersonly=false)

  Sets the given material for each Renderable visible from a given camera.

- void setMaterialForVisibles (String &materialName, Camera *cam, bool shadowcastersonly=false, bool solidsonly=false)

  Sets the given material for each Renderable visible from a given camera.

- void restoreMaterials ()

  Restores previously stored materials.

- void restoreMaterials ()

  Restores previously stored materials.

- void renderFullscreenQuad (String materialName, RenderTarget *target)

  Renderes a full screen quad on a given RendderTarget with a given material.

- void renderFullscreenQuad (String materialName, RenderTarget *target)

  Renderes a full screen quad on a given RendderTarget with a given material.

- void renderPixelSprites (String &materialName, RenderTarget *rt, int width, int height)

  Renderes sprites to pixels of the screen on a given RendderTarget with a given material.

- void renderPixelSprites (String &materialName, RenderTarget *rt, int width, int height)

Renderes sprites to pixels of the screen on a given RendderTarget with a given material.

- void renderFullscreenGrid (String &materialName, RenderTarget *rt, int width, int height)

  Renders a grid onto the screen.

Protected Attributes

- String name

  the name of the entry point texture that was created by this run

- Texture * entryPointTexture

  a pointer to the entry point texture that was created by this run

- Texture * clusterLengthTexture

  a pointer to the cluster length texture that was created by this run

- std::map< Renderable *, String > visibleObjects

  map of Renderables which will be rendered with a given material

- std::map< Renderable *, String > visibleObjects

  map of Renderables which will be rendered with a given material

- SpriteSet * pixelSprites

  SpriteSet used in pixel sprite rendering.

- BillboardSet * pixelSprites

  SpriteSet used in pixel sprite rendering.

- String spriteSetName

  unique name of the SpriteSet used in pixel sprite rendering

- Entity * fullscreenGrid

  Entity used in fullscreen grid rendering.

- unsigned long lastupdated

  The number of the last frame this run was updated.

- unsigned long startFrame

  The number of the frame this run should be updated first.

- unsigned long updateInterval

  Refresh frequency in frames.

Static Protected Attributes

- static MovablePlane * fullScreenQuad

  fulls screen quad plane used in full screen quad rendering

- static MovablePlane * fullScreenQuad

  fulls screen quad plane used in full screen quad rendering

- static Entity * fullScreenQuadEntity

  fulls screen quad Entity used in full screen quad rendering

- static Entity * fullScreenQuadEntity

  fulls screen quad Entity used in full screen quad rendering

- static SceneNode * fullScreenQuadNode

  fulls screen quad SceneNode used in full screen quad rendering

- static SceneNode * fullScreenQuadNode

  fulls screen quad SceneNode used in full screen quad rendering

5.48.1 Detailed Description

Entry point map computing run.

This map is used in the pathmap technique. It stores entry point positions and normals. This texture will be used to determine the amount of incoming light to an entry point.

See also:

OgrePMWeightComputeRenderingRun

5.48.2 Constructor & Destructor Documentation

5.48.2.1 OgrePMEntryPointMapRenderingRun::OgrePMEntryPointMapRenderingRun (String *name*)

Constructor.

Parameters:

name the name of the entry point texture created ny this run

5.48.3 Member Function Documentation

5.48.3.1 String OgrePMEntryPointMapRenderingRun::getClusterLengthTextureName () [inline]

returns the name of the cluster length texture.

This texture stores the size (number of entripoints) of each entry point cluster.

5.48.3.2 void OgrePMEntryPointMapRenderingRun::updateFrame (unsigned long *frameNum*) [protected, virtual]

This function does the actual update in a frame.

Parameters:

frameNum current frame number

Reimplemented from RenderingRun.

5.48.3.3 bool OgrePMEntryPointMapRenderingRun::needUpdate (unsigned long *frameNum*) [inline, protected, virtual]

Returns if this run needs update.

This tipically depends on the upate interval and the starting frame number.

Parameters:

frameNum current frame number

Reimplemented from RenderingRun.

5.48.3.4 OgreRenderingRun* OgreRenderingRun::asOgreRenderingRun () [inline, virtual, inherited]

Conversion to OgreRenderRun.

This function is needed because of virtual inheritance.

Reimplemented from RenderingRun.

5.48.3.5 OgreRenderingRun* OgreRenderingRun::asOgreRenderingRun () [inline, virtual, inherited]

Conversion to OgreRenderRun.

This function is needed because of virtual inheritance.

Reimplemented from RenderingRun.

5.48.3.6 Vector3 OgreRenderingRun::getCubeMapFaceDirection (unsigned char *faceId*) [protected, inherited]

Returns a direction for a cubemap face id.

This is a helper function to retrieve the normal direction of a given cubemap face.

Parameters:

faceId the number of the face

5.48.3.7 Vector3 OgreRenderingRun::getCubeMapFaceDirection (unsigned char *faceId*) [protected, inherited]

Returns a direction for a cubemap face id.

This is a helper function to retrieve the normal direction of a given cubemap face.

Parameters:

faceId the number of the face

5.48.3.8 Texture * OgreRenderingRun::createCubeRenderTexture (String *name*, const Vector3 *position*, unsigned int *resolution* = 512, PixelFormat *format* = PF_FLOAT16_RGBA, int *numMips* = 0, ColourValue *clearColor* = ColourValue::Black) [protected, inherited]

Creates a cubemap texture.

This is a helper function to easily create a cubemap texture and automaticly attach viewports to each face so it can be used as a rendertarget.

Parameters:

name the name of the texture to be created

position the initial position of the cubemap

resolution the resolution of one cubemapface

format the pixel format of the cubemap

numMips the number of mipmap levels

clearColor initial color

5.48.3.9 Texture* OgreRenderingRun::createCubeRenderTexture (String *name*, const Vector3 *position*, unsigned int *resolution* = 512, PixelFormat *format* = PF_FLOAT16_RGBA, int *numMips* = 0, ColourValue *clearColor* = ColourValue::Black) [protected, inherited]

Creates a cubemap texture.

This is a helper function to easily create a cubemap texture and automaticly attach viewports to each face so it can be used as a rendertarget.

Parameters:

name the name of the texture to be created

position the initial position of the cubemap

resolution the resolution of one cubemapface

format the pixel format of the cubemap

numMips the number of mipmap levels

clearColor initial color

5.48.3.10 void OgreRenderingRun::setMaterialForRenderables (String & *materialName*, RenderQueue * *rq*) [protected, inherited]

Sets the given material for each Renderable in a RenderQueue.

This is a helper function to set a material to each element of a previously filled Renderque. The orginal material of the Renderables are stored so they can be restored later. The function also tells the current SceneManager not to search for visible objects, as we are going to use the given RenderQueue during the next rendering.

Parameters:

materialName the name of the material to set for the Renderables

rq pointer to the filled Renderqueue instance to set material for

5.48.3.11 void OgreRenderingRun::setMaterialForRenderables (String & *materialName*, RenderQueue * *rq*, bool *solidsonly* = false) [protected, inherited]

Sets the given material for each Renderable in a RenderQueue.

This is a helper function to set a material to each element of a previously filled Renderque. The orginal material of the Renderables are stored so they can be restored later. The function also tells the current SceneManager not to search for visible objects, as we are going to use the given RenderQueue during the next rendering.

Parameters:

materialName the name of the material to set for the Renderables

rq pointer to the filled Renderqueue instance to set material for

5.48.3.12 void OgreRenderingRun::setMaterialForVisibles (String & *materialName*, Camera * *cam*, bool *shadowcastersonly* = false) [protected, inherited]

Sets the given material for each Renderable visible from a given camera.

This helper function is similar to setMaterialForRenderables but it is also responsible for filling the RenderQueue. First the RenderQueue of the current SceneManager fill be filled with the visible objects seen from the given camera. Then the required material will be set for each element of the RenderQueue. The orginal material of the Renderables are stored so they can be restored later. The function also tells the current SceneManager not to search for visible objects, as we are going to use the filled RenderQueue during the next rendering.

Parameters:

materialName the name of the material to set for the Renderables

cam pointer to the camera from which visible objects should be searched

shadowcastersonly flag to search for only shadow casters

5.48.3.13 void OgreRenderingRun::setMaterialForVisibles (String & *materialName*, Camera * *cam*, bool *shadowcastersonly* = `false`, bool *solidsonly* = `false`) [protected, inherited]

Sets the given material for each Renderable visible from a given camera.

This helper function is similar to setMaterialForRenderables but it is also responsible for filling the RenderQueue. First the RenderQueue of the current SceneManager fill be filled with the visible objects seen from the given camera. Then the required material will be set for each element of the RenderQueue. The orginal material of the Renderables are stored so they can be restored later. The function also tells the current SceneManager not to search for visible objects, as we are going to use the filled RenderQueue during the next rendering.

Parameters:

materialName the name of the material to set for the Renderables

cam pointer to the camera from which visible objects should be searched

shadowcastersonly flag to search for only shadow casters

5.48.3.14 void OgreRenderingRun::restoreMaterials () [protected, inherited]

Restores previously stored materials.

This helper function is typically used after a setMaterialForRenderables or setMaterialForVisibles call and a rendering process to restore the original material settings. The function also tells the current SceneManager to search for visible objects, as this is the default behaviour of SceneManager.

5.48.3.15 void OgreRenderingRun::restoreMaterials () [protected, inherited]

Restores previously stored materials.

This helper function is typically used after a setMaterialForRenderables or setMaterialForVisibles call and a rendering process to restore the original material settings. The function also tells the current SceneManager to search for visible objects, as this is the default behaviour of SceneManager.

5.48.3.16 void OgreRenderingRun::renderFullscreenQuad (String *materialName*, RenderTarget * *target*) [protected, inherited]

Renderes a full screen quad on a given RendderTarget with a given material.

Parameters:

materialName the name of the material bind to the quad

target the RenderTarget the quad should be rendered on

5.48.3.17 void OgreRenderingRun::renderFullscreenQuad (String *materialName*, RenderTarget * *target*) [protected, inherited]

Renderes a full screen quad on a given RendderTarget with a given material.

Parameters:

materialName the name of the material bind to the quad

target the RenderTarget the quad should be rendered on

5.48.3.18 void OgreRenderingRun::renderPixelSprites (String & *materialName*, RenderTarget * *rt*, int *width*, int *height*) [protected, inherited]

Renderes sprites to pixels of the screen on a given RendderTarget with a given material.

Pixel sprites are pixel sized quads, placed on each pixel of the RenderTarget. The number of sprites not necessary corresponds to the resolution of the rendertarget. The pixel quads will evenly fill the rendertarget's area with sizes corresponding to the given resolution. We can render fewer or more pixel quads than the number of pixels the rendertarget has (eg.: in case of caustic cubemap generation).

Parameters:

materialName the name of the material bind to the pixel sprites

rt the RenderTarget the quads should be rendered on

width the desired resolution width of the sprites

height the desired resolution height of the sprites

5.48.3.19 void OgreRenderingRun::renderPixelSprites (String & *materialName*, RenderTarget * *rt*, int *width*, int *height*) [protected, inherited]

Renderes sprites to pixels of the screen on a given RendderTarget with a given material.

Pixel sprites are pixel sized quads, placed on each pixel of the RenderTarget. The number of sprites not necessary corresponds to the resolution of the rendertarget. The pixel quads will evenly fill the rendertarget's area with sizes corresponding to the given resolution. We can render fewer or more pixel quads than the number of pixels the rendertarget has (eg.: in case of caustic cubemap generation).

Parameters:

materialName the name of the material bind to the pixel sprites

rt the RenderTarget the quads should be rendered on

width the desired resolution width of the sprites

height the desired resolution height of the sprites

5.48.3.20 void OgreRenderingRun::renderFullscreenGrid (String & *materialName*, RenderTarget * *rt*, int *width*, int *height*) [protected, inherited]

Renders a grid onto the screen.

Parameters:

rt the RenderTarget the grid should be rendered on

width the desired horizontal resolution of the grid

height the desired vertical resolution of the grid

5.48.3.21 virtual bool RenderingRun::canJoin (RenderingRun * *run*) [inline, virtual, inherited]

Returns true if two runs can be joined.

In some cases special requirements should stand to join two runs (even if they have the same type). Eg.: two caustic cube map generation technique should only be joined if they use the same material when rendering the caustic cubemap.

Reimplemented in OgreCausticCubeMapRenderingRun, OgreChildPSystemRenderingRun, and OgrePhotonMapRenderingRun.

5.48.4 Member Data Documentation

5.48.4.1 Texture* OgrePMEntryPointMapRenderingRun::clusterLengthTexture [protected]

a pointer to the cluster length texture that was created by this run

This texture stores the size (number of entripoints) of each entry point cluster.

5.48.4.2 std::map<Renderable*, String> OgreRenderingRun::visibleObjects [protected, inherited]

map of Renderables which will be rendered with a given material

The String stores the original material name that will be restored after rendering.

See also:

setMaterialForVisibles
setMaterialForRenderables
restoreMaterials

5.48.4.3 std::map<Renderable*, String> OgreRenderingRun::visibleObjects [protected, inherited]

map of Renderables which will be rendered with a given material

The String stores the original material name that will be restored after rendering.

See also:

setMaterialForVisibles
setMaterialForRenderables
restoreMaterials

5.48.4.4 MovablePlane * OgreRenderingRun::fullScreenQuad [static, protected, inherited]

fulls screen quad plane used in full screen quad rendering

See also:

renderFullscreenQuad

5.48.4.5 MovablePlane* OgreRenderingRun::fullScreenQuad [static, protected, inherited]

fulls screen quad plane used in full screen quad rendering

See also:

renderFullscreenQuad

5.48.4.6 Entity * OgreRenderingRun::fullScreenQuadEntity [static, protected, inherited]

fulls screen quad Entity used in full screen quad rendering

See also:

renderFullscreenQuad

5.48.4.7 Entity* OgreRenderingRun::fullScreenQuadEntity [static, protected, inherited]

fulls screen quad Entity used in full screen quad rendering

See also:

renderFullscreenQuad

5.48.4.8 SceneNode * OgreRenderingRun::fullScreenQuadNode [static, protected, inherited]

fulls screen quad SceneNode used in full screen quad rendering

See also:

renderFullscreenQuad

5.48.4.9 SceneNode* OgreRenderingRun::fullScreenQuadNode [static, protected, inherited]

fulls screen quad SceneNode used in full screen quad rendering

See also:

renderFullscreenQuad

5.48.4.10 SpriteSet* OgreRenderingRun::pixelSprites [protected, inherited]

SpriteSet used in pixel sprite rendering.

See also:

renderPixelSprites

5.48.4.11 BillboardSet* OgreRenderingRun::pixelSprites [protected, inherited]

SpriteSet used in pixel sprite rendering.

See also:

renderPixelSprites

5.48.4.12 String OgreRenderingRun::spriteSetName [protected, inherited]

unique name of the SpriteSet used in pixel sprite rendering

See also:

renderPixelSprites

5.48.4.13 Entity* OgreRenderingRun::fullscreenGrid [protected, inherited]

Entity used in fullscreen grid rendering.

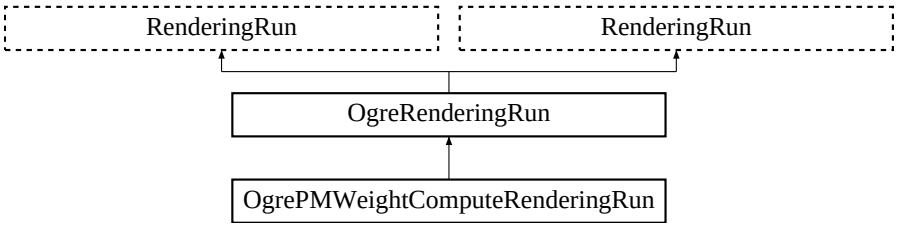
See also:

renderPixelGrid

5.49 OgrePMWeightComputeRenderingRun Class Reference

Weight computing rendering run.

Inheritance diagram for OgrePMWeightComputeRenderingRun::



Public Member Functions

- OgrePMWeightComputeRenderingRun (String name, String LightName)
  Constructor.
- String getPMWeightTetureName ()
  returns the name of the weight texture
- OgreRenderingRun * asOgreRenderingRun ()
  Conversion to OgreRenderRun.
- OgreRenderingRun * asOgreRenderingRun ()
  Conversion to OgreRenderRun.
- bool update (unsigned long frameNum)
  Calls updateFrame() if the run needs update according to its starting frame and update interval and has not been allready updated in this frame.
- virtual bool canJoin (RenderingRun *run)
  Returns true if two runs can be joined.

Protected Member Functions

- void updateFrame (unsigned long frameNum)
  This function does the actual update in a frame.
- void createWeightMap ()
  Creates the weight texture.
- Vector3 getCubeMapFaceDirection (unsigned char faceId)
  Returns a direction for a cubemap face id.

- Vector3 getCubeMapFaceDirection (unsigned char faceId)

  Returns a direction for a cubemap face id.

- Texture * createCubeRenderTexture (String name, const Vector3 position, unsigned int resolution=512, PixelFormat format=PF_FLOAT16_RGBA, int numMips=0, ColourValue clearColor=ColourValue::Black)

  Creates a cubemap texture.

- Texture * createCubeRenderTexture (String name, const Vector3 position, unsigned int resolution=512, PixelFormat format=PF_FLOAT16_RGBA, int numMips=0, ColourValue clearColor=ColourValue::Black)

  Creates a cubemap texture.

- void setMaterialForRenderables (String &materialName, RenderQueue *rq)

  Sets the given material for each Renderable in a RenderQueue.

- void setMaterialForRenderables (String &materialName, RenderQueue *rq, bool solidsonly=false)

  Sets the given material for each Renderable in a RenderQueue.

- void setMaterialForVisibles (String &materialName, Camera *cam, bool shadowcastersonly=false)

  Sets the given material for each Renderable visible from a given camera.

- void setMaterialForVisibles (String &materialName, Camera *cam, bool shadowcastersonly=false, bool solidsonly=false)

  Sets the given material for each Renderable visible from a given camera.

- void restoreMaterials ()

  Restores previously stored materials.

- void restoreMaterials ()

  Restores previously stored materials.

- void renderFullscreenQuad (String materialName, RenderTarget *target)

  Renderes a full screen quad on a given RendderTarget with a given material.

- void renderFullscreenQuad (String materialName, RenderTarget *target)

  Renderes a full screen quad on a given RendderTarget with a given material.

- void renderPixelSprites (String &materialName, RenderTarget *rt, int width, int height)

  Renderes sprites to pixels of the screen on a given RendderTarget with a given material.

- void renderPixelSprites (String &materialName, RenderTarget *rt, int width, int height)

  Renderes sprites to pixels of the screen on a given RendderTarget with a given material.

- void renderFullscreenGrid (String &materialName, RenderTarget *rt, int width, int height)

  Renders a grid onto the screen.

- virtual bool needUpdate (unsigned long frameNum)

  Returns if this run needs update.

Protected Attributes

- Matrix4 lightMatrix

  light view-projection matrix

- Matrix4 lightViewMatrix

  light view matrix

- String name

  the name of the weight texture that was created by this run

- Texture * weightTexture

  a pointer to the weight texture that was created by this run

- Texture * allWeightsTexture

  A pointer to a texture that stores weights for all the entry point.

- Light * light

  A pointer to the light source.

- std::map< Renderable *, String > visibleObjects

  map of Renderables which will be rendered with a given material

- std::map< Renderable *, String > visibleObjects

  map of Renderables which will be rendered with a given material

- SpriteSet * pixelSprites

  SpriteSet used in pixel sprite rendering.

- BillboardSet * pixelSprites

  SpriteSet used in pixel sprite rendering.

- String spriteSetName

  unique name of the SpriteSet used in pixel sprite rendering

- Entity * fullscreenGrid

  Entity used in fullscreen grid rendering.

- unsigned long lastupdated

  The number of the last frame this run was updated.

- unsigned long startFrame

  The number of the frame this run should be updated first.

- unsigned long updateInterval

  Refresh frequency in frames.

Static Protected Attributes

- static MovablePlane * fullScreenQuad

  fulls screen quad plane used in full screen quad rendering

- static MovablePlane * fullScreenQuad

  fulls screen quad plane used in full screen quad rendering

- static Entity * fullScreenQuadEntity

  fulls screen quad Entity used in full screen quad rendering

- static Entity * fullScreenQuadEntity

  fulls screen quad Entity used in full screen quad rendering

- static SceneNode * fullScreenQuadNode

  fulls screen quad SceneNode used in full screen quad rendering

- static SceneNode * fullScreenQuadNode

  fulls screen quad SceneNode used in full screen quad rendering

5.49.1 Detailed Description

Weight computing rendering run.

This run is used in the pathmap technique. It creates a texture for a light source that stores the weights that should be used for each cluster of entry points.

5.49.2 Constructor & Destructor Documentation

5.49.2.1 OgrePMWeightComputeRenderingRun::OgrePMWeightComputeRenderingRun (String *name*, String *LightName*)

Constructor.

Parameters:

name the name of the weight texture to be created

LightName the name of the light source to use

5.49.3 Member Function Documentation

5.49.3.1 void OgrePMWeightComputeRenderingRun::updateFrame (unsigned long *frameNum*) [protected, virtual]

This function does the actual update in a frame.

Parameters:

frameNum current frame number

Reimplemented from RenderingRun.

5.49.3.2 void OgrePMWeightComputeRenderingRun::createWeightMap () [inline, protected]

Creates the weight texture.

It also creates the texture storing all weights.

See also:

allWeightsTexture

5.49.3.3 OgreRenderingRun* OgreRenderingRun::asOgreRenderingRun () [inline, virtual, inherited]

Conversion to OgreRenderRun.

This function is needed because of virtual inheritance.

Reimplemented from RenderingRun.

5.49.3.4 OgreRenderingRun* OgreRenderingRun::asOgreRenderingRun () [inline, virtual, inherited]

Conversion to OgreRenderRun.

This function is needed because of virtual inheritance.

Reimplemented from RenderingRun.

5.49.3.5 Vector3 OgreRenderingRun::getCubeMapFaceDirection (unsigned char *faceId*) [protected, inherited]

Returns a direction for a cubemap face id.

This is a helper function to retrieve the normal direction of a given cubemap face.

Parameters:

faceId the number of the face

5.49.3.6 Vector3 OgreRenderingRun::getCubeMapFaceDirection (unsigned char *faceId*) [protected, inherited]

Returns a direction for a cubemap face id.

This is a helper function to retrieve the normal direction of a given cubemap face.

