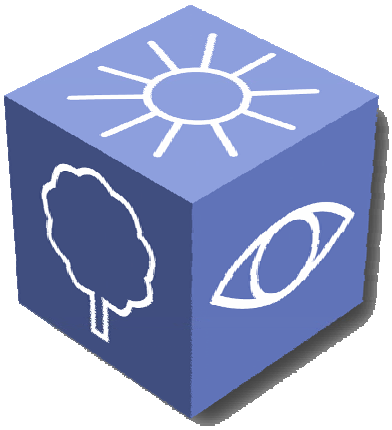


GAMETOOLS

ADVANCED TOOLS FOR DEVELOPING HIGHLY REALISTIC COMPUTER GAMES

DUMMY VISIBILITY MODULES

Document identifier:	**GameTools-3-D3.1-01-1-0-Dummy Visibility Modules**
Date: (use "update field" Word function, right mouse button)	**30/04/2005**
Work package:	**WP03: Visibility**
Partner(s):	**VUT**
Leading Partner:	**VUT**
Document status:	**APPROVED**
Deliverable identifier:	**D3.1**

Abstract: Dummy Modules Report For Visibility

Delivery Slip

	Name	Partner	Date	Signature
From	Michael Wimmer	VUT	29-04-05	
Reviewed by	Moderator and reviewers	All	02-05-05	
Approved by	Moderator and reviewers	All	02-05-05	

Document Log

Issue	Date	Comment	Author
1-0	29-04-05	First draft	Jiri Bittner (VUT)

Document Change Record

Issue	Item	Reason for Change

Files

Software Products	User files / URL
Word	gametools-ist-2-004363-3-d3.1-01-1-0-dummy visibility modules.doc (use “update field” Word function)

Contents

Chapter 1

Overview

The visibility workpackage aims to develop methods for making use of restricted visibility in large scenes. The major goal of these methods is to avoid wasting computational power on currently invisible parts of the scene. To address this goal, we develop new methods for online visibility culling and offline visibility preprocessing. The online visibility culling is either used to render only visible part of the scene or to provide queries which determine visible parts of the scene in runtime. Visibility preprocessing provides partitioning of the view space into view cells, and for each view cell it calculates a potentially visible set.

1.1 Module Class Hierarchy

The visibility workpackage consists of a module which is integrated into the game engine and an external module which deals with preprocessing. One of the goals of the design of the module class hierarchy has been easy portability of the developed methods to other game engines. The resulting class structure clearly separates the actual engine-dependent parts from the game engine independent algorithmic parts. We have used name spaces to separate the engine dependent parts (name space Ogre) from the engine independent parts (name space GtpVisibility). The external visibility preprocessing module defines its own name space (GtpVisibilityPreprocessor).

An overview of the important classes of the visibility work package and their integration into the Ogre engine is depicted in Figure 1.1.

The diagram shows that the module integrated into the engine consists of three main parts (CullingManager, QueryManager, PreprocessingManager), which are encapsulated in a helper class (Visibility Manager). The helper class then deals with the initialization of the contained parts and allows easy communication between them.

1.2 Visibility Culling

Visibility culling seamlessly integrates into the rendering loop and eliminates most invisible objects from being sent to the pipeline. Visibility culling is implemented by instances of the CullingManager class. We will provide several implementations which can be easily switched at runtime and so the best technique for the particular scene and scene representation can be selected. Note that the Ogre integration design allows to exploit all already existing methods for scene management, such as octree, BSP tree, or plain scene graph.

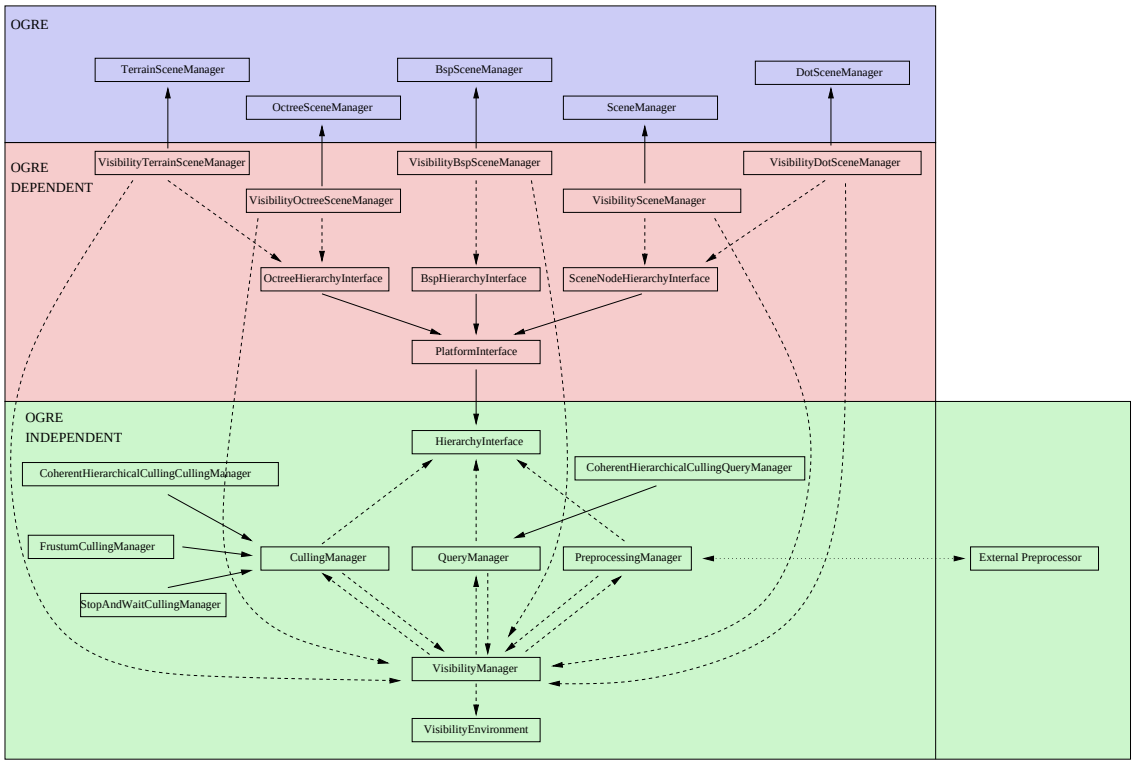


Figure 1.1: Overview of the visibility work packages classes and their integration into the Ogre game engine.

1.3 Visibility Queries

Visibility queries determine the visible geometry for a given view point in the scene. Additionally they can also report the visible scene hierarchy nodes. Visibility queries are implemented by instances of the QueryManager class. Note that some of these instances will make use of the preprocessed visibility data through the use of PreprocessingManager.

1.4 Visibility Preprocessing

Visibility preprocessing precalculates visibility for all viewpoints corresponding to cells of a view space partitioning. The visibility preprocessor is implemented as a standalone module which imports files generated by the PreprocessingManager. The results of the visibility computation will be exported to the file which can then be loaded by the PreprocessingManager and used inside the engine.

Apart from the scene definition file, the preprocessor will be able to import the view cell definition. In this case it will assume the view cells have to be described as meshes satisfying a set of requirements. Alternatively, the module can generate view cells by automatic view space partitioning.

Chapter 2

GameTools Visibility Modules Hierarchical Index

2.1 GameTools Visibility Modules Class Hierarchy

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

Chapter 3

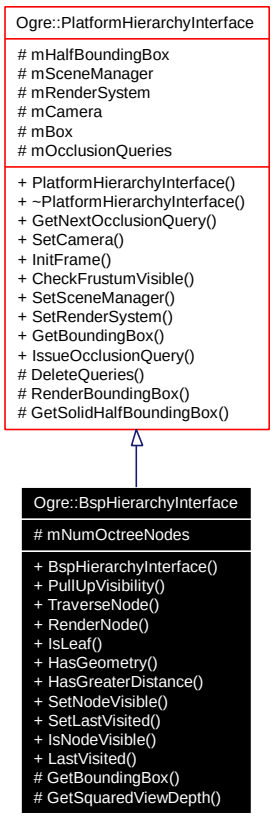
GameTools Visibility Modules Class Documentation

3.1 Ogre::BspHierarchyInterface Class Reference

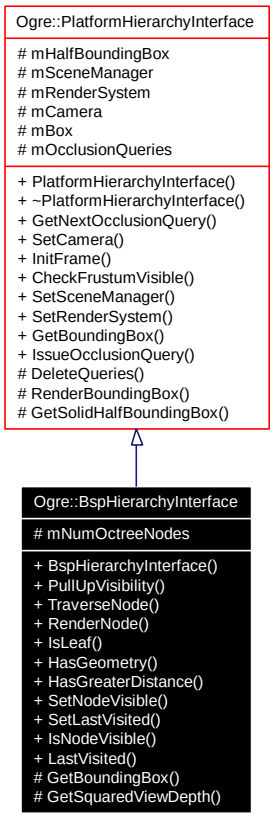
`#include <Ogre/include/OgreBspHierarchyInterface.h>`

Inherits Ogre::PlatformHierarchyInterface.

Inheritance diagram for Ogre::BspHierarchyInterface:



Collaboration diagram for Ogre::BspHierarchyInterface:



Public Member Functions

- BspHierarchyInterface (SceneManager *sm, RenderSystem *rsys)
- void PullUpVisibility (GtpVisibility::HierarchyNode *node)
- void TraverseNode (GtpVisibility::HierarchyNode *node)
- void RenderNode (GtpVisibility::HierarchyNode *node)
- bool IsLeaf (GtpVisibility::HierarchyNode *node)
- bool HasGeometry (GtpVisibility::HierarchyNode *node)
- bool HasGreaterDistance (GtpVisibility::HierarchyNode *node1, GtpVisibility::HierarchyNode *node2)
- void SetNodeVisible (GtpVisibility::HierarchyNode *node, const bool visible)
- void SetLastVisited (GtpVisibility::HierarchyNode *node, const int frameId)
- bool IsNodeVisible (GtpVisibility::HierarchyNode *node)
- int LastVisited (GtpVisibility::HierarchyNode *node)

Protected Member Functions

- AxisAlignedBox * GetBoundingBox (GtpVisibility::HierarchyNode *node)
- Real GetSquaredViewDepth (const Camera *cam, const AxisAlignedBox *box) const

Protected Attributes

- unsigned int mNumOctreeNodes

3.1.1 Detailed Description

This class implements the hierarchy interface for the Ogre bsp hierarchy.

3.1.2 Constructor & Destructor Documentation

3.1.2.1 Ogre::BspHierarchyInterface::BspHierarchyInterface (SceneManager * *sm*, RenderSystem * *rsys*)

3.1.3 Member Function Documentation

3.1.3.1 void Ogre::BspHierarchyInterface::PullUpVisibility (GtpVisibility::HierarchyNode * *node*)

Gets the given option for the scene traverser.

Remarks:
See setOption

3.1.3.2 void Ogre::BspHierarchyInterface::TraverseNode (GtpVisibility::HierarchyNode * *node*)

Traverses given node.

Parameters:
node current node

Remarks:
pushes children on distance queue

3.1.3.3 void Ogre::BspHierarchyInterface::RenderNode (GtpVisibility::HierarchyNode * *node*)

3.1.3.4 bool Ogre::BspHierarchyInterface::IsLeaf (GtpVisibility::HierarchyNode * *node*)

3.1.3.5 bool Ogre::BspHierarchyInterface::HasGeometry (GtpVisibility::HierarchyNode * *node*)

3.1.3.6 bool Ogre::BspHierarchyInterface::HasGreaterDistance (GtpVisibility::HierarchyNode * *node1*, GtpVisibility::HierarchyNode * *node2*)

3.1.3.7 void Ogre::BspHierarchyInterface::SetNodeVisible (GtpVisibility::HierarchyNode * *node*, const bool *visible*)

3.1.3.8 void Ogre::BspHierarchyInterface::SetLastVisited (GtpVisibility::HierarchyNode * *node*, const int *frameId*)

3.1.3.9 bool Ogre::BspHierarchyInterface::IsNodeVisible (GtpVisibility::HierarchyNode * *node*)

3.1.3.10 int Ogre::BspHierarchyInterface::LastVisited (GtpVisibility::HierarchyNode * *node*)

3.1.3.11 AxisAlignedBox * Ogre::BspHierarchyInterface::GetBoundingBox (GtpVisibility::HierarchyNode * *node*) [protected, virtual]

Returns pointer to the bounding box of the node.

Parameters:

node current hierarchy node

Returns:

bounding box of current node

Implements Ogre::PlatformHierarchyInterface.

3.1.3.12 Real Ogre::BspHierarchyInterface::GetSquaredViewDepth (const Camera * *cam*, const AxisAlignedBox * *box*) const [protected]

Returns squared distance of center of box with respect to the camera .

Parameters:

cam current camera

box axis aligned box

3.1.4 Member Data Documentation

3.1.4.1 unsigned int Ogre::BspHierarchyInterface::mNumOctreeNodes [protected]

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

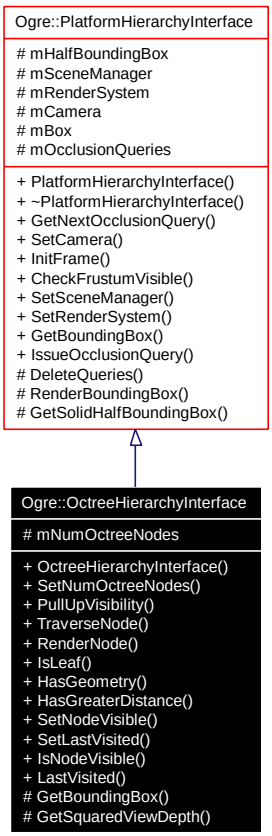
- OgreBspHierarchyInterface.h
- OgreBspHierarchyInterface.cpp

3.2 Ogre::OctreeHierarchyInterface Class Reference

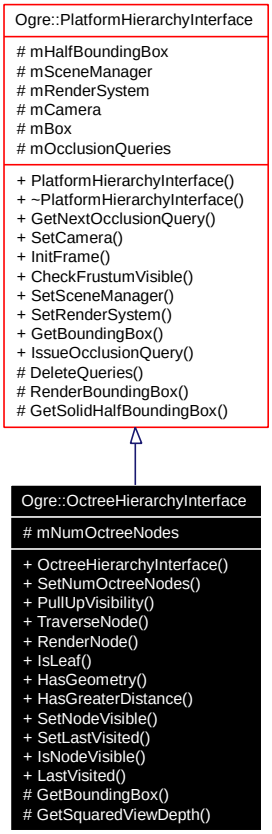
`#include <Ogre/include/OgreOctreeHierarchyInterface.h>`

Inherits Ogre::PlatformHierarchyInterface.

Inheritance diagram for Ogre::OctreeHierarchyInterface:



Collaboration diagram for Ogre::OctreeHierarchyInterface:



Public Member Functions

- OctreeHierarchyInterface (SceneManager *sm, RenderSystem *rsys)
- void SetNumOctreeNodes (unsigned int num)
- void PullUpVisibility (GtpVisibility::HierarchyNode *node)
- void TraverseNode (GtpVisibility::HierarchyNode *node)
- void RenderNode (GtpVisibility::HierarchyNode *node)
- bool IsLeaf (GtpVisibility::HierarchyNode *node)
- bool HasGeometry (GtpVisibility::HierarchyNode *node)
- bool HasGreaterDistance (GtpVisibility::HierarchyNode *node1, GtpVisibility::HierarchyNode *node2)
- void SetNodeVisible (GtpVisibility::HierarchyNode *node, const bool visible)
- void SetLastVisited (GtpVisibility::HierarchyNode *node, const int frameId)
- bool IsNodeVisible (GtpVisibility::HierarchyNode *node)
- int LastVisited (GtpVisibility::HierarchyNode *node)

Protected Member Functions

- AxisAlignedBox * GetBoundingBox (GtpVisibility::HierarchyNode *node)
- Real GetSquaredViewDepth (const Camera *cam, const AxisAlignedBox *box) const

Protected Attributes

- unsigned int mNumOctreeNodes

3.2.1 Detailed Description

This class implements the hierarchy interface for the Ogre octree hierarchy.

3.2.2 Constructor & Destructor Documentation

3.2.2.1 Ogre::OctreeHierarchyInterface::OctreeHierarchyInterface (SceneManager * *sm*, RenderSystem * *rsys*)

3.2.3 Member Function Documentation

3.2.3.1 void Ogre::OctreeHierarchyInterface::SetNumOctreeNodes (unsigned int *num*)

Sets the number of nodes in this octree

Remarks:
do not confuse this with the OctreeNode class which is derived from SceneNode

Parameters:
num number of nodes in the octree

3.2.3.2 void Ogre::OctreeHierarchyInterface::PullUpVisibility (GtpVisibility::HierarchyNode * *node*)

Gets the given option for the scene traverser.

Remarks:
See setOption

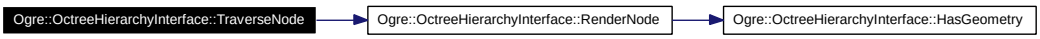
3.2.3.3 void Ogre::OctreeHierarchyInterface::TraverseNode (GtpVisibility::HierarchyNode * *node*)

Traverses given node.

