GameTools Illumination Module Reference Manual

Generated by Doxygen 1.4.6-NO

Thu Apr 27 17:29:35 2006

Contents

Chapter 1

GameTools Illumination Module Namespace Index

1.1 GameTools Illumination Module Namespace List

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

Chapter 2

GameTools Illumination Module Hierarchical Index

2.1 GameTools Illumination Module Class Hierarchy

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

Chapter 3

GameTools Illumination Module Class Index

3.1 GameTools Illumination Module Class List

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

Chapter 4

GameTools Illumination Module Namespace Documentation

4.1 CausticCasterParsers Namespace Reference

Technique Parsers.

4.1.1 Detailed Description

Technique Parsers.

4.2 CausticCubemapParsers Namespace Reference

Technique Parsers.

4.2.1 Detailed Description

Technique Parsers.

4.3 ColorCubemapParsers Namespace Reference

Technique Parsers.

4.3.1 Detailed Description

Technique Parsers.

4.4 ConvolvedCubemapParsers Namespace Reference

Technique Parsers.

4.4.1 Detailed Description

Technique Parsers.

4.5 DepthShadowRecieverParsers Namespace Reference

Technique parsers.

4.5.1 Detailed Description

Technique parsers.

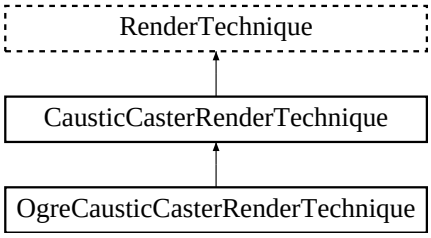
Chapter 5

GameTools 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.

Protected Member Functions

- virtual void **photonMapRunChanged** (**RenderingRun** ∗run)=0

  Called if the changed run is a **PhotonMapRenderingRun**(p. 118).

- virtual void **causticCubeMapRunChanged** (**RenderingRun** ∗run)=0

  Called if the changed run is a **CausticCubeMapRenderingRun**(p. 17).

- virtual void **distanceCubeMapRunChanged** (**RenderingRun** ∗run)=0

  Called if the changed run is a **DistanceCubeMapRenderingRun**(p. 36).

- virtual **RenderingRun** ∗ **createPhotonMapRun** ()=0

  Creates a **PhotonMapRenderingRun**(p. 118).

- virtual **RenderingRun** ∗ **createCausticCubeMapRun** ()=0

  Creates a **CausticCubeMapRenderingRun**(p. 17).

- virtual **RenderingRun** ∗ **createDistanceCubeMapRun** ()=0

  Creates a **DistanceCubeMapRenderingRun**(p. 36).

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

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. These resources will be updated by caustic recievers.

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**(p. 141) this RenderedTechnique is attached to

5.1.3 Member Function Documentation

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

Called if the changed run is a **CausticCubeMapRenderingRun**(p. 17).

Parameters:

run pointer to the changed **CausticCubeMapRenderingRun**(p. 17)

Implemented in **OgreCausticCasterRenderTechnique** (p. 49).

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

Creates a **CausticCubeMapRenderingRun**(p. 17).

Returns:

the new **CausticCubeMapRenderingRun**(p. 17) instance.

Implemented in **OgreCausticCasterRenderTechnique** (p. 49).

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

Creates a **DistanceCubeMapRenderingRun**(p. 36).

Returns:

the new **DistanceCubeMapRenderingRun**(p. 36) instance.

Implemented in **OgreCausticCasterRenderTechnique** (p. 49).

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

Creates a **PhotonMapRenderingRun**(p. 118).

Returns:
the new **PhotonMapRenderingRun**(p. 118) instance.

Implemented in **OgreCausticCasterRenderTechnique** (p. 49).

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

Called if the changed run is a **DistanceCubeMapRenderingRun**(p. 36).

Parameters:
run pointer to the changed **DistanceCubeMapRenderingRun**(p. 36)

Implemented in **OgreCausticCasterRenderTechnique** (p. 49).

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

Called if the changed run is a **PhotonMapRenderingRun**(p. 118).

Parameters:
run pointer to the changed **PhotonMapRenderingRun**(p. 118)

Implemented in **OgreCausticCasterRenderTechnique** (p. 50).

5.1.3.7 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**(p. 124)

Reimplemented from **RenderTechnique** (p. 128).

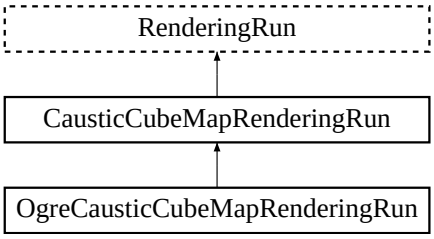
The documentation for this class was generated from the following files:

- CausticCasterRenderTechnique.h
- CausticCasterRenderTechnique.cpp

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**(p. 118).

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.

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

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 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** (p. 52).

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

Called if the changed run is a **PhotonMapRenderingRun**(p. 118).

Parameters:

run pointer to the changed **PhotonMapRenderingRun**(p. 118)

Implemented in **OgreCausticCubeMapRenderingRun** (p. 53).

5.2.3.3 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** (p. 53).

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** (p. 126).

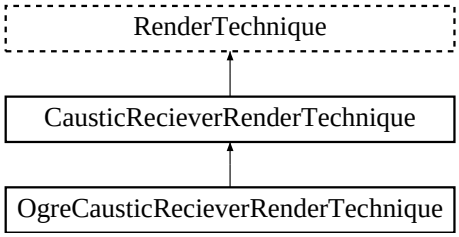
The documentation for this class was generated from the following files:

- CausticCubeMapRenderingRun.h
- CausticCubeMapRenderingRun.cpp

5.3 CausticRecieverRenderTechnique Class Reference

Base abstract class of rendering a caustic reciever object.

Inheritance diagram for CausticRecieverRenderTechnique::



Public Member Functions

- **CausticRecieverRenderTechnique** (**ElementaryRenderable** ∗**parentRenderable**, **TechniqueGroup** ∗**parentTechniqueGroup**)

  Constructor.

5.3.1 Detailed Description

Base abstract class of rendering a caustic reciever object.

5.3.2 Constructor & Destructor Documentation

5.3.2.1 CausticRecieverRenderTechnique::CausticRecieverRenderTechnique (ElementaryRenderable ∗ *parentRenderable*, TechniqueGroup ∗ *parentTechniqueGroup*)

Constructor.

Parameters:

parentRenderable the object to operate on

parentTechniqueGroup the **TechniqueGroup**(p. 141) this RenderedTechnique is attached to

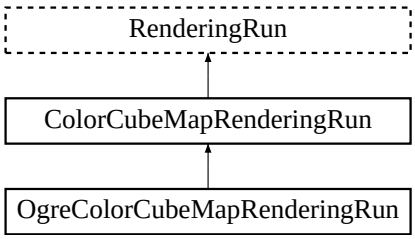
The documentation for this class was generated from the following files:

- CausticRecieverRenderTechnique.h
- CausticRecieverRenderTechnique.cpp

5.4 ColorCubeMapRenderingRun Class Reference

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

Inheritance diagram for ColorCubeMapRenderingRun::



Public Member Functions

- **ColorCubeMapRenderingRun** (unsigned long **startFrame**, unsigned long **updateInterval**, unsigned int **resolution**, bool **useDistCalc**, bool **useFaceAngleCalc**, float **distTolerance**, float **angleTolerance**, bool **updateAllFace**)

  Constructor.

Protected Member Functions

- virtual void **createColorCubeMap** ()=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.

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.

5.4.1 Detailed Description

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

A color cubemap is a cubemap of the colors of the surrounding environment.

5.4.2 Constructor & Destructor Documentation

5.4.2.1 ColorCubeMapRenderingRun::ColorCubeMapRenderingRun (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 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

5.4.3 Member Function Documentation

5.4.3.1 virtual bool ColorCubeMapRenderingRun::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 **OgreColorCubeMapRenderingRun** (p. 58).

5.4.3.2 virtual void ColorCubeMapRenderingRun::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 **OgreColorCubeMapRenderingRun** (p. 58).

5.4.3.3 void ColorCubeMapRenderingRun::updateFrame (unsigned long *frameNum*) [protected, virtual]

This function does the actual update in a frame.

Parameters:
frameNum current frame number

Reimplemented from **RenderingRun** (p. 126).

5.4.4 Member Data Documentation

5.4.4.1 float ColorCubeMapRenderingRun::angleTolerance [protected]

A value used in face skip test.

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

5.4.4.2 float ColorCubeMapRenderingRun::distTolerance [protected]

A value used in face skip test.

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

5.4.4.3 bool ColorCubeMapRenderingRun::useDistCalc [protected]

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

See also:
distTolerance(p. 23)

5.4.4.4 bool ColorCubeMapRenderingRun::useFaceAngleCalc [protected]

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

See also:
angleTolerance(p. 23)

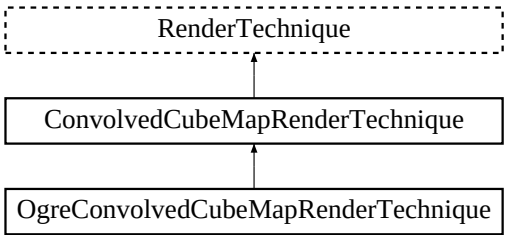
The documentation for this class was generated from the following files:

- ColorCubeMapRenderingRun.h
- ColorCubeMapRenderingRun.cpp

5.5 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**, **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.

Protected Member Functions

- virtual void **reducedCubeMapRunChanged** (**RenderingRun** ∗run)=0

  Called if the changed run is a **ColorCubeMapRenderingRun**(p. 21).

- virtual void **colorCubeMapRunChanged** (**RenderingRun** ∗run)=0

  Called if the changed run is a **ColorCubeMapRenderingRun**(p. 21).

- virtual **RenderingRun** ∗ **createColorCubeMapRun** ()=0

  Creates a **ColorCubeMapRenderingRun**(p. 21).

- virtual **RenderingRun** ∗ **createReducedCubeMapRun** ()=0

  Creates a **ColorCubeMapRenderingRun**(p. 21).

Protected Attributes

- 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 int **reducedCubeMapResolution**

  color-cubemap resolution

- 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 **startFrame**

  offset in frame number used during update

5.5.1 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.

5.5.2 Constructor & Destructor Documentation

5.5.2.1 ConvolvedCubeMapRenderTechnique::ConvolvedCubeMapRenderTechnique (unsigned long *startFrame*, unsigned long *cubeMapUpdateInterval*, unsigned int *cubeMapResolution*, unsigned int *reducedCubeMapResolution*, bool *useDistCalc*, bool *useFaceAngleCalc*, float *distTolerance*, float *angleTolerance*, bool *updateAllFace*, 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
parentRenderable the object to operate on
parentTechniqueGroup the **TechniqueGroup**(p. 141) this RenderedTechnique is attached to

5.5.3 Member Function Documentation

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

Called if the changed run is a **ColorCubeMapRenderingRun**(p. 21).

Parameters:
run pointer to the changed **ColorCubeMapRenderingRun**(p. 21)

Implemented in **OgreConvolvedCubeMapRenderTechnique** (p. 61).

5.5.3.2 virtual RenderingRun* ConvolvedCubeMapRenderTechnique::createColorCubeMapRun () [protected, pure virtual]

Creates a **ColorCubeMapRenderingRun**(p. 21).

Returns:
the new **ColorCubeMapRenderingRun**(p. 21) instance.

Implemented in **OgreConvolvedCubeMapRenderTechnique** (p. 61).

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

Creates a **ColorCubeMapRenderingRun**(p. 21).

Returns:
the new **ColorCubeMapRenderingRun**(p. 21) instance.

Implemented in **OgreConvolvedCubeMapRenderTechnique** (p. 62).

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

Called if the changed run is a **ColorCubeMapRenderingRun**(p. 21).

Parameters:
run pointer to the changed **ColorCubeMapRenderingRun**(p. 21)

Implemented in **OgreConvolvedCubeMapRenderTechnique** (p. 62).

5.5.3.5 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**(p. 124)

Reimplemented from **RenderTechnique** (p. 128).

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

Updates the resources in the given frame.

Parameters:

frameNum the actual framenumber

Reimplemented from **RenderTechnique** (p. 129).

Reimplemented in **OgreConvolvedCubeMapRenderTechnique** (p. 62).

5.5.4 Member Data Documentation

5.5.4.1 float ConvolvedCubeMapRenderTechnique::angleTolerance [protected]

A value used in face skip test.

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

5.5.4.2 float ConvolvedCubeMapRenderTechnique::distTolerance [protected]

A value used in face skip test.

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

5.5.4.3 bool ConvolvedCubeMapRenderTechnique::useDistCalc [protected]

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

See also:

distTolerance(p. 27)

5.5.4.4 bool ConvolvedCubeMapRenderTechnique::useFaceAngleCalc [protected]

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

See also:

angleTolerance(p. 27)

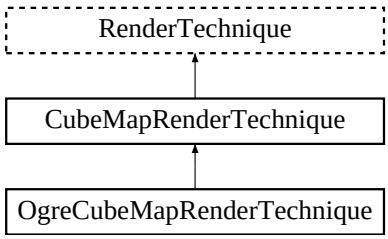
The documentation for this class was generated from the following files:

- ConvolvedCubeMapRenderTechnique.h
- ConvolvedCubeMapRenderTechnique.cpp

5.6 CubeMapRenderTechnique Class Reference

Base abstract class of rendering a color 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**, **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.

Protected Member Functions

- virtual void **colorCubeMapRunChanged** (**RenderingRun** ∗run)=0

  Called if the changed run is a **ColorCubeMapRenderingRun**(p. 21).

- virtual **RenderingRun** ∗ **createColorCubeMapRun** ()=0

  Creates a **ColorCubeMapRenderingRun**(p. 21).

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

5.6.1 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.