Parameters:

faceId the number of the face

5.49.3.7 Texture * OgreRenderingRun::createCubeRenderTexture (String *name*, const Vector3 *position*, unsigned int *resolution* = 512, PixelFormat *format* = PF_FLOAT16_RGBA, int *numMips* = 0, ColourValue *clearColor* = ColourValue::Black) [protected, inherited]

Creates a cubemap texture.

This is a helper function to easily create a cubemap texture and automaticly attach viewports to each face so it can be used as a rendertarget.

Parameters:

name the name of the texture to be created
position the initial position of the cubemap
resolution the resolution of one cubemapface
format the pixel format of the cubemap
numMips the number of mipmap levels
clearColor initial color

5.49.3.8 Texture* OgreRenderingRun::createCubeRenderTexture (String *name*, const Vector3 *position*, unsigned int *resolution* = 512, PixelFormat *format* = PF_FLOAT16_RGBA, int *numMips* = 0, ColourValue *clearColor* = ColourValue::Black) [protected, inherited]

Creates a cubemap texture.

This is a helper function to easily create a cubemap texture and automaticly attach viewports to each face so it can be used as a rendertarget.

Parameters:

name the name of the texture to be created
position the initial position of the cubemap
resolution the resolution of one cubemapface
format the pixel format of the cubemap
numMips the number of mipmap levels
clearColor initial color

5.49.3.9 void OgreRenderingRun::setMaterialForRenderables (String & *materialName*, RenderQueue * *rq*) [protected, inherited]

Sets the given material for each Renderable in a RenderQueue.

This is a helper function to set a material to each element of a previously filled Renderque. The orginal material of the Renderables are stored so they can be restored later. The function also tells the current SceneManager not to search for visible objects, as we are going to use the given RenderQueue during the next rendering.

Parameters:

materialName the name of the material to set for the Renderables

rq pointer to the filled Renderqueue instance to set material for

5.49.3.10 void OgreRenderingRun::setMaterialForRenderables (String & *materialName*, RenderQueue * *rq*, bool *solidsonly* = false) [protected, inherited]

Sets the given material for each Renderable in a RenderQueue.

This is a helper function to set a material to each element of a previously filled Renderque. The orginal material of the Renderables are stored so they can be restored later. The function also tells the current SceneManager not to search for visible objects, as we are going to use the given RenderQueue during the next rendering.

Parameters:

materialName the name of the material to set for the Renderables

rq pointer to the filled Renderqueue instance to set material for

5.49.3.11 void OgreRenderingRun::setMaterialForVisibles (String & *materialName*, Camera * *cam*, bool *shadowcastersonly* = false) [protected, inherited]

Sets the given material for each Renderable visible from a given camera.

This helper function is similar to setMaterialForRenderables but it is also responsible for filling the RenderQueue. First the RenderQueue of the current SceneManager fill be filled with the visible objects seen from the given camera. Then the required material will be set for each element of the RenderQueue. The orginal material of the Renderables are stored so they can be restored later. The function also tells the current SceneManager not to search for visible objects, as we are going to use the filled RenderQueue during the next rendering.

Parameters:

materialName the name of the material to set for the Renderables

cam pointer to the camera from which visible objects should be searched

shadowcastersonly flag to search for only shadow casters

5.49.3.12 void OgreRenderingRun::setMaterialForVisibles (String & *materialName*, Camera * *cam*, bool *shadowcastersonly* = false, bool *solidsonly* = false) [protected, inherited]

Sets the given material for each Renderable visible from a given camera.

This helper function is similar to setMaterialForRenderables but it is also responsible for filling the RenderQueue. First the RenderQueue of the current SceneManager fill be filled with the visible objects seen from the given camera. Then the required material will be set for each element of the RenderQueue. The orginal material of the Renderables are stored so they can be restored later. The function also tells the current SceneManager not to search for visible objects, as we are going to use the filled RenderQueue during the next rendering.

Parameters:

materialName the name of the material to set for the Renderables

cam pointer to the camera from which visible objects should be searched

shadowcastersonly flag to search for only shadow casters

5.49.3.13 void OgreRenderingRun::restoreMaterials () [protected, inherited]

Restores previously stored materials.

This helper function is typically used after a setMaterialForRenderables or setMaterialForVisibles call and a rendering process to restore the original material settings. The function also tells the current SceneManager to search for visible objects, as this is the default behaviour of SceneManager.

5.49.3.14 void OgreRenderingRun::restoreMaterials () [protected, inherited]

Restores previously stored materials.

This helper function is typically used after a setMaterialForRenderables or setMaterialForVisibles call and a rendering process to restore the original material settings. The function also tells the current SceneManager to search for visible objects, as this is the default behaviour of SceneManager.

5.49.3.15 void OgreRenderingRun::renderFullscreenQuad (String *materialName*, RenderTarget * *target*) [protected, inherited]

Renderes a full screen quad on a given RendderTarget with a given material.

Parameters:

materialName the name of the material bind to the quad

target the RenderTarget the quad should be rendered on

5.49.3.16 void OgreRenderingRun::renderFullscreenQuad (String *materialName*, RenderTarget * *target*) [protected, inherited]

Renderes a full screen quad on a given RendderTarget with a given material.

Parameters:

materialName the name of the material bind to the quad

target the RenderTarget the quad should be rendered on

5.49.3.17 void OgreRenderingRun::renderPixelSprites (String & *materialName*, RenderTarget * *rt*, int *width*, int *height*) [protected, inherited]

Renderes sprites to pixels of the screen on a given RendderTarget with a given material.

Pixel sprites are pixel sized quads, placed on each pixel of the RenderTarget. The number of sprites not necessary corresponds to the resolution of the rendertarget. The pixel quads will evenly fill the rendertarget's area with sizes corresponding to the given resolution. We can render fewer or more pixel quads than the number of pixels the rendertarget has (eg.: in case of caustic cubemap generation).

Parameters:

materialName the name of the material bind to the pixel sprites

rt the RenderTarget the quads should be rendered on

width the desired resolution width of the sprites

height the desired resolution height of the sprites

5.49.3.18 void OgreRenderingRun::renderPixelSprites (String & *materialName*, RenderTarget * *rt*, int *width*, int *height*) [protected, inherited]

Renderes sprites to pixels of the screen on a given RendderTarget with a given material.

Pixel sprites are pixel sized quads, placed on each pixel of the RenderTarget. The number of sprites not necessary corresponds to the resolution of the rendertarget. The pixel quads will evenly fill the rendertarget's area with sizes corresponding to the given resolution. We can render fewer or more pixel quads than the number of pixels the rendertarget has (eg.: in case of caustic cubemap generation).

Parameters:

materialName the name of the material bind to the pixel sprites

rt the RenderTarget the quads should be rendered on

width the desired resolution width of the sprites

height the desired resolution height of the sprites

5.49.3.19 void OgreRenderingRun::renderFullscreenGrid (String & *materialName*, RenderTarget * *rt*, int *width*, int *height*) [protected, inherited]

Renders a grid onto the screen.

Parameters:

rt the RenderTarget the grid should be rendered on

width the desired horizontal resolution of the grid

height the desired vertical resolution of the grid

5.49.3.20 virtual bool RenderingRun::canJoin (RenderingRun * *run*) [inline, virtual, inherited]

Returns true if two runs can be joined.

In some cases special requirements should stand to join two runs (even if they have the same type). Eg.: two caustic cube map generation technique should only be joined if they use the same material when rendering the caustic cubemap.

Reimplemented in OgreCausticCubeMapRenderingRun, OgreChildPSystemRenderingRun, and OgrePhotonMapRenderingRun.

5.49.3.21 virtual bool RenderingRun::needUpdate (unsigned long *frameNum*) [inline, protected, virtual, inherited]

Returns if this run needs update.

This tipically depends on the upate interval and the starting frame number.

Parameters:

frameNum current frame number

Reimplemented in OgreFocusingMapRenderingRun, and OgrePMEntryPointMapRenderingRun.

5.49.4 Member Data Documentation

5.49.4.1 Texture* OgrePMWeightComputeRenderingRun::allWeightsTexture [protected]

A pointer to a texture that stores weights for all the entry point.

This texture will be used to determine the weights of the clusters.

5.49.4.2 std::map<Renderable*, String> OgreRenderingRun::visibleObjects [protected, inherited]

map of Renderables which will be rendered with a given material

The String stores the original material name that will be restored after rendering.

See also:

setMaterialForVisibles
setMaterialForRenderables
restoreMaterials

5.49.4.3 std::map<Renderable*, String> OgreRenderingRun::visibleObjects [protected, inherited]

map of Renderables which will be rendered with a given material

The String stores the original material name that will be restored after rendering.

See also:

setMaterialForVisibles
setMaterialForRenderables
restoreMaterials

5.49.4.4 MovablePlane * OgreRenderingRun::fullScreenQuad [static, protected, inherited]

fulls screen quad plane used in full screen quad rendering

See also:

renderFullscreenQuad

5.49.4.5 MovablePlane* OgreRenderingRun::fullScreenQuad [static, protected, inherited]

fulls screen quad plane used in full screen quad rendering

See also:

renderFullscreenQuad

5.49.4.6 Entity * OgreRenderingRun::fullScreenQuadEntity [static, protected, inherited]

fulls screen quad Entity used in full screen quad rendering

See also:

renderFullscreenQuad

5.49.4.7 Entity* OgreRenderingRun::fullScreenQuadEntity [static, protected, inherited]

fulls screen quad Entity used in full screen quad rendering

See also:

renderFullscreenQuad

5.49.4.8 SceneNode * OgreRenderingRun::fullScreenQuadNode [static, protected, inherited]

fulls screen quad SceneNode used in full screen quad rendering

See also:

renderFullscreenQuad

5.49.4.9 SceneNode* OgreRenderingRun::fullScreenQuadNode [static, protected, inherited]

fulls screen quad SceneNode used in full screen quad rendering

See also:

renderFullscreenQuad

5.49.4.10 SpriteSet* OgreRenderingRun::pixelSprites [protected, inherited]

SpriteSet used in pixel sprite rendering.

See also:

renderPixelSprites

5.49.4.11 BillboardSet* OgreRenderingRun::pixelSprites [protected, inherited]

SpriteSet used in pixel sprite rendering.

See also:

renderPixelSprites

5.49.4.12 String OgreRenderingRun::spriteSetName [protected, inherited]

unique name of the SpriteSet used in pixel sprite rendering

See also:

renderPixelSprites

5.49.4.13 Entity* OgreRenderingRun::fullscreenGrid [protected, inherited]

Entity used in fullscreen grid rendering.

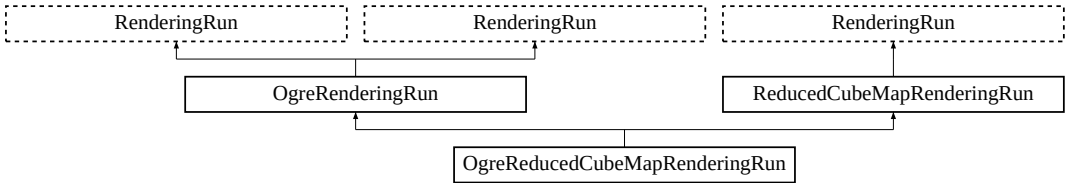
See also:

renderPixelGrid

5.50 OgreReducedCubeMapRenderingRun Class Reference

ReducedCubeMapRenderingRun used in an OGRE environment.

Inheritance diagram for OgreReducedCubeMapRenderingRun::



Public Member Functions

- OgreReducedCubeMapRenderingRun (OgreSharedRuns *sharedRuns, String name, unsigned long startFrame, unsigned long updateInterval, unsigned int resolution, bool useDistCalc=false, bool useFaceAngleCalc=false, float distTolerance=15, float angleTolerance=10, bool updateAllFace=false)

  Constructor.

- String & getReducedCubeMapTextureName ()

  returns the name of the resulting downsampled color cubemap texture

- OgreRenderingRun * asOgreRenderingRun ()

  Conversion to OgreRenderRun.

- OgreRenderingRun * asOgreRenderingRun ()

  Conversion to OgreRenderRun.

- bool update (unsigned long frameNum)

  Calls updateFrame() if the run needs update according to its starting frame and update interval and has not been allready updated in this frame.

- virtual bool canJoin (RenderingRun *run)

  Returns true if two runs can be joined.

Protected Member Functions

- void createReducedCubeMap ()

  Creates the reduced size cubemap texture.

- void updateCubeFace (int facenum)

  Updates one face of the cubemap.

- bool faceNeedsUpdate (int facenum)

Checks if a cubemap face needs to be updated.

- Vector3 getCubeMapFaceDirection (unsigned char faceId)

  Returns a direction for a cubemap face id.

- Vector3 getCubeMapFaceDirection (unsigned char faceId)

  Returns a direction for a cubemap face id.

- Texture * createCubeRenderTexture (String name, const Vector3 position, unsigned int resolution=512, PixelFormat format=PF_FLOAT16_RGBA, int numMips=0, ColourValue clearColor=ColourValue::Black)

  Creates a cubemap texture.

- Texture * createCubeRenderTexture (String name, const Vector3 position, unsigned int resolution=512, PixelFormat format=PF_FLOAT16_RGBA, int numMips=0, ColourValue clearColor=ColourValue::Black)

  Creates a cubemap texture.

- void setMaterialForRenderables (String &materialName, RenderQueue *rq)

  Sets the given material for each Renderable in a RenderQueue.

- void setMaterialForRenderables (String &materialName, RenderQueue *rq, bool solidsonly=false)

  Sets the given material for each Renderable in a RenderQueue.

- void setMaterialForVisibles (String &materialName, Camera *cam, bool shadowcastersonly=false)

  Sets the given material for each Renderable visible from a given camera.

- void setMaterialForVisibles (String &materialName, Camera *cam, bool shadowcastersonly=false, bool solidsonly=false)

  Sets the given material for each Renderable visible from a given camera.

- void restoreMaterials ()

  Restores previously stored materials.

- void restoreMaterials ()

  Restores previously stored materials.

- void renderFullscreenQuad (String materialName, RenderTarget *target)

  Renderes a full screen quad on a given RendderTarget with a given material.

- void renderFullscreenQuad (String materialName, RenderTarget *target)

  Renderes a full screen quad on a given RendderTarget with a given material.

- void renderPixelSprites (String &materialName, RenderTarget *rt, int width, int height)

  Renderes sprites to pixels of the screen on a given RendderTarget with a given material.

- void renderPixelSprites (String &materialName, RenderTarget *rt, int width, int height)

  Renderes sprites to pixels of the screen on a given RendderTarget with a given material.

- void renderFullscreenGrid (String &materialName, RenderTarget *rt, int width, int height)
  Renders a grid onto the screen.
- virtual bool needUpdate (unsigned long frameNum)
  Returns if this run needs update.
- virtual void updateFrame (unsigned long frameNum)
  This function does the actual update in a frame.
- virtual void updateFrame (unsigned long frameNum)
  This function does the actual update in a frame.

Protected Attributes

- OgreSharedRuns * sharedRuns
  a pointer to the OgreSharedRuns this run belongs to
- String name
  the name of the downsampled cubemap texture that was created by this run
- Texture * reducedCubemapTexture
  a pointer to the downsampled cubemap texture that was created by this run
- std::map< Renderable *, String > visibleObjects
  map of Renderables which will be rendered with a given material
- std::map< Renderable *, String > visibleObjects
  map of Renderables which will be rendered with a given material
- SpriteSet * pixelSprites
  SpriteSet used in pixel sprite rendering.
- BillboardSet * pixelSprites
  SpriteSet used in pixel sprite rendering.
- String spriteSetName
  unique name of the SpriteSet used in pixel sprite rendering
- Entity * fullscreenGrid
  Entity used in fullscreen grid rendering.
- unsigned long lastupdated
  The number of the last frame this run was updated.
- unsigned long startFrame
  The number of the frame this run should be updated first.
- unsigned long updateInterval

Refresh frequency in frames.

- bool updateAllFace

  defines if all cubemap faces should be updated in a frame or only one face per frame

- unsigned char currentFace

  the number of the face to be updated

- unsigned int resolution

  the resolution of the cubemap texture that was created by this run

- bool useDistCalc

  a flag to skip cube face update if object is far away or too small.

- bool useFaceAngleCalc

  a flag to skip cube face update the face is neglible.

- float distTolerance

  A value used in face skip test.

- float angleTolerance

  A value used in face skip test.

Static Protected Attributes

- static MovablePlane * fullScreenQuad

  fulls screen quad plane used in full screen quad rendering

- static MovablePlane * fullScreenQuad

  fulls screen quad plane used in full screen quad rendering

- static Entity * fullScreenQuadEntity

  fulls screen quad Entity used in full screen quad rendering

- static Entity * fullScreenQuadEntity

  fulls screen quad Entity used in full screen quad rendering

- static SceneNode * fullScreenQuadNode

  fulls screen quad SceneNode used in full screen quad rendering

- static SceneNode * fullScreenQuadNode

  fulls screen quad SceneNode used in full screen quad rendering

5.50.1 Detailed Description

ReducedCubeMapRenderingRun used in an OGRE environment.

5.50.2 Constructor & Destructor Documentation

5.50.2.1 OgreReducedCubeMapRenderingRun::OgreReducedCubeMapRenderingRun (OgreSharedRuns * *sharedRuns*, String *name*, unsigned long *startFrame*, unsigned long *updateInterval*, unsigned int *resolution*, bool *useDistCalc* = `false`, bool *useFaceAngleCalc* = `false`, float *distTolerance* = `15`, float *angleTolerance* = `10`, bool *updateAllFace* = `false`)

Constructor.

Parameters:

sharedRuns a pointer to the OgreSharedRuns this run belongs to

name the name of the cubemap texture to be created

startFrame adds an offset to the current frame number to help evenly distribute updates between frames

updateInterval update frequency

resolution cubemap resolution

useDistCalc flag to skip cube face update if object is far away

useFaceAngleCalc flag to skip cube face update if face is neglible

distTolerance distance tolerance used in face skip

angleTolerance angle tolerance used in face skip

updateAllFace defines if all cubemap faces should be updated in a frame or only one face per frame

5.50.3 Member Function Documentation

5.50.3.1 void OgreReducedCubeMapRenderingRun::updateCubeFace (int *facenum*) `[inline, protected, virtual]`

Updates one face of the cubemap.

Parameters:

facenum the number of the face to be updated

Implements ReducedCubeMapRenderingRun.

5.50.3.2 bool OgreReducedCubeMapRenderingRun::faceNeedsUpdate (int *facenum*) `[protected, virtual]`

Checks if a cubemap face needs to be updated.

If the object we are updating the cubemap for is far from the camera, or too small, or the given cubemapface does not have significant effect on the rendering the face can be skipped.

Parameters:

facenum the number of the face to be checked

Implements ReducedCubeMapRenderingRun.

5.50.3.3 OgreRenderingRun* OgreRenderingRun::asOgreRenderingRun () [inline, virtual, inherited]

Conversion to OgreRenderRun.

This function is needed because of virtual inheritance.

Reimplemented from RenderingRun.

5.50.3.4 OgreRenderingRun* OgreRenderingRun::asOgreRenderingRun () [inline, virtual, inherited]

Conversion to OgreRenderRun.

This function is needed because of virtual inheritance.

Reimplemented from RenderingRun.

5.50.3.5 Vector3 OgreRenderingRun::getCubeMapFaceDirection (unsigned char *faceId*) [protected, inherited]

Returns a direction for a cubemap face id.

This is a helper function to retrieve the normal direction of a given cubemap face.

Parameters:

faceId the number of the face

5.50.3.6 Vector3 OgreRenderingRun::getCubeMapFaceDirection (unsigned char *faceId*) [protected, inherited]

Returns a direction for a cubemap face id.

This is a helper function to retrieve the normal direction of a given cubemap face.

Parameters:

faceId the number of the face

5.50.3.7 Texture * OgreRenderingRun::createCubeRenderTexture (String *name*, const Vector3 *position*, unsigned int *resolution* = 512, PixelFormat *format* = PF_FLOAT16_RGBA, int *numMips* = 0, ColourValue *clearColor* = ColourValue::Black) [protected, inherited]

Creates a cubemap texture.

This is a helper function to easily create a cubemap texture and automaticly attach viewports to each face so it can be used as a rendertarget.

Parameters:

name the name of the texture to be created

position the initial position of the cubemap

resolution the resolution of one cubemapface

format the pixel format of the cubemap

numMips the number of mipmap levels

clearColor initial color

5.50.3.8 Texture* OgreRenderingRun::createCubeRenderTexture (String *name*, const Vector3 *position*, unsigned int *resolution* = `512`, PixelFormat *format* = `PF_FLOAT16_RGBA`, int *numMips* = `0`, ColourValue *clearColor* = `ColourValue::Black`) `[protected, inherited]`

Creates a cubemap texture.

This is a helper function to easily create a cubemap texture and automaticly attach viewports to each face so it can be used as a rendertarget.

Parameters:

name the name of the texture to be created

position the initial position of the cubemap

resolution the resolution of one cubemapface

format the pixel format of the cubemap

numMips the number of mipmap levels

clearColor initial color

5.50.3.9 void OgreRenderingRun::setMaterialForRenderables (String & *materialName*, RenderQueue * *rq*) `[protected, inherited]`

Sets the given material for each Renderable in a RenderQueue.

This is a helper function to set a material to each element of a previously filled Renderque. The orginal material of the Renderables are stored so they can be restored later. The function also tells the current SceneManager not to search for visible objects, as we are going to use the given RenderQueue during the next rendering.

Parameters:

materialName the name of the material to set for the Renderables

rq pointer to the filled Renderqueue instance to set material for

5.50.3.10 void OgreRenderingRun::setMaterialForRenderables (String & *materialName*, RenderQueue * *rq*, bool *solidsonly* = false) [protected, inherited]

Sets the given material for each Renderable in a RenderQueue.

This is a helper function to set a material to each element of a previously filled Renderque. The orginal material of the Renderables are stored so they can be restored later. The function also tells the current SceneManager not to search for visible objects, as we are going to use the given RenderQueue during the next rendering.

Parameters:

materialName the name of the material to set for the Renderables

rq pointer to the filled Renderqueue instance to set material for

5.50.3.11 void OgreRenderingRun::setMaterialForVisibles (String & *materialName*, Camera * *cam*, bool *shadowcastersonly* = false) [protected, inherited]

Sets the given material for each Renderable visible from a given camera.