Parameters:
node current node

Remarks:
pushes children on distance queue

Here is the call graph for this function:



3.2.3.4 void Ogre::OctreeHierarchyInterface::RenderNode (GtpVisibility::HierarchyNode * *node*)

Here is the call graph for this function:

3.2.3.5 bool Ogre::OctreeHierarchyInterface::IsLeaf (GtpVisibility::HierarchyNode * *node*)

3.2.3.6 bool Ogre::OctreeHierarchyInterface::HasGeometry (GtpVisibility::HierarchyNode * *node*)

3.2.3.7 bool Ogre::OctreeHierarchyInterface::HasGreaterDistance (GtpVisibility::HierarchyNode * *node1*, GtpVisibility::HierarchyNode * *node2*)

Here is the call graph for this function:

3.2.3.8 void Ogre::OctreeHierarchyInterface::SetNodeVisible (GtpVisibility::HierarchyNode * *node*, const bool *visible*)

3.2.3.9 void Ogre::OctreeHierarchyInterface::SetLastVisited (GtpVisibility::HierarchyNode * *node*, const int *frameId*)

3.2.3.10 bool Ogre::OctreeHierarchyInterface::IsNodeVisible (GtpVisibility::HierarchyNode * *node*)

3.2.3.11 int Ogre::OctreeHierarchyInterface::LastVisited (GtpVisibility::HierarchyNode * *node*)

3.2.3.12 AxisAlignedBox * Ogre::OctreeHierarchyInterface::GetBoundingBox (GtpVisibility::HierarchyNode * *node*) [protected, virtual]

Returns pointer to the bounding box of the node.

Parameters:

node current hierarchy node

Returns:

bounding box of current node

Implements Ogre::PlatformHierarchyInterface.

3.2.3.13 Real Ogre::OctreeHierarchyInterface::GetSquaredViewDepth (const Camera * *cam*, const AxisAlignedBox * *box*) const [protected]

Returns squared distance of center of box with respect to the camera .

Parameters:

cam current camera

box axis aligned box

3.2.4 Member Data Documentation

3.2.4.1 unsigned int Ogre::OctreeHierarchyInterface::mNumOctreeNodes [protected]

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

- OgreOctreeHierarchyInterface.h
- OgreOctreeHierarchyInterface.cpp

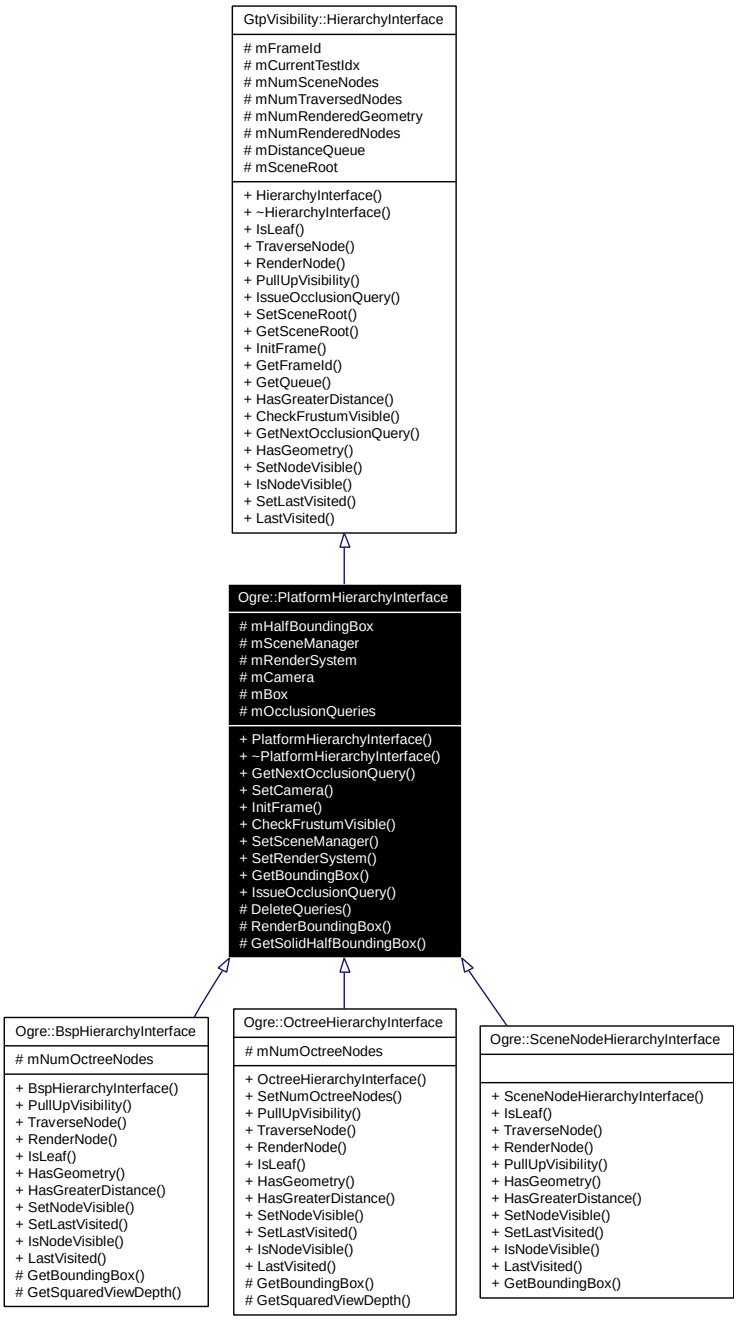
3.3 Ogre::PlatformHierarchyInterface Class Reference

`#include <Ogre/include/OgrePlatformHierarchyInterface.h>`

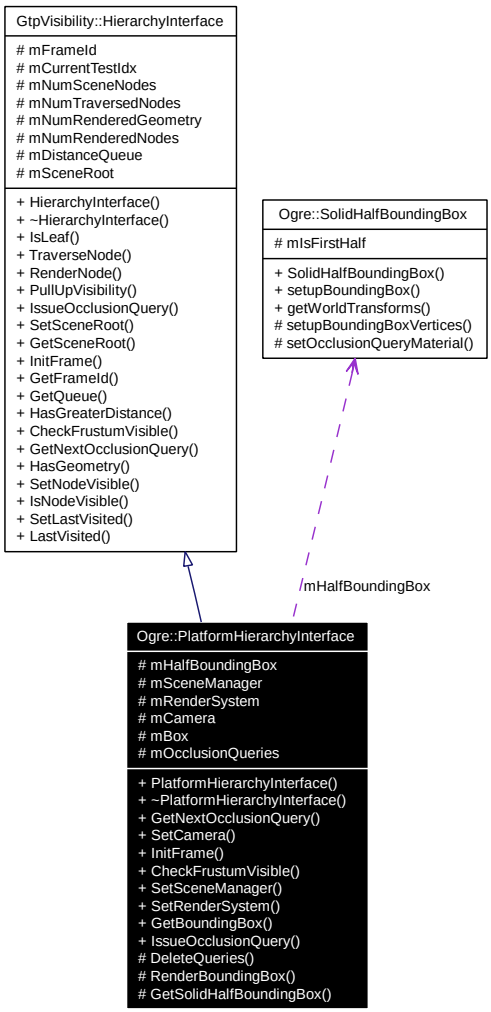
Inherits GtpVisibility::HierarchyInterface.

Inherited by Ogre::BspHierarchyInterface, Ogre::OctreeHierarchyInterface, and Ogre::SceneNodeHierarchyInterface.

Inheritance diagram for Ogre::PlatformHierarchyInterface:



Collaboration diagram for Ogre::PlatformHierarchyInterface:



Public Member Functions

- PlatformHierarchyInterface (SceneManager *sm, RenderSystem *rsys)
- ~PlatformHierarchyInterface ()
- GtpVisibility::OcclusionQuery * GetNextOcclusionQuery ()
- void SetCamera (Camera *cam)
- void InitFrame (GtpVisibility::HierarchyNode *root, Ogre::Camera *cam)
- bool CheckFrustumVisible (GtpVisibility::HierarchyNode *node, bool &intersects)
- void SetSceneManager (SceneManager *sm)
- void SetRenderSystem (RenderSystem *rsys)
- virtual AxisAlignedBox * GetBoundingBox (GtpVisibility::HierarchyNode *node)=0
- GtpVisibility::OcclusionQuery * IssueOcclusionQuery (GtpVisibility::HierarchyNode *node)

Protected Member Functions

- void DeleteQueries ()

- void RenderBoundingBox (AxisAlignedBox *box)
- SolidHalfBoundingBox * GetSolidHalfBoundingBox (int half)

Protected Attributes

- SolidHalfBoundingBox * mHalfBoundingBox [2]
- SceneManager * mSceneManager
- RenderSystem * mRenderSystem
- Camera * mCamera
- AxisAlignedBox mBox
- std::vector< PlatformOcclusionQuery * > mOcclusionQueries

3.3.1 Detailed Description

Class which implements a hierarchy interface for a specific type of hierarchy.

Remarks:
also provides methods for using occlusion queries on the hierarchy nodes

3.3.2 Constructor & Destructor Documentation

3.3.2.1 Ogre::PlatformHierarchyInterface::PlatformHierarchyInterface (SceneManager * *sm*, RenderSystem * *rsys*)

Construction taking the current scene manager and the current rendersystem as argument

Parameters:
sm current scene manager

rsys current render system

3.3.2.2 Ogre::PlatformHierarchyInterface::~PlatformHierarchyInterface ()

Here is the call graph for this function:

3.3.3 Member Function Documentation

3.3.3.1 GtpVisibility::OcclusionQuery * Ogre::PlatformHierarchyInterface::GetNextOcclusionQuery () [virtual]

Returns next available occlusion query or creates new one.

Returns:
the next occlusion query

Implements GtpVisibility::HierarchyInterface.

3.3.3.2 void Ogre::PlatformHierarchyInterface::SetCamera (Camera * *cam*)

Sets the current camera.

Parameters:

cam the current camera

3.3.3.3 void Ogre::PlatformHierarchyInterface::InitFrame (GtpVisibility::HierarchyNode * *root*, Ogre::Camera * *cam*)

Initialises this scene traverser for the current frame.

Parameters:

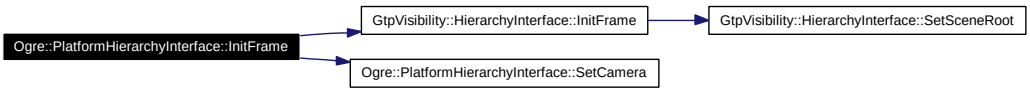
root root of the hierarchy

cam current camera

Remarks:

convenience method wich calls VisibilitySceneTraverser::initFrame, sets the current camera, and initialises the distance queue.

Here is the call graph for this function:



3.3.3.4 bool Ogre::PlatformHierarchyInterface::CheckFrustumVisible (GtpVisibility::HierarchyNode * *node*, bool & *intersects*)

Checks if the node is visible from the current view frustum.

Parameters:

node the current node

intersects returns true if the current node intersects the near plane

Here is the call graph for this function:

3.3.3.5 void Ogre::PlatformHierarchyInterface::SetSceneManager (SceneManager * *sm*)

Sets pointer to the current scene manager.

Parameters:

sm the scene manager

3.3.3.6 void Ogre::PlatformHierarchyInterface::SetRenderSystem (RenderSystem * *rsys*)

Sets pointer to the current render system

Parameters:
rsys the rendersystem

3.3.3.7 virtual AxisAlignedBox* Ogre::PlatformHierarchyInterface::GetBoundingBox (GtpVisibility::HierarchyNode * *node*) [pure virtual]

Returns pointer to bounding box of node.

Parameters:
node current hierarchy node

Returns:
bounding box of current node

Implemented in Ogre::BspHierarchyInterface, Ogre::OctreeHierarchyInterface, and Ogre::SceneNode-HierarchyInterface.

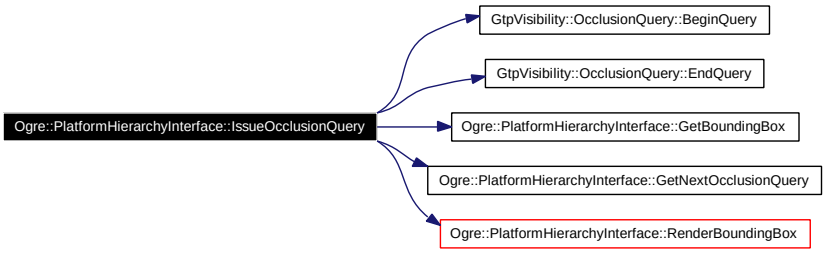
3.3.3.8 GtpVisibility::OcclusionQuery * Ogre::PlatformHierarchyInterface::IssueOcclusionQuery (GtpVisibility::HierarchyNode * *node*)

Issue a occlusion query for this node.

Parameters:
node the current hierarchy node

Returns:
occlusion query for this node

Here is the call graph for this function:



3.3.3.9 void Ogre::PlatformHierarchyInterface::DeleteQueries () [protected]

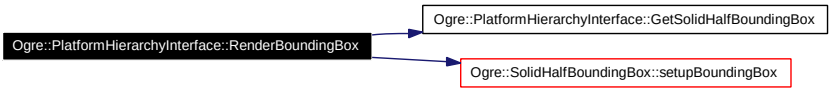
Deletes all occlusion queries.

3.3.3.10 void Ogre::PlatformHierarchyInterface::RenderBoundingBox (AxisAlignedBox * *box*) [protected]

Renders bounding box of specified node.

Parameters:
box the bounding box of the scene node to be rendered

Here is the call graph for this function:



3.3.3.11 SolidHalfBoundingBox * Ogre::PlatformHierarchyInterface::GetSolidHalfBoundingBox (int *half*) [protected]

Returns one half of the bounding box.

Parameters:
half the half index of the bouding box (0 or 1)

3.3.4 Member Data Documentation

3.3.4.1 SolidHalfBoundingBox* Ogre::PlatformHierarchyInterface::mHalfBoundingBox[2] [protected]

two halfes of an aabb.

3.3.4.2 SceneManager* Ogre::PlatformHierarchyInterface::mSceneManager [protected]

3.3.4.3 RenderSystem* Ogre::PlatformHierarchyInterface::mRenderSystem [protected]

3.3.4.4 Camera* Ogre::PlatformHierarchyInterface::mCamera [protected]

3.3.4.5 AxisAlignedBox Ogre::PlatformHierarchyInterface::mBox [protected]

3.3.4.6 std::vector<PlatformOcclusionQuery *> Ogre::PlatformHierarchyInterface::mOcclusionQueries [protected]

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

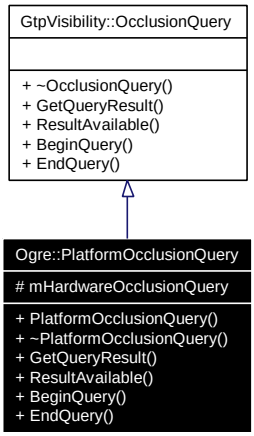
- OgrePlatformHierarchyInterface.h
- OgrePlatformHierarchyInterface.cpp

3.4 Ogre::PlatformOcclusionQuery Class Reference

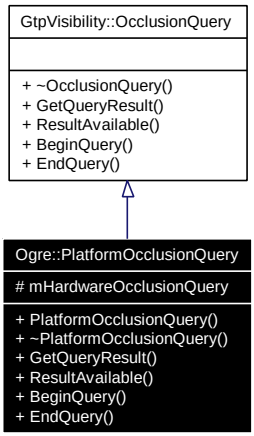
`#include <Ogre/include/OgrePlatformOcclusionQuery.h>`

Inherits GtpVisibility::OcclusionQuery.

Inheritance diagram for Ogre::PlatformOcclusionQuery:



Collaboration diagram for Ogre::PlatformOcclusionQuery:



Public Member Functions

- PlatformOcclusionQuery (RenderSystem *rsys)
- virtual ~PlatformOcclusionQuery ()
- virtual unsigned int GetQueryResult () const
- virtual bool ResultAvailable () const
- virtual void BeginQuery () const
- virtual void EndQuery () const

Protected Attributes

- HardwareOcclusionQuery * mHardwareOcclusionQuery

3.4.1 Detailed Description

This class is an implementation for occlusion queries using Ogre.

Remarks:
the class encapsulates Ogre occlusion query calls.

3.4.2 Constructor & Destructor Documentation

3.4.2.1 Ogre::PlatformOcclusionQuery::PlatformOcclusionQuery (RenderSystem * *rsys*)

3.4.2.2 Ogre::PlatformOcclusionQuery::~PlatformOcclusionQuery () [virtual]

3.4.3 Member Function Documentation

3.4.3.1 unsigned int Ogre::PlatformOcclusionQuery::GetQueryResult () const [virtual]

Returns the result of an occlusion query in terms of visible pixels.

Returns:
number of visible pixels

Implements GtpVisibility::OcclusionQuery.

3.4.3.2 bool Ogre::PlatformOcclusionQuery::ResultAvailable () const [virtual]

Returns true if the result of the query is available, false otherwise.

Returns:
if result is available

Implements GtpVisibility::OcclusionQuery.

3.4.3.3 void Ogre::PlatformOcclusionQuery::BeginQuery () const [virtual]

Begins occlusion query.

Remarks:
the query counts the number of visible pixels between it's begin and end

Implements GtpVisibility::OcclusionQuery.

3.4.3.4 void Ogre::PlatformOcclusionQuery::EndQuery () const [virtual]

Ends occlusion query.

Implements GtpVisibility::OcclusionQuery.