5.6.2 Constructor & Destructor Documentation

5.6.2.1 CubeMapRenderTechnique::CubeMapRenderTechnique (unsigned long *startFrame*, unsigned long *cubeMapUpdateInterval*, unsigned int *cubeMapResolution*, bool *useDistCalc*, bool *useFaceAngleCalc*, float *distTolerance*, float *angleTolerance*, bool *updateAllFace*, 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

parentRenderable the object to operate on

parentTechniqueGroup the **TechniqueGroup**(p. 141) this RenderedTechnique is attached to

5.6.3 Member Function Documentation

5.6.3.1 virtual void CubeMapRenderTechnique::colorCubeMapRunChanged (RenderingRun * *run*) [protected, pure virtual]

Called if the changed run is a **ColorCubeMapRenderingRun**(p. 21).

Parameters:
run pointer to the changed **ColorCubeMapRenderingRun**(p. 21)

Implemented in **OgreCubeMapRenderTechnique** (p. 64).

5.6.3.2 virtual RenderingRun* CubeMapRenderTechnique::createColorCubeMapRun () [protected, pure virtual]

Creates a **ColorCubeMapRenderingRun**(p. 21).

Returns:
the new **ColorCubeMapRenderingRun**(p. 21) instance.

Implemented in **OgreCubeMapRenderTechnique** (p. 64).

5.6.3.3 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**(p. 124)

Reimplemented from **RenderTechnique** (p. 128).

5.6.3.4 void CubeMapRenderTechnique::update (unsigned long *frameNum*) [virtual]

Updates the resources in the given frame.

Parameters:
frameNum the actual framenumber

Reimplemented from **RenderTechnique** (p. 129).

Reimplemented in **OgreCubeMapRenderTechnique** (p. 65).

5.6.4 Member Data Documentation

5.6.4.1 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.

5.6.4.2 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.

5.6.4.3 bool CubeMapRenderTechnique::useDistCalc [protected]

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

See also:
distTolerance(p. 32)

5.6.4.4 bool CubeMapRenderTechnique::useFaceAngleCalc [protected]

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

See also:
angleTolerance(p. 31)

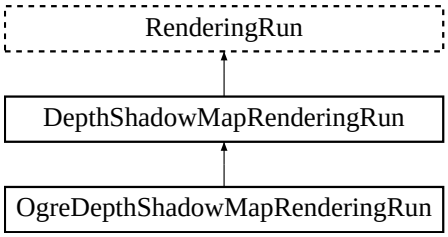
The documentation for this class was generated from the following files:

- CubeMapRenderTechnique.h
- CubeMapRenderTechnique.cpp

5.7 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.

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.

Protected Attributes

- unsigned int **resolutionX**
  width of the depth map texture
- unsigned int **resolutionY**
  height of the depth map texture

5.7.1 Detailed Description

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

A shadow map stores depth values from the lightsource.

5.7.2 Constructor & Destructor Documentation

5.7.2.1 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

5.7.3 Member Function Documentation

5.7.3.1 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** (p. 67).

5.7.3.2 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** (p. 126).

Implemented in **OgreDepthShadowMapRenderingRun** (p. 68).

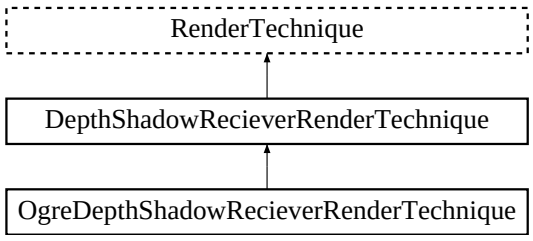
The documentation for this class was generated from the following file:

- DepthShadowMapRenderingRun.h

5.8 DepthShadowRecieverRenderTechnique Class Reference

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

Inheritance diagram for DepthShadowRecieverRenderTechnique::



Public Member Functions

- **DepthShadowRecieverRenderTechnique** (**ElementaryRenderable** ∗**parentRenderable**, **TechniqueGroup** ∗**parentTechniqueGroup**)

  Constructor.

5.8.1 Detailed Description

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

5.8.2 Constructor & Destructor Documentation

5.8.2.1 DepthShadowRecieverRenderTechnique::DepthShadowRecieverRenderTechnique (ElementaryRenderable ∗ *parentRenderable*, TechniqueGroup ∗ *parentTechniqueGroup*)

Constructor.

Parameters:

parentRenderable the object to operate on

parentTechniqueGroup the **TechniqueGroup**(p. 141) this RenderedTechnique is attached to

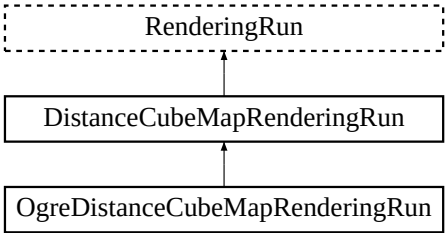
The documentation for this class was generated from the following files:

- DepthShadowRecieverRenderTechnique.h
- DepthShadowRecieverRenderTechnique.cpp

5.9 DistanceCubeMapRenderingRun Class Reference

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

Inheritance diagram for DistanceCubeMapRenderingRun::



Public Member Functions

- **DistanceCubeMapRenderingRun** (unsigned long **startFrame**, unsigned long **updateInterval**, unsigned int **resolution**, bool **useDistCalc**, bool **useFaceAngleCalc**, float **distTolerance**, float **angleTolerance**, bool **updateAllFace**)

  Constructor.

Protected Member Functions

- virtual void **createDistanceCubeMap** ()=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.

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.

5.9.1 Detailed Description

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

A distance cubemap stores the distances of the surrounding environment from the cubemap center.

5.9.2 Constructor & Destructor Documentation

5.9.2.1 DistanceCubeMapRenderingRun::DistanceCubeMapRenderingRun (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 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

5.9.3 Member Function Documentation

5.9.3.1 virtual bool DistanceCubeMapRenderingRun::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 **OgreDistanceCubeMapRenderingRun** (p. 73).

5.9.3.2 virtual void DistanceCubeMapRenderingRun::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 **OgreDistanceCubeMapRenderingRun** (p. 73).

5.9.3.3 void DistanceCubeMapRenderingRun::updateFrame (unsigned long *frameNum*)

`[protected, virtual]`

This function does the actual update in a frame.

Parameters:
frameNum current frame number

Reimplemented from **RenderingRun** (p. 126).

5.9.4 Member Data Documentation

5.9.4.1 float DistanceCubeMapRenderingRun::angleTolerance `[protected]`

A value used in face skip test.

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

5.9.4.2 float DistanceCubeMapRenderingRun::distTolerance `[protected]`

A value used in face skip test.

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

5.9.4.3 bool DistanceCubeMapRenderingRun::useDistCalc `[protected]`

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

See also:
distTolerance(p. 38)

5.9.4.4 bool DistanceCubeMapRenderingRun::useFaceAngleCalc `[protected]`

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

See also:
angleTolerance(p. 38)

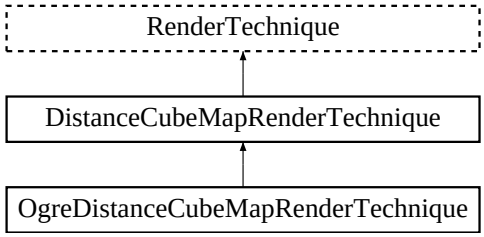
The documentation for this class was generated from the following files:

- DistanceCubeMapRenderingRun.h
- DistanceCubeMapRenderingRun.cpp

5.10 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**, **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.

Protected Member Functions

- virtual void **distanceCubeMapRunChanged** (**RenderingRun** ∗run)=0

  Called if the changed run is a **ColorCubeMapRenderingRun**(p. 21).

- virtual void **distanceCubeMapRunUpdated** (**RenderingRun** ∗run)=0

  Called if the changed run is a **ColorCubeMapRenderingRun**(p. 21).

- virtual **RenderingRun** ∗ **createDistanceCubeMapRun** ()=0

  Creates a **ColorCubeMapRenderingRun**(p. 21).

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

5.10.1 Detailed Description

Base abstract class of rendering a distance cube map.

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

5.10.2 Constructor & Destructor Documentation

5.10.2.1 DistanceCubeMapRenderTechnique::DistanceCubeMapRenderTechnique (unsigned long *startFrame*, unsigned long *cubeMapUpdateInterval*, unsigned int *cubeMapResolution*, bool *useDistCalc*, bool *useFaceAngleCalc*, float *distTolerance*, float *angleTolerance*, bool *updateAllFace*, 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

parentRenderable the object to operate on

parentTechniqueGroup the **TechniqueGroup**(p. 141) this RenderedTechnique is attached to

5.10.3 Member Function Documentation

5.10.3.1 virtual RenderingRun* DistanceCubeMapRenderTechnique::createDistanceCubeMapRun () [protected, pure virtual]

Creates a **ColorCubeMapRenderingRun**(p. 21).

Returns:
the new **ColorCubeMapRenderingRun**(p. 21) instance.

Implemented in **OgreDistanceCubeMapRenderTechnique** (p. 76).

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

Called if the changed run is a **ColorCubeMapRenderingRun**(p. 21).

Parameters:
run pointer to the changed **ColorCubeMapRenderingRun**(p. 21)

Implemented in **OgreDistanceCubeMapRenderTechnique** (p. 76).

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

Called if the changed run is a **ColorCubeMapRenderingRun**(p. 21).

Parameters:
run pointer to the changed **ColorCubeMapRenderingRun**(p. 21)

Implemented in **OgreDistanceCubeMapRenderTechnique** (p. 77).

5.10.3.4 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**(p. 124)

Reimplemented from **RenderTechnique** (p. 128).

5.10.3.5 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**(p. 124)

Reimplemented from **RenderTechnique** (p. 129).

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

Updates the resources in the given frame.

Parameters:
frameNum the actual framenumber

Reimplemented from **RenderTechnique** (p. 129).

Reimplemented in **OgreDistanceCubeMapRenderTechnique** (p. 77).

5.10.4 Member Data Documentation

5.10.4.1 float DistanceCubeMapRenderTechnique::angleTolerance [protected]

A value used in face skip test.

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

5.10.4.2 float DistanceCubeMapRenderTechnique::distTolerance [protected]

A value used in face skip test.

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

5.10.4.3 bool DistanceCubeMapRenderTechnique::useDistCalc [protected]

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

See also:
distTolerance(p. 42)

5.10.4.4 bool DistanceCubeMapRenderTechnique::useFaceAngleCalc [protected]

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

See also:
angleTolerance(p. 42)

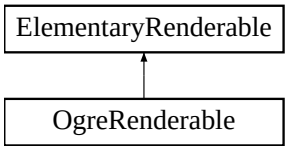
The documentation for this class was generated from the following files:

- DistanceCubeMapRenderTechnique.h
- DistanceCubeMapRenderTechnique.cpp

5.11 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.

5.11.1 Detailed Description

Base abstract class for an elementary renderable.

5.11.2 Member Function Documentation

5.11.2.1 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 recievers, shadow casters) if needed. Each group has a unique ID.

Parameters:

groupID the ID of the group to use

Implemented in **OgreRenderable** (p. 96).

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

Shows or hides the renderable.

Parameters:
visible visibility

Implemented in **OgreRenderable** (p. 96).

The documentation for this class was generated from the following file:

- ElementaryRenderable.h

5.12 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.

5.12.1 Detailed Description

Base abstract class of the illumination manager.

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

5.12.2 Member Function Documentation

5.12.2.1 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**(p. 135) instance to add

5.12.2.2 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).

5.12.2.3 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**(p. 135) instance that are joined

old2 pointer to the other **SharedRuns**(p. 135) instance that are joined

newsr pointer to the resulting parent **SharedRuns**(p. 135) instance

5.12.2.4 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**(p. 135) instance that is split

new1 pointer to one of the **SharedRuns**(p. 135) instance that remain after split

new2 pointer to the other **SharedRuns**(p. 135) instance that remain after split

5.12.2.5 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, updates render techniques and finally renders the scene to framebuffer.

Parameters:

frameNumber current framenumber

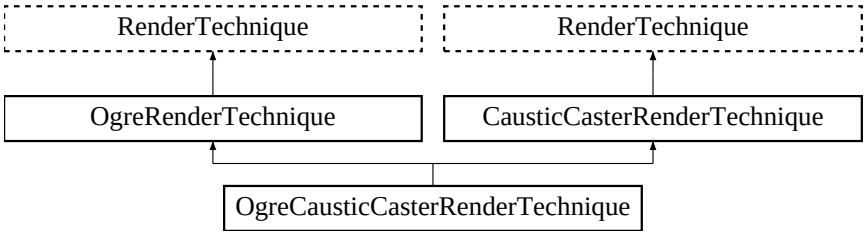
The documentation for this class was generated from the following file:

- IlluminationManager.h

5.13 OgreCausticCasterRenderTechnique Class Reference

CausticCasterRenderTechnique(p. 13) used in an OGRE environment.

Inheritance diagram for OgreCausticCasterRenderTechnique::



Public Member Functions

- **OgreCausticCasterRenderTechnique** (unsigned long **startFrame**, unsigned long **photonMapUpdateInterval**, unsigned int **photonMapResolution**, unsigned int custicCubeMapResolution, String **photonMapMaterialName**, String **causticMapMaterialName**, unsigned char **photonMapTexID**, bool **updateAllFace**, bool **useDistance**, Pass ∗**pass**, **OgreRenderable** ∗**parentRenderable**, **OgreTechniqueGroup** ∗**parentTechniqueGroup**)

  Constructor.

- ∼**OgreCausticCasterRenderTechnique** ()

  Destructor.

- String & **getCausticCubeMapName** ()

  Returns the name of the created caustic cubemap.

Protected Member Functions

- virtual void **photonMapRunChanged** (**RenderingRun** ∗run)

  Called if the changed run is a **PhotonMapRenderingRun**(p. 118).

- virtual void **causticCubeMapRunChanged** (**RenderingRun** ∗run)

  Called if the changed run is a **CausticCubeMapRenderingRun**(p. 17).

- virtual void **distanceCubeMapRunChanged** (**RenderingRun** ∗run)

  Called if the changed run is a **DistanceCubeMapRenderingRun**(p. 36).

- virtual **RenderingRun** ∗ **createPhotonMapRun** ()

  Creates a **PhotonMapRenderingRun**(p. 118).

- virtual **RenderingRun** ∗ **createCausticCubeMapRun** ()

  Creates a **CausticCubeMapRenderingRun**(p. 17).