This helper function is similar to setMaterialForRenderables but it is also responsible for filling the RenderQueue. First the RenderQueue of the current SceneManager fill be filled with the visible objects seen from the given camera. Then the required material will be set for each element of the RenderQueue. The orginal material of the Renderables are stored so they can be restored later. The function also tells the current SceneManager not to search for visible objects, as we are going to use the filled RenderQueue during the next rendering.

Parameters:

materialName the name of the material to set for the Renderables

cam pointer to the camera from which visible objects should be searched

shadowcastersonly flag to search for only shadow casters

5.50.3.12 void OgreRenderingRun::setMaterialForVisibles (String & *materialName*, Camera * *cam*, bool *shadowcastersonly* = false, bool *solidsonly* = false) [protected, inherited]

Sets the given material for each Renderable visible from a given camera.

This helper function is similar to setMaterialForRenderables but it is also responsible for filling the RenderQueue. First the RenderQueue of the current SceneManager fill be filled with the visible objects seen from the given camera. Then the required material will be set for each element of the RenderQueue. The orginal material of the Renderables are stored so they can be restored later. The function also tells the current SceneManager not to search for visible objects, as we are going to use the filled RenderQueue during the next rendering.

Parameters:

materialName the name of the material to set for the Renderables

cam pointer to the camera from which visible objects should be searched

shadowcastersonly flag to search for only shadow casters

5.50.3.13 void OgreRenderingRun::restoreMaterials () [protected, inherited]

Restores previously stored materials.

This helper function is typically used after a setMaterialForRenderables or setMaterialForVisibles call and a rendering process to restore the original material settings. The function also tells the current SceneManager to search for visible objects, as this is the default behaviour of SceneManager.

5.50.3.14 void OgreRenderingRun::restoreMaterials () [protected, inherited]

Restores previously stored materials.

This helper function is typically used after a setMaterialForRenderables or setMaterialForVisibles call and a rendering process to restore the original material settings. The function also tells the current SceneManager to search for visible objects, as this is the default behaviour of SceneManager.

5.50.3.15 void OgreRenderingRun::renderFullscreenQuad (String *materialName*, RenderTarget * *target*) [protected, inherited]

Renderes a full screen quad on a given RendderTarget with a given material.

Parameters:

materialName the name of the material bind to the quad

target the RenderTarget the quad should be rendered on

5.50.3.16 void OgreRenderingRun::renderFullscreenQuad (String *materialName*, RenderTarget * *target*) [protected, inherited]

Renderes a full screen quad on a given RendderTarget with a given material.

Parameters:

materialName the name of the material bind to the quad

target the RenderTarget the quad should be rendered on

5.50.3.17 void OgreRenderingRun::renderPixelSprites (String & *materialName*, RenderTarget * *rt*, int *width*, int *height*) [protected, inherited]

Renderes sprites to pixels of the screen on a given RendderTarget with a given material.

Pixel sprites are pixel sized quads, placed on each pixel of the RenderTarget. The number of sprites not necessary corresponds to the resolution of the rendertarget. The pixel quads will evenly fill the rendertarget's area with sizes corresponding to the given resolution. We can render fewer or more pixel quads than the number of pixels the rendertarget has (eg.: in case of caustic cubemap generation).

Parameters:

materialName the name of the material bind to the pixel sprites

rt the RenderTarget the quads should be rendered on

width the desired resolution width of the sprites

height the desired resolution height of the sprites

5.50.3.18 void OgreRenderingRun::renderPixelSprites (String & *materialName*, RenderTarget * *rt*, int *width*, int *height*) [protected, inherited]

Renderes sprites to pixels of the screen on a given RendderTarget with a given material.

Pixel sprites are pixel sized quads, placed on each pixel of the RenderTarget. The number of sprites not necessary corresponds to the resolution of the rendertarget. The pixel quads will evenly fill the rendertarget's area with sizes corresponding to the given resolution. We can render fewer or more pixel quads than the number of pixels the rendertarget has (eg.: in case of caustic cubemap generation).

Parameters:

materialName the name of the material bind to the pixel sprites

rt the RenderTarget the quads should be rendered on

width the desired resolution width of the sprites

height the desired resolution height of the sprites

5.50.3.19 void OgreRenderingRun::renderFullscreenGrid (String & *materialName*, RenderTarget * *rt*, int *width*, int *height*) [protected, inherited]

Renders a grid onto the screen.

Parameters:

rt the RenderTarget the grid should be rendered on

width the desired horizontal resolution of the grid

height the desired vertical resolution of the grid

5.50.3.20 virtual bool RenderingRun::canJoin (RenderingRun * *run*) [inline, virtual, inherited]

Returns true if two runs can be joined.

In some cases special requirements should stand to join two runs (even if they have the same type). Eg.: two caustic cube map generation technique should only be joined if they use the same material when rendering the caustic cubemap.

Reimplemented in OgreCausticCubeMapRenderingRun, OgreChildPSystemRenderingRun, and OgrePhotonMapRenderingRun.

5.50.3.21 virtual bool RenderingRun::needUpdate (unsigned long *frameNum*) [inline, protected, virtual, inherited]

Returns if this run needs update.

This tipically depends on the upate interval and the starting frame number.

Parameters:

frameNum current frame number

Reimplemented in OgreFocusingMapRenderingRun, and OgrePMEntryPointMapRenderingRun.

5.50.3.22 virtual void RenderingRun::updateFrame (unsigned long *frameNum*) [inline, protected, virtual, inherited]

This function does the actual update in a frame.

Parameters:

frameNum current frame number

Reimplemented in CausticCubeMapRenderingRun, ChildPsystemRenderingRun, CubeMapRenderingRun, DepthShadowMapRenderingRun, LightVolumeRenderingRun, PhaseTextureRenderingRun, PhotonMapRenderingRun, ReducedCubeMapRenderingRun, SceneCameraDepthRenderingRun, OgreChildPSystemRenderingRun, OgreDepthShadowMapRenderingRun, OgreFocusingMapRenderingRun, OgreLightVolumeRenderingRun, OgrePhaseTextureRenderingRun, OgrePhotonMapRenderingRun, OgrePMEntryPointMapRenderingRun, OgrePMWeightComputeRenderingRun, and OgreSceneCameraDepthRenderingRun.

5.50.3.23 void ReducedCubeMapRenderingRun::updateFrame (unsigned long *frameNum*) [protected, virtual, inherited]

This function does the actual update in a frame.

Parameters:

frameNum current frame number

Reimplemented from RenderingRun.

5.50.4 Member Data Documentation

5.50.4.1 std::map<Renderable*, String> OgreRenderingRun::visibleObjects [protected, inherited]

map of Renderables which will be rendered with a given material

The String stores the original material name that will be restored after rendering.

See also:

setMaterialForVisibles
setMaterialForRenderables
restoreMaterials

5.50.4.2 std::map<Renderable*, String> OgreRenderingRun::visibleObjects [protected, inherited]

map of Renderables which will be rendered with a given material

The String stores the original material name that will be restored after rendering.

See also:

setMaterialForVisibles
setMaterialForRenderables
restoreMaterials

5.50.4.3 MovablePlane * OgreRenderingRun::fullScreenQuad [static, protected, inherited]

fulls screen quad plane used in full screen quad rendering

See also:

renderFullscreenQuad

5.50.4.4 MovablePlane* OgreRenderingRun::fullScreenQuad [static, protected, inherited]

fulls screen quad plane used in full screen quad rendering

See also:

renderFullscreenQuad

5.50.4.5 Entity * OgreRenderingRun::fullScreenQuadEntity [static, protected, inherited]

fulls screen quad Entity used in full screen quad rendering

See also:

renderFullscreenQuad

5.50.4.6 Entity* OgreRenderingRun::fullScreenQuadEntity [static, protected, inherited]

fulls screen quad Entity used in full screen quad rendering

See also:

renderFullscreenQuad

5.50.4.7 SceneNode * OgreRenderingRun::fullScreenQuadNode [static, protected, inherited]

fulls screen quad SceneNode used in full screen quad rendering

See also:

renderFullscreenQuad

5.50.4.8 SceneNode* OgreRenderingRun::fullScreenQuadNode [static, protected, inherited]

fulls screen quad SceneNode used in full screen quad rendering

See also:

renderFullscreenQuad

5.50.4.9 SpriteSet* OgreRenderingRun::pixelSprites [protected, inherited]

SpriteSet used in pixel sprite rendering.

See also:

renderPixelSprites

5.50.4.10 BillboardSet* OgreRenderingRun::pixelSprites [protected, inherited]

SpriteSet used in pixel sprite rendering.

See also:

renderPixelSprites

5.50.4.11 String OgreRenderingRun::spriteSetName [protected, inherited]

unique name of the SpriteSet used in pixel sprite rendering

See also:

renderPixelSprites

5.50.4.12 Entity* OgreRenderingRun::fullscreenGrid [protected, inherited]

Entity used in fullscreen grid rendering.

See also:

renderPixelGrid

5.50.4.13 bool ReducedCubeMapRenderingRun::useDistCalc [protected, inherited]

a flag to skip cube face update if object is far away or too small.

See also:

distTolerance

5.50.4.14 bool ReducedCubeMapRenderingRun::useFaceAngleCalc [protected, inherited]

a flag to skip cube face update the face is neglible.

See also:

angleTolerance

5.50.4.15 float ReducedCubeMapRenderingRun::distTolerance [protected, inherited]

A value used in face skip test.

The higher this value gets the more precise, but slower the method will be.

5.50.4.16 float ReducedCubeMapRenderingRun::angleTolerance [protected, inherited]

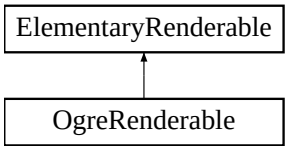
A value used in face skip test.

The higher this value gets the more precise, but slower the method will be.

5.51 OgreRenderable Class Reference

Class to wrap different Ogre Renderable types.

Inheritance diagram for OgreRenderable::



Public Member Functions

- OgreRenderable (SubEntity *sube, Entity *parentEntity)
  Constructor.
- OgreRenderable (Entity *parentEntity, int subEntityNum)
  Constructor.
- OgreRenderable (BillboardSet *billboardset, ParticleSystem *sys=0)
  Constructor.
- ~OgreRenderable (void)
  Destructor.
- void setVisible (bool visible)
  Sets the visibility of the wrapped renderable.
- void setRenderGroup (unsigned char groupID)
  Sets the rendergroup of the wrapped renderable.
- void updateRenderQueue (RenderQueue *rq)
  Updates the given renderqueue for the wrapped renderable.
- bool isVisible ()
  Retrieves if the renderable is hided or shown.
- void setMaterialName (String &name)
  Sets the material to be used by the renderable.
- const MaterialPtr & getMaterialPtr ()
  Retrieves a resource pointer to the material used by the renderable.
- const String & getMaterialName ()
  Retrieves the name of the material used by the renderable.

- Material * getMaterial ()

  Retrieves a pointer to the material used by the renderable.

- AxisAlignedBox & getBoundingBox ()

  Retrieves the axis-aligned bouding box of the renderable.

- Sphere & getBoundingSphere ()

  Retrieves the bouding sphere of the renderable.

- String & getName ()

  Retrieves the unique name assigned to the renderable.

- void updateBounds ()

  Updates bounding volumes.

- void notifyCamera (Camera *cam)

  Calls notifyCamera for the wrapped Renderable.

- Renderable * getRenderable ()

  Returns the wrapped Renderable.

Protected Attributes

- String name

  unique name assigned to the renderable

- Entity * parentEntity

  pointer to the parent Entity if the renderable is a Subentity

- SubEntity * subEntityRenderable

  pointer to the wrapped Subentity (if the renderable is a Subentity)

- BillboardSet * billboardSetRenderable

  pointer to the wrapped BillboardSet (if the renderable is a BillboardSet)

- ParticleSystem * parentParticleSystem

  pointer to the parent Particle System of the wrapped BillboardSet (if the renderable is a BillboardSet)

- AxisAlignedBox boundingBox

  axis-aligned bounding box of the wrapped renderable in world space

- Sphere boundingSphere

  bounding sphere of the wrapped renderable in world space

- Ogre_RenderableType renderableType

  type of the renderable (see Ogre_RenderableType)

5.51.1 Detailed Description

Class to wrap different Ogre Renderable types.

5.51.2 Constructor & Destructor Documentation

5.51.2.1 OgreRenderable::OgreRenderable (SubEntity * *sube*, Entity * *parentEntity*)

Constructor.

Creates an OgreRenderable from a SubEntity of an Entity.

Parameters:

sube the subentity to wrap

parentEntity the parent of the wrapped subentity

5.51.2.2 OgreRenderable::OgreRenderable (Entity * *parentEntity*, int *subEntityNum*)

Constructor.

Creates an OgreRenderable from a SubEntity of an Entity.

Parameters:

parentEntity the parent of the wrapped subentity

subEntityNum the number of the subentity to wrap

5.51.2.3 OgreRenderable::OgreRenderable (BillboardSet * *billboardset*, ParticleSystem * *sys* = 0)

Constructor.

Creates an OgreRenderable from a SubEntity of a BillboardSet.

Parameters:

billboardset the BillboardSet to wrap

5.51.3 Member Function Documentation

5.51.3.1 void OgreRenderable::setVisible (bool *visible*) [virtual]

Sets the visibility of the wrapped renderable.

Parameters:

visible visibility

See also:

ElementaryRenderable::setVisible()

Implements ElementaryRenderable.

5.51.3.2 void OgreRenderable::setRenderGroup (unsigned char *groupID*) [virtual]

Sets the rendergroup of the wrapped renderable.

Parameters:

groupID the ID of the group to use

See also:

ElementaryRenderable::setRenderGroup()

Implements ElementaryRenderable.

5.51.3.3 void OgreRenderable::updateRenderQueue (RenderQueue * *rq*)

Updates the given renderqueue for the wrapped renderable.

Parameters:

rq pointer to the renderqueue to be updated

5.51.3.4 void OgreRenderable::setMaterialName (String & *name*)

Sets the material to be used by the renderable.

Parameters:

name the name of the material to use

5.51.3.5 const MaterialPtr & OgreRenderable::getMaterialPtr ()

Retrieves a resource pointer to the material used by the renderable.

Returns:

reference to the resource pointer

5.51.3.6 const String& OgreRenderable::getMaterialName () [inline]

Retrieves the name of the material used by the renderable.

Returns:

reference to the name of the material

5.51.3.7 Material* OgreRenderable::getMaterial () [inline]

Retrieves a pointer to the material used by the renderable.

Returns:

reference to the Material pointer

5.51.3.8 AxisAlignedBox& OgreRenderable::getBoundingBox () [inline]

Retrieves the axis-aligned bouding box of the renderable.

Returns:

reference to the bouding box

5.51.3.9 Sphere& OgreRenderable::getBoundingSphere () [inline]

Retrieves the bouding sphere of the renderable.

Returns:

reference to the bouding sphere

5.51.3.10 String& OgreRenderable::getName () [inline]

Retrieves the unique name assigned to the renderable.

Returns:

reference to name of the renderable

5.51.3.11 void OgreRenderable::notifyCamera (Camera * *cam*)

Calls notifyCamera for the wrapped Renderable.

Parameters:

pointer to the Camera to pass to notifyCamera()

5.51.3.12 Renderable * OgreRenderable::getRenderable ()

Returns the wrapped Renderable.

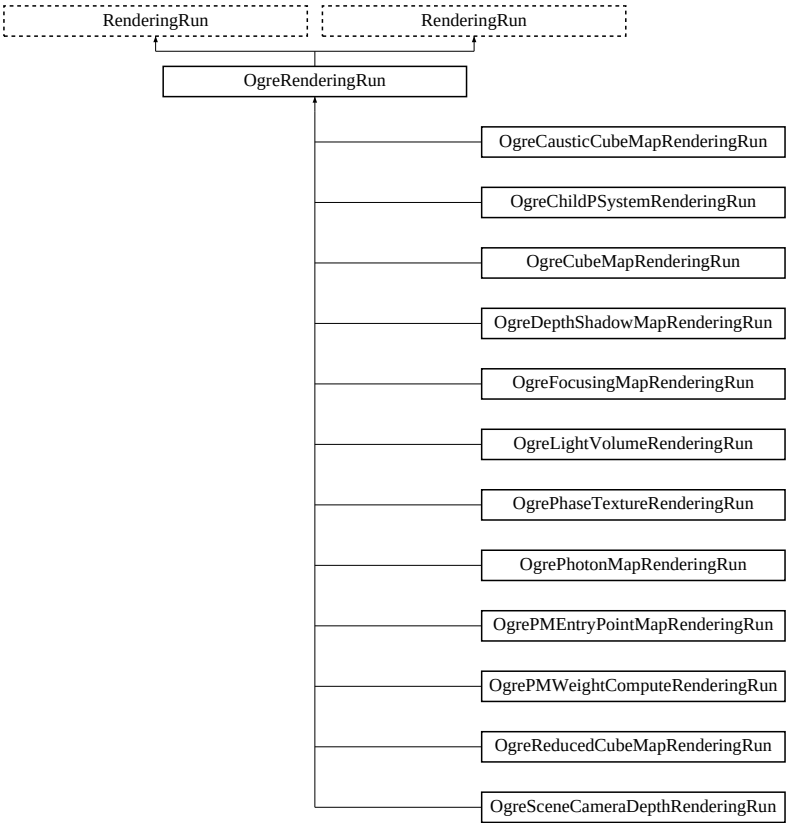
Returns:

pointer to the wrapped Renderable

5.52 OgreRenderingRun Class Reference

Base class of a RenderingRun in an OGRE environment.

Inheritance diagram for OgreRenderingRun::



Public Member Functions

- OgreRenderingRun (unsigned long startFrame, unsigned long updateInterval)

  Constructor.

- OgreRenderingRun * asOgreRenderingRun ()

  Conversion to OgreRenderRun.

- OgreRenderingRun (unsigned long startFrame, unsigned long updateInterval)

  Constructor.

- OgreRenderingRun * asOgreRenderingRun ()

  Conversion to OgreRenderRun.

- bool update (unsigned long frameNum)

Calls updateFrame() if the run needs update according to its starting frame and update interval and has not been allready updated in this frame.

- virtual bool canJoin (RenderingRun *run)

  Returns true if two runs can be joined.

Protected Member Functions

- Vector3 getCubeMapFaceDirection (unsigned char faceId)

  Returns a direction for a cubemap face id.

- Texture * createCubeRenderTexture (String name, const Vector3 position, unsigned int resolution=512, PixelFormat format=PF_FLOAT16_RGBA, int numMips=0, ColourValue clearColor=ColourValue::Black)

  Creates a cubemap texture.

- void setMaterialForRenderables (String &materialName, RenderQueue *rq)

  Sets the given material for each Renderable in a RenderQueue.

- void setMaterialForVisibles (String &materialName, Camera *cam, bool shadowcastersonly=false)

  Sets the given material for each Renderable visible from a given camera.

- void restoreMaterials ()

  Restores previously stored materials.

- void renderFullscreenQuad (String materialName, RenderTarget *target)

  Renderes a full screen quad on a given RendderTarget with a given material.

- void renderPixelSprites (String &materialName, RenderTarget *rt, int width, int height)

  Renderes sprites to pixels of the screen on a given RendderTarget with a given material.

- Vector3 getCubeMapFaceDirection (unsigned char faceId)

  Returns a direction for a cubemap face id.

- Texture * createCubeRenderTexture (String name, const Vector3 position, unsigned int resolution=512, PixelFormat format=PF_FLOAT16_RGBA, int numMips=0, ColourValue clearColor=ColourValue::Black)

  Creates a cubemap texture.

- void setMaterialForRenderables (String &materialName, RenderQueue *rq, bool solidsonly=false)

  Sets the given material for each Renderable in a RenderQueue.

- void setMaterialForVisibles (String &materialName, Camera *cam, bool shadowcastersonly=false, bool solidsonly=false)

  Sets the given material for each Renderable visible from a given camera.

- void restoreMaterials ()

  Restores previously stored materials.

- void renderFullscreenQuad (String materialName, RenderTarget *target)

  Renderes a full screen quad on a given RendderTarget with a given material.

- void renderPixelSprites (String &materialName, RenderTarget *rt, int width, int height)

  Renderes sprites to pixels of the screen on a given RendderTarget with a given material.

- void renderFullscreenGrid (String &materialName, RenderTarget *rt, int width, int height)

  Renders a grid onto the screen.

- virtual bool needUpdate (unsigned long frameNum)

  Returns if this run needs update.

- virtual void updateFrame (unsigned long frameNum)

  This function does the actual update in a frame.

Protected Attributes

- std::map< Renderable *, String > visibleObjects

  map of Renderables which will be rendered with a given material

- SpriteSet * pixelSprites

  SpriteSet used in pixel sprite rendering.

- String spriteSetName

  unique name of the SpriteSet used in pixel sprite rendering

- std::map< Renderable *, String > visibleObjects

  map of Renderables which will be rendered with a given material

- BillboardSet * pixelSprites

  SpriteSet used in pixel sprite rendering.

- Entity * fullscreenGrid

  Entity used in fullscreen grid rendering.

- unsigned long lastupdated

  The number of the last frame this run was updated.

- unsigned long startFrame

  The number of the frame this run should be updated first.

- unsigned long updateInterval

  Refresh frequency in frames.

Static Protected Attributes

- static MovablePlane * fullScreenQuad
  fulls screen quad plane used in full screen quad rendering
- static Entity * fullScreenQuadEntity
  fulls screen quad Entity used in full screen quad rendering
- static SceneNode * fullScreenQuadNode
  fulls screen quad SceneNode used in full screen quad rendering
- static MovablePlane * fullScreenQuad
  fulls screen quad plane used in full screen quad rendering
- static Entity * fullScreenQuadEntity
  fulls screen quad Entity used in full screen quad rendering
- static SceneNode * fullScreenQuadNode
  fulls screen quad SceneNode used in full screen quad rendering

5.52.1 Detailed Description

Base class of a RenderingRun in an OGRE environment.

5.52.2 Constructor & Destructor Documentation

5.52.2.1 OgreRenderingRun::OgreRenderingRun (unsigned long *startFrame*, unsigned long *updateInterval*) [inline]

Constructor.

Parameters:

startFrame adds an offset to the current frame number to help evenly distribute updates between frames

updateInterval photon map update frequency

5.52.2.2 OgreRenderingRun::OgreRenderingRun (unsigned long *startFrame*, unsigned long *updateInterval*) [inline]

Constructor.