3.4.4 Member Data Documentation

3.4.4.1 HardwareOcclusionQuery* Ogre::PlatformOcclusionQuery::mHardwareOcclusionQuery [protected]

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

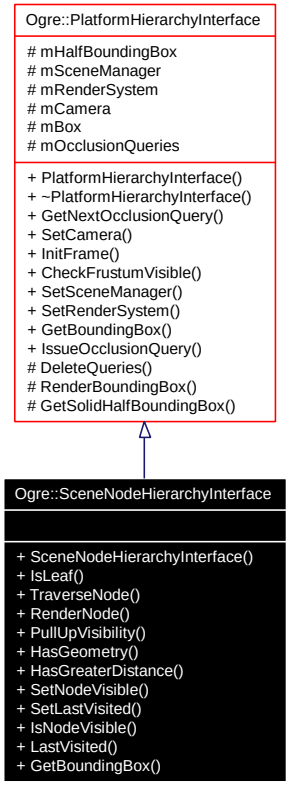
- OgrePlatformOcclusionQuery.h
- OgrePlatformOcclusionQuery.cpp

3.5 Ogre::SceneNodeHierarchyInterface Class Reference

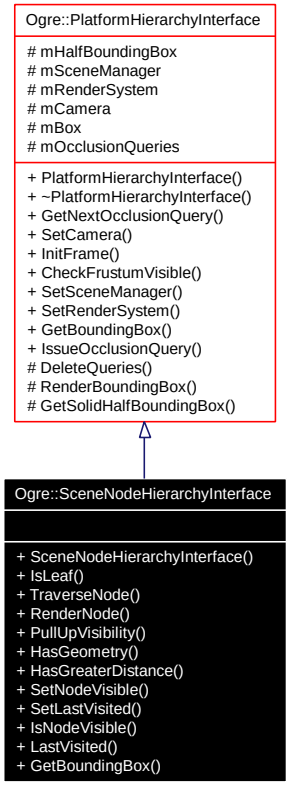
`#include <Ogre/include/OgreSceneNodeHierarchyInterface.h>`

Inherits Ogre::PlatformHierarchyInterface.

Inheritance diagram for Ogre::SceneNodeHierarchyInterface:



Collaboration diagram for Ogre::SceneNodeHierarchyInterface:



Public Member Functions

- SceneNodeHierarchyInterface (SceneManager *sm, RenderSystem *rsys)
- bool IsLeaf (GtpVisibility::HierarchyNode *node)
- void TraverseNode (GtpVisibility::HierarchyNode *node)
- void RenderNode (GtpVisibility::HierarchyNode *node)
- void PullUpVisibility (GtpVisibility::HierarchyNode *node)
- bool HasGeometry (GtpVisibility::HierarchyNode *node)
- bool HasGreaterDistance (GtpVisibility::HierarchyNode *node1, GtpVisibility::HierarchyNode *node2)
- void SetNodeVisible (GtpVisibility::HierarchyNode *node, const bool visible)
- void SetLastVisited (GtpVisibility::HierarchyNode *node, const int frameId)
- bool IsNodeVisible (GtpVisibility::HierarchyNode *node)
- int LastVisited (GtpVisibility::HierarchyNode *node)
- AxisAlignedBox * GetBoundingBox (GtpVisibility::HierarchyNode *node)

3.5.1 Detailed Description

This class implements the hierarchy interface for the Ogre scene node hierarchy.

3.5.2 Constructor & Destructor Documentation

3.5.2.1 Ogre::SceneNodeHierarchyInterface::SceneNodeHierarchyInterface (SceneManager * *sm*, RenderSystem * *rsys*)

Construction taking the current scene manager and the current rendersystem as argument

Parameters:

sm current scene manager

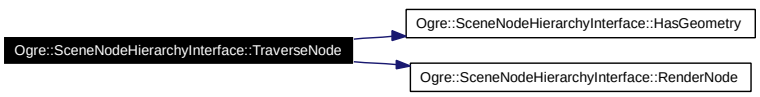
rsys current render system

3.5.3 Member Function Documentation

3.5.3.1 bool Ogre::SceneNodeHierarchyInterface::IsLeaf (GtpVisibility::HierarchyNode * *node*)

3.5.3.2 void Ogre::SceneNodeHierarchyInterface::TraverseNode (GtpVisibility::HierarchyNode * *node*)

Here is the call graph for this function:



3.5.3.3 void Ogre::SceneNodeHierarchyInterface::RenderNode (GtpVisibility::HierarchyNode * *node*)

3.5.3.4 void Ogre::SceneNodeHierarchyInterface::PullUpVisibility (GtpVisibility::HierarchyNode * *node*)

3.5.3.5 bool Ogre::SceneNodeHierarchyInterface::HasGeometry (GtpVisibility::HierarchyNode * *node*)

3.5.3.6 bool Ogre::SceneNodeHierarchyInterface::HasGreaterDistance (GtpVisibility::HierarchyNode * *node1*, GtpVisibility::HierarchyNode * *node2*)

3.5.3.7 void Ogre::SceneNodeHierarchyInterface::SetNodeVisible (GtpVisibility::HierarchyNode * *node*, const bool *visible*)

3.5.3.8 void Ogre::SceneNodeHierarchyInterface::SetLastVisited (GtpVisibility::HierarchyNode * *node*, const int *frameId*)

3.5.3.9 bool Ogre::SceneNodeHierarchyInterface::IsNodeVisible (GtpVisibility::HierarchyNode * *node*)

3.5.3.10 int Ogre::SceneNodeHierarchyInterface::LastVisited (GtpVisibility::HierarchyNode * *node*)

3.5.3.11 AxisAlignedBox * Ogre::SceneNodeHierarchyInterface::GetBoundingBox (GtpVisibility::HierarchyNode * *node*) [virtual]

Returns pointer to bounding box of node.

Parameters:

node current hierarchy node

Returns:

bounding box of current node

Implements Ogre::PlatformHierarchyInterface.

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

- OgreSceneNodeHierarchyInterface.h
- OgreSceneNodeHierarchyInterface.cpp

3.6 Ogre::SolidHalfBoundingBox Class Reference

`#include <Ogre/include/OgreSolidHalfBoundingBox.h>`

Public Member Functions

- SolidHalfBoundingBox (bool isFirstHalf)
- void setupBoundingBox (const AxisAlignedBox &aabb)
- void getWorldTransforms (Matrix4 *xform) const

Protected Member Functions

- void setupBoundingBoxVertices (const AxisAlignedBox &aab)
- void setOcclusionQueryMaterial ()

Protected Attributes

- bool mIsFirstHalf

3.6.1 Detailed Description

Allows the rendering of one half of a solid bounding box.

Remarks:

This class builds a wireframe renderable from a given aabb. A pointer to this class can be added to a render queue to display the bounding box of an object.

3.6.2 Constructor & Destructor Documentation

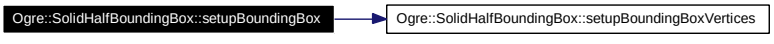
3.6.2.1 Ogre::SolidHalfBoundingBox::SolidHalfBoundingBox (bool *isFirstHalf*)

Here is the call graph for this function:

3.6.3 Member Function Documentation

3.6.3.1 void Ogre::SolidHalfBoundingBox::setupBoundingBox (const AxisAlignedBox & *aabb*)

Here is the call graph for this function:



3.6.3.2 void Ogre::SolidHalfBoundingBox::getWorldTransforms (Matrix4 * *xform*) const

Override this method to prevent parent transforms (rotation,translation,scale) and to make it public.

3.6.3.3 void Ogre::SolidHalfBoundingBox::setupBoundingBoxVertices (const AxisAlignedBox & *aab*) [protected]

Builds the wireframe line list.

Parameters:

aab the axis aligned bounding box for setting up the list.

3.6.3.4 void Ogre::SolidHalfBoundingBox::setOcclusionQueryMaterial () [protected]

Sets the material used for occlusion queries.

Remarks:

the material is called "OcclusionQuery" and uses no lighting, no depth write, and no colours

3.6.4 Member Data Documentation

3.6.4.1 bool Ogre::SolidHalfBoundingBox::mIsFirstHalf [protected]

Whether this half box is the first or the second half of the bounding box.

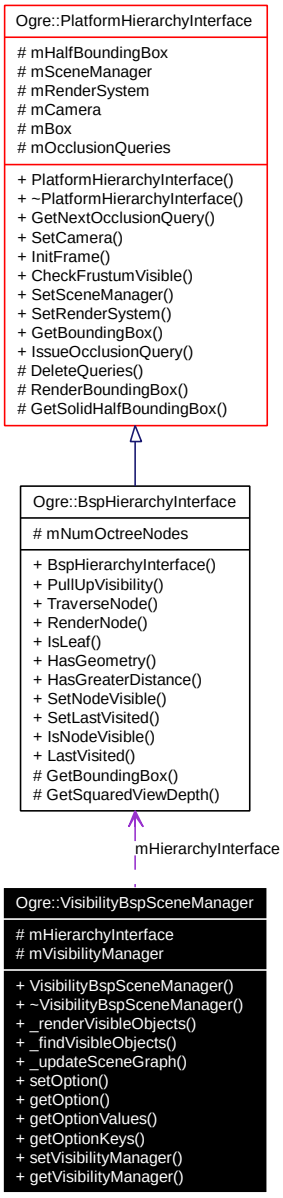
The documentation for this class was generated from the following files:

- OgreSolidHalfBoundingBox.h
- OgreSolidHalfBoundingBox.cpp

3.7 Ogre::VisibilityBspSceneManager Class Reference

`#include <Ogre/include/OgreVisibilityBspSceneManager.h>`

Collaboration diagram for Ogre::VisibilityBspSceneManager:



Public Member Functions

- VisibilityBspSceneManager (GtpVisibility::Manager *visManager)
- ~VisibilityBspSceneManager ()
- void _renderVisibleObjects ()
- void _findVisibleObjects (Camera *cam, bool onlyShadowCasters)
- void _updateSceneGraph (Camera *cam)

- virtual bool setOption (const String &, const void *)
- virtual bool getOption (const String &, void *)
- bool getOptionValues (const String &key, StringVector &refValueList)
- bool getOptionKeys (StringVector &refKeys)
- void setVisibilityManager (GtpVisibility::Manager *visManager)
- GtpVisibility::Manager * getVisibilityManager (void)

Protected Attributes

- BspHierarchyInterface * mHierarchyInterface
- GtpVisibility::Manager * mVisibilityManager

3.7.1 Detailed Description

This class extends the octree scene manager, using occlusion queries for visibility culling.

3.7.2 Constructor & Destructor Documentation

3.7.2.1 Ogre::VisibilityBspSceneManager::VisibilityBspSceneManager (GtpVisibility::Manager * *visManager*)

3.7.2.2 Ogre::VisibilityBspSceneManager::~VisibilityBspSceneManager ()

3.7.3 Member Function Documentation

3.7.3.1 void Ogre::VisibilityBspSceneManager::_renderVisibleObjects ()

Here is the call graph for this function:

3.7.3.2 void Ogre::VisibilityBspSceneManager::_findVisibleObjects (Camera * *cam*, bool *onlyShadowCasters*)

3.7.3.3 void Ogre::VisibilityBspSceneManager::_updateSceneGraph (Camera * *cam*)

Here is the call graph for this function:

3.7.3.4 bool Ogre::VisibilityBspSceneManager::setOption (const String &, const void *) [virtual]

Sets the given option for the SceneManager

Remarks:
Options are: "Algorithm", int *;

3.7.3.5 bool Ogre::VisibilityBspSceneManager::getOption (const String &, void *) [virtual]

Gets the given option for the Scene Manager.

Remarks:
See setOption

3.7.3.6 bool Ogre::VisibilityBspSceneManager::getOptionValues (const String & *key*, StringVector & *refValueList*)

3.7.3.7 bool Ogre::VisibilityBspSceneManager::getOptionKeys (StringVector & *refKeys*)

3.7.3.8 void Ogre::VisibilityBspSceneManager::setVisibilityManager (GtpVisibility::Manager * *visManager*)

Sets the visibility manager.

Parameters:
visManager the visibility manager

3.7.3.9 GtpVisibility::Manager * Ogre::VisibilityBspSceneManager::getVisibilityManager (void)

See set.

3.7.4 Member Data Documentation

3.7.4.1 BspHierarchyInterface* Ogre::VisibilityBspSceneManager::mHierarchyInterface [protected]

3.7.4.2 GtpVisibility::Manager* Ogre::VisibilityBspSceneManager::mVisibilityManager [protected]

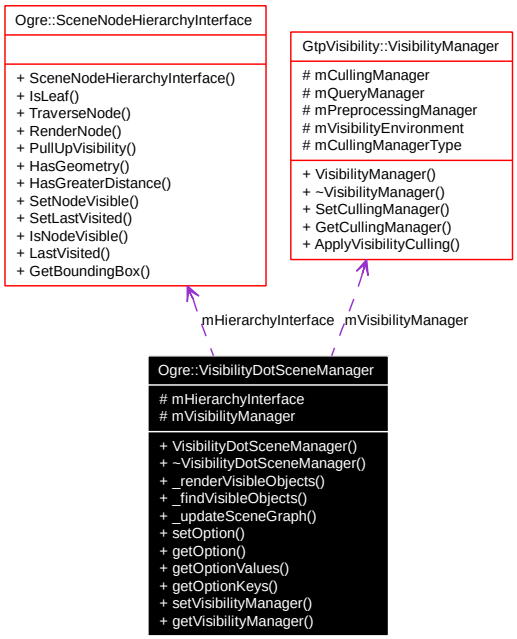
The documentation for this class was generated from the following files:

- OgreVisibilityBspSceneManager.h
- OgreVisibilityBspSceneManager.cpp

3.8 Ogre::VisibilityDotSceneManager Class Reference

`#include <Ogre/include/OgreVisibilityDotSceneManager.h>`

Collaboration diagram for Ogre::VisibilityDotSceneManager:



Public Member Functions

- VisibilityDotSceneManager (GtpVisibility::VisibilityManager *visManager)
- ~VisibilityDotSceneManager ()
- void _renderVisibleObjects ()
- void _findVisibleObjects (Camera *cam, bool onlyShadowCasters)
- void _updateSceneGraph (Camera *cam)
- virtual bool setOption (const String &, const void *)
- virtual bool getOption (const String &, void *)
- bool getOptionValues (const String &key, StringVector &refValueList)
- bool getOptionKeys (StringVector &refKeys)
- void setVisibilityManager (GtpVisibility::VisibilityManager *visManager)
- GtpVisibility::VisibilityManager * getVisibilityManager ()

Protected Attributes

- Ogre::SceneNodeHierarchyInterface * mHierarchyInterface
- GtpVisibility::VisibilityManager * mVisibilityManager

3.8.1 Detailed Description

This class extends the dot scene manager, using occlusion queries for visibility culling.

Remarks:
the scene manager can operate on Ogre scene descriptions in XML defined by the ".scene" format

3.8.2 Constructor & Destructor Documentation

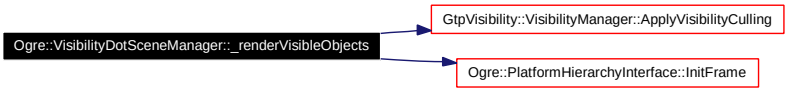
3.8.2.1 Ogre::VisibilityDotSceneManager::VisibilityDotSceneManager (GtpVisibility::VisibilityManager * *visManager*)

3.8.2.2 Ogre::VisibilityDotSceneManager::~VisibilityDotSceneManager ()

3.8.3 Member Function Documentation

3.8.3.1 void Ogre::VisibilityDotSceneManager::_renderVisibleObjects ()

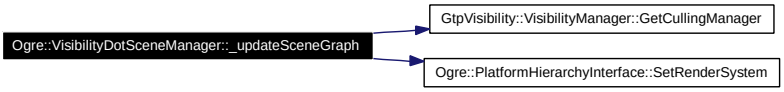
Here is the call graph for this function:



3.8.3.2 void Ogre::VisibilityDotSceneManager::_findVisibleObjects (Camera * *cam*, bool *onlyShadowCasters*)

3.8.3.3 void Ogre::VisibilityDotSceneManager::_updateSceneGraph (Camera * *cam*)

Here is the call graph for this function:



3.8.3.4 bool Ogre::VisibilityDotSceneManager::setOption (const String &, const void *) [virtual]

Sets the given option for the SceneManager

Remarks:
Options are: "Algorithm", int *;

3.8.3.5 bool Ogre::VisibilityDotSceneManager::getOption (const String &, void *) [virtual]

Gets the given option for the Scene VisibilityManager.

Remarks:
See setOption

3.8.3.6 bool Ogre::VisibilityDotSceneManager::getOptionValues (const String & *key*, StringVector & *refValueList*)

3.8.3.7 bool Ogre::VisibilityDotSceneManager::getOptionKeys (StringVector & *refKeys*)

3.8.3.8 void Ogre::VisibilityDotSceneManager::setVisibilityManager (GtpVisibility::VisibilityManager * *visManager*)

Sets the visibility manager.

Parameters:

visManager the visibility manager

3.8.3.9 GtpVisibility::VisibilityManager * Ogre::VisibilityDotSceneManager::getVisibilityManager ()

See set.

3.8.4 Member Data Documentation

3.8.4.1 Ogre::SceneNodeHierarchyInterface* Ogre::VisibilityDotSceneManager::mHierarchyInterface [protected]

3.8.4.2 GtpVisibility::VisibilityManager* Ogre::VisibilityDotSceneManager::mVisibilityManager [protected]

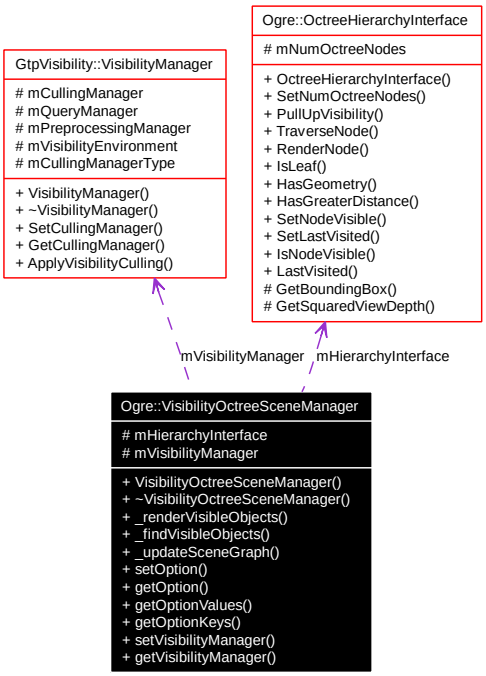
The documentation for this class was generated from the following files:

- OgreVisibilityDotSceneManager.h
- OgreVisibilityDotSceneManager.cpp

3.9 Ogre::VisibilityOctreeSceneManager Class Reference

`#include <Ogre/include/OgreVisibilityOctreeSceneManager.h>`

Collaboration diagram for Ogre::VisibilityOctreeSceneManager:



Public Member Functions

- VisibilityOctreeSceneManager (GtpVisibility::VisibilityManager *visManager)
- ~VisibilityOctreeSceneManager ()
- void _renderVisibleObjects ()
- void _findVisibleObjects (Camera *cam, bool onlyShadowCasters)
- void _updateSceneGraph (Camera *cam)
- virtual bool setOption (const String &, const void *)
- virtual bool getOption (const String &, void *)
- bool getOptionValues (const String &key, StringVector &refValueList)
- bool getOptionKeys (StringVector &refKeys)
- void setVisibilityManager (GtpVisibility::VisibilityManager *visManager)
- GtpVisibility::VisibilityManager * getVisibilityManager (void)

Protected Attributes

- OctreeHierarchyInterface * mHierarchyInterface
- GtpVisibility::VisibilityManager * mVisibilityManager

3.9.1 Detailed Description

This class extends the octree scene manager, using occlusion queries for visibility culling.