- virtual **RenderingRun** ∗ **createDistanceCubeMapRun** ()

  Creates a **DistanceCubeMapRenderingRun**(p. 36).

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.

5.13.1 Detailed Description

CausticCasterRenderTechnique(p. 13) used in an OGRE environment.

5.13.2 Constructor & Destructor Documentation

5.13.2.1 OgreCausticCasterRenderTechnique::OgreCausticCasterRenderTechnique (unsigned long *startFrame*, unsigned long *photonMapUpdateInterval*, unsigned int *photonMapResolution*, unsigned int *custicCubeMapResolution*, String *photonMapMaterialName*, String *causticMapMaterialName*, unsigned char *photonMapTexID*, bool *updateAllFace*, bool *useDistance*, 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 choton 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)

pass the pass to operate on

parentRenderable the object to operate on

parentTechniqueGroup the **TechniqueGroup**(p. 141) this RenderedTechnique is attached to

5.13.3 Member Function Documentation

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

Called if the changed run is a **CausticCubeMapRenderingRun**(p. 17).

Parameters:
run pointer to the changed **CausticCubeMapRenderingRun**(p. 17)

Implements **CausticCasterRenderTechnique** (p. 15).

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

Creates a **CausticCubeMapRenderingRun**(p. 17).

Returns:
the new **CausticCubeMapRenderingRun**(p. 17) instance.

Implements **CausticCasterRenderTechnique** (p. 15).

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

Creates a **DistanceCubeMapRenderingRun**(p. 36).

Returns:
the new **DistanceCubeMapRenderingRun**(p. 36) instance.

Implements **CausticCasterRenderTechnique** (p. 15).

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

Creates a **PhotonMapRenderingRun**(p. 118).

Returns:
the new **PhotonMapRenderingRun**(p. 118) instance.

Implements **CausticCasterRenderTechnique** (p. 16).

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

Called if the changed run is a **DistanceCubeMapRenderingRun**(p. 36).

Parameters:
run pointer to the changed **DistanceCubeMapRenderingRun**(p. 36)

Implements **CausticCasterRenderTechnique** (p. 16).

5.13.3.6 String & OgreCausticCasterRenderTechnique::getCausticCubeMapName ()

Returns the name of the created caustic cubemap.

Returns:
name of the caustic cubemap texture

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

Called if the changed run is a **PhotonMapRenderingRun**(p. 118).

Parameters:
run pointer to the changed **PhotonMapRenderingRun**(p. 118)

Implements **CausticCasterRenderTechnique** (p. 16).

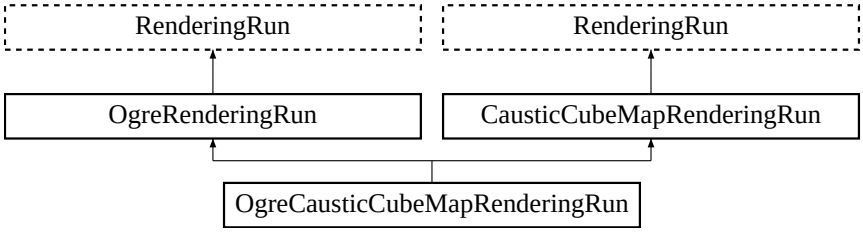
The documentation for this class was generated from the following files:

- OgreCausticCasterRenderTechnique.h
- OgreCausticCasterRenderTechnique.cpp

5.14 OgreCausticCubeMapRenderingRun Class Reference

ColorCubeMapRenderingRun(p. 21) 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**)

  Constructor.

- String & **getCausticCubeMapTextureName** ()

  returns the name of the resulting caustic cubemap texture

- void **photonMapChanged** (**RenderingRun** ∗run)

  Called if the changed run is a **PhotonMapRenderingRun**(p. 118).

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.

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**(p. 108) *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

- String **materialName**

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

5.14.1 Detailed Description

ColorCubeMapRenderingRun(p. 21) used in an OGRE environment.

5.14.2 Constructor & Destructor Documentation

5.14.2.1 OgreCausticCubeMapRenderingRun::OgreCausticCubeMapRenderingRun (OgreSharedRuns * *sharedRuns*, String *name*, unsigned long *startFrame*, unsigned long *updateInterval*, unsigned int *resolution*, String *materialName*, unsigned char *photonMapTexId*, bool *updateAllFace*)

Constructor.

Parameters:

sharedRuns a pointer to the **OgreSharedRuns**(p. 108) 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

5.14.3 Member Function Documentation

5.14.3.1 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** (p. 18).

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

Called if the changed run is a **PhotonMapRenderingRun**(p. 118).

Parameters:

run pointer to the changed **PhotonMapRenderingRun**(p. 118)

Implements **CausticCubeMapRenderingRun** (p. 18).

5.14.3.3 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** (p. 18).

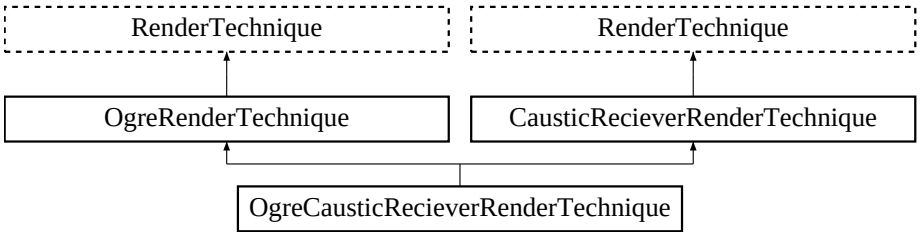
The documentation for this class was generated from the following files:

- OgreCausticCubeMapRenderingRun.h
- OgreCausticCubeMapRenderingRun.cpp

5.15 OgreCausticRecieverRenderTechnique Class Reference

CausticRecieverRenderTechnique(p. 20) used in an OGRE environment.

Inheritance diagram for OgreCausticRecieverRenderTechnique::



Public Member Functions

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

  Constructor.

- ∼**OgreCausticRecieverRenderTechnique** ()

  Destructor.

- virtual void **update** (unsigned long frameNum)

  Updates the resources in the given frame.

Protected Attributes

- int **maxcasters**

  the maximum number of caustic casters from which this reciever 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

5.15.1 Detailed Description

CausticRecieverRenderTechnique(p. 20) 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 reciever 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.

5.15.2 Constructor & Destructor Documentation

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

Constructor.

Parameters:

maxcasters the maximum number of caustic casters from which this reciever 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**(p. 141) this RenderedTechnique is attached to

5.15.3 Member Function Documentation

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

Updates the resources in the given frame.

A **RenderTechnique**(p. 127) is usually need some resources from several runs, so these runs will be updated.

Parameters:

frameNum the actual framenumber

Reimplemented from **RenderTechnique** (p. 129).

5.15.4 Member Data Documentation

5.15.4.1 String OgreCausticRecieverRenderTechnique::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.

5.15.4.2 std::vector<Pass*> OgreCausticRecieverRenderTechnique::passes [protected]

new passes created by this technique

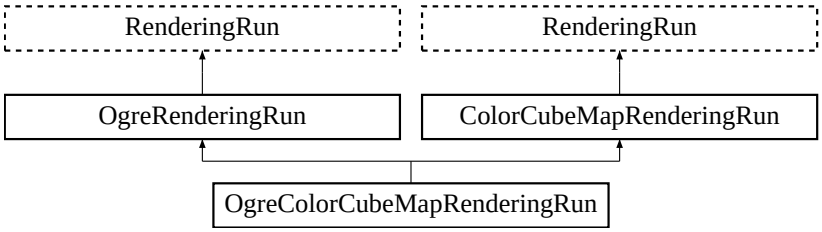
The documentation for this class was generated from the following files:

- OgreCausticRecieverRenderTechnique.h
- OgreCausticRecieverRenderTechnique.cpp

5.16 OgreColorCubeMapRenderingRun Class Reference

ColorCubeMapRenderingRun(p. 21) used in an OGRE environment.

Inheritance diagram for OgreColorCubeMapRenderingRun::



Public Member Functions

- **OgreColorCubeMapRenderingRun** (**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 **getColorCubeMapTextureName** ()

  returns the name of the resulting color cubemap texture

Protected Member Functions

- void **createColorCubeMap** ()

  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.

Protected Attributes

- **OgreSharedRuns** ∗ **sharedRuns**

  a pointer to the **OgreSharedRuns**(p. 108) *this run belongs to*

- String **name**

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

- Texture ∗ **colorCubemapTexture**

  a pointer to the cubemap texture that was created by this run

5.16.1 Detailed Description

ColorCubeMapRenderingRun(p. 21) used in an OGRE environment.

5.16.2 Constructor & Destructor Documentation

5.16.2.1 OgreColorCubeMapRenderingRun::OgreColorCubeMapRenderingRun (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**(p. 108) 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

5.16.3 Member Function Documentation

5.16.3.1 bool OgreColorCubeMapRenderingRun::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 **ColorCubeMapRenderingRun** (p. 22).

5.16.3.2 void OgreColorCubeMapRenderingRun::updateCubeFace (int *facenum*) [inline, protected, virtual]

Updates one face of the cubemap.

Parameters:

facenum the number of the face to be updated

Implements **ColorCubeMapRenderingRun** (p. 23).

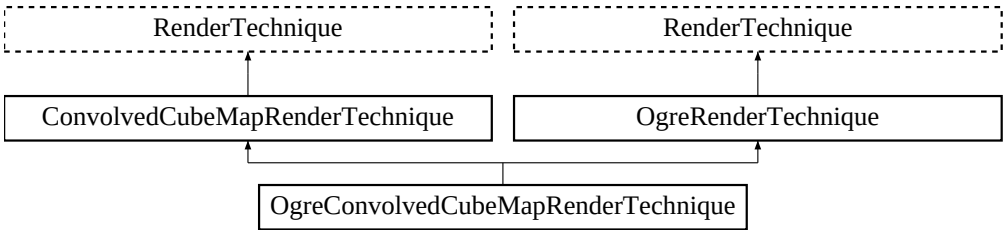
The documentation for this class was generated from the following files:

- OgreColorCubeMapRenderingRun.h
- OgreColorCubeMapRenderingRun.cpp

5.17 OgreConvolvedCubeMapRenderTechnique Class Reference

ConvolvedCubeMapRenderTechnique(p. 24) 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 **reducedTexID**, bool **useDistCalc**, bool **useFaceAngleCalc**, float **distTolerance**, float **angleTolerance**, bool **updateAllFace**, Pass ∗**pass**, **OgreRenderable** ∗**parentRenderable**, **OgreTechniqueGroup** ∗**parentTechniqueGroup**)

  Constructor.

- ∼**OgreConvolvedCubeMapRenderTechnique** ()

  Destructor.

- void **update** (unsigned long frameNum)

  Updates the resources in the given frame.

Protected Member Functions

- void **reducedCubeMapRunChanged** (**RenderingRun** ∗run)

  Called if the changed run is a **ColorCubeMapRenderingRun**(p. 21).

- void **colorCubeMapRunChanged** (**RenderingRun** ∗run)

  Called if the changed run is a **ColorCubeMapRenderingRun**(p. 21).

- **RenderingRun** ∗ **createColorCubeMapRun** ()

  Creates a **ColorCubeMapRenderingRun**(p. 21).

- **RenderingRun** ∗ **createReducedCubeMapRun** ()

  Creates a **ColorCubeMapRenderingRun**(p. 21).

Protected Attributes

- unsigned char **reducedTexID**

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

5.17.1 Detailed Description

ConvolvedCubeMapRenderTechnique(p. 24) used in an Ogre environment.

5.17.2 Constructor & Destructor Documentation

5.17.2.1 OgreConvolvedCubeMapRenderTechnique::OgreConvolvedCubeMapRenderTechnique (unsigned long *startFrame*, unsigned long *cubeMapUpdateInterval*, unsigned int *cubeMapResolution*, unsigned int *reducedCubeMapResolution*, unsigned char *reducedTexID*, bool *useDistCalc*, bool *useFaceAngleCalc*, float *distTolerance*, float *angleTolerance*, bool *updateAllFace*, 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

pass the pass to operate on

parentRenderable the object to operate on

parentTechniqueGroup the **TechniqueGroup**(p. 141) this RenderedTechnique is attached to

5.17.3 Member Function Documentation

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

Called if the changed run is a **ColorCubeMapRenderingRun**(p. 21).

Parameters:

run pointer to the changed **ColorCubeMapRenderingRun**(p. 21)

Implements **ConvolvedCubeMapRenderTechnique** (p. 26).

5.17.3.2 RenderingRun * OgreConvolvedCubeMapRenderTechnique::createColorCubeMapRun () [protected, virtual]

Creates a **ColorCubeMapRenderingRun**(p. 21).

Returns:
the new **ColorCubeMapRenderingRun**(p. 21) instance.

Implements **ConvolvedCubeMapRenderTechnique** (p. 26).

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

Creates a **ColorCubeMapRenderingRun**(p. 21).

Returns:
the new **ColorCubeMapRenderingRun**(p. 21) instance.

Implements **ConvolvedCubeMapRenderTechnique** (p. 26).

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

Called if the changed run is a **ColorCubeMapRenderingRun**(p. 21).

Parameters:
run pointer to the changed **ColorCubeMapRenderingRun**(p. 21)

Implements **ConvolvedCubeMapRenderTechnique** (p. 26).

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

Updates the resources in the given frame.

Parameters:
frameNum the actual framenumber

Reimplemented from **ConvolvedCubeMapRenderTechnique** (p. 27).

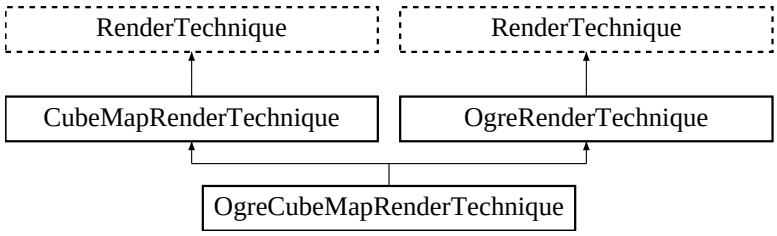
The documentation for this class was generated from the following files:

- OgreConvolvedCubeMapRenderTechnique.h
- OgreConvolvedCubeMapRenderTechnique.cpp

5.18 OgreCubeMapRenderTechnique Class Reference

CubeMapRenderTechnique(p. 29) 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**, Pass ∗**pass**, **OgreRenderable** ∗**parentRenderable**, **OgreTechniqueGroup** ∗**parentTechniqueGroup**)

  Constructor.