Parameters:

startFrame adds an offset to the current frame number to help evenly distribute updates between frames

updateInterval photon map update frequency

5.52.3 Member Function Documentation

5.52.3.1 OgreRenderingRun* OgreRenderingRun::asOgreRenderingRun () [inline, virtual]

Conversion to OgreRenderRun.

This function is needed because of virtual inheritance.

Reimplemented from RenderingRun.

5.52.3.2 Vector3 OgreRenderingRun::getCubeMapFaceDirection (unsigned char *faceId*) [protected]

Returns a direction for a cubemap face id.

This is a helper function to retrieve the normal direction of a given cubemap face.

Parameters:

faceId the number of the face

5.52.3.3 Texture * OgreRenderingRun::createCubeRenderTexture (String *name*, const Vector3 *position*, unsigned int *resolution* = 512, PixelFormat *format* = PF_FLOAT16_RGBA, int *numMips* = 0, ColourValue *clearColor* = ColourValue::Black) [protected]

Creates a cubemap texture.

This is a helper function to easily create a cubemap texture and automaticly attach viewports to each face so it can be used as a rendertarget.

Parameters:

name the name of the texture to be created

position the initial position of the cubemap

resolution the resolution of one cubemapface

format the pixel format of the cubemap

numMips the number of mipmap levels

clearColor initial color

5.52.3.4 void OgreRenderingRun::setMaterialForRenderables (String & *materialName*, RenderQueue * *rq*) [protected]

Sets the given material for each Renderable in a RenderQueue.

This is a helper function to set a material to each element of a previously filled Renderque. The orginal material of the Renderables are stored so they can be restored later. The function also tells the current

SceneManager not to search for visible objects, as we are going to use the given RenderQueue during the next rendering.

Parameters:

materialName the name of the material to set for the Renderables

rq pointer to the filled Renderqueue instance to set material for

5.52.3.5 void OgreRenderingRun::setMaterialForVisibles (String & *materialName*, Camera * *cam*, bool *shadowcastersonly* = `false`) `[protected]`

Sets the given material for each Renderable visible from a given camera.

This helper function is similar to setMaterialForRenderables but it is also responsible for filling the RenderQueue. First the RenderQueue of the current SceneManager fill be filled with the visible objects seen from the given camera. Then the required material will be set for each element of the RenderQueue. The orginal material of the Renderables are stored so they can be restored later. The function also tells the current SceneManager not to search for visible objects, as we are going to use the filled RenderQueue during the next rendering.

Parameters:

materialName the name of the material to set for the Renderables

cam pointer to the camera from which visible objects should be searched

shadowcastersonly flag to search for only shadow casters

5.52.3.6 void OgreRenderingRun::restoreMaterials () `[protected]`

Restores previously stored materials.

This helper function is typically used after a setMaterialForRenderables or setMaterialForVisibles call and a rendering process to restore the original material settings. The function also tells the current SceneManager to search for visible objects, as this is the default behaviour of SceneManager.

5.52.3.7 void OgreRenderingRun::renderFullscreenQuad (String *materialName*, RenderTarget * *target*) `[protected]`

Renderes a full screen quad on a given RendderTarget with a given material.

Parameters:

materialName the name of the material bind to the quad

target the RenderTarget the quad should be rendered on

5.52.3.8 void OgreRenderingRun::renderPixelSprites (String & *materialName*, RenderTarget * *rt*, int *width*, int *height*) [protected]

Renderes sprites to pixels of the screen on a given RendderTarget with a given material.

Pixel sprites are pixel sized quads, placed on each pixel of the RenderTarget. The number of sprites not necessary corresponds to the resolution of the rendertarget. The pixel quads will evenly fill the rendertarget's area with sizes corresponding to the given resolution. We can render fewer or more pixel quads than the number of pixels the rendertarget has (eg.: in case of caustic cubemap generation).

Parameters:

materialName the name of the material bind to the pixel sprites

rt the RenderTarget the quads should be rendered on

width the desired resolution width of the sprites

height the desired resolution height of the sprites

5.52.3.9 OgreRenderingRun* OgreRenderingRun::asOgreRenderingRun () [inline, virtual]

Conversion to OgreRenderRun.

This function is needed because of virtual inheritance.

Reimplemented from RenderingRun.

5.52.3.10 Vector3 OgreRenderingRun::getCubeMapFaceDirection (unsigned char *faceId*) [protected]

Returns a direction for a cubemap face id.

This is a helper function to retrieve the normal direction of a given cubemap face.

Parameters:

faceId the number of the face

5.52.3.11 Texture* OgreRenderingRun::createCubeRenderTexture (String *name*, const Vector3 *position*, unsigned int *resolution* = 512, PixelFormat *format* = PF_FLOAT16_RGBA, int *numMips* = 0, ColourValue *clearColor* = ColourValue::Black) [protected]

Creates a cubemap texture.

This is a helper function to easily create a cubemap texture and automaticly attach viewports to each face so it can be used as a rendertarget.

Parameters:

name the name of the texture to be created

position the initial position of the cubemap

resolution the resolution of one cubemapface

format the pixel format of the cubemap

numMips the number of mipmap levels

clearColor initial color

5.52.3.12 void OgreRenderingRun::setMaterialForRenderables (String & *materialName*, RenderQueue * *rq*, bool *solidsonly* = `false`) [protected]

Sets the given material for each Renderable in a RenderQueue.

This is a helper function to set a material to each element of a previously filled Renderque. The orginal material of the Renderables are stored so they can be restored later. The function also tells the current SceneManager not to search for visible objects, as we are going to use the given RenderQueue during the next rendering.

Parameters:

materialName the name of the material to set for the Renderables

rq pointer to the filled Renderqueue instance to set material for

5.52.3.13 void OgreRenderingRun::setMaterialForVisibles (String & *materialName*, Camera * *cam*, bool *shadowcastersonly* = `false`, bool *solidsonly* = `false`) [protected]

Sets the given material for each Renderable visible from a given camera.

This helper function is similar to setMaterialForRenderables but it is also responsible for filling the RenderQueue. First the RenderQueue of the current SceneManager fill be filled with the visible objects seen from the given camera. Then the required material will be set for each element of the RenderQueue. The orginal material of the Renderables are stored so they can be restored later. The function also tells the current SceneManager not to search for visible objects, as we are going to use the filled RenderQueue during the next rendering.

Parameters:

materialName the name of the material to set for the Renderables

cam pointer to the camera from which visible objects should be searched

shadowcastersonly flag to search for only shadow casters

5.52.3.14 void OgreRenderingRun::restoreMaterials () [protected]

Restores previously stored materials.

This helper function is typically used after a setMaterialForRenderables or setMaterialForVisibles call and a rendering process to restore the original material settings. The function also tells the current SceneManager to search for visible objects, as this is the default behaviour of SceneManager.

5.52.3.15 void OgreRenderingRun::renderFullscreenQuad (String *materialName*, RenderTarget * *target*) [protected]

Renderes a full screen quad on a given RendderTarget with a given material.

Parameters:

materialName the name of the material bind to the quad

target the RenderTarget the quad should be rendered on

5.52.3.16 void OgreRenderingRun::renderPixelSprites (String & *materialName*, RenderTarget * *rt*, int *width*, int *height*) [protected]

Renderes sprites to pixels of the screen on a given RendderTarget with a given material.

Pixel sprites are pixel sized quads, placed on each pixel of the RenderTarget. The number of sprites not necessary corresponds to the resolution of the rendertarget. The pixel quads will evenly fill the rendertarget's area with sizes corresponding to the given resolution. We can render fewer or more pixel quads than the number of pixels the rendertarget has (eg.: in case of caustic cubemap generation).

Parameters:

materialName the name of the material bind to the pixel sprites

rt the RenderTarget the quads should be rendered on

width the desired resolution width of the sprites

height the desired resolution height of the sprites

5.52.3.17 void OgreRenderingRun::renderFullscreenGrid (String & *materialName*, RenderTarget * *rt*, int *width*, int *height*) [protected]

Renders a grid onto the screen.

Parameters:

rt the RenderTarget the grid should be rendered on

width the desired horizontal resolution of the grid

height the desired vertical resolution of the grid

5.52.3.18 virtual bool RenderingRun::canJoin (RenderingRun * *run*) [inline, virtual, inherited]

Returns true if two runs can be joined.

In some cases special requirements should stand to join two runs (even if they have the same type). Eg.: two caustic cube map generation technique should only be joined if they use the same material when rendering the caustic cubemap.

Reimplemented in OgreCausticCubeMapRenderingRun, OgreChildPSystemRenderingRun, and OgrePhotonMapRenderingRun.

5.52.3.19 virtual bool RenderingRun::needUpdate (unsigned long *frameNum*) [inline, protected, virtual, inherited]

Returns if this run needs update.

This tipically depends on the upate interval and the starting frame number.

Parameters:

frameNum current frame number

Reimplemented in OgreFocusingMapRenderingRun, and OgrePMEntryPointMapRenderingRun.

5.52.3.20 virtual void RenderingRun::updateFrame (unsigned long *frameNum*) [inline, protected, virtual, inherited]

This function does the actual update in a frame.

Parameters:

frameNum current frame number

Reimplemented in CausticCubeMapRenderingRun, ChildPsystemRenderingRun, CubeMapRenderingRun, DepthShadowMapRenderingRun, LightVolumeRenderingRun, PhaseTextureRenderingRun, PhotonMapRenderingRun, ReducedCubeMapRenderingRun, SceneCameraDepthRenderingRun, OgreChildPSystemRenderingRun, OgreDepthShadowMapRenderingRun, OgreFocusingMapRenderingRun, OgreLightVolumeRenderingRun, OgrePhaseTextureRenderingRun, OgrePhotonMapRenderingRun, OgrePMEntryPointMapRenderingRun, OgrePMWeightComputeRenderingRun, and OgreSceneCameraDepthRenderingRun.

5.52.4 Member Data Documentation

5.52.4.1 std::map<Renderable*, String> OgreRenderingRun::visibleObjects [protected]

map of Renderables which will be rendered with a given material

The String stores the original material name that will be restored after rendering.

See also:

setMaterialForVisibles
setMaterialForRenderables
restoreMaterials

5.52.4.2 MovablePlane * OgreRenderingRun::fullScreenQuad [static, protected]

fulls screen quad plane used in full screen quad rendering

See also:

renderFullscreenQuad

5.52.4.3 Entity * OgreRenderingRun::fullScreenQuadEntity [static, protected]

fulls screen quad Entity used in full screen quad rendering

See also:

renderFullscreenQuad

5.52.4.4 SceneNode * OgreRenderingRun::fullScreenQuadNode [static, protected]

fulls screen quad SceneNode used in full screen quad rendering

See also:

renderFullscreenQuad

5.52.4.5 SpriteSet* OgreRenderingRun::pixelSprites [protected]

SpriteSet used in pixel sprite rendering.

See also:

renderPixelSprites

5.52.4.6 String OgreRenderingRun::spriteSetName [protected]

unique name of the SpriteSet used in pixel sprite rendering

See also:

renderPixelSprites

5.52.4.7 std::map<Renderable*, String> OgreRenderingRun::visibleObjects [protected]

map of Renderables which will be rendered with a given material
The String stores the original material name that will be restored after rendering.

See also:

setMaterialForVisibles
setMaterialForRenderables
restoreMaterials

5.52.4.8 MovablePlane* OgreRenderingRun::fullScreenQuad [static, protected]

fulls screen quad plane used in full screen quad rendering

See also:

renderFullscreenQuad

5.52.4.9 Entity* OgreRenderingRun::fullScreenQuadEntity [static, protected]

fulls screen quad Entity used in full screen quad rendering

See also:

renderFullscreenQuad

5.52.4.10 SceneNode* OgreRenderingRun::fullScreenQuadNode [static, protected]

fulls screen quad SceneNode used in full screen quad rendering

See also:

renderFullscreenQuad

5.52.4.11 BillboardSet* OgreRenderingRun::pixelSprites [protected]

SpriteSet used in pixel sprite rendering.

See also:

renderPixelSprites

5.52.4.12 Entity* OgreRenderingRun::fullscreenGrid [protected]

Entity used in fullscreen grid rendering.

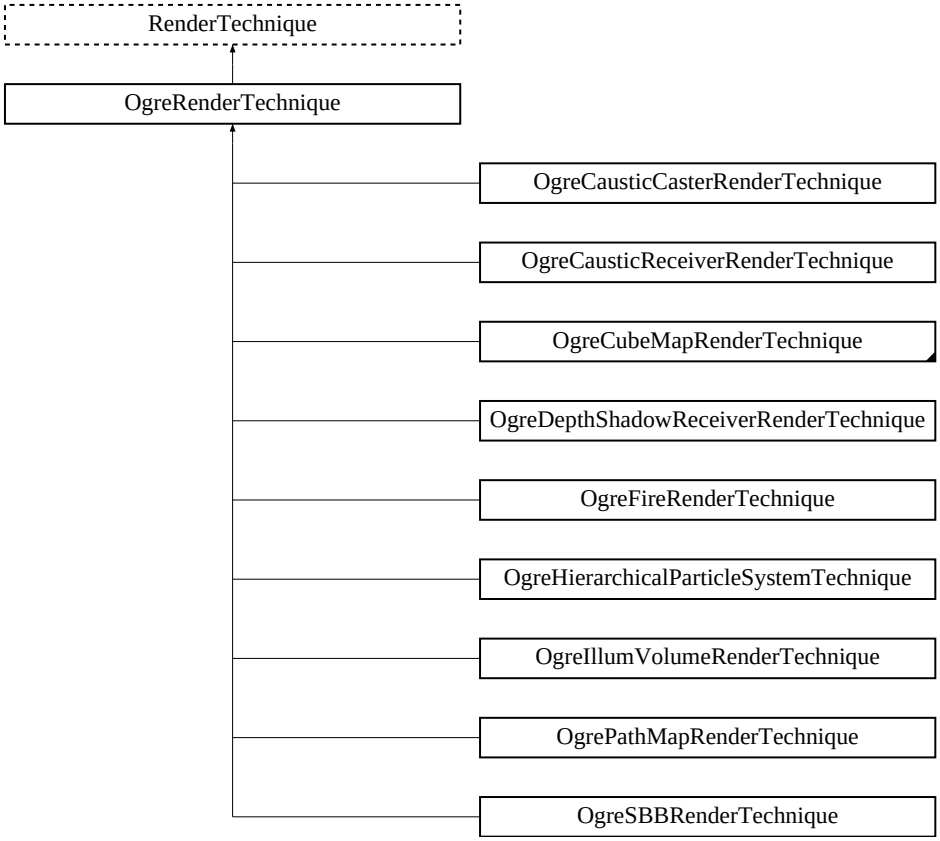
See also:

renderPixelGrid

5.53 OgreRenderTechnique Class Reference

Class of RenderTechniques used in an OGRE environment.

Inheritance diagram for OgreRenderTechnique::



Public Member Functions

- OgreRenderTechnique (Pass *pass, OgreRenderable *parentRenderable, OgreTechniqueGroup *parentTechniqueGroup)

  Constructor.

- virtual OgreRenderTechnique * asOgreRenderTechnique ()

  Conversion to OgreRenderTechnique.

- virtual void update (unsigned long frameNum)

  Updates the resources in the given frame.

- virtual void runChanged (RenderingRunType runType, RenderingRun *run)

  Called after one of he shared runs changes.

- virtual void runUpdated (RenderingRunType runType, RenderingRun *run)

  Called after one of he shared runs updates.

- ElementaryRenderable * getParentRenderable ()

  Retrieves the renderable this technique operates on.

Protected Attributes

- OgreRenderable * parentOgreRenderable

  a OgreRenderable pointer to the renderable this technique operates on.

- OgreTechniqueGroup * parentOgreTechniqueGroup

  a OgreTechniqueGroup pointer to the TechniqueGroup this technique is attached to.

- Pass * pass

  a pointer to the pass this technique operates on.

- ElementaryRenderable * parentRenderable

  The renderable this technique operates on.

- TechniqueGroup * parentTechniqueGroup

  The TechniqueGroup this RenderedTechnique is attached to.

- SharedRuns * sharedRuns

  The SharedRuns this RenderedTechnique is attached to.

5.53.1 Detailed Description

Class of RenderTechniques used in an OGRE environment.

5.53.2 Constructor & Destructor Documentation

5.53.2.1 OgreRenderTechnique::OgreRenderTechnique (Pass * *pass*, OgreRenderable * *parentRenderable*, OgreTechniqueGroup * *parentTechniqueGroup*)

Constructor.

Parameters:

the pass to operate on

parentRenderable the object to operate on

parentTechniqueGroup the TechniqueGroup this RenderedTechnique is attached to

5.53.3 Member Function Documentation

5.53.3.1 virtual OgreRenderTechnique* OgreRenderTechnique::asOgreRenderTechnique () [inline, virtual]

Conversion to OgreRenderTechnique.

This function is needed because of virtual inheritance.

Reimplemented from RenderTechnique.

5.53.3.2 virtual void RenderTechnique::update (unsigned long *frameNum*) [inline, virtual, inherited]

Updates the resources in the given frame.

A RenderTechnique is usually need some resources from several runs, so these runs will be updated.

Parameters:

frameNum the actual framenumber

Reimplemented in ColorCubeMapRenderTechnique, ConvolvedCubeMapRenderTechnique, DistanceCubeMapRenderTechnique, HierarchicalParticleSystemTechnique, IllumVolumeRenderTechnique, OgreCausticReceiverRenderTechnique, OgreColorCubeMapRenderTechnique, OgreConvolvedCubeMapRenderTechnique, OgreDepthShadowReceiverRenderTechnique, OgreDistanceCubeMapRenderTechnique, OgreFireRenderTechnique, OgrePathMapRenderTechnique, and OgreSBBRenderTechnique.

5.53.3.3 virtual void RenderTechnique::runChanged (RenderingRunType *runType*, RenderingRun * *run*) [inline, virtual, inherited]

Called after one of he shared runs changes.

Parameters:

runType enum describing the type of the changed run

run pointer to the changed RenderingRun

Reimplemented in CausticCasterRenderTechnique, ColorCubeMapRenderTechnique, ConvolvedCubeMapRenderTechnique, CubeMapRenderTechnique, DistanceCubeMapRenderTechnique, HierarchicalParticleSystemTechnique, and IllumVolumeRenderTechnique.

5.53.3.4 virtual void RenderTechnique::runUpdated (RenderingRunType *runType*, RenderingRun * *run*) [inline, virtual, inherited]

Called after one of he shared runs updates.

Parameters:

runType enum describing the type of the updated run

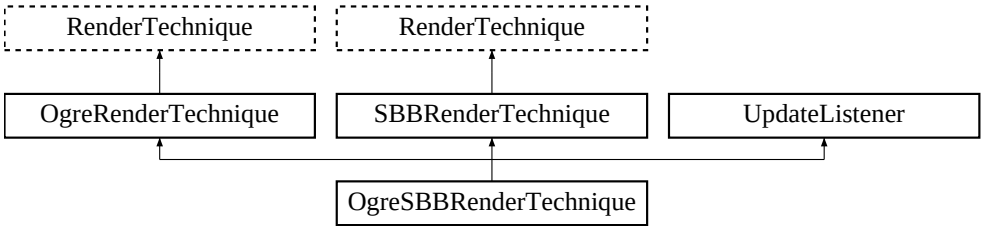
run pointer to the updated RenderingRun

Reimplemented in CausticCasterRenderTechnique, ColorCubeMapRenderTechnique, CubeMapRenderTechnique, DistanceCubeMapRenderTechnique, HierarchicalParticleSystemTechnique, and IllumVolumeRenderTechnique.

5.54 OgreSBBRenderTechnique Class Reference

SBBRenderTechnique used in an OGRE environment.

Inheritance diagram for OgreSBBRenderTechnique::



Public Member Functions

- OgreSBBRenderTechnique (unsigned char depthTexID, Pass *pass, OgreRenderable *parentRenderable, OgreTechniqueGroup *parentTechniqueGroup)

  Constructor.

- virtual void update (unsigned long frameNum)

  Updates the resources in the given frame.

- void preAllUpdates ()

  Called before RenderTechnique updates.

- void postAllUpdates ()

  Called after RenderTechnique updates.

- virtual OgreRenderTechnique * asOgreRenderTechnique ()

  Conversion to OgreRenderTechnique.

- virtual void runChanged (RenderingRunType runType, RenderingRun *run)

  Called after one of he shared runs changes.

- virtual void runUpdated (RenderingRunType runType, RenderingRun *run)

  Called after one of he shared runs updates.

- ElementaryRenderable * getParentRenderable ()

  Retrieves the renderable this technique operates on.

Protected Attributes

- unsigned char depthTexID
- OgreRenderable * parentOgreRenderable

a OgreRenderable pointer to the renderable this technique operates on.

- OgreTechniqueGroup * parentOgreTechniqueGroup

  a OgreTechniqueGroup pointer to the TechniqueGroup this technique is attached to.

- Pass * pass

  a pointer to the pass this technique operates on.

- ElementaryRenderable * parentRenderable

  The renderable this technique operates on.

- TechniqueGroup * parentTechniqueGroup

  The TechniqueGroup this RenderedTechnique is attached to.

- SharedRuns * sharedRuns

  The SharedRuns this RenderedTechnique is attached to.

5.54.1 Detailed Description

SBBRenderTechnique used in an OGRE environment.

5.54.2 Constructor & Destructor Documentation

5.54.2.1 OgreSBBRenderTechnique::OgreSBBRenderTechnique (unsigned char *depthTexID*, Pass * *pass*, OgreRenderable * *parentRenderable*, OgreTechniqueGroup * *parentTechniqueGroup*)

Constructor.

Parameters:

depthTexID the id of the texture unit state the resulting scene depth map should be bound to

pass the pass to operate on

parentRenderable the object to operate on

parentTechniqueGroup the TechniqueGroup this RenderedTechnique is attached to

5.54.3 Member Function Documentation

5.54.3.1 void OgreSBBRenderTechnique::update (unsigned long *frameNum*) [virtual]

Updates the resources in the given frame.

A RenderTechnique is usually need some resources from several runs, so these runs will be updated.

Parameters:

frameNum the actual framenumber

Reimplemented from RenderTechnique.

5.54.3.2 virtual OgreRenderTechnique* OgreRenderTechnique::asOgreRenderTechnique ()

`[inline, virtual, inherited]`

Conversion to OgreRenderTechnique.

This function is needed because of virtual inheritance.

Reimplemented from RenderTechnique.