3.9.2 Constructor & Destructor Documentation

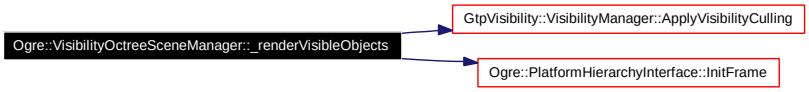
3.9.2.1 Ogre::VisibilityOctreeSceneManager::VisibilityOctreeSceneManager (GtpVisibility::VisibilityManager * *visManager*)

3.9.2.2 Ogre::VisibilityOctreeSceneManager::~VisibilityOctreeSceneManager ()

3.9.3 Member Function Documentation

3.9.3.1 void Ogre::VisibilityOctreeSceneManager::_renderVisibleObjects ()

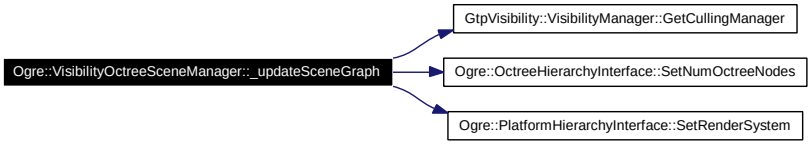
Here is the call graph for this function:



3.9.3.2 void Ogre::VisibilityOctreeSceneManager::_findVisibleObjects (Camera * *cam*, bool *onlyShadowCasters*)

3.9.3.3 void Ogre::VisibilityOctreeSceneManager::_updateSceneGraph (Camera * *cam*)

Here is the call graph for this function:



3.9.3.4 bool Ogre::VisibilityOctreeSceneManager::setOption (const String &, const void *) [virtual]

Sets the given option for the SceneManager

Remarks:
Options are: "Algorithm", int *;

3.9.3.5 bool Ogre::VisibilityOctreeSceneManager::getOption (const String &, void *) [virtual]

Gets the given option for the Scene VisibilityManager.

Remarks:
See setOption

3.9.3.6 bool Ogre::VisibilityOctreeSceneManager::getOptionValues (const String & *key*, StringVector & *refValueList*)

3.9.3.7 bool Ogre::VisibilityOctreeSceneManager::getOptionKeys (StringVector & *refKeys*)

3.9.3.8 void Ogre::VisibilityOctreeSceneManager::setVisibilityManager (GtpVisibility::VisibilityManager * *visManager*)

Sets the visibility manager.

Parameters:

visManager the visibility manager

3.9.3.9 GtpVisibility::VisibilityManager * Ogre::VisibilityOctreeSceneManager::getVisibilityManager (void)

See set.

3.9.4 Member Data Documentation

3.9.4.1 OctreeHierarchyInterface* Ogre::VisibilityOctreeSceneManager::mHierarchyInterface [protected]

3.9.4.2 GtpVisibility::VisibilityManager* Ogre::VisibilityOctreeSceneManager::mVisibilityManager [protected]

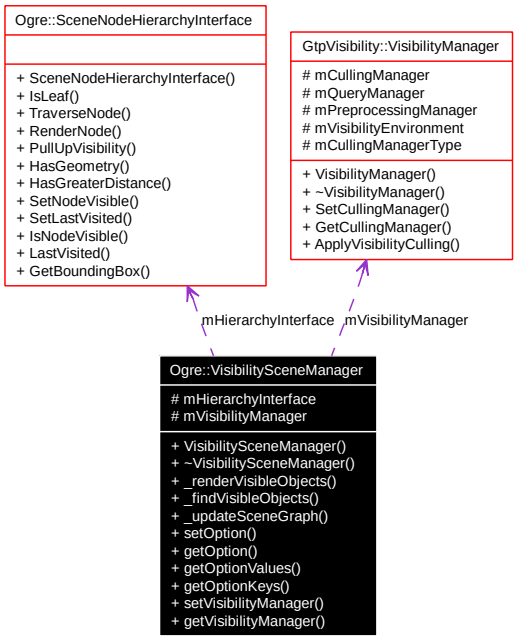
The documentation for this class was generated from the following files:

- OgreVisibilityOctreeSceneManager.h
- OgreVisibilityOctreeSceneManager.cpp

3.10 Ogre::VisibilitySceneManager Class Reference

`#include <Ogre/include/OgreVisibilitySceneManager.h>`

Collaboration diagram for Ogre::VisibilitySceneManager:



Public Member Functions

- VisibilitySceneManager (GtpVisibility::VisibilityManager *visManager)
- ~VisibilitySceneManager ()
- void _renderVisibleObjects ()
- void _findVisibleObjects (Camera *cam, bool onlyShadowCasters)
- void _updateSceneGraph (Camera *cam)
- virtual bool setOption (const String &, const void *)
- virtual bool getOption (const String &, void *)
- bool getOptionValues (const String &key, StringVector &refValueList)
- bool getOptionKeys (StringVector &refKeys)
- void setVisibilityManager (GtpVisibility::VisibilityManager *visManager)
- GtpVisibility::VisibilityManager * getVisibilityManager ()

Protected Attributes

- SceneNodeHierarchyInterface * mHierarchyInterface
- GtpVisibility::VisibilityManager * mVisibilityManager

3.10.1 Detailed Description

This class implements the default scene manager, using occlusion queries for visibility culling.

3.10.2 Constructor & Destructor Documentation

3.10.2.1 Ogre::VisibilitySceneManager::VisibilitySceneManager (GtpVisibility::VisibilityManager * *visManager*)

3.10.2.2 Ogre::VisibilitySceneManager::~VisibilitySceneManager ()

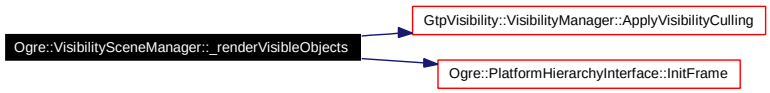
3.10.3 Member Function Documentation

3.10.3.1 void Ogre::VisibilitySceneManager::_renderVisibleObjects ()

Overriden from SceneManager. Renders the scene using occlusion culling.

Remarks:
the type of algorithm is specified by the occlusion culling manager type using by the visibility manager.

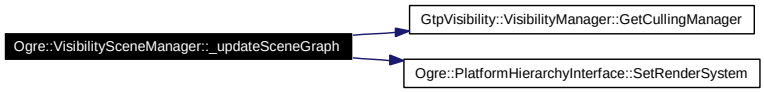
Here is the call graph for this function:



3.10.3.2 void Ogre::VisibilitySceneManager::_findVisibleObjects (Camera * *cam*, bool *onlyShadowCasters*)

3.10.3.3 void Ogre::VisibilitySceneManager::_updateSceneGraph (Camera * *cam*)

Here is the call graph for this function:



3.10.3.4 bool Ogre::VisibilitySceneManager::setOption (const String &, const void *) [virtual]

Sets the given option for the SceneManager

Remarks:
Options are: "Algorithm", int *;

3.10.3.5 bool Ogre::VisibilitySceneManager::getOption (const String &, void *) [virtual]

Gets the given option for the Scene VisibilityManager.

Remarks:
See setOption

3.10.3.6 bool Ogre::VisibilitySceneManager::getOptionValues (const String & *key*, StringVector & *refValueList*)

3.10.3.7 bool Ogre::VisibilitySceneManager::getOptionKeys (StringVector & *refKeys*)

3.10.3.8 void Ogre::VisibilitySceneManager::setVisibilityManager (GtpVisibility::VisibilityManager * *visManager*)

Sets the visibility manager.

Parameters:

visManager the visibility manager

3.10.3.9 GtpVisibility::VisibilityManager * Ogre::VisibilitySceneManager::getVisibilityManager ()

See set.

3.10.4 Member Data Documentation

3.10.4.1 SceneNodeHierarchyInterface* Ogre::VisibilitySceneManager::mHierarchyInterface [protected]

3.10.4.2 GtpVisibility::VisibilityManager* Ogre::VisibilitySceneManager::mVisibilityManager [protected]

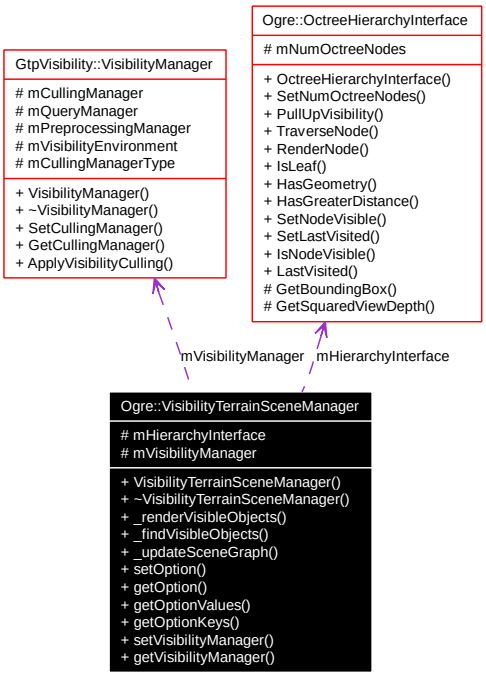
The documentation for this class was generated from the following files:

- OgreVisibilitySceneManager.h
- OgreVisibilitySceneManager.cpp

3.11 Ogre::VisibilityTerrainSceneManager Class Reference

`#include <Ogre/include/OgreVisibilityTerrainSceneManager.h>`

Collaboration diagram for Ogre::VisibilityTerrainSceneManager:



Public Member Functions

- VisibilityTerrainSceneManager (GtpVisibility::VisibilityManager *visManager)
- ~VisibilityTerrainSceneManager ()
- void _renderVisibleObjects (void)
- void _findVisibleObjects (Camera *cam, bool onlyShadowCasters)
- void _updateSceneGraph (Camera *cam)
- virtual bool setOption (const String &, const void *)
- virtual bool getOption (const String &, void *)
- bool getOptionValues (const String &key, StringVector &refValueList)
- bool getOptionKeys (StringVector &refKeys)
- void setVisibilityManager (GtpVisibility::VisibilityManager *visManager)
- GtpVisibility::VisibilityManager * getVisibilityManager ()

Protected Attributes

- OctreeHierarchyInterface * mHierarchyInterface
- GtpVisibility::VisibilityManager * mVisibilityManager

3.11.1 Detailed Description

This class extends the terrain scene manager, using occlusion queries for visibility culling.

3.11.2 Constructor & Destructor Documentation

3.11.2.1 Ogre::VisibilityTerrainSceneManager::VisibilityTerrainSceneManager (GtpVisibility::VisibilityManager * *visManager*)

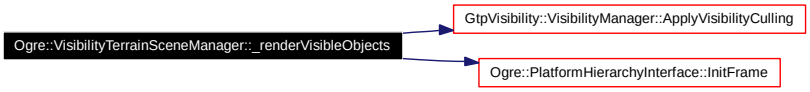
Here is the call graph for this function:

3.11.2.2 Ogre::VisibilityTerrainSceneManager::~VisibilityTerrainSceneManager ()

3.11.3 Member Function Documentation

3.11.3.1 void Ogre::VisibilityTerrainSceneManager::_renderVisibleObjects (void)

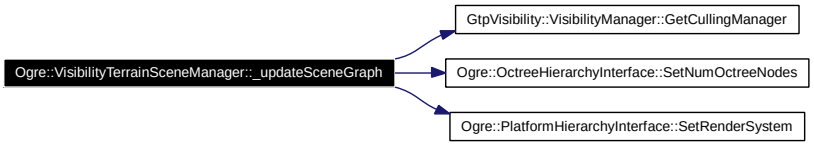
Here is the call graph for this function:



3.11.3.2 void Ogre::VisibilityTerrainSceneManager::_findVisibleObjects (Camera * *cam*, bool *onlyShadowCasters*)

3.11.3.3 void Ogre::VisibilityTerrainSceneManager::_updateSceneGraph (Camera * *cam*)

Here is the call graph for this function:



3.11.3.4 bool Ogre::VisibilityTerrainSceneManager::setOption (const String &, const void *) [virtual]

Sets the given option for the SceneManager

Remarks:

Options are: "Algorithm", int *;

3.11.3.5 bool Ogre::VisibilityTerrainSceneManager::getOption (const String &, void *) [virtual]

Gets the given option for the Scene VisibilityManager.

Remarks:
See setOption

3.11.3.6 bool Ogre::VisibilityTerrainSceneManager::getOptionValues (const String & *key*, StringVector & *refValueList*)

3.11.3.7 bool Ogre::VisibilityTerrainSceneManager::getOptionKeys (StringVector & *refKeys*)

3.11.3.8 void Ogre::VisibilityTerrainSceneManager::setVisibilityManager (GtpVisibility::VisibilityManager * *visManager*)

Sets the visibility manager.

Parameters:
visManager the visibility manager

3.11.3.9 GtpVisibility::VisibilityManager * Ogre::VisibilityTerrainSceneManager::getVisibilityManager ()

See set.

3.11.4 Member Data Documentation

3.11.4.1 OctreeHierarchyInterface* Ogre::VisibilityTerrainSceneManager::mHierarchyInterface [protected]

3.11.4.2 GtpVisibility::VisibilityManager* Ogre::VisibilityTerrainSceneManager::mVisibilityManager [protected]

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

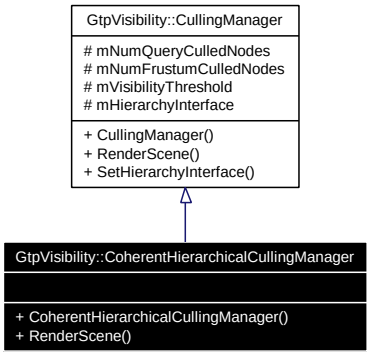
- OgreVisibilityTerrainSceneManager.h
- OgreVisibilityTerrainSceneManager.cpp

3.12 GtpVisibility::CoherentHierarchicalCullingManager Class Reference

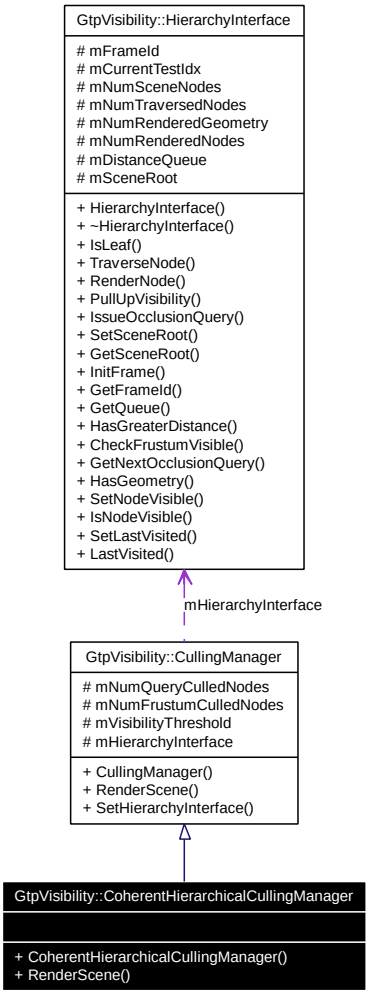
`#include <GtpVisibility/include/CoherentHierarchicalCullingManager.h>`

Inherits GtpVisibility::CullingManager.

Inheritance diagram for GtpVisibility::CoherentHierarchicalCullingManager:



Collaboration diagram for GtpVisibility::CoherentHierarchicalCullingManager:



Public Member Functions

- CoherentHierarchicalCullingManager (HierarchyInterface *hierarchyInterface)
- void RenderScene ()

3.12.1 Detailed Description

Renders the scene with the coherent hierarchical culling algorithm.

3.12.2 Constructor & Destructor Documentation

3.12.2.1 GtpVisibility::CoherentHierarchicalCullingManager::CoherentHierarchicalCullingManager (HierarchyInterface * *hierarchyInterface*)

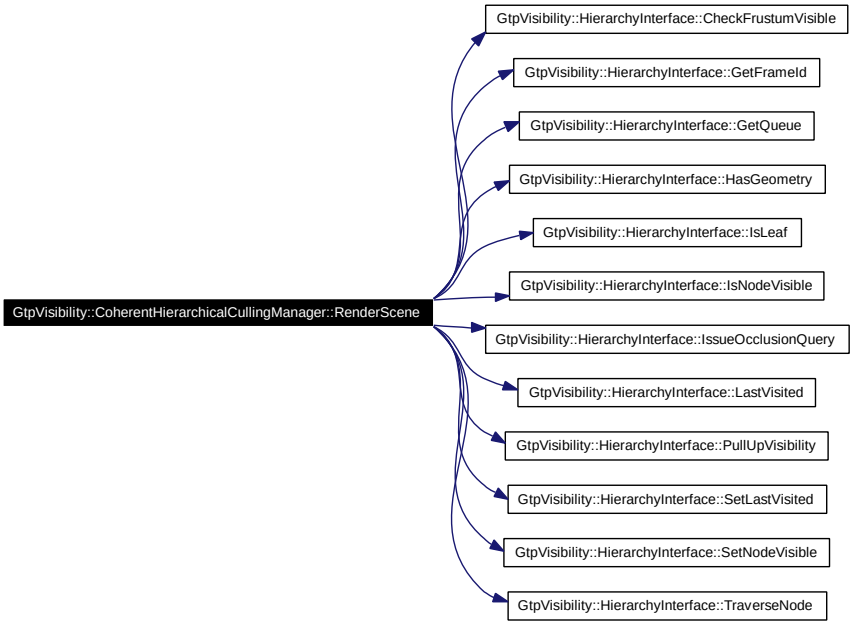
3.12.3 Member Function Documentation

3.12.3.1 void GtpVisibility::CoherentHierarchicalCullingManager::RenderScene () [virtual]

Renders the scene using a specific occlusion culling algorithm, e.g., coherent hierarchical culling or stop and wait.