- ∼**OgreCubeMapRenderTechnique** ()

  Destructor.

- void **update** (unsigned long frameNum)

  Updates the resources in the given frame.

Protected Member Functions

- void **colorCubeMapRunChanged** (**RenderingRun** ∗run)

  Called if the changed run is a **ColorCubeMapRenderingRun**(p. 21).

- **RenderingRun** ∗ **createColorCubeMapRun** ()

  Creates a **ColorCubeMapRenderingRun**(p. 21).

Protected Attributes

- unsigned char **texID**

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

5.18.1 Detailed Description

CubeMapRenderTechnique(p. 29) used in an Ogre environment.

5.18.2 Constructor & Destructor Documentation

5.18.2.1 OgreCubeMapRenderTechnique::OgreCubeMapRenderTechnique (unsigned long *startFrame*, unsigned long *cubeMapUpdateInterval*, unsigned int *cubeMapResolution*, unsigned char *texID*, bool *useDistCalc*, bool *useFaceAngleCalc*, float *distTolerance*, float *angleTolerance*, bool *updateAllFace*, 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

pass the pass to operate on

parentRenderable the object to operate on

parentTechniqueGroup the **TechniqueGroup**(p. 141) this RenderedTechnique is attached to

5.18.3 Member Function Documentation

5.18.3.1 void OgreCubeMapRenderTechnique::colorCubeMapRunChanged (RenderingRun * *run*) [protected, virtual]

Called if the changed run is a **ColorCubeMapRenderingRun**(p. 21).

Parameters:

run pointer to the changed **ColorCubeMapRenderingRun**(p. 21)

Implements **CubeMapRenderTechnique** (p. 31).

5.18.3.2 RenderingRun * OgreCubeMapRenderTechnique::createColorCubeMapRun () [protected, virtual]

Creates a **ColorCubeMapRenderingRun**(p. 21).

Returns:

the new **ColorCubeMapRenderingRun**(p. 21) instance.

Implements **CubeMapRenderTechnique** (p. 31).

5.18.3.3 void OgreCubeMapRenderTechnique::update (unsigned long *frameNum*) [virtual]

Updates the resources in the given frame.

Parameters:
frameNum the actual framenumber

Reimplemented from **CubeMapRenderTechnique** (p. 31).

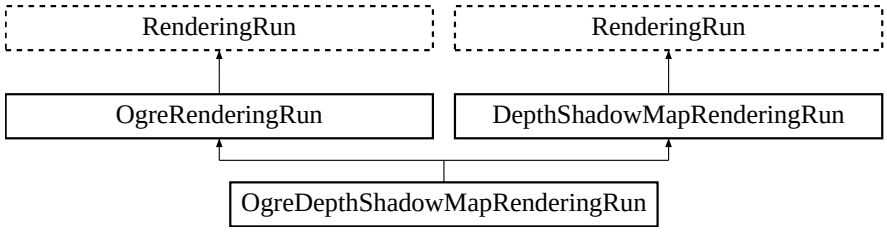
The documentation for this class was generated from the following files:

- OgreCubeMapRenderTechnique.h
- OgreCubeMapRenderTechnique.cpp

5.19 OgreDepthShadowMapRenderingRun Class Reference

ColorCubeMapRenderingRun(p. 21) 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.

- String **getDepthMapTextureName** ()

  returns the depth shadow map texture created by this run

- void **refreshLight** ()

  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

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).

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**(p. 108) *this run belongs to*

- String **name**

  the name of the 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

5.19.1 Detailed Description

ColorCubeMapRenderingRun(p. 21) used in an OGRE environment.

5.19.2 Constructor & Destructor Documentation

5.19.2.1 OgreDepthShadowMapRenderingRun::OgreDepthShadowMapRenderingRun (OgreSharedRuns * *sharedRuns*, String *name*, Light * *light*, unsigned int *resolutionX*, unsigned int *resolutionY*, String *materialName*)

Constructor.

Parameters:

sharedRuns a pointer to the **OgreSharedRuns**(p. 108) 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

5.19.3 Member Function Documentation

5.19.3.1 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** (p. 34).

5.19.3.2 void OgreDepthShadowMapRenderingRun::updateFrame (unsigned long *frameNum*) [protected, virtual]

This function does the actual update in a frame.

Parameters:
frameNum current frame number

Implements **DepthShadowMapRenderingRun** (p. 34).

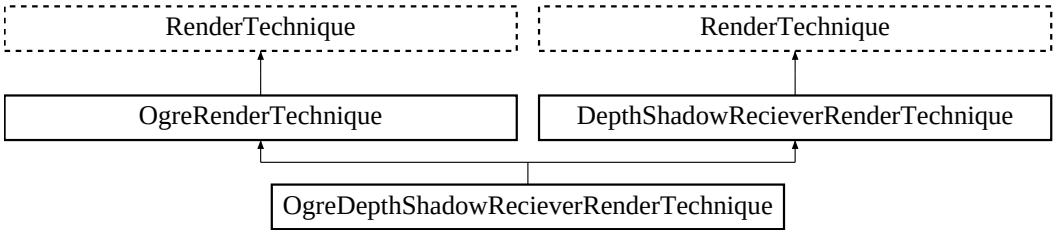
The documentation for this class was generated from the following files:

- OgreDepthShadowMapRenderingRun.h
- OgreDepthShadowMapRenderingRun.cpp

5.20 OgreDepthShadowRecieverRenderTechnique Class Reference

DepthShadowRecieverRenderTechnique(p. 35) used in an OGRE environment.

Inheritance diagram for OgreDepthShadowRecieverRenderTechnique::



Public Member Functions

- **OgreDepthShadowRecieverRenderTechnique** (int **maxlights**, String **shadowVertexProgram**, String **shadowFragmentProgram**, Pass ∗**pass**, **OgreRenderable** ∗**parentRenderable**, **OgreTechniqueGroup** ∗**parentTechniqueGroup**)

  Constructor.

- ∼**OgreDepthShadowRecieverRenderTechnique** ()

  Destructor.

- virtual void **update** (unsigned long frameNum)

  Updates the resources in the given frame.

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**

5.20.1 Detailed Description

DepthShadowRecieverRenderTechnique(p. 35) 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 recievers are visible. It is the shadow reciever 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.

5.20.2 Constructor & Destructor Documentation

5.20.2.1 OgreDepthShadowRecieverRenderTechnique::OgreDepthShadowRecieverRenderTechnique (int *maxlights*, String *shadowVertexProgram*, String *shadowFragmentProgram*, 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.

pass the pass after which shadowing passes should be added

parentRenderable the object to operate on

parentTechniqueGroup the **TechniqueGroup**(p. 141) this RenderedTechnique is attached to

5.20.3 Member Function Documentation

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

Updates the resources in the given frame.

A **RenderTechnique**(p. 127) is usually need some resources from several runs, so these runs will be updated.

Parameters:

frameNum the actual framenumber

Reimplemented from **RenderTechnique** (p. 129).

5.20.4 Member Data Documentation

5.20.4.1 int OgreDepthShadowRecieverRenderTechnique::maxlights [protected]

the maximum number of light sources to recieve shadow from

During update the nearest light sources will be found and used.

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

new passes created by this technique

5.20.4.3 String OgreDepthShadowRecieverRenderTechnique::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.

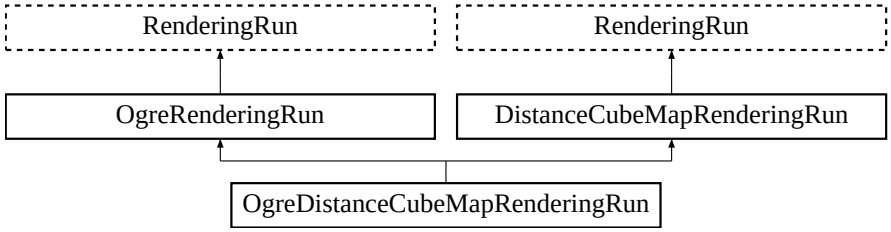
The documentation for this class was generated from the following files:

- OgreDepthShadowRecieverRenderTechnique.h
- OgreDepthShadowRecieverRenderTechnique.cpp

5.21 OgreDistanceCubeMapRenderingRun Class Reference

ColorCubeMapRenderingRun(p. 21) used in an OGRE environment.

Inheritance diagram for OgreDistanceCubeMapRenderingRun::



Public Member Functions

- **OgreDistanceCubeMapRenderingRun** (**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 **getDistanceCubeMapTextureName** ()

  returns the name of the resulting distance cubemap texture

Protected Member Functions

- void **createDistanceCubeMap** ()

  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.

Protected Attributes

- **OgreSharedRuns** ∗ **sharedRuns**

  a pointer to the **OgreSharedRuns**(p. 108) *this run belongs to*

- String **name**

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

- Texture ∗ **distanceCubemapTexture**

  a pointer to the cubemap texture that was created by this run

5.21.1 Detailed Description

ColorCubeMapRenderingRun(p. 21) used in an OGRE environment.

5.21.2 Constructor & Destructor Documentation

5.21.2.1 OgreDistanceCubeMapRenderingRun::OgreDistanceCubeMapRenderingRun (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**(p. 108) 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

5.21.3 Member Function Documentation

5.21.3.1 bool OgreDistanceCubeMapRenderingRun::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 **DistanceCubeMapRenderingRun** (p. 37).

5.21.3.2 void OgreDistanceCubeMapRenderingRun::updateCubeFace (int *facenum*) [inline, protected, virtual]

Updates one face of the cubemap.

Parameters:

facenum the number of the face to be updated

Implements **DistanceCubeMapRenderingRun** (p. 38).

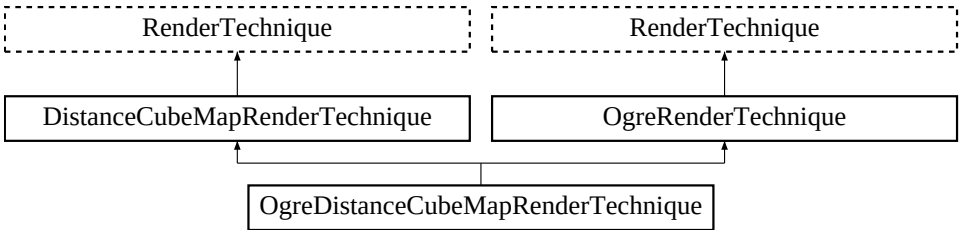
The documentation for this class was generated from the following files:

- OgreDistanceCubeMapRenderingRun.h
- OgreDistanceCubeMapRenderingRun.cpp

5.22 OgreDistanceCubeMapRenderTechnique Class Reference

DistanceCubeMapRenderTechnique(p. 39) 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**, Pass ∗**pass**, **OgreRenderable** ∗**parentRenderable**, **OgreTechniqueGroup** ∗**parentTechniqueGroup**)

  Constructor.

- ∼**OgreDistanceCubeMapRenderTechnique** ()

  Destructor.

- void **update** (unsigned long frameNum)

  Updates the resources in the given frame.

Protected Member Functions

- void **distanceCubeMapRunChanged** (**RenderingRun** ∗run)

  Called if the changed run is a **ColorCubeMapRenderingRun**(p. 21).

- **RenderingRun** ∗ **createDistanceCubeMapRun** ()

  Creates a **ColorCubeMapRenderingRun**(p. 21).

- void **distanceCubeMapRunUpdated** (**RenderingRun** ∗run)

  Called if the changed run is a **ColorCubeMapRenderingRun**(p. 21).

Protected Attributes

- unsigned char **texID**

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

5.22.1 Detailed Description

DistanceCubeMapRenderTechnique(p. 39) used in an Ogre environment.

5.22.2 Constructor & Destructor Documentation

5.22.2.1 OgreDistanceCubeMapRenderTechnique::OgreDistanceCubeMapRenderTechnique (unsigned long *startFrame*, unsigned long *cubeMapUpdateInterval*, unsigned int *cubeMapResolution*, unsigned char *texID*, bool *useDistCalc*, bool *useFaceAngleCalc*, float *distTolerance*, float *angleTolerance*, bool *updateAllFace*, 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

pass the pass to operate on

parentRenderable the object to operate on

parentTechniqueGroup the **TechniqueGroup**(p. 141) this RenderedTechnique is attached to

5.22.3 Member Function Documentation

5.22.3.1 RenderingRun * OgreDistanceCubeMapRenderTechnique::createDistanceCubeMapRun () [protected, virtual]

Creates a **ColorCubeMapRenderingRun**(p. 21).

Returns:

the new **ColorCubeMapRenderingRun**(p. 21) instance.

Implements **DistanceCubeMapRenderTechnique** (p. 41).

5.22.3.2 void OgreDistanceCubeMapRenderTechnique::distanceCubeMapRunChanged (RenderingRun * *run*) [protected, virtual]

Called if the changed run is a **ColorCubeMapRenderingRun**(p. 21).

Parameters:

run pointer to the changed **ColorCubeMapRenderingRun**(p. 21)

Implements **DistanceCubeMapRenderTechnique** (p. 41).

5.22.3.3 void OgreDistanceCubeMapRenderTechnique::distanceCubeMapRunUpdated (RenderingRun * *run*) [protected, virtual]

Called if the changed run is a **ColorCubeMapRenderingRun**(p. 21).

Parameters:

run pointer to the changed **ColorCubeMapRenderingRun**(p. 21)

Implements **DistanceCubeMapRenderTechnique** (p. 41).

5.22.3.4 void OgreDistanceCubeMapRenderTechnique::update (unsigned long *frameNum*) [virtual]

Updates the resources in the given frame.

Parameters:

frameNum the actual framenumber

Reimplemented from **DistanceCubeMapRenderTechnique** (p. 42).