5.54.3.3 virtual void RenderTechnique::runChanged (RenderingRunType *runType*, RenderingRun * *run*)

`[inline, virtual, inherited]`

Called after one of he shared runs changes.

Parameters:

runType enum describing the type of the changed run

run pointer to the changed RenderingRun

Reimplemented in CausticCasterRenderTechnique, ColorCubeMapRenderTechnique, ConvolvedCubeMapRenderTechnique, CubeMapRenderTechnique, DistanceCubeMapRenderTechnique, HierarchicalParticleSystemTechnique, and IllumVolumeRenderTechnique.

5.54.3.4 virtual void RenderTechnique::runUpdated (RenderingRunType *runType*, RenderingRun * *run*)

`[inline, virtual, inherited]`

Called after one of he shared runs updates.

Parameters:

runType enum describing the type of the updated run

run pointer to the updated RenderingRun

Reimplemented in CausticCasterRenderTechnique, ColorCubeMapRenderTechnique, CubeMapRenderTechnique, DistanceCubeMapRenderTechnique, HierarchicalParticleSystemTechnique, and IllumVolumeRenderTechnique.

5.54.4 Member Data Documentation

5.54.4.1 unsigned char OgreSBBRenderTechnique::depthTexID

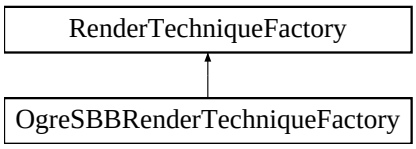
`[protected]`

&brief the id of the texture unit state the resulting scene depth map should be bound to

5.55 OgreSBBRenderTechniqueFactory Class Reference

RenderTechniqueFactory to create OgreSBBRenderTechnique instances.

Inheritance diagram for OgreSBBRenderTechniqueFactory::



Public Member Functions

- OgreRenderTechnique * createInstance (IllumTechniqueParams *params, Pass *pass, OgreRenderable *parentRenderable, OgreTechniqueGroup *parentTechniqueGroup)

  Creates a RenderTechnique of the factory type.

- bool isType (String type)

  Returns if this factory can create a RenderTechnique of the given type.

- virtual void parseParams (IllumTechniqueParams *params)

  parses parameters from the material file.

Protected Types

- typedef void(*) ILLUM_ATTRIBUTE_PARSER (String &params, RenderTechniqueFactory *factory)

  function for parsing RenderTechnique attributes

- typedef std::map< String, ILLUM_ATTRIBUTE_PARSER > AttribParserList

  Keyword-mapped attribute parsers.

Protected Attributes

- AttribParserList attributeParsers

  map of parser functions

- String typeName

  factoryname

5.55.1 Detailed Description

RenderTechniqueFactory to create OgreSBBRenderTechnique instances.

5.55.2 Member Typedef Documentation

5.55.2.1 typedef void(∗) RenderTechniqueFactory::ILLUM_ATTRIBUTE_PARSER(String &params, RenderTechniqueFactory ∗factory) [protected, inherited]

function for parsing RenderTechnique attributes

Parameters:

params attribute value stored in a String

5.55.3 Member Function Documentation

5.55.3.1 OgreRenderTechnique ∗ OgreSBBRenderTechniqueFactory::createInstance (IllumTechniqueParams ∗ *params*, Pass ∗ *pass*, OgreRenderable ∗ *parentRenderable*, OgreTechniqueGroup ∗ *parentTechniqueGroup*) [virtual]

Creates a RenderTechnique of the factory type.

Parameters:

params containes constructor parameters as NameValuePairList

pass the Pass to use in RenderTechnique constructor

pass the parentRenderable to pass to RenderTechnique constructor

pass the parentTechniqueGroup to pass to RenderTechnique constructor

Implements RenderTechniqueFactory.

5.55.3.2 bool RenderTechniqueFactory::isType (String *type*) [inline, inherited]

Returns if this factory can create a RenderTechnique of the given type.

Parameters:

type RenderTechnique type

5.55.3.3 void RenderTechniqueFactory::parseParams (IllumTechniqueParams * *params*) [virtual, inherited]

parses parameters from the material file.

The parsed parameters will be passed to the new RenderTechnique's constructor.

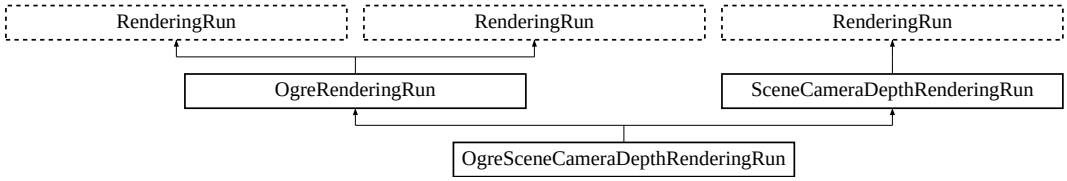
Parameters:

params pointer to the IllumTechniqueParams structure that was read from the material script and containes the parameters to be parsed.

5.56 OgreSceneCameraDepthRenderingRun Class Reference

SceneCameraDepthRenderingRun used in an OGRE environment.

Inheritance diagram for OgreSceneCameraDepthRenderingRun::



Public Member Functions

- OgreSceneCameraDepthRenderingRun (OgreSharedRuns *sharedRuns, String name, Viewport *playerView)

  Constructor.

- String getDepthTextureName ()

  returns the name of the camera depth texture

- OgreRenderingRun * asOgreRenderingRun ()

  Conversion to OgreRenderRun.

- OgreRenderingRun * asOgreRenderingRun ()

  Conversion to OgreRenderRun.

- bool update (unsigned long frameNum)

  Calls updateFrame() if the run needs update according to its starting frame and update interval and has not been allready updated in this frame.

- virtual bool canJoin (RenderingRun *run)

  Returns true if two runs can be joined.

Protected Member Functions

- void updateFrame (unsigned long frameNum)

  This function does the actual update in a frame.

- void createDepthMap ()

  Creates the depth map texture.

- Vector3 getCubeMapFaceDirection (unsigned char faceId)

  Returns a direction for a cubemap face id.

- Vector3 getCubeMapFaceDirection (unsigned char faceId)

  Returns a direction for a cubemap face id.

- Texture * createCubeRenderTexture (String name, const Vector3 position, unsigned int resolution=512, PixelFormat format=PF_FLOAT16_RGBA, int numMips=0, ColourValue clearColor=ColourValue::Black)

  Creates a cubemap texture.

- Texture * createCubeRenderTexture (String name, const Vector3 position, unsigned int resolution=512, PixelFormat format=PF_FLOAT16_RGBA, int numMips=0, ColourValue clearColor=ColourValue::Black)

  Creates a cubemap texture.

- void setMaterialForRenderables (String &materialName, RenderQueue *rq)

  Sets the given material for each Renderable in a RenderQueue.

- void setMaterialForRenderables (String &materialName, RenderQueue *rq, bool solidsonly=false)

  Sets the given material for each Renderable in a RenderQueue.

- void setMaterialForVisibles (String &materialName, Camera *cam, bool shadowcastersonly=false)

  Sets the given material for each Renderable visible from a given camera.

- void setMaterialForVisibles (String &materialName, Camera *cam, bool shadowcastersonly=false, bool solidsonly=false)

  Sets the given material for each Renderable visible from a given camera.

- void restoreMaterials ()

  Restores previously stored materials.

- void restoreMaterials ()

  Restores previously stored materials.

- void renderFullscreenQuad (String materialName, RenderTarget *target)

  Renderes a full screen quad on a given RendderTarget with a given material.

- void renderFullscreenQuad (String materialName, RenderTarget *target)

  Renderes a full screen quad on a given RendderTarget with a given material.

- void renderPixelSprites (String &materialName, RenderTarget *rt, int width, int height)

  Renderes sprites to pixels of the screen on a given RendderTarget with a given material.

- void renderPixelSprites (String &materialName, RenderTarget *rt, int width, int height)

  Renderes sprites to pixels of the screen on a given RendderTarget with a given material.

- void renderFullscreenGrid (String &materialName, RenderTarget *rt, int width, int height)

  Renders a grid onto the screen.

- virtual bool needUpdate (unsigned long frameNum)

  Returns if this run needs update.

Protected Attributes

- Viewport * playerView
  pointer to the player's viewport
- Camera * playerCamera
  pointer to the player's camera
- OgreSharedRuns * sharedRuns
  a pointer to the OgreSharedRuns this run belongs to
- String name
  the name of the depth texture that was created by this run
- Texture * depthTexture
  a pointer to the scene depth texture that was created by this run
- std::map< Renderable *, String > visibleObjects
  map of Renderables which will be rendered with a given material
- std::map< Renderable *, String > visibleObjects
  map of Renderables which will be rendered with a given material
- SpriteSet * pixelSprites
  SpriteSet used in pixel sprite rendering.
- BillboardSet * pixelSprites
  SpriteSet used in pixel sprite rendering.
- String spriteSetName
  unique name of the SpriteSet used in pixel sprite rendering
- Entity * fullscreenGrid
  Entity used in fullscreen grid rendering.
- unsigned long lastupdated
  The number of the last frame this run was updated.
- unsigned long startFrame
  The number of the frame this run should be updated first.
- unsigned long updateInterval
  Refresh frequency in frames.

Static Protected Attributes

- static MovablePlane * fullScreenQuad
  fulls screen quad plane used in full screen quad rendering

- static MovablePlane * fullScreenQuad
  fulls screen quad plane used in full screen quad rendering
- static Entity * fullScreenQuadEntity
  fulls screen quad Entity used in full screen quad rendering
- static Entity * fullScreenQuadEntity
  fulls screen quad Entity used in full screen quad rendering
- static SceneNode * fullScreenQuadNode
  fulls screen quad SceneNode used in full screen quad rendering
- static SceneNode * fullScreenQuadNode
  fulls screen quad SceneNode used in full screen quad rendering

5.56.1 Detailed Description

SceneCameraDepthRenderingRun used in an OGRE environment.

5.56.2 Constructor & Destructor Documentation

5.56.2.1 OgreSceneCameraDepthRenderingRun::OgreSceneCameraDepthRenderingRun (OgreSharedRuns * *sharedRuns*, String *name*, Viewport * *playerView*)

Constructor.

Parameters:

sharedRuns a pointer to the OgreSharedRuns this run belongs to

name the name of the depth texture to be created

playerView pointer to the player's viewport

5.56.3 Member Function Documentation

5.56.3.1 void OgreSceneCameraDepthRenderingRun::updateFrame (unsigned long *frameNum*) [protected, virtual]

This function does the actual update in a frame.

Parameters:

frameNum current frame number

Implements SceneCameraDepthRenderingRun.

5.56.3.2 OgreRenderingRun* OgreRenderingRun::asOgreRenderingRun () [inline, virtual, inherited]

Conversion to OgreRenderRun.

This function is needed because of virtual inheritance.

Reimplemented from RenderingRun.

5.56.3.3 OgreRenderingRun* OgreRenderingRun::asOgreRenderingRun () [inline, virtual, inherited]

Conversion to OgreRenderRun.

This function is needed because of virtual inheritance.

Reimplemented from RenderingRun.

5.56.3.4 Vector3 OgreRenderingRun::getCubeMapFaceDirection (unsigned char *faceId*) [protected, inherited]

Returns a direction for a cubemap face id.

This is a helper function to retrieve the normal direction of a given cubemap face.

Parameters:

faceId the number of the face

5.56.3.5 Vector3 OgreRenderingRun::getCubeMapFaceDirection (unsigned char *faceId*) [protected, inherited]

Returns a direction for a cubemap face id.

This is a helper function to retrieve the normal direction of a given cubemap face.

Parameters:

faceId the number of the face

5.56.3.6 Texture * OgreRenderingRun::createCubeRenderTexture (String *name*, const Vector3 *position*, unsigned int *resolution* = 512, PixelFormat *format* = PF_FLOAT16_RGBA, int *numMips* = 0, ColourValue *clearColor* = ColourValue::Black) [protected, inherited]

Creates a cubemap texture.

This is a helper function to easily create a cubemap texture and automaticly attach viewports to each face so it can be used as a rendertarget.

Parameters:

name the name of the texture to be created

position the initial position of the cubemap

resolution the resolution of one cubemapface

format the pixel format of the cubemap

numMips the number of mipmap levels

clearColor initial color

5.56.3.7 Texture* OgreRenderingRun::createCubeRenderTexture (String *name*, const Vector3 *position*, unsigned int *resolution* = 512, PixelFormat *format* = PF_FLOAT16_RGBA, int *numMips* = 0, ColourValue *clearColor* = ColourValue::Black) [protected, inherited]

Creates a cubemap texture.

This is a helper function to easily create a cubemap texture and automaticly attach viewports to each face so it can be used as a rendertarget.

Parameters:

name the name of the texture to be created

position the initial position of the cubemap

resolution the resolution of one cubemapface

format the pixel format of the cubemap

numMips the number of mipmap levels

clearColor initial color

5.56.3.8 void OgreRenderingRun::setMaterialForRenderables (String & *materialName*, RenderQueue * *rq*) [protected, inherited]

Sets the given material for each Renderable in a RenderQueue.

This is a helper function to set a material to each element of a previously filled Renderque. The orginal material of the Renderables are stored so they can be restored later. The function also tells the current SceneManager not to search for visible objects, as we are going to use the given RenderQueue during the next rendering.

Parameters:

materialName the name of the material to set for the Renderables

rq pointer to the filled Renderqueue instance to set material for

5.56.3.9 void OgreRenderingRun::setMaterialForRenderables (String & *materialName*, RenderQueue * *rq*, bool *solidsonly* = false) [protected, inherited]

Sets the given material for each Renderable in a RenderQueue.

This is a helper function to set a material to each element of a previously filled Renderque. The orginal material of the Renderables are stored so they can be restored later. The function also tells the current SceneManager not to search for visible objects, as we are going to use the given RenderQueue during the next rendering.

Parameters:

materialName the name of the material to set for the Renderables

rq pointer to the filled Renderqueue instance to set material for

5.56.3.10 void OgreRenderingRun::setMaterialForVisibles (String & *materialName*, Camera * *cam*, bool *shadowcastersonly* = false) [protected, inherited]

Sets the given material for each Renderable visible from a given camera.

This helper function is similar to setMaterialForRenderables but it is also responsible for filling the RenderQueue. First the RenderQueue of the current SceneManager fill be filled with the visible objects seen from the given camera. Then the required material will be set for each element of the RenderQueue. The orginal material of the Renderables are stored so they can be restored later. The function also tells the current SceneManager not to search for visible objects, as we are going to use the filled RenderQueue during the next rendering.

Parameters:

materialName the name of the material to set for the Renderables

cam pointer to the camera from which visible objects should be searched

shadowcastersonly flag to search for only shadow casters

5.56.3.11 void OgreRenderingRun::setMaterialForVisibles (String & *materialName*, Camera * *cam*, bool *shadowcastersonly* = false, bool *solidsonly* = false) [protected, inherited]

Sets the given material for each Renderable visible from a given camera.

This helper function is similar to setMaterialForRenderables but it is also responsible for filling the RenderQueue. First the RenderQueue of the current SceneManager fill be filled with the visible objects seen from the given camera. Then the required material will be set for each element of the RenderQueue. The orginal material of the Renderables are stored so they can be restored later. The function also tells the current SceneManager not to search for visible objects, as we are going to use the filled RenderQueue during the next rendering.

Parameters:

materialName the name of the material to set for the Renderables

cam pointer to the camera from which visible objects should be searched

shadowcastersonly flag to search for only shadow casters

5.56.3.12 void OgreRenderingRun::restoreMaterials () [protected, inherited]

Restores previously stored materials.

This helper function is typically used after a setMaterialForRenderables or setMaterialForVisibles call and a rendering process to restore the original material settings. The function also tells the current SceneManager to search for visible objects, as this is the default behaviour of SceneManager.

5.56.3.13 void OgreRenderingRun::restoreMaterials () [protected, inherited]

Restores previously stored materials.

This helper function is typically used after a setMaterialForRenderables or setMaterialForVisibles call and a rendering process to restore the original material settings. The function also tells the current SceneManager to search for visible objects, as this is the default behaviour of SceneManager.

5.56.3.14 void OgreRenderingRun::renderFullscreenQuad (String *materialName*, RenderTarget * *target*) [protected, inherited]

Renderes a full screen quad on a given RendderTarget with a given material.

Parameters:

materialName the name of the material bind to the quad

target the RenderTarget the quad should be rendered on

5.56.3.15 void OgreRenderingRun::renderFullscreenQuad (String *materialName*, RenderTarget * *target*) [protected, inherited]

Renderes a full screen quad on a given RendderTarget with a given material.

Parameters:

materialName the name of the material bind to the quad

target the RenderTarget the quad should be rendered on

5.56.3.16 void OgreRenderingRun::renderPixelSprites (String & *materialName*, RenderTarget * *rt*, int *width*, int *height*) [protected, inherited]

Renderes sprites to pixels of the screen on a given RendderTarget with a given material.

Pixel sprites are pixel sized quads, placed on each pixel of the RenderTarget. The number of sprites not necessary corresponds to the resolution of the rendertarget. The pixel quads will evenly fill the rendertarget's area with sizes corresponding to the given resolution. We can render fewer or more pixel quads than the number of pixels the rendertarget has (eg.: in case of caustic cubemap generation).

Parameters:

materialName the name of the material bind to the pixel sprites

rt the RenderTarget the quads should be rendered on

width the desired resolution width of the sprites

height the desired resolution height of the sprites

5.56.3.17 void OgreRenderingRun::renderPixelSprites (String & *materialName*, RenderTarget * *rt*, int *width*, int *height*) [protected, inherited]

Renderes sprites to pixels of the screen on a given RendderTarget with a given material.

Pixel sprites are pixel sized quads, placed on each pixel of the RenderTarget. The number of sprites not necessary corresponds to the resolution of the rendertarget. The pixel quads will evenly fill the rendertarget's area with sizes corresponding to the given resolution. We can render fewer or more pixel quads than the number of pixels the rendertarget has (eg.: in case of caustic cubemap generation).

Parameters:

materialName the name of the material bind to the pixel sprites

rt the RenderTarget the quads should be rendered on

width the desired resolution width of the sprites

height the desired resolution height of the sprites

5.56.3.18 void OgreRenderingRun::renderFullscreenGrid (String & *materialName*, RenderTarget * *rt*, int *width*, int *height*) [protected, inherited]

Renders a grid onto the screen.

Parameters:

rt the RenderTarget the grid should be rendered on

width the desired horizontal resolution of the grid

height the desired vertical resolution of the grid

5.56.3.19 virtual bool RenderingRun::canJoin (RenderingRun * *run*) [inline, virtual, inherited]

Returns true if two runs can be joined.

In some cases special requirements should stand to join two runs (even if they have the same type). Eg.: two caustic cube map generation technique should only be joined if they use the same material when rendering the caustic cubemap.

Reimplemented in OgreCausticCubeMapRenderingRun, OgreChildPSystemRenderingRun, and OgrePhotonMapRenderingRun.

5.56.3.20 virtual bool RenderingRun::needUpdate (unsigned long *frameNum*) [inline, protected, virtual, inherited]

Returns if this run needs update.

This tipically depends on the upate interval and the starting frame number.

Parameters:

frameNum current frame number

Reimplemented in OgreFocusingMapRenderingRun, and OgrePMEntryPointMapRenderingRun.

5.56.4 Member Data Documentation

5.56.4.1 std::map<Renderable*, String> OgreRenderingRun::visibleObjects [protected, inherited]

map of Renderables which will be rendered with a given material

The String stores the original material name that will be restored after rendering.

See also:

setMaterialForVisibles
setMaterialForRenderables
restoreMaterials

5.56.4.2 std::map<Renderable*, String> OgreRenderingRun::visibleObjects [protected, inherited]

map of Renderables which will be rendered with a given material

The String stores the original material name that will be restored after rendering.

See also:

setMaterialForVisibles
setMaterialForRenderables
restoreMaterials

5.56.4.3 MovablePlane * OgreRenderingRun::fullScreenQuad [static, protected, inherited]

fulls screen quad plane used in full screen quad rendering

See also:

renderFullscreenQuad

5.56.4.4 MovablePlane* OgreRenderingRun::fullScreenQuad [static, protected, inherited]

fulls screen quad plane used in full screen quad rendering

See also:

renderFullscreenQuad

5.56.4.5 Entity * OgreRenderingRun::fullScreenQuadEntity [static, protected, inherited]

fulls screen quad Entity used in full screen quad rendering

See also:

renderFullscreenQuad

5.56.4.6 Entity* OgreRenderingRun::fullScreenQuadEntity [static, protected, inherited]

fulls screen quad Entity used in full screen quad rendering

See also:

renderFullscreenQuad

5.56.4.7 SceneNode * OgreRenderingRun::fullScreenQuadNode [static, protected, inherited]

fulls screen quad SceneNode used in full screen quad rendering

See also:

renderFullscreenQuad

5.56.4.8 SceneNode* OgreRenderingRun::fullScreenQuadNode [static, protected, inherited]

fulls screen quad SceneNode used in full screen quad rendering

See also:

renderFullscreenQuad

5.56.4.9 SpriteSet* OgreRenderingRun::pixelSprites [protected, inherited]

SpriteSet used in pixel sprite rendering.

See also:

renderPixelSprites

5.56.4.10 BillboardSet* OgreRenderingRun::pixelSprites [protected, inherited]

SpriteSet used in pixel sprite rendering.

See also:

renderPixelSprites

5.56.4.11 String OgreRenderingRun::spriteSetName [protected, inherited]

unique name of the SpriteSet used in pixel sprite rendering

See also:

renderPixelSprites

5.56.4.12 Entity* OgreRenderingRun::fullscreenGrid [protected, inherited]

Entity used in fullscreen grid rendering.

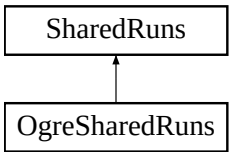
See also:

renderPixelGrid

5.57 OgreSharedRuns Class Reference

Class of SharedRuns used in an OGRE environment.

Inheritance diagram for OgreSharedRuns::



Public Member Functions

- OgreSharedRuns ()

  Constructor.

- std::map< RenderingRunType, RenderingRun * > & getSharedRuns ()

  Retrieves the contaied RenderingRuns with their type information.

- void setBoundingSphere (Sphere &sphere)

  Sets the bounding sphere of the node.

- void setBoundingBox (AxisAlignedBox &box)

  Sets the axis-aligned bounding box of the node.