Implements GtpVisibility::CullingManager.

Here is the call graph for this function:



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

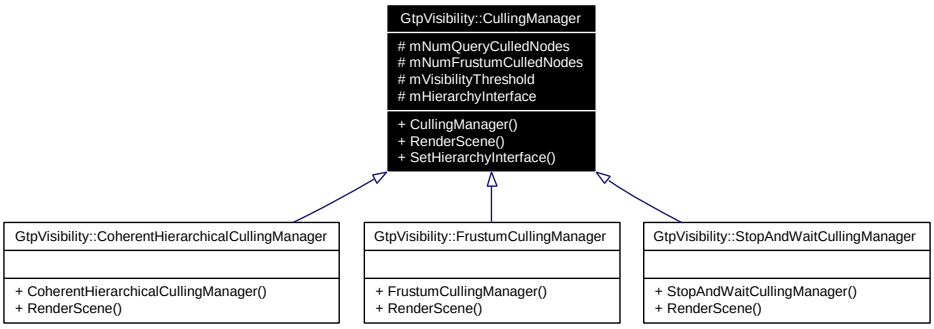
- CoherentHierarchicalCullingManager.h
- CoherentHierarchicalCullingManager.cpp

3.13 GtpVisibility::CullingManager Class Reference

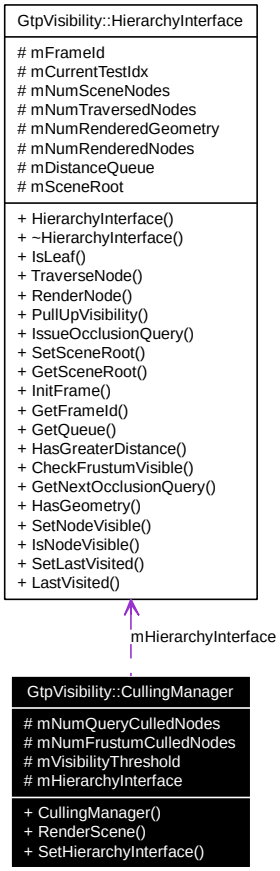
`#include <GtpVisibility/include/CullingManager.h>`

Inherited by GtpVisibility::CoherentHierarchicalCullingManager, GtpVisibility::FrustumCullingManager, and GtpVisibility::StopAndWaitCullingManager.

Inheritance diagram for GtpVisibility::CullingManager:



Collaboration diagram for GtpVisibility::CullingManager:



Public Member Functions

- CullingManager (HierarchyInterface *hierarchyInterface)
- virtual void RenderScene ()=0
- void SetHierarchyInterface (HierarchyInterface *hierarchyInterface)

Protected Attributes

- unsigned int mNumQueryCulledNodes
- unsigned int mNumFrustumCulledNodes
- unsigned int mVisibilityThreshold
- HierarchyInterface * mHierarchyInterface

3.13.1 Detailed Description

This abstract class implements an interface for a specific culling algorithm. The algorithm is either used to render a scene or to make a visibility query.

3.13.2 Constructor & Destructor Documentation

3.13.2.1 GtpVisibility::CullingManager::CullingManager (HierarchyInterface * *hierarchyInterface*)

Constructor taking a scene traverser for a specific type of hierarchy as argument.

3.13.3 Member Function Documentation

3.13.3.1 virtual void GtpVisibility::CullingManager::RenderScene () [pure virtual]

Renders the scene using a specific occlusion culling algorithm, e.g., coherent hierarchical culling or stop and wait.

Implemented in GtpVisibility::CoherentHierarchicalCullingManager, GtpVisibility::FrustumCullingManager, and GtpVisibility::StopAndWaitCullingManager.

3.13.3.2 void GtpVisibility::CullingManager::SetHierarchyInterface (HierarchyInterface * *hierarchyInterface*)

Sets the hierarchy interface.

Parameters:
hierarchyInterface

Remarks:
the hierarchy interface encapsulates the hierarchy we are working on

3.13.4 Member Data Documentation

3.13.4.1 unsigned int GtpVisibility::CullingManager::mNumQueryCulledNodes [protected]

3.13.4.2 unsigned int GtpVisibility::CullingManager::mNumFrustumCulledNodes [protected]

3.13.4.3 unsigned int GtpVisibility::CullingManager::mVisibilityThreshold [protected]

3.13.4.4 HierarchyInterface* GtpVisibility::CullingManager::mHierarchyInterface [protected]

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

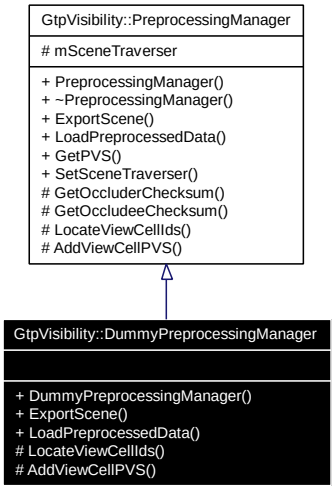
- CullingManager.h
- CullingManager.cpp

3.14 GtpVisibility::DummyPreprocessingManager Class Reference

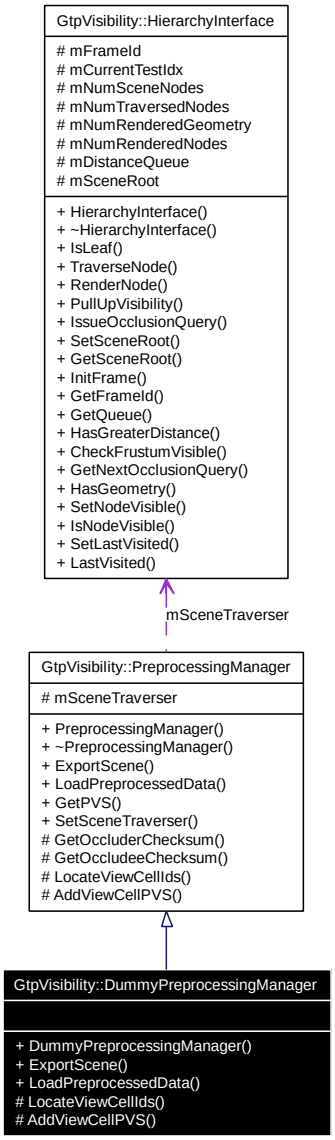
`#include <GtpVisibility/include/DummyPreprocessingManager.h>`

Inherits GtpVisibility::PreprocessingManager.

Inheritance diagram for GtpVisibility::DummyPreprocessingManager:



Collaboration diagram for GtpVisibility::DummyPreprocessingManager:



Public Member Functions

- DummyPreprocessingManager (HierarchyInterface *hierarchyInterface)
- virtual bool ExportScene (const string filename)
- virtual bool LoadPreprocessedData (const string filename)

Protected Member Functions

- virtual bool LocateViewCellIds (const Vector3 &center, const float radius, vector< int > *viewCellIds)
- virtual int AddViewCellPVS (const int cellID, InfoContainer< NodeInfo > *visibleNodes, InfoContainer< MeshInfo > *visibleMeshes)

3.14.1 Detailed Description

Implements a dummy interface to the external visibility preprocessing.

3.14.2 Constructor & Destructor Documentation

3.14.2.1 GtpVisibility::DummyPreprocessingManager::DummyPreprocessingManager (HierarchyInterface * *hierarchyInterface*)

Constructor taking a scene traverser for a specific type of hierarchyInterface as argument.

3.14.3 Member Function Documentation

3.14.3.1 bool GtpVisibility::DummyPreprocessingManager::ExportScene (const string *filename*) [virtual]

Export the scene for visibility preprocessing. Exports the hierarchyInterface including its bounding boxes + mesh data for occluders. The viewcell data will either be supplied directly to the visibility preprocessing standalone module

Implements GtpVisibility::PreprocessingManager.

3.14.3.2 bool GtpVisibility::DummyPreprocessingManager::LoadPreprocessedData (const string *filename*) [virtual]

Load preprocessed visibility information

Implements GtpVisibility::PreprocessingManager.

3.14.3.3 bool GtpVisibility::DummyPreprocessingManager::LocateViewCellIds (const Vector3 & *center*, const float *radius*, vector< int > * *viewCellIds*) [protected, virtual]

Find viewcells intersecting a spherical neighborhood of a point. Returns false if no viewcell was found.

Implements GtpVisibility::PreprocessingManager.

3.14.3.4 int GtpVisibility::DummyPreprocessingManager::AddViewCellPVS (const int *cellID*, InfoContainer< NodeInfo > * *visibleNodes*, InfoContainer< MeshInfo > * *visibleMeshes*) [protected, virtual]

Add a PVS of the given viewcell to the already evaluated PVS by computing a union with visibleNodes and visibleMeshes. Returns the number of added entries.

Implements GtpVisibility::PreprocessingManager.

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

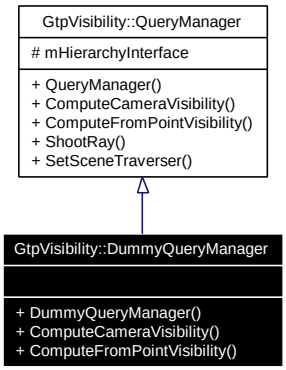
- DummyPreprocessingManager.h
- DummyPreprocessingManager.cpp

3.15 GtpVisibility::DummyQueryManager Class Reference

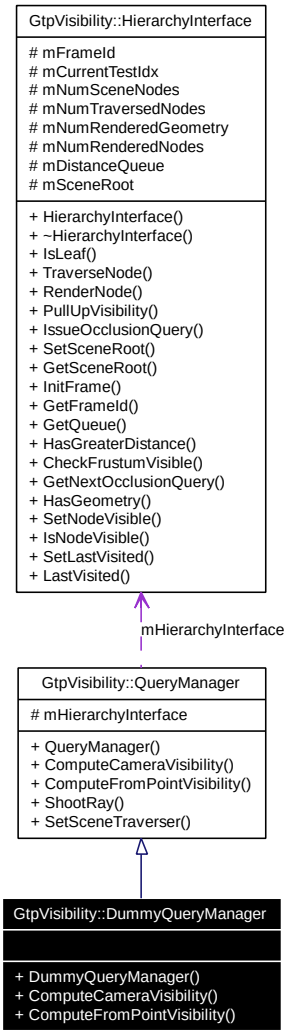
`#include <GtpVisibility/include/DummyQueryManager.h>`

Inherits GtpVisibility::QueryManager.

Inheritance diagram for GtpVisibility::DummyQueryManager:



Collaboration diagram for GtpVisibility::DummyQueryManager:



Public Member Functions

- DummyQueryManager (HierarchyInterface *hierarchyInterface)
- virtual void ComputeCameraVisibility (const Camera &camera, InfoContainer< NodeInfo > *visibleNodes, InfoContainer< MeshInfo > *visibleGeometry, bool relativeVisibility=false)
- virtual void ComputeFromPointVisibility (const Vector3 &point, InfoContainer< NodeInfo > *visibleNodes, InfoContainer< MeshInfo > *visibleGeometry, bool relativeVisibility=false)

3.15.1 Detailed Description

This implements dummy visibility queries. The queries return only the root of the hierarchy as visible node.

3.15.2 Constructor & Destructor Documentation

3.15.2.1 GtpVisibility::DummyQueryManager::DummyQueryManager (HierarchyInterface * *hierarchyInterface*) [inline]

Constructor taking a scene traverser for a specific type of hierarchyInterface as argument.

3.15.3 Member Function Documentation

3.15.3.1 void GtpVisibility::DummyQueryManager::ComputeCameraVisibility (const Camera & *camera*, InfoContainer< NodeInfo > * *visibleNodes*, InfoContainer< MeshInfo > * *visibleGeometry*, bool *relativeVisibility* = false) [virtual]

See also:
QueryManager::ComputeCameraVisibility()

Implements GtpVisibility::QueryManager.

Here is the call graph for this function:

3.15.3.2 void GtpVisibility::DummyQueryManager::ComputeFromPointVisibility (const Vector3 & *point*, InfoContainer< NodeInfo > * *visibleNodes*, InfoContainer< MeshInfo > * *visibleGeometry*, bool *relativeVisibility* = false) [virtual]

See also:
QueryManager::ComputeFromPointVisibility()

Implements GtpVisibility::QueryManager.

Here is the call graph for this function:

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

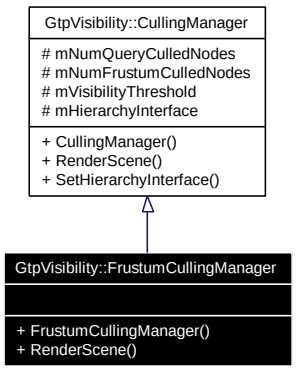
- DummyQueryManager.h
- DummyQueryManager.cpp

3.16 GtpVisibility::FrustumCullingManager Class Reference

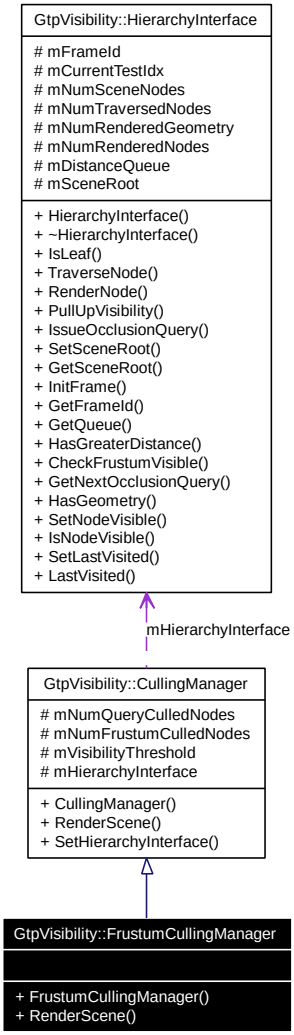
`#include <GtpVisibility/include/FrustumCullingManager.h>`

Inherits GtpVisibility::CullingManager.

Inheritance diagram for GtpVisibility::FrustumCullingManager:



Collaboration diagram for GtpVisibility::FrustumCullingManager:



Public Member Functions

- FrustumCullingManager (HierarchyInterface *hierarchyInterface)
- void RenderScene ()

3.16.1 Detailed Description

Renders the scene, applies only view frustum culling.

3.16.2 Constructor & Destructor Documentation

3.16.2.1 GtpVisibility::FrustumCullingManager::FrustumCullingManager (HierarchyInterface * *hierarchyInterface*)

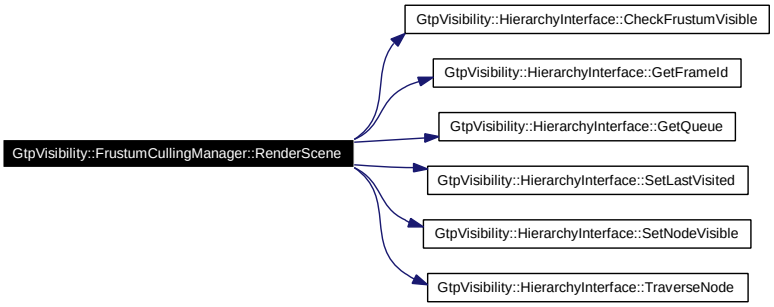
3.16.3 Member Function Documentation

3.16.3.1 void GtpVisibility::FrustumCullingManager::RenderScene () [virtual]

Renders the scene using a specific occlusion culling algorithm, e.g., coherent hierarchical culling or stop and wait.

Implements GtpVisibility::CullingManager.

Here is the call graph for this function:



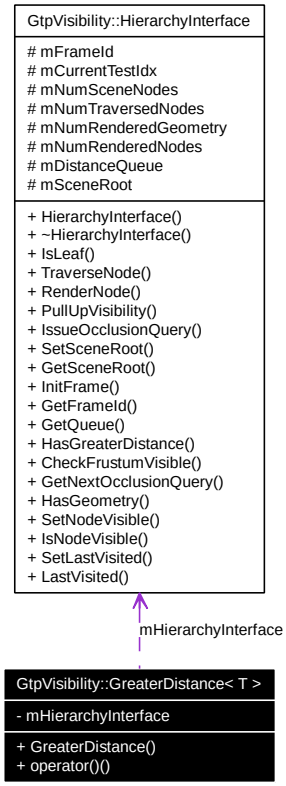
The documentation for this class was generated from the following files:

- FrustumCullingManager.h
- FrustumCullingManager.cpp

3.17 GtpVisibility::GreaterDistance< T > Class Template Reference

`#include <GtpVisibility/include/DistanceQueue.h>`

Collaboration diagram for GtpVisibility::GreaterDistance< T >:



Public Member Functions

- GreaterDistance (HierarchyInterface *hierarchyInterface)
- bool operator() (T v1, T v2) const

Private Attributes

- HierarchyInterface * mHierarchyInterface

3.17.1 Detailed Description

template<typename T> class GtpVisibility::GreaterDistance< T >

This class implements the less operator for the priority queue, i.e., a greater distance has a lower priority in the queue.