The documentation for this class was generated from the following files:

- OgreDistanceCubeMapRenderTechnique.h
- OgreDistanceCubeMapRenderTechnique.cpp

5.23 OgreIlluminationManager Class Reference

Implementation of **IlluminationManager**(p. 45) in an OGRE environment.

Public Member Functions

- void **addRenderTechniqueFactory** (**RenderTechniqueFactory** ∗factory)
  registers a **RenderTechniqueFactory**(p. 130)
- float **getMaxJoinRadius** ()
  retirieves the maximum bounding sphere radius with two **SharedRuns**(p. 135) *can be joined.*
- void **setMaxJoinRadius** (float rad)
  sets the maximum bounding sphere radius with two **SharedRuns**(p. 135) *can be joined.*
- 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.
- 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**(p. 135)*) that are caustic casters from a given point.*

- void **createGlobalRun** (RenderingRunType runType)

  Creates a global **RenderingRun**(p. 124) *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**(p. 124) *with the given type.*

- void **createPerLightRun** (String lightName, RenderingRunType runType)

  Creates a **RenderingRun**(p. 124) *attached to a lightsource with the given type.*

- **RenderingRun** ∗ **getPerLightRun** (String lightName, RenderingRunType runType)

  Retuns a **RenderingRun**(p. 124) *attached to a lightsource with the given type.*

- void **updatePerLightRun** (String lightName, RenderingRunType runType, unsigned long frameNum)

  Updates a **RenderingRun**(p. 124) *attached to a lightsource with the given type.*

Static Public Member Functions

- static **OgreIlluminationManager** & **getSingleton** ()

  Returns the one and only **OgreIlluminationManager**(p. 78) *instance.*

Protected Member Functions

- **OgreIlluminationManager** ()

  *Protected constructor (***OgreIlluminationManager**(p. 78) *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**(p. 127) *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).

Protected Attributes

- std::list< **RenderTechniqueFactory** * > **techniqueFactories**
  registered RenderTechniqueFactories
- float **maxRad**
  The maximum bounding sphere radius that groupped objects (see **SharedRuns**(p. 135) *class) can have.*
- 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**(p. 135) *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.

Static Protected Attributes

- static **OgreIlluminationManager** * **instance** = NULL
  The one and only **OgreIlluminationManager**(p. 78) *instance.*

5.23.1 Detailed Description

Implementation of **IlluminationManager**(p. 45) in an OGRE environment.

5.23.2 Member Function Documentation

5.23.2.1 void OgreIlluminationManager::addSharedRuns (SharedRuns * *runs*)

Register a shared run object.

Only called when new techniques are created.

Parameters:
runs pointer to the **SharedRuns**(p. 135) instance to add

5.23.2.2 void OgreIlluminationManager::createGlobalRun (RenderingRunType *runType*)

Creates a global **RenderingRun**(p. 124) of the given type.

If a **RenderingRun**(p. 124) with the given type already exist there is nothing to do.

Parameters:

runType type enum of the **RenderingRun**(p. 124) to create

5.23.2.3 void OgreIlluminationManager::createPerLightRun (String *lightName*, RenderingRunType *runType*)

Creates a **RenderingRun**(p. 124) attached to a lightsource with the given type.

Parameters:

lightName name of the lightsource

runType type enum of the **RenderingRun**(p. 124) to create

5.23.2.4 void OgreIlluminationManager::createTechnique (IllumTechniqueParams * *params*, Pass * *pass*, OgreRenderable * *rend*, OgreSharedRuns * *sRuns*) [protected]

creates a specific type of **RenderTechnique**(p. 127) for a Renderable's pass.

It searches all registered RenderTechniqueFactories.

5.23.2.5 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

5.23.2.6 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

5.23.2.7 RenderingRun * OgreIlluminationManager::getGlobalRun (RenderingRunType *runType*)

Returns the global RendderingRun with the given type.

Parameters:
runType type enum of the **RenderingRun**(p. 124) to retrieve

Returns:
pointer to the **RenderingRun**(p. 124), NULL if no **RenderingRun**(p. 124) with the given type exists

5.23.2.8 Camera* OgreIlluminationManager::getMainCamera () [inline]

Returns a pointer to the player camera.

Returns:
pointer to the main player camera. Needed by **RenderTechnique**(p. 127) and **RenderingRun**(p. 124) classes.

5.23.2.9 Viewport* OgreIlluminationManager::getMainViewport () [inline]

Returns a pointer to the viewport attached to the player camera.

Returns:
pointer to the viewport. Needed by **RenderTechnique**(p. 127) and **RenderingRun**(p. 124) classes.

5.23.2.10 void OgreIlluminationManager::getNearestCausticCasters (Vector3 *position*, std::vector< OgreSharedRuns * > * *nearestcasters*, unsigned int *maxCount*)

Searches for the nearest object groups (**SharedRuns**(p. 135)) 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**(p. 135) to
maxCount the maximum number of nearest casters to search for

5.23.2.11 RenderingRun * OgreIlluminationManager::getPerLightRun (String *lightName*, RenderingRunType *runType*)

Retuns a **RenderingRun**(p. 124) attached to a lightsource with the given type.

Parameters:
lightName name of the lightsource
runType type enum of the **RenderingRun**(p. 124) to return

Returns:
pointer to the **RenderingRun**(p. 124), NULL if no **RenderingRun**(p. 124) with the given type exists

5.23.2.12 void OgreIlluminationManager::joinSharedRuns ()

Joins shared runs if needed.

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

5.23.2.13 void OgreIlluminationManager::setMainCamera (Camera * *camera*) [inline]

Sets the player camera.

Parameters:

camera pointer to the main player camera

5.23.2.14 void OgreIlluminationManager::setMainViewport (Viewport * *viewport*) [inline]

Sets the viewport attached to the player camera.

Parameters:

viewport pointer to the viewport

5.23.2.15 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**(p. 135) instance that are joined

old2 pointer to the other **SharedRuns**(p. 135) instance that are joined

newsr pointer to the resulting parent **SharedRuns**(p. 135) instance

5.23.2.16 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**(p. 135) instance that is split

new1 pointer to one of the **SharedRuns**(p. 135) instance that remain after split

new2 pointer to the other **SharedRuns**(p. 135) instance that remain after split

5.23.2.17 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.

5.23.2.18 void OgreIlluminationManager::updateGlobalRun (RenderingRunType *runType*, unsigned long *frameNum*)

Updates a global **RenderingRun**(p. 124) with the given type.

Parameters:

runType type enum of the **RenderingRun**(p. 124) to update

frameNum current framenumber

5.23.2.19 void OgreIlluminationManager::updatePerLightRun (String *lightName*, RenderingRunType *runType*, unsigned long *frameNum*)

Updates a **RenderingRun**(p. 124) attached to a lightsource with the given type.

Parameters:

lightName name of the lightsource

runType type enum of the **RenderingRun**(p. 124) to update

frameNum current framenumber

5.23.3 Member Data Documentation

5.23.3.1 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.

5.23.3.2 float OgreIlluminationManager::maxRad [protected]

The maximum bounding sphere radius that groupped objects (see **SharedRuns**(p. 135) class) can have.

See also:

canJoin
joinRuns

5.23.3.3 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.

5.23.3.4 std::list<SharedRuns*> OgreIlluminationManager::sharedRunRoots [protected]

List containing **SharedRuns**(p. 135) roots.

It is the IlluminationManager's task to find the **SharedRuns**(p. 135) which can be joined. Only the root **SharedRuns**(p. 135) needs to be checked.

5.23.3.5 class VisibleFinderVisitor* OgreIlluminationManager::visitor [protected]

VisibleFinderVisitor instance.

Used for adding visible renderables with valid TechniqueGroups to the visibleObjects vector.

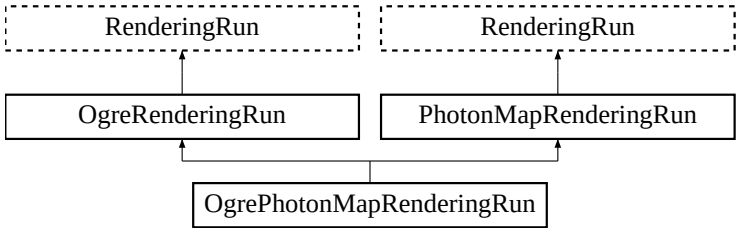
The documentation for this class was generated from the following files:

- OgreIlluminationManager.h
- OgreIlluminationManager.cpp

5.24 OgrePhotonMapRenderingRun Class Reference

ColorCubeMapRenderingRun(p. 21) 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

- void **refreshLight** ()

  Refreshes light camera matrices, called in each update.

Protected Member Functions

- void **updateFrame** (unsigned long frameNum)

  This function does the actual update in a frame.

- void **createPhotonMap** ()

  Creates a photon hit map.

Protected Attributes

- String **materialName**

  the name of the material should be used when rendering the choton hit map

- Light ∗ **light**

  pointer to the nearest light source from the caster object

- Camera ∗ **photonMapCamera**

  the created photon hit map texture

- bool **useDistance**

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

- **OgreSharedRuns** ∗ **sharedRuns**

  a pointer to the **OgreSharedRuns**(p. 108) *this run belongs to*

- String **name**

  the name of the photonmap texture that was created by this run

- Texture ∗ **photonMapTexture**

  a pointer to the photonmap texture that was created by this run

- unsigned int **resolution**

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

5.24.1 Detailed Description

ColorCubeMapRenderingRun(p. 21) used in an OGRE environment.

5.24.2 Constructor & Destructor Documentation

5.24.2.1 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**(p. 108) 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 color cubemap resolution

materialName the name of the material should be used when rendering the choton hit map

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

5.24.3 Member Function Documentation

5.24.3.1 void OgrePhotonMapRenderingRun::refreshLight ()

Refreshes light camera matrices, called in each update.

TODO: search nearest light, set light params

5.24.3.2 void OgrePhotonMapRenderingRun::updateFrame (unsigned long *frameNum*) [protected, virtual]

This function does the actual update in a frame.

Parameters:

frameNum current frame number

Implements **PhotonMapRenderingRun** (p. 119).

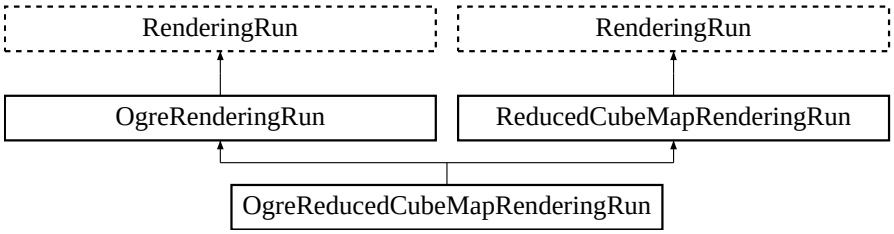
The documentation for this class was generated from the following files:

- OgrePhotonMapRenderingRun.h
- OgrePhotonMapRenderingRun.cpp

5.25 OgreReducedCubeMapRenderingRun Class Reference

ColorCubeMapRenderingRun(p. 21) 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 reduced sized color cubemap texture

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.

Protected Attributes

- **OgreSharedRuns** ∗ **sharedRuns**

  a pointer to the **OgreSharedRuns**(p. 108) *this run belongs to*

- String **name**

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

- Texture ∗ **reducedCubemapTexture**

  a pointer to the cubemap texture that was created by this run

5.25.1 Detailed Description

ColorCubeMapRenderingRun(p. 21) used in an OGRE environment.

5.25.2 Constructor & Destructor Documentation

5.25.2.1 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**(p. 108) 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

5.25.3 Member Function Documentation

5.25.3.1 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** (p. 121).

5.25.3.2 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** (p. 122).

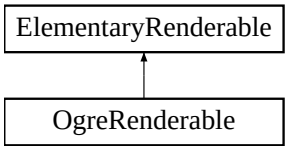
The documentation for this class was generated from the following files:

- OgreReducedCubeMapRenderingRun.h
- OgreReducedCubeMapRenderingRun.cpp

5.26 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)
  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.
- 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)

- 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)

5.26.1 Detailed Description

Class to wrap different Ogre Renderable types.

5.26.2 Constructor & Destructor Documentation

5.26.2.1 OgreRenderable::OgreRenderable (SubEntity * *sube*, Entity * *parentEntity*)

Constructor.

Creates an **OgreRenderable**(p. 92) from a SubEntity of an Entity.

Parameters:

sube the subentity to wrap

parentEntity the parent of the wrapped subentity

5.26.2.2 OgreRenderable::OgreRenderable (Entity * *parentEntity*, int *subEntityNum*)

Constructor.

Creates an **OgreRenderable**(p. 92) from a SubEntity of an Entity.

Parameters:

parentEntity the parent of the wrapped subentity

subEntityNum the number of the subentity to wrap

5.26.2.3 OgreRenderable::OgreRenderable (BillboardSet * *billboardset*)

Constructor.

Creates an **OgreRenderable**(p. 92) from a SubEntity of a BillboardSet.

Parameters:

billboardset the BillboardSet to wrap

5.26.3 Member Function Documentation

5.26.3.1 AxisAlignedBox& OgreRenderable::getBoundingBox () [inline]

Retrieves the axis-aligned bouding box of the renderable.

Returns:

reference to the bouding box

5.26.3.2 Sphere& OgreRenderable::getBoundingSphere () [inline]

Retrieves the bouding sphere of the renderable.

Returns:

reference to the bouding sphere

5.26.3.3 Material* OgreRenderable::getMaterial () [inline]

Retrieves a pointer to the material used by the renderable.

Returns:
reference to the Material pointer

5.26.3.4 const String& OgreRenderable::getMaterialName () [inline]

Retrieves the name of the material used by the renderable.

Returns:
reference to the name of the material

5.26.3.5 const MaterialPtr & OgreRenderable::getMaterialPtr ()

Retrieves a resource pointer to the material used by the renderable.

Returns:
reference to the resource pointer

5.26.3.6 String& OgreRenderable::getName () [inline]

Retrieves the unique name assigned to the renderable.

Returns:
reference to name of the renderable

5.26.3.7 Renderable * OgreRenderable::getRenderable ()

Returns the wrapped Renderable.