- Sphere & getBoundingSphere ()

  Returns the bounding sphere of the node.

- AxisAlignedBox & getBoundingBox ()

  Returns the axis-aligned bounding box of the node.

- Sphere & getRootBoundingSphere ()

  Returns the bounding sphere of the root parent node.

- Sphere & getRootBoundingSphere (RenderingRunType runType)

  Retrieves the bounding sphere of the topmost parent node of this SharedRuns node, which have a specified RenderingRun type.

- AxisAlignedBox & getRootBoundingBox ()

  Returns the axis-aligned bounding box of the root parent node.

- AxisAlignedBox & getRootBoundingBox (RenderingRunType runType)

  Retrieves the axis-aligned bounding box of the topmost parent node of this SharedRuns node, which have a specified RenderingRun type.

- const Vector3 & getRootPosition ()

Returns the world space center position of the root parent node.

- const Vector3 & getRootPosition (RenderingRunType runType)

  Retrieves the world space center position of the topmost parent node of this SharedRuns node, which have a specified RenderingRun type.

- bool hasOwnRun (RenderingRunType runType)

  Checks if this node has a resource with the given type.

- void addRenderablesToQueue (RenderQueue *rq, bool checkVisible=true)

  Adds all the Renderables connected to this node to a given RenderQueue.

- void notifyCamera (Camera *cam)

  Calls notifyCamera() to all the Renderables connected to this node.

- void findSharedRootsForType (RenderingRunType runType, std::vector< OgreSharedRuns * > &roots)

  Finds all the topmost nodes which have resources of the given type.

- RenderingRun * getRun (RenderingRunType runType)
- void addRun (RenderingRunType runType, RenderingRun *run)
- void updateRun (RenderingRunType runType, unsigned long frameNum)
- void updateBounds ()

  Updates the boundary of this SharedRuns (and also it's parent).

- void validate ()

  Validate this SharedRuns (and also all childs).

- void destroy ()

  Destroys the node (and all parents recursively).

- void runUpdated (RenderingRunType runType, RenderingRun *run)

  Called after one of he shared runs updates.

- void runChanged (RenderingRunType runType, RenderingRun *run)

  Called after one of he shared runs changes.

- virtual void addTechniqueGroup (TechniqueGroup *group)

  Adds a child TechniqueGroup.

- void setMaterial (String materialName)

  Sets the given material for all connected renderables.

- void restoreMaterial ()

  Restores the prevoius materials for the connected renderables.

- virtual SharedRuns * joinRuns (SharedRuns *otherRuns)
- virtual void setVisible (bool visible)

  Shows or hides this SharedRuns (and also all childnodes).

- virtual void hide ()

  Hides this SharedRuns (and also all childs).

- virtual void restoreVisibility ()

  Restores the visibility of this SharedRuns (and also all childs).

- virtual SharedRuns * getRoot ()

  Retrieves the root node of this SharedRuns node.

- virtual SharedRuns * getRoot (RenderingRunType runType)

  Retrieves the topmost parent node of this SharedRuns node, which have a specified RenderingRun type.

- virtual void unbindParent ()

  Unbinds the parent of the node, called at splitting.

- virtual void unbindAndKillParent ()

  Unbinds the deletes the parent of the node, called at splitting.

Static Public Member Functions

- static bool canJoin (SharedRuns *r1, SharedRuns *r2)

  Checks if two SharedRuns node can be joined.

- static bool haveCommonRuns (SharedRuns *r1, SharedRuns *r2, std::vector< RenderingRunType > &commonruns)

  Checks if two SharedRuns have common resources so that they can be joined.

Protected Member Functions

- void gatherRuns ()

  Collects RenderingRuns references from the child nodes, used when joining.

- void fireRunChanges ()

  Sends runChanged events for each RenderingRun type, used after join and split.

- SharedRuns * createInstance ()

  Creates a new SharedRuns instance. All derivatives should implement this.

- void hideRenderables ()

  Hides all the connected renderables, only used if this is a leaf.

- void restoreRenderableVisibility ()

  Restires visibility of all the connected renderables, only used if this is a leaf.

- void setRenderablesVisible (bool visible)

  Set visibility of connected renderables, only used if this is a leaf.

Protected Attributes

- std::vector< TechniqueGroup * > childTechniqueGroups
  child TechniqueGroup instance.
- std::map< RenderingRunType, RenderingRun * > sharedRuns
  map of contained RenderingRuns
- std::map< OgreRenderable *, bool > renderables
  map of connected renderables with visibility information
- std::map< OgreRenderable *, String > renderableMaterials
  map of connected renderables with material name information
- Sphere boundingSphere
  the bounding sphere of all the objects connected to this node.
- AxisAlignedBox boundingBox
  the axis aligned bounding box of all the objects connected to this node.
- SharedRuns * parent
  parent SharedRuns instance
- SharedRuns * child1
  child SharedRuns instance
- SharedRuns * child2
  child SharedRuns instance

5.57.1 Detailed Description

Class of SharedRuns used in an OGRE environment.

5.57.2 Member Function Documentation

5.57.2.1 std::map<RenderingRunType, RenderingRun*>& OgreSharedRuns::getSharedRuns () [inline]

Retrieves the contaied RenderingRuns with their type information.

Returns:

map of renderables

5.57.2.2 void OgreSharedRuns::setBoundingSphere (Sphere & *sphere*) [inline]

Sets the bounding sphere of the node.

Parameters:

sphere bounding sphere

5.57.2.3 void OgreSharedRuns::setBoundingBox (AxisAlignedBox & *box*) [inline]

Sets the axis-aligned bounding box of the node.

Parameters:

box bounding box

5.57.2.4 Sphere& OgreSharedRuns::getBoundingSphere () [inline]

Returns the bounding sphere of the node.

Returns:

a reference to the bounding sphere

5.57.2.5 AxisAlignedBox& OgreSharedRuns::getBoundingBox () [inline]

Returns the axis-aligned bounding box of the node.

Returns:

a reference to the bounding box

5.57.2.6 Sphere& OgreSharedRuns::getRootBoundingSphere () [inline]

Returns the bounding sphere of the root parent node.

Returns:

a reference to the bounding sphere

5.57.2.7 Sphere& OgreSharedRuns::getRootBoundingSphere (RenderingRunType *runType*) [inline]

Retrieves the bounding sphere of the topmost parent node of this SharedRuns node, which have a specified RenderingRun type.

Parameters:

runType the RenderingRun type

Returns:

a reference to the bounding sphere

5.57.2.8 AxisAlignedBox& OgreSharedRuns::getRootBoundingBox () [inline]

Returns the axis-aligned bounding box of the root parent node.

Returns:

a reference to the bounding box

5.57.2.9 AxisAlignedBox& OgreSharedRuns::getRootBoundingBox (RenderingRunType *runType*) [inline]

Retrieves the axis-aligned bounding box of the topmost parent node of this SharedRuns node, which have a specified RenderingRun type.

Parameters:

runType the RenderingRun type

Returns:

a reference to the bounding box

5.57.2.10 const Vector3& OgreSharedRuns::getRootPosition () [inline]

Returns the world space center position of the root parent node.

Returns:

a reference to the center position

5.57.2.11 const Vector3& OgreSharedRuns::getRootPosition (RenderingRunType *runType*) [inline]

Retrieves the world space center position of the topmost parent node of this SharedRuns node, which have a specified RenderingRun type.

Parameters:

runType the RenderingRun type

Returns:

a reference to the center position

5.57.2.12 bool OgreSharedRuns::canJoin (SharedRuns * *r1*, SharedRuns * *r2*) [static]

Checks if two SharedRuns node can be joined.

Parameters:

r1 pointer to one of the SharedRuns instance

r2 pointer to the other SharedRuns instance

5.57.2.13 bool OgreSharedRuns::haveCommonRuns (SharedRuns * *r1*, SharedRuns * *r2*, std::vector< RenderingRunType > & *commonruns*) [static]

Checks if two SharedRuns have common resources so that they can be joined.

Parameters:

r1 pointer to one of the SharedRuns instance

r2 pointer to the other SharedRuns instance

5.57.2.14 bool OgreSharedRuns::hasOwnRun (RenderingRunType *runType*) [virtual]

Checks if this node has a resource with the given type.

Only this node none of the child nodes are checked.

Parameters:

the type of the RenderingRun to look for

Implements SharedRuns.

5.57.2.15 void OgreSharedRuns::addRenderablesToQueue (RenderQueue * *rq*, bool *checkVisible* = `true`)

Adds all the Renderables connected to this node to a given RenderQueue.

The function is called recoursively for all child nodes.

Parameters:

pointer to the RenderQueue to add the Renderables to

5.57.2.16 void OgreSharedRuns::notifyCamera (Camera * *cam*)

Calls notifyCamera() to all the Renderables connected to this node.

The function is called recoursively for all child nodes.

Parameters:

pointer to the Camera instance to call notifyCamera() with

5.57.2.17 void OgreSharedRuns::findSharedRootsForType (RenderingRunType *runType*, std::vector< OgreSharedRuns * > & *roots*)

Finds all the topmost nodes which have resources of the given type.

This function is called for the root node, and will be called recoursively downwards in the tree for all childs. If a node with with the given resource is found, the node can be added and it's childs don't need to be checked anymore.

From a given group of objects (root node) a new set of groups will be created (they will be the members of the given vector). Each new group will be a subtree of the original tree. They will form groups that contain the maximum number of objects that can be joined by the the given resource type.

Parameters:

runType the type of RenderingRun to look for

roots reference to the collection to add the new groups to

5.57.2.18 RenderingRun * OgreSharedRuns::getRun (RenderingRunType *runType*) `[virtual]`

&brief Retrieves a shared resource.

Parameters:

runType enum, type of the RenderingRun to be retrieved

Returns:

pointer to the RenderingRun of type "runType", null if this type does not exists

Implements SharedRuns.

5.57.2.19 void OgreSharedRuns::addRun (RenderingRunType *runType*, RenderingRun * *run*) [virtual]

&brief Adds a RenderingRun instance to the shared resources.

Parameters:

runType enum, type of the RenderingRun to add

run pointer to the RenderingRun instance to add

Implements SharedRuns.

5.57.2.20 void OgreSharedRuns::updateRun (RenderingRunType *runType*, unsigned long *frameNum*) [virtual]

&brief Updates a shared RenderingRun.

Parameters:

runType enum, type of the RenderingRun to update

frameNum current framenumber

Implements SharedRuns.

5.57.2.21 void OgreSharedRuns::validate () [virtual]

Validate this SharedRuns (and also all childs).

Validation meens that all the SharedRuns that are joined will be examined if the sharing is still valid. If it finds out that two SharedRuns can't be joined anymore (eg.: they moved far from each other), their parent will be split and destroyed (all parent of this node also should be deleted recursively).

Implements SharedRuns.

5.57.2.22 void OgreSharedRuns::runUpdated (RenderingRunType *runType*, RenderingRun * *run*) [virtual]

Called after one of he shared runs updates.

This message will be forwarded to each child.

Parameters:

runType enum describing the type of the updated run

run pointer to the updated RenderingRun

Reimplemented from SharedRuns.

5.57.2.23 void OgreSharedRuns::runChanged (RenderingRunType *runType*, RenderingRun * *run*) [virtual]

Called after one of he shared runs changes.

This message will be forwarded to each child.

Parameters:

runType enum describing the type of the changed run

run pointer to the changed RenderingRun

Reimplemented from SharedRuns.

5.57.2.24 virtual void OgreSharedRuns::addTechniqueGroup (TechniqueGroup * *group*) [inline, virtual]

Adds a child TechniqueGroup.

Parameters:

group pointer to the TechniqueGroup instance to add.

Implements SharedRuns.

5.57.2.25 void OgreSharedRuns::setMaterial (String *materialName*)

Sets the given material for all connected renderables.

The previous materials will be stored so later can be restored.

See also:

restoreMaterial

Parameters:

name of the material to be set.

5.57.2.26 void OgreSharedRuns::restoreMaterial ()

Restores the prevoius materials for the connected renderables.

See also:

setMaterial

5.57.2.27 SharedRuns * OgreSharedRuns::createInstance () [protected, virtual]

Creates a new SharedRuns instance. All derivatives should implement this.

Returns:

a new SharedRuns instance

Implements SharedRuns.

5.57.2.28 void OgreSharedRuns::setRenderablesVisible (bool *visible*) [inline, protected, virtual]

Set visibility of connected renderables, only used if this is a leaf.

Parameters:

visible visibility

Implements SharedRuns.

5.57.2.29 SharedRuns * SharedRuns::joinRuns (SharedRuns * *otherRuns*) [virtual, inherited]

&brief Joines two SharedRuns.

The resulting SharedRuns become the parent of the two SharedRuns.

Parameters:

otherRuns pointer to the SharedRuns instance to join with

Returns:

the new parent SharedRuns instance

5.57.2.30 void SharedRuns::setVisible (bool *visible*) [virtual, inherited]

Shows or hides this SharedRuns (and also all childnodes).

Parameters:

visible visibility

5.57.2.31 void SharedRuns::hide () [virtual, inherited]

Hides this SharedRuns (and also all childs).

The previous visibility is saved.

5.57.2.32 SharedRuns * SharedRuns::getRoot () [virtual, inherited]

Retrieves the root node of this SharedRuns node.

Returns:

pointer to the root SharedRuns instance

5.57.2.33 SharedRuns * SharedRuns::getRoot (RenderingRunType *runType*) [virtual, inherited]

Retrieves the topmost parent node of this SharedRuns node, which have a specified RenderingRun type.

Parameters:

runType the RenderingRun type

Returns:

pointer to the parent SharedRuns instance

5.57.3 Member Data Documentation

5.57.3.1 std::vector<TechniqueGroup*> OgreSharedRuns::childTechniqueGroups [protected]

child TechniqueGroup instance.

If this SharedRuns node is a leaf, it containes a reference to a TechniqueGroup instance. All messages will be transfered to this object, and bounding information will be retrieved from this TechniqueGroup

5.57.3.2 std::map<OgreRenderable*, bool> OgreSharedRuns::renderables [protected]

map of connected renderables with visibility information

Used to show or hide the renderables connected to a leaf OgreSharedRuns node.

5.57.3.3 std::map<OgreRenderable*, String> OgreSharedRuns::renderableMaterials [protected]

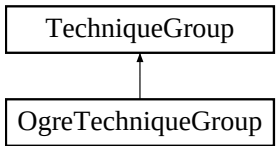
map of connected renderables with material name information

Used to resture the original materials of the renderables connected to a leaf OgreSharedRuns node.

5.58 OgreTechniqueGroup Class Reference

Base class of a SharedRuns in an OGRE environment.

Inheritance diagram for OgreTechniqueGroup::



Public Member Functions

- void addRenderTechnique (RenderTechnique *technique)
  Adds a rendertechnique to the group.
- void update (unsigned long frameNum)
  Updates all rendertechniques.
- void runChanged (RenderingRunType runType, RenderingRun *run)
  Called after one of the rendering runs changes.
- void runUpdated (RenderingRunType runType, RenderingRun *run)
  Called after one of the rendering runs updates.
- void updateBounds ()
  Updates the connected SharedRuns boundary.
- void addSharedRun (SharedRuns *sharedRuns)
  Adds an empty SharedRuns parent.
- SharedRuns * getSharedRuns ()
  Retrieves the shared runs.
- virtual void validateSharedRuns ()
  Validates the connected SharedRuns instance.

Protected Attributes

- std::vector< OgreRenderTechnique * > renderTechniques
  Collection of OgreRenderTechniques.
- SharedRuns * parentSharedRuns
  Pointer to the connected SharedRuns instance each technique uses.

5.58.1 Detailed Description

Base class of a SharedRuns in an OGRE environment.

5.58.2 Member Function Documentation

5.58.2.1 void OgreTechniqueGroup::addRenderTechnique (RenderTechnique * *technique*) [virtual]

Adds a rendertechnique to the group.

Parameters:

technique the RenderTechnique instance to add.

Implements TechniqueGroup.

5.58.2.2 void OgreTechniqueGroup::update (unsigned long *frameNum*) [virtual]

Updates all rendertechniques.

Parameters:

framenum current framenumber

Implements TechniqueGroup.

5.58.2.3 void OgreTechniqueGroup::runChanged (RenderingRunType *runType*, RenderingRun * *run*) [virtual]

Called after one of the rendering runs changes.

This message will be forwarded to each RenderTechique.

Parameters:

runType enum describing the type of the changed run

run pointer to the changed RenderingRun

Implements TechniqueGroup.

5.58.2.4 void OgreTechniqueGroup::runUpdated (RenderingRunType *runType*, RenderingRun * *run*) [virtual]

Called after one of the rendering runs updates.

This message will be forwarded to each RenderTechique.

Parameters:

runType enum describing the type of the updated run

run pointer to the updated RenderingRun

Implements TechniqueGroup.

5.58.2.5 void TechniqueGroup::addSharedRun (SharedRuns * *sharedRuns*) [inline, inherited]

Adds an empty SharedRuns parent.

Used after creating a new TechniqueGroup.

Parameters:

sharedRuns pointer to the SharedRun instance this RenderTechniques will use.

5.58.2.6 SharedRuns* TechniqueGroup::getSharedRuns () [inline, inherited]

Retrieves the shared runs.

Returns:

pointer to the SharedRun instance this RenderTechniques uses.

5.58.3 Member Data Documentation

5.58.3.1 std::vector<OgreRenderTechnique*> OgreTechniqueGroup::renderTechniques [protected]

Collection of OgreRenderTechniques.

All messages will be forwarded to each element of this vector.

5.59 PathMapClusters Struct Reference

Structure to store path map cluster information for a subentity.

Public Attributes

- unsigned int count
  the number of clusters this subentity belongs to
- unsigned int * clusters
  the indices of the cluster this subentity belongs to.
- String pathMapTextureFilename
  the name of the path map file this subentity uses
- unsigned int pathMapResolution
  the resolution of the path map file.

5.59.1 Detailed Description

Structure to store path map cluster information for a subentity.

5.60 PathMapEntryPoint Struct Reference

Structure of a path map entry point.

Public Attributes

- Vector3 position
  the position of the entry point.
- Vector3 normal
  the normal of the entry point.
- float prob
  the probability of the entry point.

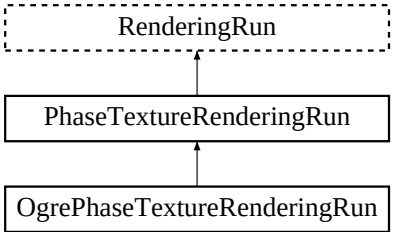
5.60.1 Detailed Description

Structure of a path map entry point.

5.61 PhaseTextureRenderingRun Class Reference

Base abstract class that defines a rendering process that creates phase texture.

Inheritance diagram for PhaseTextureRenderingRun::



Public Member Functions

- PhaseTextureRenderingRun (unsigned int resolutionX, unsigned int resolutionY)

  Constructor.

- bool update (unsigned long frameNum)

  Calls updateFrame() if the run needs update according to its starting frame and update interval and has not been allready updated in this frame.

- virtual class OgreRenderingRun * asOgreRenderingRun ()

  Conversion to OgreRenderRun.

- virtual bool canJoin (RenderingRun *run)

  Returns true if two runs can be joined.

Protected Member Functions

- virtual void createPhaseTexture ()=0

  Creates the depth map texture.

- virtual void updateFrame (unsigned long frameNum)=0

  This function does the actual update in a frame.

- virtual bool needUpdate (unsigned long frameNum)

  Returns if this run needs update.

Protected Attributes

- unsigned int resolutionX

  width of the texture

- unsigned int resolutionY
  height of the texture
- unsigned long lastupdated
  The number of the last frame this run was updated.
- unsigned long startFrame
  The number of the frame this run should be updated first.
- unsigned long updateInterval
  Refresh frequency in frames.

5.61.1 Detailed Description

Base abstract class that defines a rendering process that creates phase texture.

The phase texture can be used as a look-up table during rendering of partcipating media. If the texture is created once it can be saved and reused, so this run is usually not needed, only runned once. This texture is addressed as the following: the u coordinates represents the symmetry of scattering (negative if backward scattering, positive if forward scattering and zero if equally scaterring in the forward and in the backward directions) ; the v coordinates represent the cosine of the angle between the incoming and outgoing directions.

5.61.2 Constructor & Destructor Documentation

5.61.2.1 PhaseTextureRenderingRun::PhaseTextureRenderingRun (unsigned int *resolutionX*, unsigned int *resolutionY*) [inline]

Constructor.

Parameters:

resolutionX width of the texture
resolutionY height of the texture

5.61.3 Member Function Documentation

5.61.3.1 virtual void PhaseTextureRenderingRun::updateFrame (unsigned long *frameNum*) [protected, pure virtual]

This function does the actual update in a frame.

Parameters:

frameNum current frame number

Reimplemented from RenderingRun.

Implemented in OgrePhaseTextureRenderingRun.

5.61.3.2 virtual class OgreRenderingRun* RenderingRun::asOgreRenderingRun () [inline, virtual, inherited]

Conversion to OgreRenderRun.

This function is needed because of virtual inheritance.

Reimplemented in OgreRenderingRun, and OgreRenderingRun.

5.61.3.3 virtual bool RenderingRun::canJoin (RenderingRun * *run*) [inline, virtual, inherited]

Returns true if two runs can be joined.

In some cases special requirements should stand to join two runs (even if they have the same type). Eg.: two caustic cube map generation technique should only be joined if they use the same material when rendering the caustic cubemap.

Reimplemented in OgreCausticCubeMapRenderingRun, OgreChildPSystemRenderingRun, and OgrePhotonMapRenderingRun.

5.61.3.4 virtual bool RenderingRun::needUpdate (unsigned long *frameNum*) [inline, protected, virtual, inherited]

Returns if this run needs update.

This tipically depends on the upate interval and the starting frame number.

Parameters:

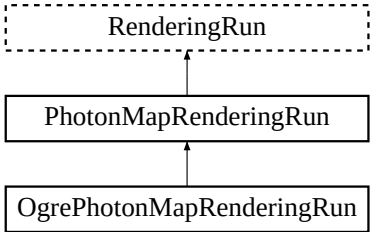
frameNum current frame number

Reimplemented in OgreFocusingMapRenderingRun, and OgrePMEntryPointMapRenderingRun.

5.62 PhotonMapRenderingRun Class Reference

Base abstract class that defines a rendering process of a photon hit map.