3.17.2 Constructor & Destructor Documentation

3.17.2.1 template<typename T> GtpVisibility::GreaterDistance< T >::GreaterDistance (HierarchyInterface * *hierarchyInterface*) [inline]

3.17.3 Member Function Documentation

3.17.3.1 template<typename T> bool GtpVisibility::GreaterDistance< T >::operator() (T *v1*, T *v2*) const [inline]

Here is the call graph for this function:

3.17.4 Member Data Documentation

3.17.4.1 template<typename T> HierarchyInterface* GtpVisibility::GreaterDistance< T >::mHierarchyInterface [private]

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

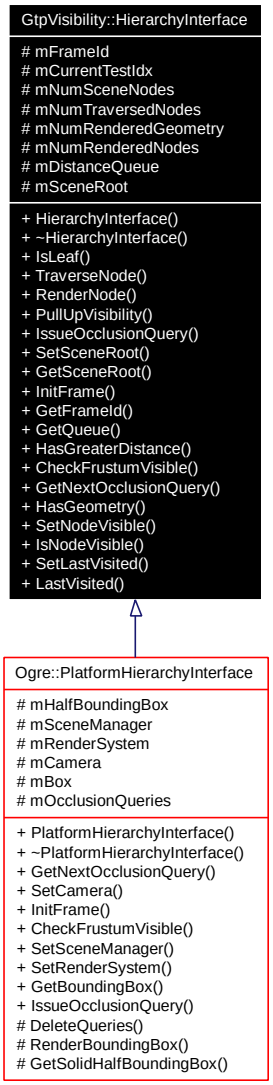
- DistanceQueue.h

3.18 GtpVisibility::HierarchyInterface Class Reference

`#include <GtpVisibility/include/HierarchyInterface.h>`

Inherited by Ogre::PlatformHierarchyInterface.

Inheritance diagram for GtpVisibility::HierarchyInterface:



Public Member Functions

- HierarchyInterface ()
- virtual ~HierarchyInterface ()
- virtual bool IsLeaf (HierarchyNode *node)=0
- virtual void TraverseNode (HierarchyNode *node)=0
- virtual void RenderNode (HierarchyNode *node)=0
- virtual void PullUpVisibility (HierarchyNode *node)=0
- virtual OcclusionQuery * IssueOcclusionQuery (HierarchyNode *node)=0

- void SetSceneRoot (HierarchyNode *root)
- HierarchyNode * GetSceneRoot () const
- void InitFrame (HierarchyNode *root)
- int GetFrameId ()
- DistanceQueue * GetQueue ()
- virtual bool HasGreaterDistance (HierarchyNode *node1, HierarchyNode *node2)=0
- virtual bool CheckFrustumVisible (HierarchyNode *node, bool &intersects)=0
- virtual OcclusionQuery * GetNextOcclusionQuery ()=0
- virtual bool HasGeometry (HierarchyNode *node)=0
- virtual void SetNodeVisible (HierarchyNode *node, const bool visible)=0
- virtual bool IsNodeVisible (HierarchyNode *node)=0
- virtual void SetLastVisited (HierarchyNode *node, const int frameId)=0
- virtual int LastVisited (HierarchyNode *node)=0

Protected Attributes

- unsigned int mFrameId
- int mCurrentTestIdx
- unsigned int mNumSceneNodes
- unsigned int mNumTraversedNodes
- unsigned int mNumRenderedGeometry
- unsigned int mNumRenderedNodes
- DistanceQueue * mDistanceQueue
- HierarchyNode * mSceneRoot

3.18.1 Detailed Description

Class which implements a hierarchy interface for a scene hierarchy.

3.18.2 Constructor & Destructor Documentation

3.18.2.1 GtpVisibility::HierarchyInterface::HierarchyInterface ()

Default constructor.

3.18.2.2 GtpVisibility::HierarchyInterface::~HierarchyInterface () [virtual]

3.18.3 Member Function Documentation

3.18.3.1 virtual bool GtpVisibility::HierarchyInterface::IsLeaf (HierarchyNode * *node*) [pure virtual]

Returns true if current node is leaf of the hierarchy.

Parameters:

node hierarchy node

Returns:

true if node is leaf

3.18.3.2 virtual void GtpVisibility::HierarchyInterface::TraverseNode (HierarchyNode * *node*) [pure virtual]

Traverses the given node.

Parameters:
node the hierarchy node

3.18.3.3 virtual void GtpVisibility::HierarchyInterface::RenderNode (HierarchyNode * *node*) [pure virtual]

Renders current scene node .

Parameters:
node current scene node to be rendered

3.18.3.4 virtual void GtpVisibility::HierarchyInterface::PullUpVisibility (HierarchyNode * *node*) [pure virtual]

Pulls up the visibility from the current node recursively to the parent nodes.

Parameters:
node the current node

3.18.3.5 virtual OcclusionQuery* GtpVisibility::HierarchyInterface::IssueOcclusionQuery (HierarchyNode * *node*) [pure virtual]

Issue a occlusion query for this node.

Parameters:
node the current hierarchy node

Returns:
occlusion query for this node

3.18.3.6 void GtpVisibility::HierarchyInterface::SetSceneRoot (HierarchyNode * *root*)

Sets the root of the scene hierarchy.

Parameters:
root the hierarchy root

3.18.3.7 HierarchyNode* GtpVisibility::HierarchyInterface::GetSceneRoot () const [inline]

Get the root of the scene hierarchy.

Returns:
the hierarchy root

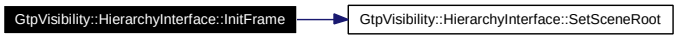
3.18.3.8 void GtpVisibility::HierarchyInterface::InitFrame (HierarchyNode * *root*)

Sets the scene root and initialises this scene traverser for a traversal.

Parameters:
root current scene root

Remarks:
initialises some parameters, and also the statistics.

Here is the call graph for this function:



3.18.3.9 int GtpVisibility::HierarchyInterface::GetFrameId ()

Returns current frame id.

Returns:
frame id

3.18.3.10 DistanceQueue * GtpVisibility::HierarchyInterface::GetQueue ()

Returns the current distance queue.

Returns:
current distance queue

3.18.3.11 virtual bool GtpVisibility::HierarchyInterface::HasGreaterDistance (HierarchyNode * *node1*, HierarchyNode * *node2*) [pure virtual]

Returns true if node 1 has greater distance to the view plane than node 2.

Parameters:
node1 the first node to be compared

node2 the second node to be compared

3.18.3.12 virtual bool GtpVisibility::HierarchyInterface::CheckFrustumVisible (HierarchyNode * *node*, bool & *intersects*) [pure virtual]

Checks if the node is visible from the current view frustum.

Parameters:
node the current node

intersects returns true if the current node intersects the near plane

3.18.3.13 virtual OcclusionQuery* GtpVisibility::HierarchyInterface::GetNextOcclusionQuery () [pure virtual]

Returns next available occlusion query or creates new one.

Returns:
the next occlusion query

Implemented in Ogre::PlatformHierarchyInterface.

3.18.3.14 virtual bool GtpVisibility::HierarchyInterface::HasGeometry (HierarchyNode * *node*) [pure virtual]

Returns true if there is renderable geometry attached to this node

Parameters:
node the current node

Returns:
if the node has renderable geometry

3.18.3.15 virtual void GtpVisibility::HierarchyInterface::SetNodeVisible (HierarchyNode * *node*, const bool *visible*) [pure virtual]

Sets the visible flag for this node.

Parameters:
node the current node

visible the visible flag

3.18.3.16 virtual bool GtpVisibility::HierarchyInterface::IsNodeVisible (HierarchyNode * *node*) [pure virtual]

Returns true if node has the visible flag set. See set

3.18.3.17 virtual void GtpVisibility::HierarchyInterface::SetLastVisited (HierarchyNode * *node*, const int *frameId*) [pure virtual]

Sets the last visited frame id for this node.

Parameters:
node the current node

frameId the current frame id

3.18.3.18 virtual int GtpVisibility::HierarchyInterface::LastVisited (HierarchyNode * *node*) [pure virtual]

Returns frame id when this node was last visited by the traverser. See set

3.18.4 Member Data Documentation

3.18.4.1 unsigned int GtpVisibility::HierarchyInterface::mFrameId [protected]

3.18.4.2 int GtpVisibility::HierarchyInterface::mCurrentTestIdx [protected]

3.18.4.3 unsigned int GtpVisibility::HierarchyInterface::mNumSceneNodes [protected]

3.18.4.4 unsigned int GtpVisibility::HierarchyInterface::mNumTraversedNodes [protected]

3.18.4.5 unsigned int GtpVisibility::HierarchyInterface::mNumRenderedGeometry [protected]

3.18.4.6 unsigned int GtpVisibility::HierarchyInterface::mNumRenderedNodes [protected]

3.18.4.7 DistanceQueue* GtpVisibility::HierarchyInterface::mDistanceQueue [protected]

3.18.4.8 HierarchyNode* GtpVisibility::HierarchyInterface::mSceneRoot [protected]

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

- HierarchyInterface.h
- HierarchyInterface.cpp

3.19 GtpVisibility::MeshInfo Class Reference

`#include <GtpVisibility/include/VisibilityInfo.h>`

Public Member Functions

- MeshInfo (Mesh *mesh, const float v)

Protected Attributes

- Mesh * mMesh
- float mVisibility

3.19.1 Detailed Description

Class storing the visibility information of a mesh.

3.19.2 Constructor & Destructor Documentation

3.19.2.1 GtpVisibility::MeshInfo::MeshInfo (Mesh * *mesh*, const float *v*) [inline]

3.19.3 Member Data Documentation

3.19.3.1 Mesh* GtpVisibility::MeshInfo::mMesh [protected]

pointer to the scene node

3.19.3.2 float GtpVisibility::MeshInfo::mVisibility [protected]

node visibility can either be a number of visible pixels or relative number of visible pixels (if the hardware queries will provide the total number of ratsterized pixels

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

- VisibilityInfo.h

3.20 GtpVisibility::NodeInfo Class Reference

`#include <GtpVisibility/include/VisibilityInfo.h>`

Public Member Functions

- NodeInfo (HierarchyNode *node, const float v)

Protected Attributes

- HierarchyNode * mNode
- float mVisibility

3.20.1 Detailed Description

Class storing the visibility information of a scene node.

3.20.2 Constructor & Destructor Documentation

3.20.2.1 GtpVisibility::NodeInfo::NodeInfo (HierarchyNode * *node*, const float *v*) [inline]

3.20.3 Member Data Documentation

3.20.3.1 HierarchyNode* GtpVisibility::NodeInfo::mNode [protected]

pointer to the scene node

3.20.3.2 float GtpVisibility::NodeInfo::mVisibility [protected]

node visibility can either be a number of visible pixels or relative number of visible pixels (if the hardware queries will provide the total number of ratsterized pixels

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

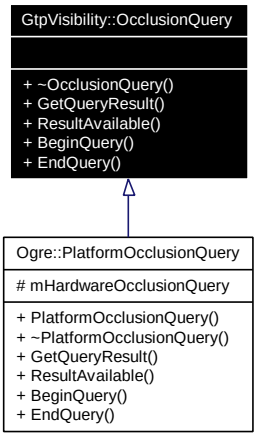
- VisibilityInfo.h

3.21 GtpVisibility::OcclusionQuery Class Reference

`#include <GtpVisibility/include/OcclusionQuery.h>`

Inherited by Ogre::PlatformOcclusionQuery.

Inheritance diagram for GtpVisibility::OcclusionQuery:



Public Member Functions

- virtual ~OcclusionQuery ()
- virtual unsigned int GetQueryResult () const =0
- virtual bool ResultAvailable () const =0
- virtual void BeginQuery () const =0
- virtual void EndQuery () const =0

3.21.1 Detailed Description

This class is an abstract interface for occlusion queries.

3.21.2 Constructor & Destructor Documentation

3.21.2.1 virtual GtpVisibility::OcclusionQuery::~OcclusionQuery () [inline, virtual]

3.21.3 Member Function Documentation

3.21.3.1 virtual unsigned int GtpVisibility::OcclusionQuery::GetQueryResult () const [pure virtual]

Returns the result of an occlusion query in terms of visible pixels.

Returns:
number of visible pixels

Implemented in Ogre::PlatformOcclusionQuery.

3.21.3.2 virtual bool GtpVisibility::OcclusionQuery::ResultAvailable () const [pure virtual]

Returns true if the result of the query is available, false otherwise.

Returns:
if result is available

Implemented in Ogre::PlatformOcclusionQuery.

3.21.3.3 virtual void GtpVisibility::OcclusionQuery::BeginQuery () const [pure virtual]

Begins occlusion query.

Remarks:
the query counts the number of visible pixels between it's begin and end

Implemented in Ogre::PlatformOcclusionQuery.

3.21.3.4 virtual void GtpVisibility::OcclusionQuery::EndQuery () const [pure virtual]

Ends occlusion query.

Implemented in Ogre::PlatformOcclusionQuery.

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

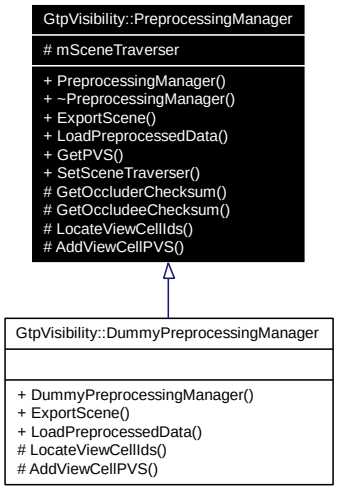
- OcclusionQuery.h

3.22 GtpVisibility::PreprocessingManager Class Reference

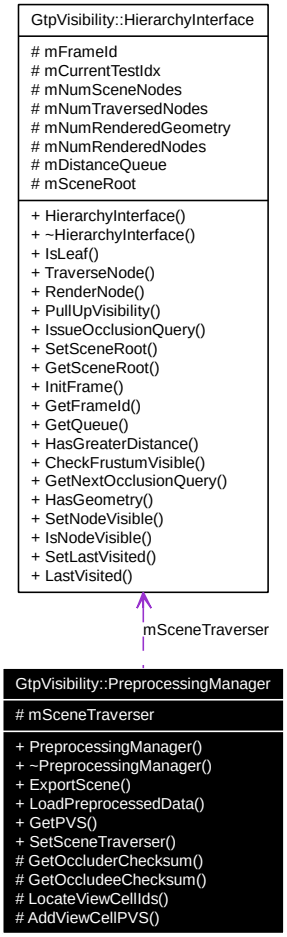
`#include <GtpVisibility/include/PreprocessingManager.h>`

Inherited by GtpVisibility::DummyPreprocessingManager.

Inheritance diagram for GtpVisibility::PreprocessingManager:



Collaboration diagram for GtpVisibility::PreprocessingManager:



Public Member Functions

- PreprocessingManager (HierarchyInterface ∗hierarchyInterface)
- virtual ∼PreprocessingManager ()
- virtual bool ExportScene (const string filename)=0
- virtual bool LoadPreprocessedData (const string filename)=0
- virtual bool GetPVS (const Vector3 &point, const float radius, InfoContainer< NodeInfo > ∗visibleNodes, InfoContainer< MeshInfo > ∗visibleMeshes)
- virtual void SetSceneTraverser (HierarchyInterface ∗hierarchyInterface)

Protected Member Functions

- long GetOccluderChecksum () const
- long GetOccludeeChecksum () const
- virtual bool LocateViewCellIds (const Vector3 &center, const float radius, vector< int > ∗viewCellIds)=0
- virtual int AddViewCellPVS (const int cellID, InfoContainer< NodeInfo > ∗visibleNodes, InfoContainer< MeshInfo > ∗visibleMeshes)=0

Protected Attributes

- HierarchyInterface * mSceneTraverser

3.22.1 Detailed Description

This class defines an interface to the external visibility preprocessor. It allows to export the static part of the scene to a file in the XML format understood by the preprocessor. By default all the exported meshes are considered as scene occluders, whereas occludees are formed by their bounding boxes, and bounding boxes of the scene hierarchy.

This class also allows to import the preprocessed data with PVS information for the viewcells (note that the viewcells are either generated automatically by the preprocessor or loaded from a polyhedral definition stored in a file (see documentation for the Preprocessor class).

3.22.2 Constructor & Destructor Documentation

3.22.2.1 GtpVisibility::PreprocessingManager::PreprocessingManager (HierarchyInterface * *hierarchyInterface*)

Constructor taking a HierarchyInterface as argument. The HierarchyInterface makes the PreprocessingManager independent from the actual scene representation as long as it supports a required set of methods.

3.22.2.2 virtual GtpVisibility::PreprocessingManager::~PreprocessingManager () [virtual]

Destructor which deletes all data describing static scene visibility.

3.22.3 Member Function Documentation

3.22.3.1 virtual bool GtpVisibility::PreprocessingManager::ExportScene (const string *filename*) [pure virtual]

Export the scene for visibility preprocessing. Exports the hierarchyInterface including its bounding boxes + mesh data. Note that the bounding boxes can be used as occludees by the extrenal preprocessor. The viewcell data will either be supplied directly to the visibility preprocessing standalone module.

Parameters:

filename name of the file to export.

Returns:

true if the export was succesful.

Implemented in GtpVisibility::DummyPreprocessingManager.

3.22.3.2 virtual bool GtpVisibility::PreprocessingManager::LoadPreprocessedData (const string *filename*) [pure virtual]

Load preprocessed visibility information. The loaded data is matched with the current scene graph. The topology of the current scene graph has to match with the loaded data. There is also geometrical check of

the bounding boxes. Once loading is succesful the PreprocessingManager establishes links from its internal visibility representation to the scene graph: no change of this part of the scene graph is allowed; this would violate the static scene assumption!