Returns:
pointer to the wrapped Renderable

5.26.3.8 void OgreRenderable::notifyCamera (Camera * *cam*)

Calls notifyCamera for the wrapped Renderable.

Parameters:
pointer to the Camera to pass to **notifyCamera()**(p. 95)

5.26.3.9 void OgreRenderable::setMaterialName (String & *name*)

Sets the material to be used by the renderable.

Parameters:
name the name of the material to use

5.26.3.10 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()(p. 43)

Implements **ElementaryRenderable** (p. 43).

5.26.3.11 void OgreRenderable::setVisible (bool *visible*) [virtual]

Sets the visibility of the wrapped renderable.

Parameters:
visible visibility

See also:
ElementaryRenderable::setVisible()(p. 43)

Implements **ElementaryRenderable** (p. 43).

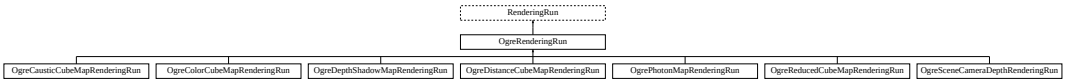
The documentation for this class was generated from the following files:

- OgreRenderable.h
- OgreRenderable.cpp

5.27 OgreRenderingRun Class Reference

Base class of a **RenderingRun**(p. 124) in an OGRE environment.

Inheritance diagram for OgreRenderingRun::



Public Member Functions

- **OgreRenderingRun** (unsigned long **startFrame**, unsigned long **updateInterval**)

  Constructor.

- **OgreRenderingRun** ∗ **asOgreRenderingRun** ()

  Conversion to OgreRenderRun.

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.

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

Static Protected Attributes

- static MovablePlane * **fullScreenQuad** = NULL

  fulls screen quad plane used in full screen quad rendering

- static Entity * **fullScreenQuadEntity** = NULL

  fulls screen quad Entity used in full screen quad rendering

- static SceneNode * **fullScreenQuadNode** = NULL

  fulls screen quad SceneNode used in full screen quad rendering

5.27.1 Detailed Description

Base class of a **RenderingRun**(p. 124) in an OGRE environment.

5.27.2 Constructor & Destructor Documentation

5.27.2.1 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

5.27.3 Member Function Documentation

5.27.3.1 OgreRenderingRun* OgreRenderingRun::asOgreRenderingRun () [inline, virtual]

Conversion to OgreRenderRun.

This function is needed because of virtual inheritance.

Reimplemented from **RenderingRun** (p. 125).

5.27.3.2 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

5.27.3.3 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

5.27.3.4 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

5.27.3.5 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

5.27.3.6 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.

5.27.3.7 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

5.27.3.8 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

5.27.4 Member Data Documentation

5.27.4.1 MovablePlane * OgreRenderingRun::fullScreenQuad = NULL [static, protected]

fulls screen quad plane used in full screen quad rendering

See also:

renderFullscreenQuad(p. 99)

5.27.4.2 Entity * OgreRenderingRun::fullScreenQuadEntity = NULL [static, protected]

fulls screen quad Entity used in full screen quad rendering

See also:
renderFullscreenQuad(p. 99)

5.27.4.3 SceneNode * OgreRenderingRun::fullScreenQuadNode = NULL [static, protected]

fulls screen quad SceneNode used in full screen quad rendering

See also:
renderFullscreenQuad(p. 99)

5.27.4.4 SpriteSet* OgreRenderingRun::pixelSprites [protected]

SpriteSet used in pixel sprite rendering.

See also:
renderPixelSprites(p. 99)

5.27.4.5 String OgreRenderingRun::spriteSetName [protected]

unique name of the SpriteSet used in pixel sprite rendering

See also:
renderPixelSprites(p. 99)

5.27.4.6 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(p. 100)
setMaterialForRenderables(p. 100)
restoreMaterials(p. 100)

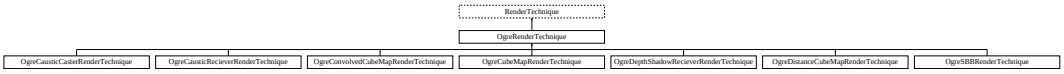
The documentation for this class was generated from the following files:

- OgreRenderingRun.h
- OgreRenderingRun.cpp

5.28 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**(p. 102).

Protected Attributes

- **OgreRenderable** ∗ **parentOgreRenderable**

  a **OgreRenderable**(p. 92) *pointer to the renderable this technique operates on.*

- **OgreTechniqueGroup** ∗ **parentOgreTechniqueGroup**

  a **OgreTechniqueGroup**(p. 116) *pointer to the* **TechniqueGroup**(p. 141) *this technique is attached to.*

- Pass ∗ **pass**

  a pointer to the pass this technique operates on.

5.28.1 Detailed Description

Class of RenderTechniques used in an OGRE environment.

5.28.2 Constructor & Destructor Documentation

5.28.2.1 OgreRenderTechnique::OgreRenderTechnique (Pass ∗ *pass*, OgreRenderable ∗ *parentRenderable*, OgreTechniqueGroup ∗ *parentTechniqueGroup*)

Constructor.

Parameters:

the pass to operate on

parentRenderable the object to operate on

parentTechniqueGroup the **TechniqueGroup**(p. 141) this RenderedTechnique is attached to

5.28.3 Member Function Documentation

5.28.3.1 virtual OgreRenderTechnique* OgreRenderTechnique::asOgreRenderTechnique ()

`[inline, virtual]`

Conversion to **OgreRenderTechnique**(p. 102).

This function is needed because of virtual inheritance.

Reimplemented from **RenderTechnique** (p. 128).

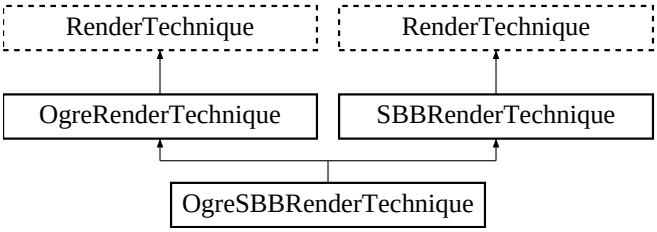
The documentation for this class was generated from the following files:

- OgreRenderTechnique.h
- OgreRenderTechnique.cpp

5.29 OgreSBBRenderTechnique Class Reference

SBBRenderTechnique(p. 132) 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.

Protected Attributes

- unsigned char **depthTexID**

5.29.1 Detailed Description

SBBRenderTechnique(p. 132) used in an OGRE environment.

5.29.2 Constructor & Destructor Documentation

5.29.2.1 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**(p. 141) this RenderedTechnique is attached to

5.29.3 Member Function Documentation

5.29.3.1 void OgreSBBRenderTechnique::update (unsigned long *frameNum*) [virtual]

Updates the resources in the given frame.

A **RenderTechnique**(p. 127) is usually need some resources from several runs, so these runs will be updated.

Parameters:
frameNum the actual framenumber

Reimplemented from **RenderTechnique** (p. 129).

5.29.4 Member Data Documentation

5.29.4.1 unsigned char OgreSBBRenderTechnique::depthTexID [protected]

&brief the id of the texture unit state the resulting scene depth map should be bound to

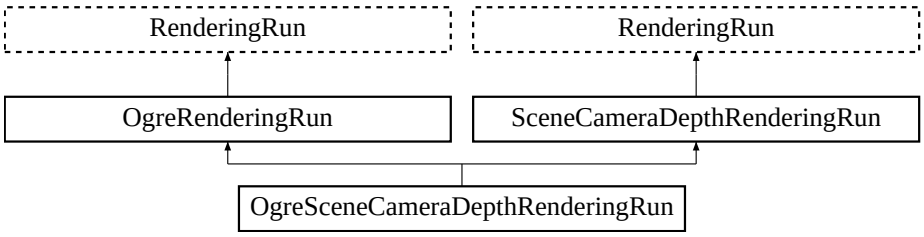
The documentation for this class was generated from the following files:

- OgreSBBRenderTechnique.h
- OgreSBBRenderTechnique.cpp

5.30 OgreSceneCameraDepthRenderingRun Class Reference

ColorCubeMapRenderingRun(p. 21) 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

Protected Member Functions

- void **updateFrame** (unsigned long frameNum)

  This function does the actual update in a frame.

- void **createDepthMap** ()

  Creates the depth map texture.

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**(p. 108) *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

5.30.1 Detailed Description

ColorCubeMapRenderingRun(p. 21) used in an OGRE environment.

5.30.2 Constructor & Destructor Documentation

5.30.2.1 OgreSceneCameraDepthRenderingRun::OgreSceneCameraDepthRenderingRun (OgreSharedRuns * *sharedRuns*, String *name*, Viewport * *playerView*)

Constructor.

Parameters:

sharedRuns a pointer to the **OgreSharedRuns**(p. 108) this run belongs to

name the name of the scene depth texture to be created

playerView pointer to the player's viewport

5.30.3 Member Function Documentation

5.30.3.1 void OgreSceneCameraDepthRenderingRun::updateFrame (unsigned long *frameNum*) [protected, virtual]

This function does the actual update in a frame.

Parameters:

frameNum current frame number

Implements **SceneCameraDepthRenderingRun** (p. 133).

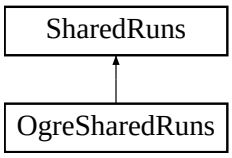
The documentation for this class was generated from the following files:

- OgreSceneCameraDepthRenderingRun.h
- OgreSceneCameraDepthRenderingRun.cpp

5.31 OgreSharedRuns Class Reference

Class of **SharedRuns**(p. 135) 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**(p. 135) *node, which have a specified* **RenderingRun**(p. 124) *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**(p. 135) *node, which have a specified* **RenderingRun**(p. 124) *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**(p. 135) *node, which have a specified* **RenderingRun**(p. 124) *type.*

- bool **hasOwnRun** (RenderingRunType runType)

  Checks if this node has a resource with the given type.

- void **addRenderablesToQueue** (RenderQueue *rq)

  Adds all the Renderables connected to this node to a given RenderQueue.

- void **notifyCamera** (Camera *cam)

  Calls **notifyCamera**()(p. 114) *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**(p. 135) *(and also it's parent).*

- void **validate** ()

  Validate this **SharedRuns**(p. 135) *(and also all childs).*

- void **destroy** ()

  Destroys the node (and all parents recursively).

Static Public Member Functions

- static bool **canJoin** (**SharedRuns** *r1, **SharedRuns** *r2)

  Checks if two **SharedRuns**(p. 135) *node can be joined.*

- static bool **haveCommonRuns** (**SharedRuns** *r1, **SharedRuns** *r2)

  Checks if two **SharedRuns**(p. 135) *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**(p. 124) *type, used after join and split.*

- **SharedRuns** * **createInstance** ()

  Creates a new **SharedRuns**(p. 135) *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::map< RenderingRunType, **RenderingRun** * > **sharedRuns**
  map of contained RenderingRuns
- std::map< **OgreRenderable** *, bool > **renderables**
  map of connected renderablis with visibility 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.

5.31.1 Detailed Description

Class of **SharedRuns**(p. 135) used in an OGRE environment.

5.31.2 Member Function Documentation

5.31.2.1 void OgreSharedRuns::addRenderablesToQueue (RenderQueue * *rq*)

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

5.31.2.2 void OgreSharedRuns::addRun (RenderingRunType *runType*, RenderingRun * *run*) [virtual]

&brief Adds a **RenderingRun**(p. 124) instance to the shared resources.

Parameters:
runType enum, type of the **RenderingRun**(p. 124) to add
run pointer to the **RenderingRun**(p. 124) instance to add

Implements **SharedRuns** (p. 137).

5.31.2.3 bool OgreSharedRuns::canJoin (SharedRuns * *r1*, SharedRuns * *r2*) [static]

Checks if two **SharedRuns**(p. 135) node can be joined.

Parameters:

r1 pointer to one of the **SharedRuns**(p. 135) instance

r2 pointer to the other **SharedRuns**(p. 135) instance

5.31.2.4 SharedRuns * OgreSharedRuns::createInstance () [protected, virtual]

Creates a new **SharedRuns**(p. 135) instance. All derivatives should implement this.

Returns:

a new **SharedRuns**(p. 135) instance

Implements **SharedRuns** (p. 137).

5.31.2.5 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**(p. 124) to look for

roots reference to the collection to add the new groups to

5.31.2.6 AxisAlignedBox& OgreSharedRuns::getBoundingBox () [inline]

Returns the axis-aligned bounding box of the node.

Returns:

a reference to the bounding box

5.31.2.7 Sphere& OgreSharedRuns::getBoundingSphere () [inline]

Returns the bounding sphere of the node.

Returns:

a reference to the bounding sphere

5.31.2.8 AxisAlignedBox& OgreSharedRuns::getRootBoundingBox (RenderingRunType *runType*) [inline]

Retrieves the axis-aligned bounding box of the topmost parent node of this **SharedRuns**(p. 135) node, which have a specified **RenderingRun**(p. 124) type.

Parameters:
runType the **RenderingRun**(p. 124) type

Returns:
a reference to the bounding box

5.31.2.9 AxisAlignedBox& OgreSharedRuns::getRootBoundingBox () [inline]

Returns the axis-aligned bounding box of the root parent node.

Returns:
a reference to the bounding box

5.31.2.10 Sphere& OgreSharedRuns::getRootBoundingSphere (RenderingRunType *runType*) [inline]

Retrieves the bounding sphere of the topmost parent node of this **SharedRuns**(p. 135) node, which have a specified **RenderingRun**(p. 124) type.

Parameters:
runType the **RenderingRun**(p. 124) type

Returns:
a reference to the bounding sphere

5.31.2.11 Sphere& OgreSharedRuns::getRootBoundingSphere () [inline]

Returns the bounding sphere of the root parent node.

Returns:
a reference to the bounding sphere

5.31.2.12 const Vector3& OgreSharedRuns::getRootPosition (RenderingRunType *runType*) [inline]

Retrieves the world space center position of the topmost parent node of this **SharedRuns**(p. 135) node, which have a specified **RenderingRun**(p. 124) type.