Inheritance diagram for PhotonMapRenderingRun::



Public Member Functions

- PhotonMapRenderingRun (unsigned long startFrame, unsigned long updateInterval)

  Constructor.

- bool update (unsigned long frameNum)

  Calls updateFrame() if the run needs update according to its starting frame and update interval and has not been allready updated in this frame.

- virtual class OgreRenderingRun * asOgreRenderingRun ()

  Conversion to OgreRenderRun.

- virtual bool canJoin (RenderingRun *run)

  Returns true if two runs can be joined.

Protected Member Functions

- virtual void updateFrame (unsigned long frameNum)=0

  This function does the actual update in a frame.

- virtual void createPhotonMap ()=0

  Creates a photon hit map.

- virtual bool needUpdate (unsigned long frameNum)

  Returns if this run needs update.

Protected Attributes

- unsigned long lastupdated

  The number of the last frame this run was updated.

- unsigned long startFrame

  The number of the frame this run should be updated first.

- unsigned long updateInterval

  Refresh frequency in frames.

5.62.1 Detailed Description

Base abstract class that defines a rendering process of a photon hit map.

A photon hit map stores the directions where the incoming photons are refracted by a caustic emitter object. One pixel of the photon map represents one photon hit, hte direction is encoded in the RGB channels. If the alpha channel has zero value, the hit is invalid.

5.62.2 Constructor & Destructor Documentation

5.62.2.1 PhotonMapRenderingRun::PhotonMapRenderingRun (unsigned long *startFrame*, unsigned long *updateInterval*) [inline]

Constructor.

Parameters:

startFrame adds an offset to the current frame number to help evenly distribute updates between frames

updateInterval update frequency

5.62.3 Member Function Documentation

5.62.3.1 virtual void PhotonMapRenderingRun::updateFrame (unsigned long *frameNum*) [protected, pure virtual]

This function does the actual update in a frame.

Parameters:

frameNum current frame number

Reimplemented from RenderingRun.

Implemented in OgrePhotonMapRenderingRun.

5.62.3.2 virtual class OgreRenderingRun* RenderingRun::asOgreRenderingRun () [inline, virtual, inherited]

Conversion to OgreRenderRun.

This function is needed because of virtual inheritance.

Reimplemented in OgreRenderingRun, and OgreRenderingRun.

5.62.3.3 virtual bool RenderingRun::canJoin (RenderingRun * *run*) [inline, virtual, inherited]

Returns true if two runs can be joined.

In some cases special requirements should stand to join two runs (even if they have the same type). Eg.: two caustic cube map generation technique should only be joined if they use the same material when rendering the caustic cubemap.

Reimplemented in OgreCausticCubeMapRenderingRun, OgreChildPSystemRenderingRun, and OgrePhotonMapRenderingRun.

5.62.3.4 virtual bool RenderingRun::needUpdate (unsigned long *frameNum*) [inline, protected, virtual, inherited]

Returns if this run needs update.

This tipically depends on the upate interval and the starting frame number.

Parameters:

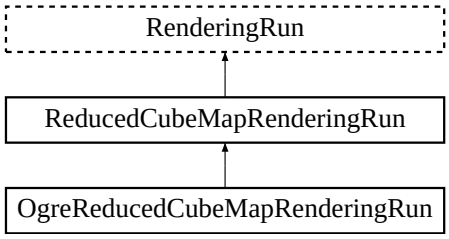
frameNum current frame number

Reimplemented in OgreFocusingMapRenderingRun, and OgrePMEntryPointMapRenderingRun.

5.63 ReducedCubeMapRenderingRun Class Reference

Base abstract class that defines a rendering process of a downsampled color-cubemap.

Inheritance diagram for ReducedCubeMapRenderingRun::



Public Member Functions

- ReducedCubeMapRenderingRun (unsigned long startFrame, unsigned long updateInterval, unsigned int resolution, bool useDistCalc, bool useFaceAngleCalc, float distTolerance, float angleTolerance, bool updateAllFace)

  Constructor.

- bool update (unsigned long frameNum)

  Calls updateFrame() if the run needs update according to its starting frame and update interval and has not been allready updated in this frame.

- virtual class OgreRenderingRun * asOgreRenderingRun ()

  Conversion to OgreRenderRun.

- virtual bool canJoin (RenderingRun *run)

  Returns true if two runs can be joined.

Protected Member Functions

- virtual void createReducedCubeMap ()=0

  Creates the reduced size cubemap texture.

- virtual void updateCubeFace (int facenum)=0

  Updates one face of the cubemap.

- virtual bool faceNeedsUpdate (int facenum)=0

  Checks if a cubemap face needs to be updated.

- virtual void updateFrame (unsigned long frameNum)

  This function does the actual update in a frame.

- virtual bool needUpdate (unsigned long frameNum)

  Returns if this run needs update.

Protected Attributes

- bool updateAllFace

  defines if all cubemap faces should be updated in a frame or only one face per frame

- unsigned char currentFace

  the number of the face to be updated

- unsigned int resolution

  the resolution of the cubemap texture that was created by this run

- bool useDistCalc

  a flag to skip cube face update if object is far away or too small.

- bool useFaceAngleCalc

  a flag to skip cube face update the face is neglible.

- float distTolerance

  A value used in face skip test.

- float angleTolerance

  A value used in face skip test.

- unsigned long lastupdated

  The number of the last frame this run was updated.

- unsigned long startFrame

  The number of the frame this run should be updated first.

- unsigned long updateInterval

  Refresh frequency in frames.

5.63.1 Detailed Description

Base abstract class that defines a rendering process of a downsampled color-cubemap.

The resulting cubemap is a lower resolution variation of the color cube map. It is created with averaging the original cubemap. The lower resolution cubemap can be convolved faster and can efficiently be used in effects like diffuse reflection.

5.63.2 Constructor & Destructor Documentation

5.63.2.1 ReducedCubeMapRenderingRun::ReducedCubeMapRenderingRun (unsigned long *startFrame*, unsigned long *updateInterval*, unsigned int *resolution*, bool *useDistCalc*, bool *useFaceAngleCalc*, float *distTolerance*, float *angleTolerance*, bool *updateAllFace*)

Constructor.

Parameters:

startFrame adds an offset to the current frame number to help evenly distribute updates between frames

updateInterval update frequency

resolution reduced cubemap resolution

useDistCalc flag to skip cube face update if object is far away

useFaceAngleCalc flag to skip cube face update if face is neglible

distTolerance distance tolerance used in face skip

angleTolerance angle tolerance used in face skip

updateAllFace defines if all cubemap faces should be updated in a frame or only one face per frame

5.63.3 Member Function Documentation

5.63.3.1 virtual void ReducedCubeMapRenderingRun::updateCubeFace (int *facenum*) [inline, protected, pure virtual]

Updates one face of the cubemap.

Parameters:

facenum the number of the face to be updated

Implemented in OgreReducedCubeMapRenderingRun.

5.63.3.2 virtual bool ReducedCubeMapRenderingRun::faceNeedsUpdate (int *facenum*) [protected, pure virtual]

Checks if a cubemap face needs to be updated.

If the object we are updating the cubemap for is far from the camera, or too small, or the given cubemapface does not have significant effect on the rendering the face can be skipped.

Parameters:

facenum the number of the face to be checked

Implemented in OgreReducedCubeMapRenderingRun.

5.63.3.3 void ReducedCubeMapRenderingRun::updateFrame (unsigned long *frameNum*) [protected, virtual]

This function does the actual update in a frame.

Parameters:

frameNum current frame number

Reimplemented from RenderingRun.

5.63.3.4 virtual class OgreRenderingRun* RenderingRun::asOgreRenderingRun () [inline, virtual, inherited]

Conversion to OgreRenderRun.

This function is needed because of virtual inheritance.

Reimplemented in OgreRenderingRun, and OgreRenderingRun.

5.63.3.5 virtual bool RenderingRun::canJoin (RenderingRun * *run*) [inline, virtual, inherited]

Returns true if two runs can be joined.

In some cases special requirements should stand to join two runs (even if they have the same type). Eg.: two caustic cube map generation technique should only be joined if they use the same material when rendering the caustic cubemap.

Reimplemented in OgreCausticCubeMapRenderingRun, OgreChildPSystemRenderingRun, and OgrePhotonMapRenderingRun.

5.63.3.6 virtual bool RenderingRun::needUpdate (unsigned long *frameNum*) [inline, protected, virtual, inherited]

Returns if this run needs update.

This tipically depends on the upate interval and the starting frame number.

Parameters:

frameNum current frame number

Reimplemented in OgreFocusingMapRenderingRun, and OgrePMEntryPointMapRenderingRun.

5.63.4 Member Data Documentation

5.63.4.1 bool ReducedCubeMapRenderingRun::useDistCalc [protected]

a flag to skip cube face update if object is far away or too small.

See also:

distTolerance

5.63.4.2 bool ReducedCubeMapRenderingRun::useFaceAngleCalc [protected]

a flag to skip cube face update the face is neglible.

See also:

angleTolerance

5.63.4.3 float ReducedCubeMapRenderingRun::distTolerance [protected]

A value used in face skip test.

The higher this value gets the more precise, but slower the method will be.

5.63.4.4 float ReducedCubeMapRenderingRun::angleTolerance [protected]

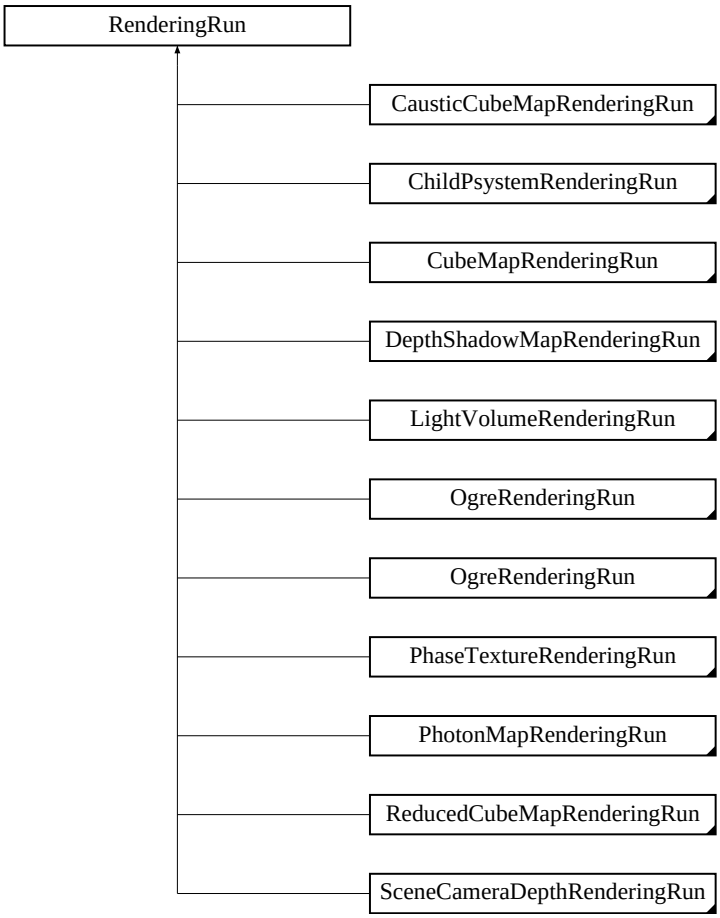
A value used in face skip test.

The higher this value gets the more precise, but slower the method will be.

5.64 RenderingRun Class Reference

Base class for a computation module.

Inheritance diagram for RenderingRun::



Public Member Functions

- RenderingRun (unsigned long startFrame, unsigned long updateInterval)

  Constructor.

- bool update (unsigned long frameNum)

  Calls updateFrame() if the run needs update according to its starting frame and update interval and has not been allready updated in this frame.

- virtual class OgreRenderingRun * asOgreRenderingRun ()

  Conversion to OgreRenderRun.

- virtual bool canJoin (RenderingRun *run)

Returns true if two runs can be joined.

Protected Member Functions

- virtual bool needUpdate (unsigned long frameNum)

  Returns if this run needs update.

- virtual void updateFrame (unsigned long frameNum)

  This function does the actual update in a frame.

Protected Attributes

- unsigned long lastupdated

  The number of the last frame this run was updated.

- unsigned long startFrame

  The number of the frame this run should be updated first.

- unsigned long updateInterval

  Refresh frequency in frames.

5.64.1 Detailed Description

Base class for a computation module.

A run typically - but not necessarily or exclusively - consists of a series of rendering passes. A run is alway created to compute some kind of resource for a RenderTechnique. The type of the resource depends on the type of the run (typically it is a Texture). Runs can be attached to only one Technique (if only one of the Techniques attached to a Renderable can use this resource, and each Renderable requires a unique one), or can be shared between several Techniques and Renderables (for example a cube-map). Runs are updated only once in a frame, but not necessary in each frame.

5.64.2 Constructor & Destructor Documentation

5.64.2.1 RenderingRun::RenderingRun (unsigned long *startFrame*, unsigned long *updateInterval*)

Constructor.

Parameters:

startFrame adds an offset to the current frame number to help evenly distribute updates between frames

updateInterval photon map update frequency

5.64.3 Member Function Documentation

5.64.3.1 virtual class OgreRenderingRun* RenderingRun::asOgreRenderingRun () [inline, virtual]

Conversion to OgreRenderRun.

This function is needed because of virtual inheritance.

Reimplemented in OgreRenderingRun, and OgreRenderingRun.

5.64.3.2 virtual bool RenderingRun::canJoin (RenderingRun * *run*) [inline, virtual]

Returns true if two runs can be joined.

In some cases special requirements should stand to join two runs (even if they have the same type). Eg.: two caustic cube map generation technique should only be joined if they use the same material when rendering the caustic cubemap.

Reimplemented in OgreCausticCubeMapRenderingRun, OgreChildPSystemRenderingRun, and OgrePhotonMapRenderingRun.

5.64.3.3 virtual bool RenderingRun::needUpdate (unsigned long *frameNum*) [inline, protected, virtual]

Returns if this run needs update.

This tipically depends on the upate interval and the starting frame number.

Parameters:

frameNum current frame number

Reimplemented in OgreFocusingMapRenderingRun, and OgrePMEntryPointMapRenderingRun.

5.64.3.4 virtual void RenderingRun::updateFrame (unsigned long *frameNum*) [inline, protected, virtual]

This function does the actual update in a frame.

Parameters:

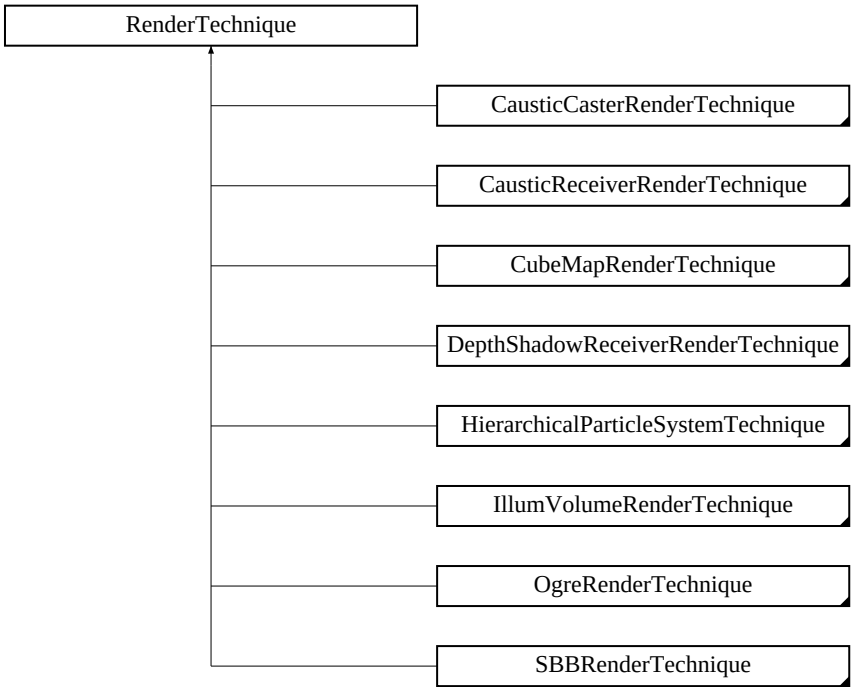
frameNum current frame number

Reimplemented in CausticCubeMapRenderingRun, ChildPsystemRenderingRun, CubeMapRenderingRun, DepthShadowMapRenderingRun, LightVolumeRenderingRun, PhaseTextureRenderingRun, PhotonMapRenderingRun, ReducedCubeMapRenderingRun, SceneCameraDepthRenderingRun, OgreChildPSystemRenderingRun, OgreDepthShadowMapRenderingRun, OgreFocusingMapRenderingRun, OgreLightVolumeRenderingRun, OgrePhaseTextureRenderingRun, OgrePhotonMapRenderingRun, OgrePMEntryPointMapRenderingRun, OgrePMWeightComputeRenderingRun, and OgreSceneCameraDepthRenderingRun.

5.65 RenderTechnique Class Reference

Base class for a rendering technique.

Inheritance diagram for RenderTechnique::



Public Member Functions

- RenderTechnique (ElementaryRenderable *parentRenderable, TechniqueGroup *parentTechniqueGroup)

  Constructor.

- virtual void update (unsigned long frameNum)

  Updates the resources in the given frame.

- virtual void runChanged (RenderingRunType runType, RenderingRun *run)

  Called after one of he shared runs changes.

- virtual void runUpdated (RenderingRunType runType, RenderingRun *run)

  Called after one of he shared runs updates.

- virtual class OgreRenderTechnique * asOgreRenderTechnique ()

  Conversion to OgreRenderTechnique.

- ElementaryRenderable * getParentRenderable ()

Retrieves the renderable this technique operates on.

Protected Attributes

- ElementaryRenderable * parentRenderable

  The renderable this technique operates on.

- TechniqueGroup * parentTechniqueGroup

  The TechniqueGroup this RenderedTechnique is attached to.

- SharedRuns * sharedRuns

  The SharedRuns this RenderedTechnique is attached to.

5.65.1 Detailed Description

Base class for a rendering technique.

A RenderTechnique gives a description about how to render an object, and what kind of resources are needed to do this. A RenderTechnique does not define the whole process of rendering only one property of the display, example: this object will need a cubemap or this object is going to be a caustic caster. RenderTechniques usually operate on one pass of the rendering, and bind some resources to this pass (for example it can bind a texture or cubemap tho the given pass).

RenderTechniques are always bound to a Renderable, and a SharedRuns object. The Renderable defines the object to operate on. SharedRuns is to register RenderingRuns which can be shared between other RenderTechniques (example: a cubemap can be used by other techniques too, or even more than one object can share a single cubemap).

5.65.2 Constructor & Destructor Documentation

5.65.2.1 RenderTechnique::RenderTechnique (ElementaryRenderable * *parentRenderable*, TechniqueGroup * *parentTechniqueGroup*)

Constructor.

Parameters:

parentRenderable the object to operate on

parentTechniqueGroup the TechniqueGroup this RenderedTechnique is attached to

5.65.3 Member Function Documentation

5.65.3.1 virtual void RenderTechnique::update (unsigned long *frameNum*) [inline, virtual]

Updates the resources in the given frame.

A RenderTechnique is usually need some resources from several runs, so these runs will be updated.

Parameters:

frameNum the actual framenumber

Reimplemented in ColorCubeMapRenderTechnique, ConvolvedCubeMapRenderTechnique, DistanceCubeMapRenderTechnique, HierarchicalParticleSystemTechnique, IllumVolumeRenderTechnique, OgreCausticReceiverRenderTechnique, OgreColorCubeMapRenderTechnique, OgreConvolvedCubeMapRenderTechnique, OgreDepthShadowReceiverRenderTechnique, OgreDistanceCubeMapRenderTechnique, OgreFireRenderTechnique, OgrePathMapRenderTechnique, and OgreSBBRenderTechnique.

5.65.3.2 virtual void RenderTechnique::runChanged (RenderingRunType *runType*, RenderingRun * *run*) [inline, virtual]

Called after one of he shared runs changes.

Parameters:

runType enum describing the type of the changed run

run pointer to the changed RenderingRun

Reimplemented in CausticCasterRenderTechnique, ColorCubeMapRenderTechnique, ConvolvedCubeMapRenderTechnique, CubeMapRenderTechnique, DistanceCubeMapRenderTechnique, HierarchicalParticleSystemTechnique, and IllumVolumeRenderTechnique.

5.65.3.3 virtual void RenderTechnique::runUpdated (RenderingRunType *runType*, RenderingRun * *run*) [inline, virtual]

Called after one of he shared runs updates.

Parameters:

runType enum describing the type of the updated run

run pointer to the updated RenderingRun

Reimplemented in CausticCasterRenderTechnique, ColorCubeMapRenderTechnique, CubeMapRenderTechnique, DistanceCubeMapRenderTechnique, HierarchicalParticleSystemTechnique, and IllumVolumeRenderTechnique.

5.65.3.4 virtual class OgreRenderTechnique* RenderTechnique::asOgreRenderTechnique () [inline, virtual]

Conversion to OgreRenderTechnique.

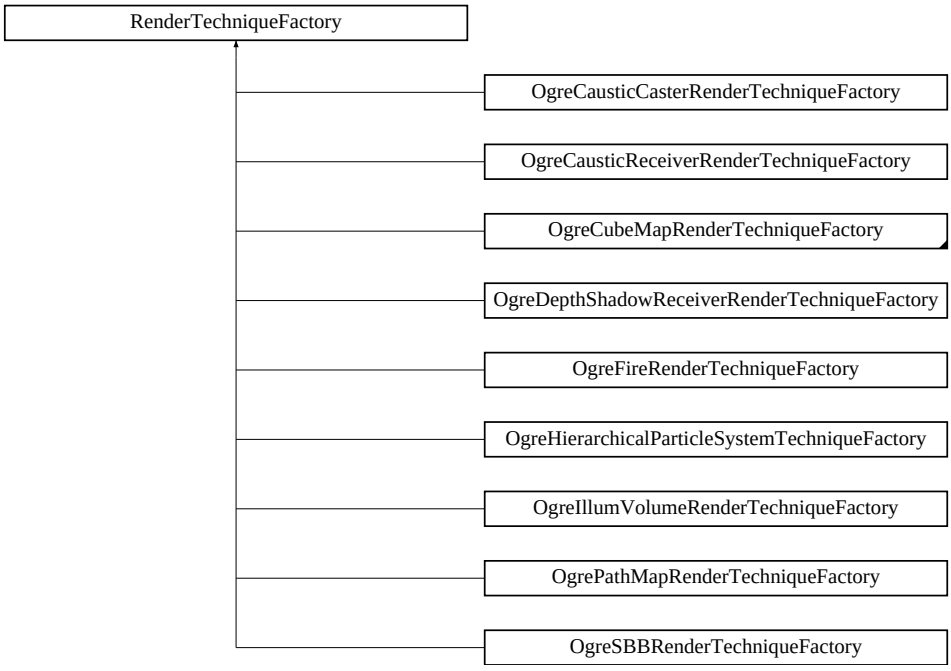
This function is needed because of virtual inheritance.