Parameters:

filename name of the file to load.

Returns:

true if the loading was succesful.

Implemented in GtpVisibility::DummyPreprocessingManager.

3.22.3.3 bool GtpVisibility::PreprocessingManager::GetPVS (const Vector3 & *point*, const float *radius*, InfoContainer< NodeInfo > * *visibleNodes*, InfoContainer< MeshInfo > * *visibleMeshes*) [virtual]

Retrieve a PVS corresponding to the given spherical neighborhood of the point. Typically the implementation of this method will firs locate all viewcells intersecting the sphere using efficient logarithmic search enhanced with caching the last query. Then it computes a union of the PVS for the found viewcells.

Parameters:

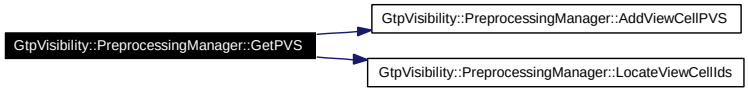
point the center point of the spherical neighborhood

radius the radius of the spherical neighborhood. Note that if radius==0 a more efficient implementation of the method for this special case can be used.

visibleNodes of not NULL a set of visible hierarchy nodes is added to the visibleNodes container. This set is formed of visible leafs or fully visible interior nodes.

visibleMeshes if not NULL the set of visible meshes is added to the container. Returns true if the corresponding PVS exists.

Here is the call graph for this function:



3.22.3.4 virtual void GtpVisibility::PreprocessingManager::SetSceneTraverser (HierarchyInterface * *hierarchyInterface*) [inline, virtual]

Sets the scene traverser.

Remarks:

the scene traverser is dependent on the type of hierarchyInterface the scene consists of.

3.22.3.5 long GtpVisibility::PreprocessingManager::GetOccluderChecksum () const [inline, protected]

Get checksum of the current occluder set in the scene graph. This method is used to check if the preprocessed data matches the current scene graph -> $$ could be moved to VisibilitySceneTraverser

3.22.3.6 long GtpVisibility::PreprocessingManager::GetOccludeeChecksum () const [inline, protected]

Get checksum of the current occludee set in the scene graph. This method is used to check if the preprocessed data matches the current scene graph -> $$ could be moved to VisibilitySceneTraverser

3.22.3.7 virtual bool GtpVisibility::PreprocessingManager::LocateViewCellIds (const Vector3 & *center*, const float *radius*, vector< int > * *viewCellIds*) [protected, pure virtual]

Find viewcells intersecting a spherical neighborhood of a point. Returns false if no viewcell was found.

Implemented in GtpVisibility::DummyPreprocessingManager.

3.22.3.8 virtual int GtpVisibility::PreprocessingManager::AddViewCellPVS (const int *cellID*, InfoContainer< NodeInfo > * *visibleNodes*, InfoContainer< MeshInfo > * *visibleMeshes*) [protected, pure virtual]

Add a PVS of the given viewcell to the already evaluated PVS by computing a union with visibleNodes and visibleMeshes. Returns the number of added entries.

Implemented in GtpVisibility::DummyPreprocessingManager.

3.22.4 Member Data Documentation

3.22.4.1 HierarchyInterface* GtpVisibility::PreprocessingManager::mSceneTraverser [protected]

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

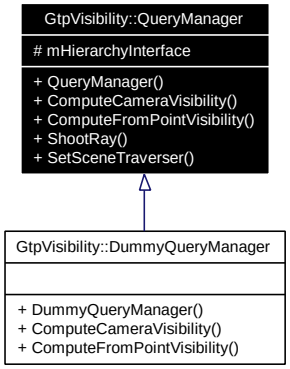
- PreprocessingManager.h
- PreprocessingManager.cpp

3.23 GtpVisibility::QueryManager Class Reference

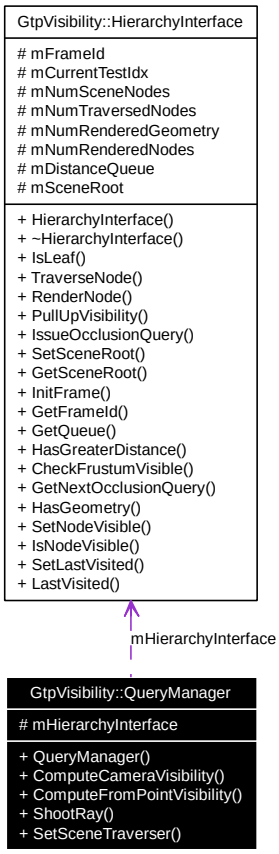
`#include <GtpVisibility/include/QueryManager.h>`

Inherited by GtpVisibility::DummyQueryManager.

Inheritance diagram for GtpVisibility::QueryManager:



Collaboration diagram for GtpVisibility::QueryManager:



Public Member Functions

- QueryManager (HierarchyInterface *hierarchyInterface)
- virtual void ComputeCameraVisibility (const Camera &camera, InfoContainer< NodeInfo > *visibleNodes, InfoContainer< MeshInfo > *visibleGeometry, bool relativeVisibility=false)=0
- virtual void ComputeFromPointVisibility (const Vector3 &point, InfoContainer< NodeInfo > *visibleNodes, InfoContainer< MeshInfo > *visibleGeometry, bool relativeVisibility=false)=0
- virtual bool ShootRay (const Ray &ray, std::vector< Mesh * > *visibleMeshes, bool isGlobal-Line=false)
- void SetSceneTraverser (HierarchyInterface *hierarchyInterface)

Protected Attributes

- HierarchyInterface * mHierarchyInterface

3.23.1 Detailed Description

This abstract class defines interface for a specific visibility query algorithm. The interface supports two from point visibility queries and the ray shooting query. The output of the queries consists of list of visible meshes and hierarchy nodes. The from point queries will be implemented either with item bufferring or based purelly on occlusion queries. Note that the actuall implementation can also exploit the output of the visibility preprocessor.

3.23.2 Constructor & Destructor Documentation

3.23.2.1 GtpVisibility::QueryManager::QueryManager (HierarchyInterface * *hierarchyInterface*)

Constructor taking a hierarchy interface as an argument. This allows to operate onm different hierarchy types, while reusing the implementation of the query methods.

3.23.3 Member Function Documentation

3.23.3.1 virtual void GtpVisibility::QueryManager::ComputeCameraVisibility (const Camera & *camera*, InfoContainer< NodeInfo > * *visibleNodes*, InfoContainer< MeshInfo > * *visibleGeometry*, bool *relativeVisibility* = `false`) `[pure virtual]`

Computes restricted visibility from point by using an explicit camera to execute the visibility query.

Parameters:

camera The camera to be used

visibleNodes Pointer to the container where visible nodes should be added. This set is formed of visible leafs or fully visible interior nodes. If NULL no visible nodes are not evaluated.

visibleGeometry Pointer to the container where visible meshes should be added. If NULL no visible meshes are not evaluated.

relativeVisibility If true the visibility member for NodeInfo and MeshInfo represent relative visibility; i.e. the number of visible pixels divided by the the number of projected pixels.

Returns:

true if the corresponding PVS exists.

Implemented in GtpVisibility::DummyQueryManager.

3.23.3.2 virtual void GtpVisibility::QueryManager::ComputeFromPointVisibility (const Vector3 & *point*, InfoContainer< NodeInfo > * *visibleNodes*, InfoContainer< MeshInfo > * *visibleGeometry*, bool *relativeVisibility* = false) [pure virtual]

Uses the specified point to execute the visibility query in all directions.

See also:
ComputeCameraVisibility()

Implemented in GtpVisibility::DummyQueryManager.

3.23.3.3 bool GtpVisibility::QueryManager::ShootRay (const Ray & *ray*, std::vector< Mesh * > * *visibleMeshes*, bool *isGlobalLine* = false) [virtual]

Ray shooting intreface: finds an intersection with objects in the scene.

Parameters:
ray The given input ray (assuming the ray direction is normalized)

visibleMeshes List of meshes intersecting the ray

isGlobalLine If false only first intersection with opaque object is returned. Otherwise all intersections of the ray with the scene are found.

Returns:
true if there is any intersection.

3.23.3.4 void GtpVisibility::QueryManager::SetSceneTraverser (HierarchyInterface * *hierarchyInterface*)

Sets the scene traverser.

Remarks:
the scene traverser depends on the type of hierarchyInterface the scene consists of.

3.23.4 Member Data Documentation

3.23.4.1 HierarchyInterface* GtpVisibility::QueryManager::mHierarchyInterface [protected]

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

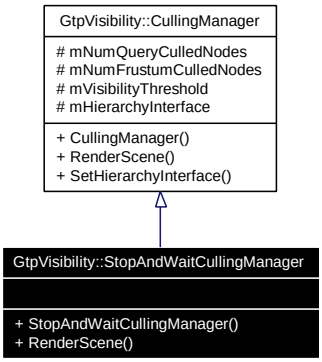
- QueryManager.h
- QueryManager.cpp

3.24 GtpVisibility::StopAndWaitCullingManager Class Reference

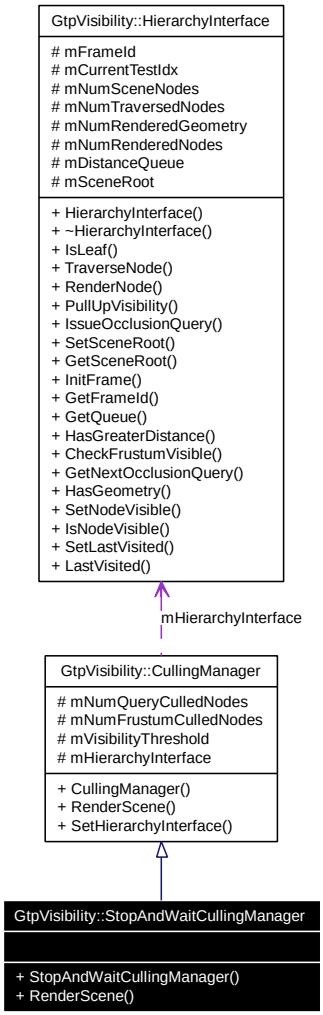
`#include <GtpVisibility/include/StopAndWaitCullingManager.h>`

Inherits GtpVisibility::CullingManager.

Inheritance diagram for GtpVisibility::StopAndWaitCullingManager:



Collaboration diagram for GtpVisibility::StopAndWaitCullingManager:



Public Member Functions

- StopAndWaitCullingManager (HierarchyInterface *hierarchyInterface)
- void RenderScene ()

3.24.1 Detailed Description

Renders the scene with the hierarchical stop and wait algorithm.

3.24.2 Constructor & Destructor Documentation

3.24.2.1 GtpVisibility::StopAndWaitCullingManager::StopAndWaitCullingManager (HierarchyInterface * *hierarchyInterface*)

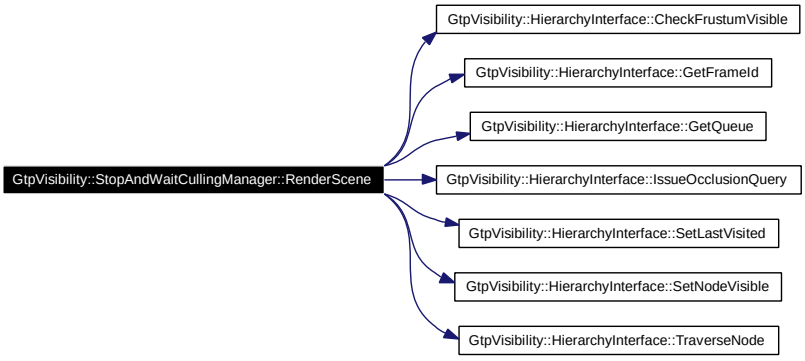
3.24.3 Member Function Documentation

3.24.3.1 void GtpVisibility::StopAndWaitCullingManager::RenderScene () [virtual]

Renders the scene using a specific occlusion culling algorithm, e.g., coherent hierarchical culling or stop and wait.

Implements GtpVisibility::CullingManager.

Here is the call graph for this function:



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

- StopAndWaitCullingManager.h
- StopAndWaitCullingManager.cpp

3.25 GtpVisibility::VisibilityEnvironment Class Reference

`#include <GtpVisibility/include/VisibilityEnvironment.h>`

Public Types

- enum CullingManagerType { FRUSTUM_CULLING, STOP_AND_WAIT, COHERENT_-HIERARCHICAL_CULLING }

Public Member Functions

- VisibilityEnvironment ()
- void LoadEnvironment ()

3.25.1 Detailed Description

This class provides different parameters for the visibility manager.

3.25.2 Member Enumeration Documentation

3.25.2.1 enum GtpVisibility::VisibilityEnvironment::CullingManagerType

Different types of occlusion culling algorithms

Enumeration values:

FRUSTUM_CULLING

STOP_AND_WAIT

COHERENT_HIERARCHICAL_CULLING

3.25.3 Constructor & Destructor Documentation

3.25.3.1 GtpVisibility::VisibilityEnvironment::VisibilityEnvironment ()

3.25.4 Member Function Documentation

3.25.4.1 void GtpVisibility::VisibilityEnvironment::LoadEnvironment ()

Loads an environment from disk.

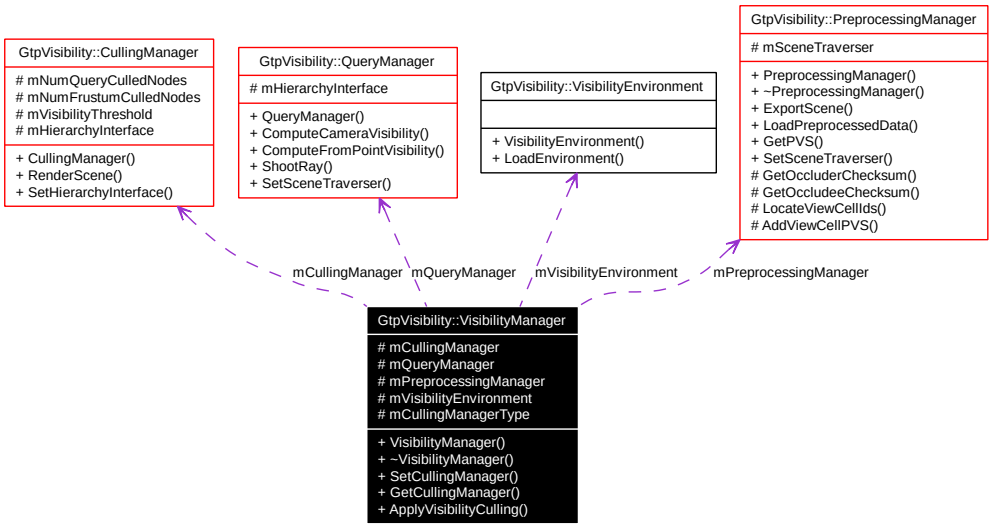
The documentation for this class was generated from the following files:

- VisibilityEnvironment.h
- VisibilityEnvironment.cpp

3.26 GtpVisibility::VisibilityManager Class Reference

`#include <GtpVisibility/include/VisibilityManager.h>`

Collaboration diagram for GtpVisibility::VisibilityManager:



Public Member Functions

- VisibilityManager (VisibilityEnvironment *visEnvironment)
- ~VisibilityManager ()
- void SetCullingManager (VisibilityEnvironment::CullingManagerType ocmType)
- CullingManager * GetCullingManager ()
- void ApplyVisibilityCulling ()

Protected Attributes

- CullingManager * mCullingManager
- QueryManager * mQueryManager
- PreprocessingManager * mPreprocessingManager
- VisibilityEnvironment * mVisibilityEnvironment
- VisibilityEnvironment::CullingManagerType mCullingManagerType

3.26.1 Detailed Description

This class manages all forms of visibility. It is the main class of our visibility module and manages online occlusion culling, offline culling, and visibility queries.

3.26.2 Constructor & Destructor Documentation

3.26.2.1 GtpVisibility::VisibilityManager::VisibilityManager (VisibilityEnvironment * *visEnvironment*)

Constructor taking the visibility environment object as parameter

Parameters:
visEnvironment the visibility environment

3.26.2.2 GtpVisibility::VisibilityManager::~VisibilityManager ()

3.26.3 Member Function Documentation

3.26.3.1 void GtpVisibility::VisibilityManager::SetCullingManager (VisibilityEnvironment::CullingManagerType *ocmType*)

Sets the current online occlusion culling manager, e.g., the stop and wait algorithm or coherent hierarchical culling.

Parameters:
ocmType the online occlusion culling manager type

3.26.3.2 CullingManager * GtpVisibility::VisibilityManager::GetCullingManager ()

Returns the current online occlusion culling manager. See set

3.26.3.3 void GtpVisibility::VisibilityManager::ApplyVisibilityCulling ()

Applies the online visibility culling algorithm on a scene.

Remarks:
the algorithm depends on the current culling manager.