Parameters:
runType the **RenderingRun**(p. 124) type

Returns:
a reference to the center position

5.31.2.13 const Vector3& OgreSharedRuns::getRootPosition () [inline]

Returns the world space center position of the root parent node.

Returns:
a reference to the center position

5.31.2.14 RenderingRun * OgreSharedRuns::getRun (RenderingRunType *runType*) [virtual]

&brief Retrieves a shared resource.

Parameters:
runType enum, type of the **RenderingRun**(p. 124) to be retrieved

Returns:
pointer to the **RenderingRun**(p. 124) of type "runType", null if this type does not exists

Implements **SharedRuns** (p. 138).

5.31.2.15 std::map<RenderingRunType, RenderingRun*>& OgreSharedRuns::getSharedRuns () [inline]

Retrieves the contaied RenderingRuns with their type information.

Returns:
map of renderables

5.31.2.16 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**(p. 124) to look for

Implements **SharedRuns** (p. 135).

5.31.2.17 bool OgreSharedRuns::haveCommonRuns (SharedRuns * *r1*, SharedRuns * *r2*) [static]

Checks if two **SharedRuns**(p. 135) have common resources so that they can be joined.

Parameters:
r1 pointer to one of the **SharedRuns**(p. 135) instance

r2 pointer to the other **SharedRuns**(p. 135) instance

5.31.2.18 void OgreSharedRuns::notifyCamera (Camera * *cam*)

Calls **notifyCamera**()(p. 114) 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**()(p. 114) with

5.31.2.19 void OgreSharedRuns::setBoundingBox (AxisAlignedBox & *box*) [inline]

Sets the axis-aligned bounding box of the node.

Parameters:
box bounding box

5.31.2.20 void OgreSharedRuns::setBoundingSphere (Sphere & *sphere*) [inline]

Sets the bounding sphere of the node.

Parameters:
sphere bounding sphere

5.31.2.21 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** (p. 139).

5.31.2.22 void OgreSharedRuns::updateRun (RenderingRunType *runType*, unsigned long *frameNum*) [virtual]

&brief Updates a shared **RenderingRun**(p. 124).

Parameters:
runType enum, type of the **RenderingRun**(p. 124) to update

frameNum current framenumber

Implements **SharedRuns** (p. 139).

5.31.2.23 void OgreSharedRuns::validate () [virtual]

Validate this **SharedRuns**(p. 135) (and also all childs).

Validation meens that all the **SharedRuns**(p. 135) that are joined will be examined if the sharing is still valid. If it finds out that two **SharedRuns**(p. 135) 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** (p. 140).

5.31.3 Member Data Documentation

5.31.3.1 std::map<OgreRenderable*, bool> OgreSharedRuns::renderables [protected]

map of connected renderablis with visibility information

Used to show or hide the renderables connected to a leaf **OgreSharedRuns**(p. 108) node.

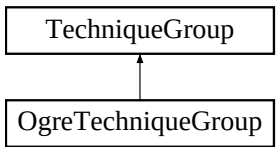
The documentation for this class was generated from the following files:

- OgreSharedRuns.h
- OgreSharedRuns.cpp

5.32 OgreTechniqueGroup Class Reference

Base class of a **SharedRuns**(p. 135) 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 he shared runs changes.
- void **runUpdated** (RenderingRunType runType, **RenderingRun** ∗run)
  Called after one of he shared runs updates.
- void **updateBounds** ()
  Updates the connected **SharedRuns**(p. 135) *boundary.*

Protected Attributes

- std::vector< **OgreRenderTechnique** ∗ > **renderTechniques**
  Collection of OgreRenderTechniques.

5.32.1 Detailed Description

Base class of a **SharedRuns**(p. 135) in an OGRE environment.

5.32.2 Member Function Documentation

5.32.2.1 void OgreTechniqueGroup::addRenderTechnique (RenderTechnique ∗ *technique*) [virtual]

Adds a rendertechnique to the group.

Parameters:
technique the **RenderTechnique**(p. 127) instance to add.

Implements **TechniqueGroup** (p. 142).

5.32.2.2 void OgreTechniqueGroup::runChanged (RenderingRunType *runType*, RenderingRun * *run*) [virtual]

Called after one of he shared 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**(p. 124)

Implements **TechniqueGroup** (p. 142).

5.32.2.3 void OgreTechniqueGroup::runUpdated (RenderingRunType *runType*, RenderingRun * *run*) [virtual]

Called after one of he shared 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**(p. 124)

Implements **TechniqueGroup** (p. 143).

5.32.2.4 void OgreTechniqueGroup::update (unsigned long *frameNum*) [virtual]

Updates all rendertechniques.

Parameters:

framenum current framenumber

Implements **TechniqueGroup** (p. 143).

5.32.3 Member Data Documentation

5.32.3.1 std::vector<OgreRenderTechnique*> OgreTechniqueGroup::renderTechniques [protected]

Collection of OgreRenderTechniques.

All messages will be forwarded to each element of this vector.

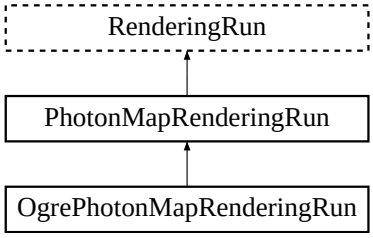
The documentation for this class was generated from the following files:

- OgreTechniqueGroup.h
- OgreTechniqueGroup.cpp

5.33 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.

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.

5.33.1 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.

5.33.2 Constructor & Destructor Documentation

5.33.2.1 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

5.33.3 Member Function Documentation

5.33.3.1 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** (p. 126).

Implemented in **OgrePhotonMapRenderingRun** (p. 88).

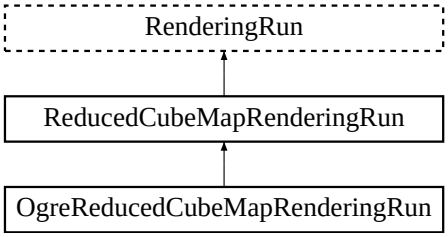
The documentation for this class was generated from the following file:

- PhotonMapRenderingRun.h

5.34 ReducedCubeMapRenderingRun Class Reference

Base abstract class that defines a rendering process of a reduced sized 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.

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.

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.

5.34.1 Detailed Description

Base abstract class that defines a rendering process of a reduced sized color-cubemap.

The resulting cubemap is a lower resolution variation of a 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.

5.34.2 Constructor & Destructor Documentation

5.34.2.1 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

5.34.3 Member Function Documentation

5.34.3.1 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** (p. 90).

5.34.3.2 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** (p. 90).

5.34.3.3 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** (p. 126).

5.34.4 Member Data Documentation

5.34.4.1 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.

5.34.4.2 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.

5.34.4.3 bool ReducedCubeMapRenderingRun::useDistCalc [protected]

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

See also:
distTolerance(p. 122)

5.34.4.4 bool ReducedCubeMapRenderingRun::useFaceAngleCalc [protected]

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

See also:
angleTolerance(p. 122)

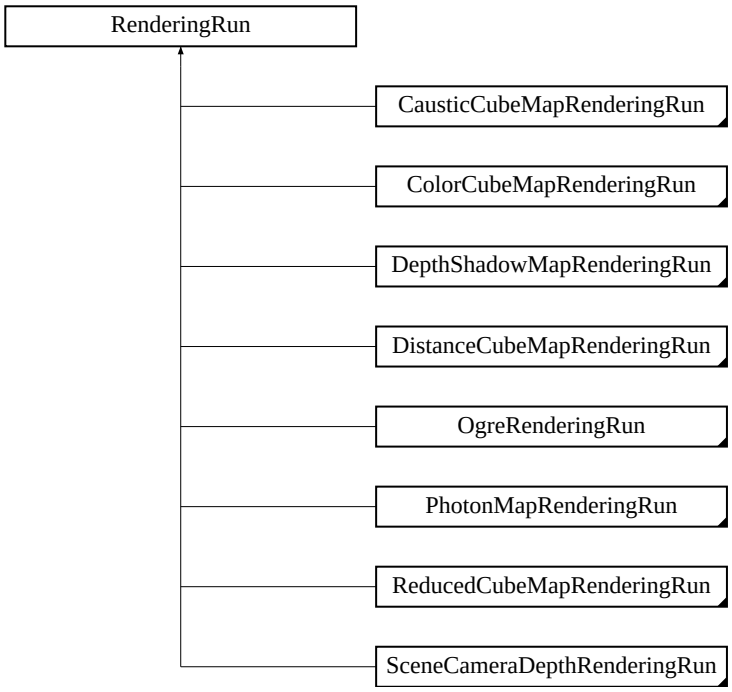
The documentation for this class was generated from the following files:

- ReducedCubeMapRenderingRun.h
- ReducedCubeMapRenderingRun.cpp

5.35 RenderingRun Class Reference

Base class for a computation module.

Inheritance diagram for RenderingRun::



Public Member Functions

- **RenderingRun** (unsigned long **startFrame**, unsigned long **updateInterval**)

  Constructor.

- void **update** (unsigned long frameNum)

  Calls **updateFrame**()(p. 126) *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.

Protected Member Functions

- 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.

5.35.1 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**(p. 127). 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.

5.35.2 Constructor & Destructor Documentation

5.35.2.1 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

5.35.3 Member Function Documentation

5.35.3.1 virtual class OgreRenderingRun* RenderingRun::asOgreRenderingRun () `[inline, virtual]`

Conversion to OgreRenderRun.

This function is needed because of virtual inheritance.

Reimplemented in **OgreRenderingRun** (p. 98).

5.35.3.2 bool RenderingRun::needUpdate (unsigned long *frameNum*) `[inline, protected]`

Returns if this run needs update.

Parameters:

frameNum current frame number

5.35.3.3 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** (p. 19), **ColorCubeMapRenderingRun** (p. 23), **DepthShadowMapRenderingRun** (p. 34), **DistanceCubeMapRenderingRun** (p. 38), **PhotonMapRenderingRun** (p. 119), **ReducedCubeMapRenderingRun** (p. 122), **SceneCameraDepthRenderingRun** (p. 133), **OgreDepthShadowMapRenderingRun** (p. 68), **OgrePhotonMapRenderingRun** (p. 88), and **OgreSceneCameraDepthRenderingRun** (p. 107).

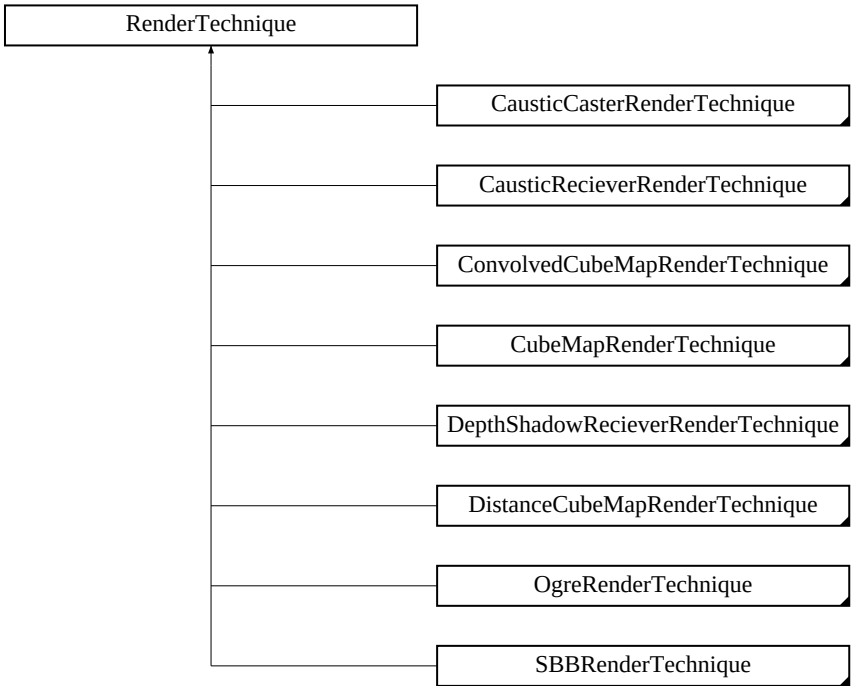
The documentation for this class was generated from the following files:

- RenderingRun.h
- RenderingRun.cpp

5.36 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**(p. 102).

- **ElementaryRenderable** ∗ **getParentRenderable** ()

  Retrieves the renderable this technique operates on.

Protected Attributes

- **ElementaryRenderable ∗ parentRenderable**
  The renderable this technique operates on.
- **TechniqueGroup ∗ parentTechniqueGroup**
  The **TechniqueGroup**(p. 141) *this RenderedTechnique is attached to.*
- **SharedRuns ∗ sharedRuns**
  The **SharedRuns**(p. 135) *this RenderedTechnique is attached to.*

5.36.1 Detailed Description

Base class for a rendering technique.

A **RenderTechnique**(p. 127) gives a description about how to render an object, and what kind of resources are needed to do this. A **RenderTechnique**(p. 127) 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**(p. 135) object. The Renderable defines the object to operate on. **SharedRuns**(p. 135) 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).

5.36.2 Constructor & Destructor Documentation

5.36.2.1 RenderTechnique::RenderTechnique (ElementaryRenderable ∗ *parentRenderable*, TechniqueGroup ∗ *parentTechniqueGroup*)

Constructor.

Parameters:

parentRenderable the object to operate on

parentTechniqueGroup the **TechniqueGroup**(p. 141) this RenderedTechnique is attached to

5.36.3 Member Function Documentation

5.36.3.1 virtual class OgreRenderTechnique∗ RenderTechnique::asOgreRenderTechnique () `[inline, virtual]`

Conversion to **OgreRenderTechnique**(p. 102).

This function is needed because of virtual inheritance.

Reimplemented in **OgreRenderTechnique** (p. 103).

5.36.3.2 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**(p. 124)

Reimplemented in **CausticCasterRenderTechnique** (p. 16), **ConvolvedCubeMapRenderTechnique** (p. 27), **CubeMapRenderTechnique** (p. 31), and **DistanceCubeMapRenderTechnique** (p. 41).

5.36.3.3 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**(p. 124)

Reimplemented in **DistanceCubeMapRenderTechnique** (p. 41).