Reimplemented in OgreRenderTechnique.

5.66 RenderTechniqueFactory Class Reference

Base abstract class for creating RenderTechnique instances.

Inheritance diagram for RenderTechniqueFactory::



Public Member Functions

- bool isType (String type)

  Returns if this factory can create a RenderTechnique of the given type.

- virtual OgreRenderTechnique * createInstance (IllumTechniqueParams *params, Pass *pass, OgreRenderable *parentRenderable, OgreTechniqueGroup *parentTechniqueGroup)=0

  Creates a RenderTechnique of the factory type.

- virtual void parseParams (IllumTechniqueParams *params)

  parses parameters from the material file.

Protected Types

- typedef void(*) ILLUM_ATTRIBUTE_PARSER (String &params, RenderTechniqueFactory *factory)

  function for parsing RenderTechnique attributes

- typedef std::map< String, ILLUM_ATTRIBUTE_PARSER > AttribParserList

Keyword-mapped attribute parsers.

Protected Attributes

- AttribParserList attributeParsers
  map of parser functions
- String typeName
  factoryname

5.66.1 Detailed Description

Base abstract class for creating RenderTechnique instances.

5.66.2 Member Typedef Documentation

5.66.2.1 typedef void(∗) RenderTechniqueFactory::ILLUM_ATTRIBUTE_PARSER(String &params, RenderTechniqueFactory ∗factory) [protected]

function for parsing RenderTechnique attributes

Parameters:

params attribute value stored in a String

5.66.3 Member Function Documentation

5.66.3.1 bool RenderTechniqueFactory::isType (String *type*) [inline]

Returns if this factory can create a RenderTechnique of the given type.

Parameters:

type RenderTechnique type

5.66.3.2 virtual OgreRenderTechnique∗ RenderTechniqueFactory::createInstance (IllumTechniqueParams ∗ *params*, Pass ∗ *pass*, OgreRenderable ∗ *parentRenderable*, OgreTechniqueGroup ∗ *parentTechniqueGroup*) [pure virtual]

Creates a RenderTechnique of the factory type.

Parameters:

params containes constructor parameters as NameValuePairList

pass the Pass to use in RenderTechnique constructor

pass the parentRenderable to pass to RenderTechnique constructor

pass the parentTechniqueGroup to pass to RenderTechnique constructor

Implemented in OgreCausticCasterRenderTechniqueFactory, OgreCausticReceiverRenderTechniqueFactory, OgreColorCubeMapRenderTechniqueFactory, OgreConvoledCubeMapRenderTechniqueFactory, OgreCubeMapRenderTechniqueFactory, OgreDepthShadowReceiverRenderTechniqueFactory, OgreDistanceCubeMapRenderTechniqueFactory, OgreFireRenderTechniqueFactory, OgreHierarchicalParticleSystemTechniqueFactory, OgreIllumVolumeRenderTechniqueFactory, OgrePathMapRenderTechniqueFactory, and OgreSBBRenderTechniqueFactory.

5.66.3.3 void RenderTechniqueFactory::parseParams (IllumTechniqueParams * *params*) [virtual]

parses parameters from the material file.

The parsed parameters will be passed to the new RenderTechnique's constructor.

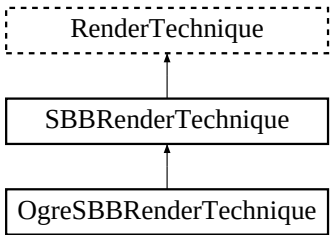
Parameters:

params pointer to the IllumTechniqueParams structure that was read from the material script and containes the parameters to be parsed.

5.67 SBBRenderTechnique Class Reference

Base abstract class of rendering a particle system with the spherical billboard method.

Inheritance diagram for SBBRenderTechnique::



Public Member Functions

- SBBRenderTechnique (ElementaryRenderable *parentRenderable, TechniqueGroup *parentTechniqueGroup)
  Constructor.
- virtual void update (unsigned long frameNum)
  Updates the resources in the given frame.
- virtual void runChanged (RenderingRunType runType, RenderingRun *run)
  Called after one of he shared runs changes.
- virtual void runUpdated (RenderingRunType runType, RenderingRun *run)
  Called after one of he shared runs updates.
- virtual class OgreRenderTechnique * asOgreRenderTechnique ()
  Conversion to OgreRenderTechnique.
- ElementaryRenderable * getParentRenderable ()
  Retrieves the renderable this technique operates on.

Protected Attributes

- ElementaryRenderable * parentRenderable
  The renderable this technique operates on.
- TechniqueGroup * parentTechniqueGroup
  The TechniqueGroup this RenderedTechnique is attached to.
- SharedRuns * sharedRuns
  The SharedRuns this RenderedTechnique is attached to.

5.67.1 Detailed Description

Base abstract class of rendering a particle system with the spherical billboard method.

This technique only defines that the rendering of the object needs depth information of the scene from the player camera's view.

5.67.2 Constructor & Destructor Documentation

5.67.2.1 SBBRenderTechnique::SBBRenderTechnique (ElementaryRenderable * *parentRenderable*, TechniqueGroup * *parentTechniqueGroup*)

Constructor.

Parameters:

parentRenderable the object to operate on

parentTechniqueGroup the TechniqueGroup this RenderedTechnique is attached to

5.67.3 Member Function Documentation

5.67.3.1 virtual void RenderTechnique::update (unsigned long *frameNum*) [inline, virtual, inherited]

Updates the resources in the given frame.

A RenderTechnique is usually need some resources from several runs, so these runs will be updated.

Parameters:

frameNum the actual framenumber

Reimplemented in ColorCubeMapRenderTechnique, ConvolvedCubeMapRenderTechnique, DistanceCubeMapRenderTechnique, HierarchicalParticleSystemTechnique, IllumVolumeRenderTechnique, OgreCausticReceiverRenderTechnique, OgreColorCubeMapRenderTechnique, OgreConvolvedCubeMapRenderTechnique, OgreDepthShadowReceiverRenderTechnique, OgreDistanceCubeMapRenderTechnique, OgreFireRenderTechnique, OgrePathMapRenderTechnique, and OgreSBBRenderTechnique.

5.67.3.2 virtual void RenderTechnique::runChanged (RenderingRunType *runType*, RenderingRun * *run*) [inline, virtual, inherited]

Called after one of he shared runs changes.

Parameters:

runType enum describing the type of the changed run

run pointer to the changed RenderingRun

Reimplemented in CausticCasterRenderTechnique, ColorCubeMapRenderTechnique, ConvolvedCubeMapRenderTechnique, CubeMapRenderTechnique, DistanceCubeMapRenderTechnique, HierarchicalParticleSystemTechnique, and IllumVolumeRenderTechnique.

5.67.3.3 virtual void RenderTechnique::runUpdated (RenderingRunType *runType*, RenderingRun * ***run***) [inline, virtual, inherited]

Called after one of he shared runs updates.

Parameters:

runType enum describing the type of the updated run

run pointer to the updated RenderingRun

Reimplemented in CausticCasterRenderTechnique, ColorCubeMapRenderTechnique, CubeMapRenderTechnique, DistanceCubeMapRenderTechnique, HierarchicalParticleSystemTechnique, and IllumVolumeRenderTechnique.

5.67.3.4 virtual class OgreRenderTechnique* RenderTechnique::asOgreRenderTechnique () [inline, virtual, inherited]

Conversion to OgreRenderTechnique.

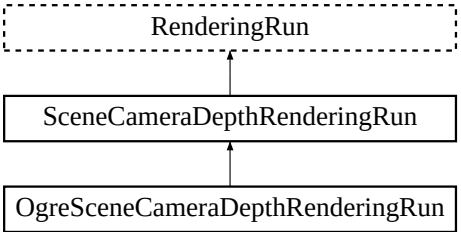
This function is needed because of virtual inheritance.

Reimplemented in OgreRenderTechnique.

5.68 SceneCameraDepthRenderingRun Class Reference

Base abstract class that defines a rendering process that creates depth map fro the camera.

Inheritance diagram for SceneCameraDepthRenderingRun::



Public Member Functions

- SceneCameraDepthRenderingRun ()

  Constructor.

- bool update (unsigned long frameNum)

  Calls updateFrame() if the run needs update according to its starting frame and update interval and has not been allready updated in this frame.

- virtual class OgreRenderingRun * asOgreRenderingRun ()

  Conversion to OgreRenderRun.

- virtual bool canJoin (RenderingRun *run)

  Returns true if two runs can be joined.

Protected Member Functions

- virtual void createDepthMap ()=0

  Creates the depth map texture.

- virtual void updateFrame (unsigned long frameNum)=0

  This function does the actual update in a frame.

- virtual bool needUpdate (unsigned long frameNum)

  Returns if this run needs update.

Protected Attributes

- unsigned long lastupdated

  The number of the last frame this run was updated.

- unsigned long startFrame

  The number of the frame this run should be updated first.

- unsigned long updateInterval

  Refresh frequency in frames.

5.68.1 Detailed Description

Base abstract class that defines a rendering process that creates depth map fro the camera.

The depth map stores the scene's camera space z coordinates (rendered from the player's view).

5.68.2 Member Function Documentation

5.68.2.1 virtual void SceneCameraDepthRenderingRun::updateFrame (unsigned long *frameNum*) [protected, pure virtual]

This function does the actual update in a frame.

Parameters:

frameNum current frame number

Reimplemented from RenderingRun.

Implemented in OgreSceneCameraDepthRenderingRun.

5.68.2.2 virtual class OgreRenderingRun* RenderingRun::asOgreRenderingRun () [inline, virtual, inherited]

Conversion to OgreRenderRun.

This function is needed because of virtual inheritance.

Reimplemented in OgreRenderingRun, and OgreRenderingRun.

5.68.2.3 virtual bool RenderingRun::canJoin (RenderingRun * *run*) [inline, virtual, inherited]

Returns true if two runs can be joined.

In some cases special requirements should stand to join two runs (even if they have the same type). Eg.: two caustic cube map generation technique should only be joined if they use the same material when rendering the caustic cubemap.

Reimplemented in OgreCausticCubeMapRenderingRun, OgreChildPSystemRenderingRun, and OgrePhotonMapRenderingRun.

5.68.2.4 virtual bool RenderingRun::needUpdate (unsigned long *frameNum*) [inline, protected, virtual, inherited]

Returns if this run needs update.

This tipically depends on the upate interval and the starting frame number.

Parameters:

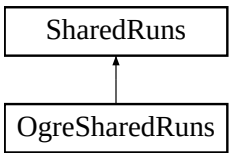
frameNum current frame number

Reimplemented in OgreFocusingMapRenderingRun, and OgrePMEntryPointMapRenderingRun.

5.69 SharedRuns Class Reference

Base abstract class for a collection of shared resources (RenderingRuns).

Inheritance diagram for SharedRuns::



Public Member Functions

- SharedRuns (void)
- virtual bool hasOwnRun (RenderingRunType runType)=0
- virtual RenderingRun * getRun (RenderingRunType runType)=0
- virtual void addRun (RenderingRunType runType, RenderingRun *run)=0
- virtual void updateRun (RenderingRunType runType, unsigned long frameNum)=0
- virtual SharedRuns * joinRuns (SharedRuns *otherRuns)
- virtual void runChanged (RenderingRunType runType, RenderingRun *run)

  Called after one of he shared runs changes.

- virtual void runUpdated (RenderingRunType runType, RenderingRun *run)

  Called after one of he shared runs updates.

- virtual void addTechniqueGroup (TechniqueGroup *group)=0

  Adds a child TechniqueGroup.

- virtual void setVisible (bool visible)

  Shows or hides this SharedRuns (and also all childnodes).

- virtual void hide ()

  Hides this SharedRuns (and also all childs).

- virtual void restoreVisibility ()

  Restores the visibility of this SharedRuns (and also all childs).

- virtual SharedRuns * getRoot ()

  Retrieves the root node of this SharedRuns node.

- virtual SharedRuns * getRoot (RenderingRunType runType)

  Retrieves the topmost parent node of this SharedRuns node, which have a specified RenderingRun type.

- virtual void updateBounds ()=0

  Updates the boundary of this SharedRuns (and also it's parent).

- virtual void validate ()=0

  Validate this SharedRuns (and also all childs).

- virtual void destroy ()=0

  Destroys the node (and all parents recursively).

- virtual void unbindParent ()

  Unbinds the parent of the node, called at splitting.

- virtual void unbindAndKillParent ()

  Unbinds the deletes the parent of the node, called at splitting.

Protected Member Functions

- virtual void gatherRuns ()=0

  Collects RenderingRuns references from the child nodes, used when joining.

- virtual void fireRunChanges ()=0

  Sends runChanged events for each RenderingRun type, used after join and split.

- virtual SharedRuns * createInstance ()=0

  Creates a new SharedRuns instance. All derivatives should implement this.

- virtual void setRenderablesVisible (bool visible)=0

  Set visibility of connected renderables, only used if this is a leaf.

- virtual void hideRenderables ()=0

  Hides all the connected renderables, only used if this is a leaf.

- virtual void restoreRenderableVisibility ()=0

  Restires visibility of all the connected renderables, only used if this is a leaf.

Protected Attributes

- SharedRuns * parent

  parent SharedRuns instance

- SharedRuns * child1

  child SharedRuns instance

- SharedRuns * child2

  child SharedRuns instance

5.69.1 Detailed Description

Base abstract class for a collection of shared resources (RenderingRuns).

Technique resources which can be shared between several techniques or objects are managed by SharedRuns. These SharedRuns store the shared resources. They also act like nodes of a binary tree, as separate SharedRuns can also be joined if for example the objects for which they store resources are close enough so even one shared resources is enough for the given objects.

5.69.2 Constructor & Destructor Documentation

5.69.2.1 SharedRuns::SharedRuns (void)

&brief Constructor.

5.69.3 Member Function Documentation

5.69.3.1 virtual bool SharedRuns::hasOwnRun (RenderingRunType *runType*) [pure virtual]

&brief Returns true if this shared run object has a given run type.

It returns true if the shared run object node has a rendering run object with the given type. It checks only the current node of the tree, no children nodes are searched for existing rendering runs.

Parameters:

runType enum, type of the RenderingRun to search for

Returns:

search result

Implemented in OgreSharedRuns.

5.69.3.2 virtual RenderingRun* SharedRuns::getRun (RenderingRunType *runType*) [pure virtual]

&brief Retrieves a shared resource.

Parameters:

runType enum, type of the RenderingRun to be retrieved

Returns:

pointer to the RenderingRun of type "runType", null if this type does not exists

Implemented in OgreSharedRuns.

5.69.3.3 virtual void SharedRuns::addRun (RenderingRunType *runType*, RenderingRun * *run*) [pure virtual]

&brief Adds a RenderingRun instance to the shared resources.

Parameters:

runType enum, type of the RenderingRun to add

run pointer to the RenderingRun instance to add

Implemented in OgreSharedRuns.

5.69.3.4 virtual void SharedRuns::updateRun (RenderingRunType *runType*, unsigned long *frameNum*) [pure virtual]

&brief Updates a shared RenderingRun.

Parameters:

runType enum, type of the RenderingRun to update

frameNum current framenumber

Implemented in OgreSharedRuns.

5.69.3.5 SharedRuns * SharedRuns::joinRuns (SharedRuns * *otherRuns*) [virtual]

&brief Joines two SharedRuns.

The resulting SharedRuns become the parent of the two SharedRuns.

Parameters:

otherRuns pointer to the SharedRuns instance to join with

Returns:

the new parent SharedRuns instance

5.69.3.6 void SharedRuns::runChanged (RenderingRunType *runType*, RenderingRun * *run*) [virtual]

Called after one of he shared runs changes.

This message will be forwarded to each child.

Parameters:

runType enum describing the type of the changed run

run pointer to the changed RenderingRun

Reimplemented in OgreSharedRuns.

5.69.3.7 void SharedRuns::runUpdated (RenderingRunType *runType*, RenderingRun * *run*) [virtual]

Called after one of he shared runs updates.

This message will be forwarded to each child.

Parameters:

runType enum describing the type of the updated run

run pointer to the updated RenderingRun

Reimplemented in OgreSharedRuns.

5.69.3.8 virtual void SharedRuns::addTechniqueGroup (TechniqueGroup * *group*) [pure virtual]

Adds a child TechniqueGroup.

Parameters:

group pointer to the TechniqueGroup instance to add.

Implemented in OgreSharedRuns.

5.69.3.9 void SharedRuns::setVisible (bool *visible*) [virtual]

Shows or hides this SharedRuns (and also all childnodes).

Parameters:

visible visibility

5.69.3.10 void SharedRuns::hide () [virtual]

Hides this SharedRuns (and also all childs).

The previous visibility is saved.

5.69.3.11 SharedRuns * SharedRuns::getRoot () [virtual]

Retrieves the root node of this SharedRuns node.

Returns:

pointer to the root SharedRuns instance

5.69.3.12 SharedRuns * SharedRuns::getRoot (RenderingRunType *runType*) [virtual]

Retrieves the topmost parent node of this SharedRuns node, which have a specified RenderingRun type.

Parameters:

runType the RenderingRun type

Returns:

pointer to the parent SharedRuns instance

5.69.3.13 virtual void SharedRuns::validate () [pure virtual]

Validate this SharedRuns (and also all childs).

Validation meens that all the SharedRuns that are joined will be examined if the sharing is still valid. If it finds out that two SharedRuns can't be joined anymore (eg.: they moved far from each other), their parent will be split and destroyed (all parent of this node also should be deleted recursively).

Implemented in OgreSharedRuns.

5.69.3.14 virtual SharedRuns* SharedRuns::createInstance () [protected, pure virtual]

Creates a new SharedRuns instance. All derivatives should implement this.

Returns:

a new SharedRuns instance

Implemented in OgreSharedRuns.

5.69.3.15 virtual void SharedRuns::setRenderablesVisible (bool *visible*) [protected, pure virtual]

Set visibility of connected renderables, only used if this is a leaf.

Parameters:

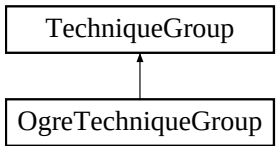
visible visibility

Implemented in OgreSharedRuns.

5.70 TechniqueGroup Class Reference

Base abstract class for a collection of techniques.

Inheritance diagram for TechniqueGroup::



Public Member Functions

- TechniqueGroup (void)
  Constructor.
- void addSharedRun (SharedRuns *sharedRuns)
  Adds an empty SharedRuns parent.
- SharedRuns * getSharedRuns ()
  Retrieves the shared runs.
- virtual void addRenderTechnique (RenderTechnique *technique)=0
  Adds a rendertechnique to the group.
- virtual void update (unsigned long frameNum)=0
  Updates all rendertechniques.
- virtual void runChanged (RenderingRunType runType, RenderingRun *run)=0
  Called after one of the rendering runs changes.
- virtual void runUpdated (RenderingRunType runType, RenderingRun *run)=0
  Called after one of the rendering runs updates.
- virtual void updateBounds ()
  Updates the connected SharedRuns boundary.
- virtual void validateSharedRuns ()
  Validates the connected SharedRuns instance.

Protected Attributes

- SharedRuns * parentSharedRuns
  Pointer to the connected SharedRuns instance each technique uses.

5.70.1 Detailed Description

Base abstract class for a collection of techniques.

This is a helper class, to collect RenderTechnique instances bound to a single renderable. It's main task is to recieve and forward messages to each RenderTechnique.

5.70.2 Member Function Documentation

5.70.2.1 void TechniqueGroup::addSharedRun (SharedRuns * *sharedRuns*) [inline]

Adds an empty SharedRuns parent.

Used after creating a new TechniqueGroup.

Parameters:

sharedRuns pointer to the SharedRun instance this RenderTechniques will use.

5.70.2.2 SharedRuns* TechniqueGroup::getSharedRuns () [inline]

Retrieves the shared runs.

Returns:

pointer to the SharedRun instance this RenderTechniques uses.

5.70.2.3 virtual void TechniqueGroup::addRenderTechnique (RenderTechnique * *technique*) [pure virtual]

Adds a rendertechnique to the group.

Parameters:

technique the RenderTechnique instance to add.

Implemented in OgreTechniqueGroup.

5.70.2.4 virtual void TechniqueGroup::update (unsigned long *frameNum*) [pure virtual]

Updates all rendertechniques.

Parameters:

framenum current framenumber

Implemented in OgreTechniqueGroup.

5.70.2.5 virtual void TechniqueGroup::runChanged (RenderingRunType *runType*, RenderingRun * *run*) [pure virtual]

Called after one of the rendering runs changes.

This message will be forwarded to each RenderTechique.

Parameters:

runType enum describing the type of the changed run

run pointer to the changed RenderingRun

Implemented in OgreTechniqueGroup.

5.70.2.6 virtual void TechniqueGroup::runUpdated (RenderingRunType *runType*, RenderingRun * *run*) [pure virtual]

Called after one of the rendering runs updates.

This message will be forwarded to each RenderTechique.

Parameters:

runType enum describing the type of the updated run

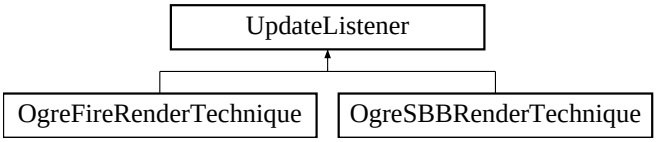
run pointer to the updated RenderingRun

Implemented in OgreTechniqueGroup.

5.71 UpdateListener Class Reference

Event handler class.

Inheritance diagram for UpdateListener::



Public Member Functions

- virtual void preAllUpdates ()

  Called before RenderTechnique updates.

- virtual void postAllUpdates ()

  Called after RenderTechnique updates.

5.71.1 Detailed Description

Event handler class.

The derived calsses of this class can register task that should be done before and/or after updating all RenderTechniques. UpdateListeners can be registered to the OgreIlluminationMager.

See also:

addUpdateListener