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

Updates the resources in the given frame.

A **RenderTechnique**(p. 127) is usually need some resources from several runs, so these runs will be updated.

Parameters:

frameNum the actual framenumber

Reimplemented in **ConvolvedCubeMapRenderTechnique** (p. 27), **CubeMapRenderTechnique** (p. 31), **DistanceCubeMapRenderTechnique** (p. 42), **OgreCausticRecieverRenderTechnique** (p. 55), **OgreConvolvedCubeMapRenderTechnique** (p. 62), **OgreCubeMapRenderTechnique** (p. 65), **OgreDepthShadowRecieverRenderTechnique** (p. 70), **OgreDistanceCubeMapRenderTechnique** (p. 77), and **OgreSBBRenderTechnique** (p. 105).

The documentation for this class was generated from the following files:

- RenderTechnique.h
- RenderTechnique.cpp

5.37 RenderTechniqueFactory Class Reference

Base abstract class for creating **RenderTechnique**(p. 127) instances.

Public Member Functions

- bool **isType** (String type)

  Returns if this factory can create a **RenderTechnique**(p. 127) *of the given type.*

- virtual **OgreRenderTechnique** ∗ **createInstance** (IllumTechniqueParams ∗params, Pass ∗pass, **OgreRenderable** ∗parentRenderable, **OgreTechniqueGroup** ∗parentTechniqueGroup)=0

  Creates a **RenderTechnique**(p. 127) *of the factory type.*

Protected Types

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

  function for parsing **RenderTechnique**(p. 127) *attributes*

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

  Keyword-mapped attribute parsers.

Protected Attributes

- **AttribParserList attributeParsers**

  map of parser functions

- String **typeName**

  factoryname

5.37.1 Detailed Description

Base abstract class for creating **RenderTechnique**(p. 127) instances.

5.37.2 Member Typedef Documentation

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

function for parsing **RenderTechnique**(p. 127) attributes

Parameters:

params attribute value stored in a String

5.37.3 Member Function Documentation

5.37.3.1 virtual OgreRenderTechnique* RenderTechniqueFactory::createInstance (IllumTechniqueParams * *params*, Pass * *pass*, OgreRenderable * *parentRenderable*, OgreTechniqueGroup * *parentTechniqueGroup*) [pure virtual]

Creates a **RenderTechnique**(p. 127) of the factory type.

Parameters:

params containes constructor parameters as NameValuePairList

pass the Pass to use in **RenderTechnique**(p. 127) constructor

pass the parentRenderable to pass to **RenderTechnique**(p. 127) constructor

pass the parentTechniqueGroup to pass to **RenderTechnique**(p. 127) constructor

5.37.3.2 bool RenderTechniqueFactory::isType (String *type*) [inline]

Returns if this factory can create a **RenderTechnique**(p. 127) of the given type.

Parameters:

type **RenderTechnique**(p. 127) type

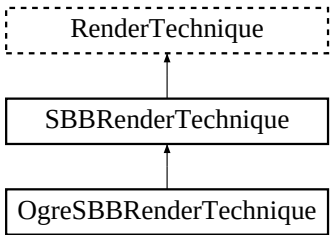
The documentation for this class was generated from the following files:

- OgreRenderTechnique.h
- OgreRenderTechnique.cpp

5.38 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.

5.38.1 Detailed Description

Base abstract class of rendering a particle system with the spherical billboard method.

5.38.2 Constructor & Destructor Documentation

5.38.2.1 SBBRenderTechnique::SBBRenderTechnique (ElementaryRenderable ∗ *parentRenderable*, TechniqueGroup ∗ *parentTechniqueGroup*)

Constructor.

Parameters:

parentRenderable the object to operate on

parentTechniqueGroup the **TechniqueGroup**(p. 141) this RenderedTechnique is attached to

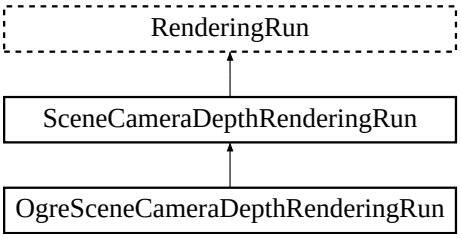
The documentation for this class was generated from the following files:

- SBBRenderTechnique.h
- SBBRenderTechnique.cpp

5.39 SceneCameraDepthRenderingRun Class Reference

Base abstract class that defines a rendering process that creates depth map.

Inheritance diagram for SceneCameraDepthRenderingRun::



Public Member Functions

- **SceneCameraDepthRenderingRun** ()

  Constructor.

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.

5.39.1 Detailed Description

Base abstract class that defines a rendering process that creates depth map.

The depth map stores the scene's camera space z coordinates (rendered from the player's view).

5.39.2 Member Function Documentation

5.39.2.1 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** (p. 126).

Implemented in **OgreSceneCameraDepthRenderingRun** (p. 107).

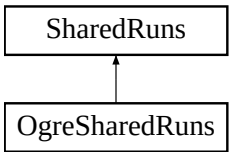
The documentation for this class was generated from the following file:

- SceneCameraDepthRenderingRun.h

5.40 SharedRuns Class Reference

Base abstract class for a collection of shared resources (RenderingRuns).

Inheritance diagram for SharedRuns::



Public Member Functions

- **SharedRuns** (void)
- 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.

- void **setTechniqueGroup** (**TechniqueGroup** ∗group)

  Sets child **TechniqueGroup**(p. 141)*.*

- virtual void **setVisible** (bool visible)

  Shows or hides this **SharedRuns**(p. 135) *(and also all childnodes).*

- virtual void **hide** ()

  Hides this **SharedRuns**(p. 135) *(and also all childs).*

- virtual void **restoreVisibility** ()

  Restores the visibility of this **SharedRuns**(p. 135) *(and also all childs).*

- virtual **SharedRuns** ∗ **getRoot** ()

  Retrieves the root node of this **SharedRuns**(p. 135) *node.*

- virtual **SharedRuns** ∗ **getRoot** (RenderingRunType runType)

  Retrieves the topmost parent node of this **SharedRuns**(p. 135) *node, which have a specified* **Rendering-Run**(p. 124) *type.*

- virtual void **updateBounds** ()=0

  Updates the boundary of this **SharedRuns**(p. 135) *(and also it's parent).*

- virtual void **validate** ()=0

  Validate this **SharedRuns**(p. 135) *(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**(p. 124) *type, used after join and split.*

- virtual **SharedRuns** ∗ **createInstance** ()=0

  Creates a new **SharedRuns**(p. 135) *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**(p. 135) *instance*

- **SharedRuns** ∗ **child1**

  child **SharedRuns**(p. 135) *instance*

- **SharedRuns** ∗ **child2**

  child **SharedRuns**(p. 135) *instance*

- **TechniqueGroup** ∗ **childTechniques**

  child **TechniqueGroup**(p. 141) *instance.*

5.40.1 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**(p. 135). These **SharedRuns**(p. 135) store the shared resources. They also act like nodes of a binary tree, as separate **SharedRuns**(p. 135) can also be joined if for example the objects for witch they store resources are close enough so even one shared resources is enough for the given objects.

5.40.2 Constructor & Destructor Documentation

5.40.2.1 SharedRuns::SharedRuns (void)

&brief Constructor.

5.40.3 Member Function Documentation

5.40.3.1 virtual void SharedRuns::addRun (RenderingRunType *runType*, RenderingRun * *run*) [pure virtual]

&brief Adds a **RenderingRun**(p. 124) instance to the shared resources.

Parameters:

runType enum, type of the **RenderingRun**(p. 124) to add

run pointer to the **RenderingRun**(p. 124) instance to add

Implemented in **OgreSharedRuns** (p. 110).

5.40.3.2 virtual SharedRuns* SharedRuns::createInstance () [protected, pure virtual]

Creates a new **SharedRuns**(p. 135) instance. All derivatives should implement this.

Returns:

a new **SharedRuns**(p. 135) instance

Implemented in **OgreSharedRuns** (p. 111).

5.40.3.3 SharedRuns * SharedRuns::getRoot (RenderingRunType *runType*) [virtual]

Retrieves the topmost parent node of this **SharedRuns**(p. 135) node, which have a specified **RenderingRun**(p. 124) type.

Parameters:

runType the **RenderingRun**(p. 124) type

Returns:

pointer to the parent **SharedRuns**(p. 135) instance

5.40.3.4 SharedRuns * SharedRuns::getRoot () [virtual]

Retrieves the root node of this **SharedRuns**(p. 135) node.

Returns:
pointer to the root **SharedRuns**(p. 135) instance

5.40.3.5 virtual RenderingRun* SharedRuns::getRun (RenderingRunType *runType*) [pure virtual]

&brief Retrieves a shared resource.

Parameters:
runType enum, type of the **RenderingRun**(p. 124) to be retrieved

Returns:
pointer to the **RenderingRun**(p. 124) of type "runType", null if this type does not exists

Implemented in **OgreSharedRuns** (p. 113).

5.40.3.6 void SharedRuns::hide () [virtual]

Hides this **SharedRuns**(p. 135) (and also all childs).

The previous visibility is saved.

5.40.3.7 SharedRuns * SharedRuns::joinRuns (SharedRuns * *otherRuns*) [virtual]

&brief Joines two **SharedRuns**(p. 135).

The resulting **SharedRuns**(p. 135) become the parent of the two **SharedRuns**(p. 135).

Parameters:
otherRuns pointer to the **SharedRuns**(p. 135) instance to join with

Returns:
the new parent **SharedRuns**(p. 135) instance

5.40.3.8 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**(p. 124)

5.40.3.9 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**(p. 124)

5.40.3.10 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** (p. 114).

5.40.3.11 void SharedRuns::setTechniqueGroup (TechniqueGroup * *group*) [inline]

Sets child **TechniqueGroup**(p. 141).

Parameters:

group pointer to the **TechniqueGroup**(p. 141) instance to add.

5.40.3.12 void SharedRuns::setVisible (bool *visible*) [virtual]

Shows or hides this **SharedRuns**(p. 135) (and also all childnodes).

Parameters:

visible visibility

5.40.3.13 virtual void SharedRuns::updateRun (RenderingRunType *runType*, unsigned long *frameNum*) [pure virtual]

&brief Updates a shared **RenderingRun**(p. 124).

Parameters:

runType enum, type of the **RenderingRun**(p. 124) to update

frameNum current framenumber

Implemented in **OgreSharedRuns** (p. 114).

5.40.3.14 virtual void SharedRuns::validate () [pure virtual]

Validate this **SharedRuns**(p. 135) (and also all childs).

Validation meens that all the **SharedRuns**(p. 135) that are joined will be examined if the sharing is still valid. If it finds out that two **SharedRuns**(p. 135) 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** (p. 115).

5.40.4 Member Data Documentation

5.40.4.1 TechniqueGroup* SharedRuns::childTechniques [protected]

child **TechniqueGroup**(p. 141) instance.

If this **SharedRuns**(p. 135) node is a leaf, it containes a reference to a **TechniqueGroup**(p. 141) instance. All messages will be transfered to this object, and bounding information will be retrieved from this **TechniqueGroup**(p. 141)

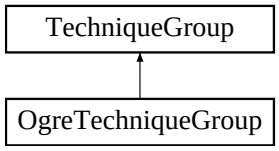
The documentation for this class was generated from the following files:

- SharedRuns.h
- SharedRuns.cpp

5.41 TechniqueGroup Class Reference

Base abstract class for a collection techniques.

Inheritance diagram for TechniqueGroup::



Public Member Functions

- **TechniqueGroup** (void)

  Constructor.

- void **addSharedRun** (**SharedRuns** ∗sharedRuns)

  Adds an empty **SharedRuns**(p. 135) *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 he shared runs changes.

- virtual void **runUpdated** (RenderingRunType runType, **RenderingRun** ∗run)=0

  Called after one of he shared runs updates.

- virtual void **updateBounds** ()

  Updates the connected **SharedRuns**(p. 135) *boundary.*

- virtual void **validateSharedRuns** ()

  Validates the connected **SharedRuns**(p. 135) *instance.*

Protected Attributes

- **SharedRuns** ∗ **parentSharedRuns**

  Pointer to the connected **SharedRuns**(p. 135) *instance each technique uses.*

5.41.1 Detailed Description

Base abstract class for a collection techniques.

This is a helper class, to collect **RenderTechnique**(p. 127) instances bound to a single renderable. It's main task is to recieve and forward messages to each **RenderTechnique**(p. 127).

5.41.2 Member Function Documentation

5.41.2.1 virtual void TechniqueGroup::addRenderTechnique (RenderTechnique * *technique*) [pure virtual]

Adds a rendertechnique to the group.

Parameters:
technique the **RenderTechnique**(p. 127) instance to add.

Implemented in **OgreTechniqueGroup** (p. 116).

5.41.2.2 void TechniqueGroup::addSharedRun (SharedRuns * *sharedRuns*) [inline]

Adds an empty **SharedRuns**(p. 135) parent.

Used after creating a new **TechniqueGroup**(p. 141).

Parameters:
sharedRuns the SharedRun instance the RenderTechniques will use.

5.41.2.3 SharedRuns* TechniqueGroup::getSharedRuns () [inline]

Retrieves the shared runs.

Returns:
the SharedRun instance the RenderTechniques use.

5.41.2.4 virtual void TechniqueGroup::runChanged (RenderingRunType *runType*, RenderingRun * *run*) [pure virtual]

Called after one of he shared 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**(p. 124)

Implemented in **OgreTechniqueGroup** (p. 117).

5.41.2.5 virtual void TechniqueGroup::runUpdated (RenderingRunType *runType*, RenderingRun * *run*) [pure virtual]

Called after one of he shared 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**(p. 124)

Implemented in **OgreTechniqueGroup** (p. 117).

5.41.2.6 virtual void TechniqueGroup::update (unsigned long *frameNum*) [pure virtual]

Updates all rendertechniques.

Parameters:

framenum current framenumber

Implemented in **OgreTechniqueGroup** (p. 117).

The documentation for this class was generated from the following files:

- TechniqueGroup.h
- TechniqueGroup.cpp