Here is the call graph for this function:

3.26.4 Member Data Documentation

3.26.4.1 CullingManager* GtpVisibility::VisibilityManager::mCullingManager [protected]

3.26.4.2 QueryManager* GtpVisibility::VisibilityManager::mQueryManager [protected]

3.26.4.3 PreprocessingManager* GtpVisibility::VisibilityManager::mPreprocessingManager [protected]

3.26.4.4 VisibilityEnvironment* GtpVisibility::VisibilityManager::mVisibilityEnvironment [protected]

3.26.4.5 VisibilityEnvironment::CullingManagerType GtpVisibility::VisibilityManager::mCullingManagerType [protected]

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

- VisibilityManager.h
- VisibilityManager.cpp

3.27 GtpVisibilityPreprocessor::AxisAlignedBox3 Class Reference

`#include <GtpVisibilityPreprocessor/include/AxisAlignedBox3.h>`

Protected Attributes

- Vector3 min
- Vector3 max

3.27.1 Detailed Description

Axis aligned box

3.27.2 Member Data Documentation

3.27.2.1 Vector3 GtpVisibilityPreprocessor::AxisAlignedBox3::min [protected]

3.27.2.2 Vector3 GtpVisibilityPreprocessor::AxisAlignedBox3::max [protected]

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

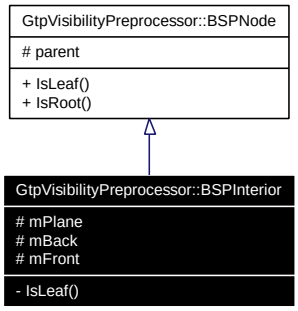
- AxisAlignedBox3.h

3.28 GtpVisibilityPreprocessor::BSPInterior Class Reference

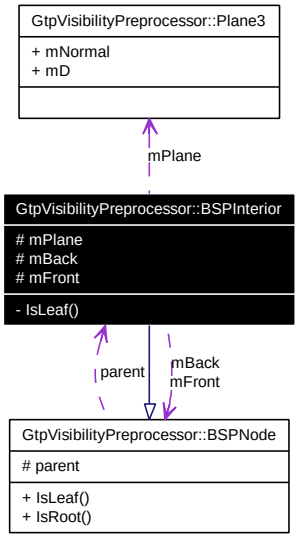
`#include <GtpVisibilityPreprocessor/include/ViewCellBSP.h>`

Inherits GtpVisibilityPreprocessor::BSPNode.

Inheritance diagram for GtpVisibilityPreprocessor::BSPInterior:



Collaboration diagram for GtpVisibilityPreprocessor::BSPInterior:



Protected Attributes

- Plane3 mPlane

  Splitting plane corresponding to this node.

- BSPNode * mBack

  back node

- BSPNode * mFront

  front node

Private Member Functions

- virtual bool IsLeaf () const

3.28.1 Detailed Description

BSP interior node implementation

3.28.2 Member Function Documentation

3.28.2.1 virtual bool GtpVisibilityPreprocessor::BSPInterior::IsLeaf () const [inline, private, virtual]

Returns:
false since it is an interior node

Implements GtpVisibilityPreprocessor::BSPNode.

3.28.3 Member Data Documentation

3.28.3.1 Plane3 GtpVisibilityPreprocessor::BSPInterior::mPlane [protected]

Splitting plane corresponding to this node.

3.28.3.2 BSPNode* GtpVisibilityPreprocessor::BSPInterior::mBack [protected]

back node

3.28.3.3 BSPNode* GtpVisibilityPreprocessor::BSPInterior::mFront [protected]

front node

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

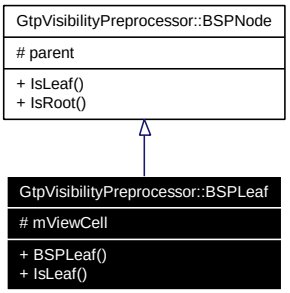
- ViewCellBSP.h

3.29 GtpVisibilityPreprocessor::BSPLeaf Class Reference

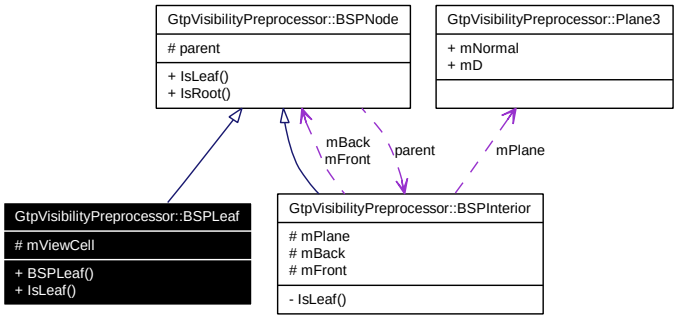
`#include <GtpVisibilityPreprocessor/include/ViewCellBSP.h>`

Inherits GtpVisibilityPreprocessor::BSPNode.

Inheritance diagram for GtpVisibilityPreprocessor::BSPLeaf:



Collaboration diagram for GtpVisibilityPreprocessor::BSPLeaf:



Public Member Functions

- BSPLeaf (ViewCell *viewCell=NULL)
- virtual bool IsLeaf () const

Protected Attributes

- ViewCell * mViewCell

3.29.1 Detailed Description

BSP leaf node implementation

3.29.2 Constructor & Destructor Documentation

3.29.2.1 GtpVisibilityPreprocessor::BSPLeaf::BSPLeaf (ViewCell * *viewCell* = NULL) [inline]

3.29.3 Member Function Documentation

3.29.3.1 virtual bool GtpVisibilityPreprocessor::BSPLeaf::IsLeaf () const [inline, virtual]

Returns:
true since it is an interior node

Implements GtpVisibilityPreprocessor::BSPNode.

3.29.4 Member Data Documentation

3.29.4.1 ViewCell* GtpVisibilityPreprocessor::BSPLeaf::mViewCell [protected]

polygonal representation of this viewcell if NULL this note does not correspond to feasible viewcell

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

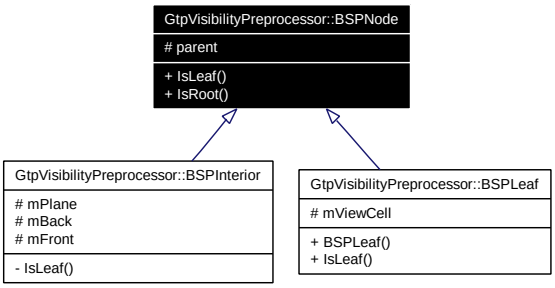
- ViewCellBSP.h

3.30 GtpVisibilityPreprocessor::BSPNode Class Reference

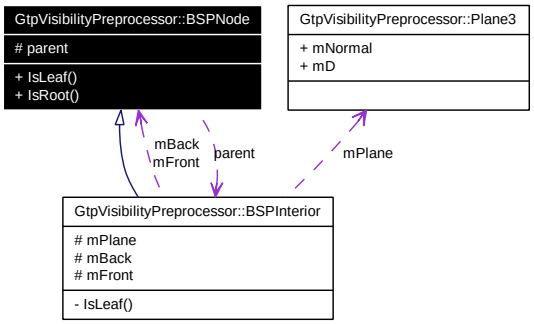
`#include <GtpVisibilityPreprocessor/include/ViewCellBSP.h>`

Inherited by GtpVisibilityPreprocessor::BSPInterior, and GtpVisibilityPreprocessor::BSPLeaf.

Inheritance diagram for GtpVisibilityPreprocessor::BSPNode:



Collaboration diagram for GtpVisibilityPreprocessor::BSPNode:



Public Member Functions

- virtual bool IsLeaf () const =0
- virtual bool IsRoot () const

Protected Attributes

- BSPInterior * parent

  parent of this node

3.30.1 Detailed Description

BSPNode abstract class serving for interior and leaf node implementation

3.30.2 Member Function Documentation

3.30.2.1 virtual bool GtpVisibilityPreprocessor::BSPNode::IsLeaf () const [pure virtual]

Determines whether this node is a leaf or not

Returns:
true if leaf

Implemented in GtpVisibilityPreprocessor::BSPInterior, and GtpVisibilityPreprocessor::BSPLeaf.

3.30.2.2 virtual bool GtpVisibilityPreprocessor::BSPNode::IsRoot () const [inline, virtual]

Determines whether this node is a root

Returns:
true if root

3.30.3 Member Data Documentation

3.30.3.1 BSPInterior* GtpVisibilityPreprocessor::BSPNode::parent [protected]

parent of this node

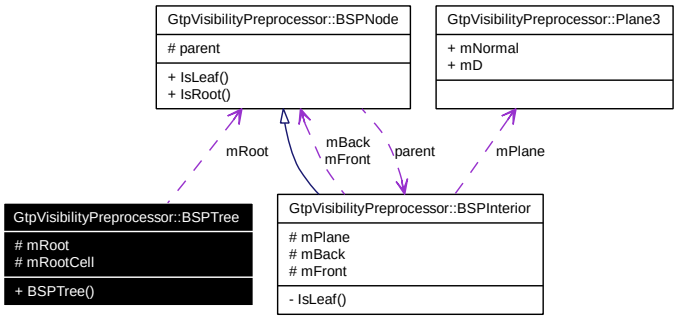
The documentation for this class was generated from the following file:

- ViewCellBSP.h

3.31 GtpVisibilityPreprocessor::BSPTree Class Reference

`#include <GtpVisibilityPreprocessor/include/ViewCellBSP.h>`

Collaboration diagram for GtpVisibilityPreprocessor::BSPTree:



Public Member Functions

- BSPTree (ViewCell ∗cell)

Protected Attributes

- BSPNode ∗ mRoot

  Pointer to the root of the tree.

- ViewCell ∗ mRootCell

  Pointer to the root cell of the viewspace ∗/.

3.31.1 Detailed Description

Implementation of the ViewCell BSP tree

3.31.2 Constructor & Destructor Documentation

3.31.2.1 GtpVisibilityPreprocessor::BSPTree::BSPTree (ViewCell ∗ *cell*) [inline]

Constructor takes a pointer to the cell corresponding to the whole viewspace

3.31.3 Member Data Documentation

3.31.3.1 BSPNode∗ GtpVisibilityPreprocessor::BSPTree::mRoot [protected]

Pointer to the root of the tree.

3.31.3.2 ViewCell* GtpVisibilityPreprocessor::BSPTree::mRootCell [protected]

Pointer to the root cell of the viewspace */.

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

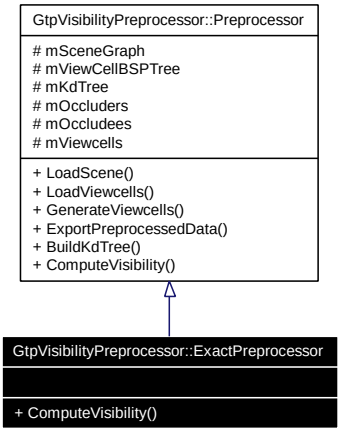
- ViewCellBSP.h

3.32 GtpVisibilityPreprocessor::ExactPreprocessor Class Reference

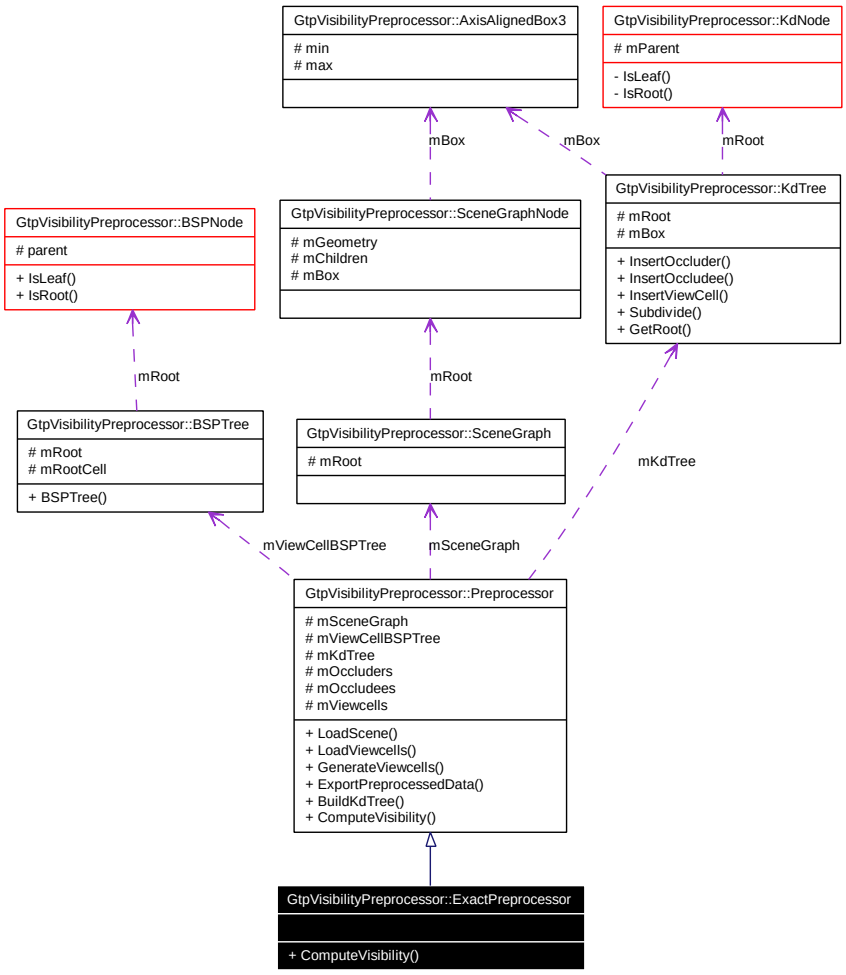
`#include <GtpVisibilityPreprocessor/include/ExactPreprocessor.h>`

Inherits GtpVisibilityPreprocessor::Preprocessor.

Inheritance diagram for GtpVisibilityPreprocessor::ExactPreprocessor:



Collaboration diagram for GtpVisibilityPreprocessor::ExactPreprocessor:



Public Member Functions

- virtual bool ComputeVisibility ()

3.32.1 Detailed Description

Exact visibility preprocessing. The implementation is based on 6D BSP and Plucker coordinates

3.32.2 Member Function Documentation

3.32.2.1 bool GtpVisibilityPreprocessor::ExactPreprocessor::ComputeVisibility () [virtual]

Compute visibility method. This method has to be reimplemented by the actual Preprocessor implementation (e.g. SamplingPreprocessor, ExactPreprocessor, GlobalSamplingpreprocessor)

Implements GtpVisibilityPreprocessor::Preprocessor.

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

- ExactPreprocessor.h

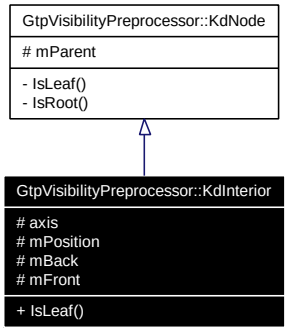
- ExactPreprocessor.cpp

3.33 GtpVisibilityPreprocessor::KdInterior Class Reference

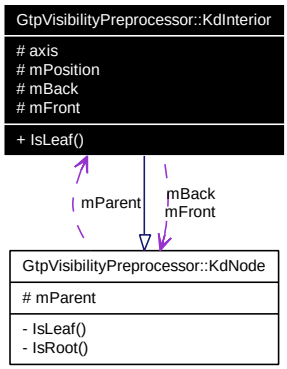
`#include <GtpVisibilityPreprocessor/include/KdTree.h>`

Inherits GtpVisibilityPreprocessor::KdNode.

Inheritance diagram for GtpVisibilityPreprocessor::KdInterior:



Collaboration diagram for GtpVisibilityPreprocessor::KdInterior:



Public Member Functions

- virtual bool IsLeaf () const

Protected Attributes

- int axis
- float mPosition
- KdNode * mBack
- KdNode * mFront

3.33.1 Detailed Description

Implementation of the kd-tree interior node

3.33.2 Member Function Documentation

3.33.2.1 virtual bool GtpVisibilityPreprocessor::KdInterior::IsLeaf () const [inline, virtual]

See also:
KdNode::IsLeaf()

Implements GtpVisibilityPreprocessor::KdNode.

3.33.3 Member Data Documentation

3.33.3.1 int GtpVisibilityPreprocessor::KdInterior::axis [protected]

splitting axis

3.33.3.2 float GtpVisibilityPreprocessor::KdInterior::mPosition [protected]

splitting position, absolute position within the bounding box of this node

3.33.3.3 KdNode* GtpVisibilityPreprocessor::KdInterior::mBack [protected]

back node

3.33.3.4 KdNode* GtpVisibilityPreprocessor::KdInterior::mFront [protected]

front node

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

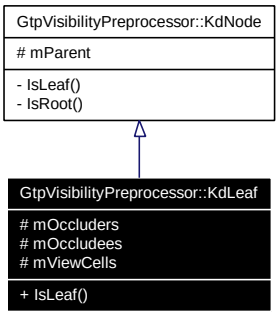
- KdTree.h

3.34 GtpVisibilityPreprocessor::KdLeaf Class Reference

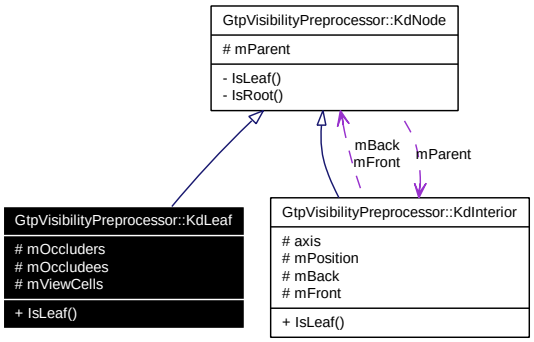
`#include <GtpVisibilityPreprocessor/include/KdTree.h>`

Inherits GtpVisibilityPreprocessor::KdNode.

Inheritance diagram for GtpVisibilityPreprocessor::KdLeaf:



Collaboration diagram for GtpVisibilityPreprocessor::KdLeaf:



Public Member Functions

- virtual bool IsLeaf () const

Protected Attributes

- MeshContainer mOccluders
- MeshContainer mOccludees
- ViewCellContainer mViewCells

3.34.1 Detailed Description

Implementation of the kd-tree leaf node

3.34.2 Member Function Documentation

3.34.2.1 virtual bool GtpVisibilityPreprocessor::KdLeaf::IsLeaf () const [inline, virtual]

See also:
KdNode::IsLeaf()

Implements GtpVisibilityPreprocessor::KdNode.

3.34.3 Member Data Documentation

3.34.3.1 MeshContainer GtpVisibilityPreprocessor::KdLeaf::mOccluders [protected]

pointers to occluders contained in this node

3.34.3.2 MeshContainer GtpVisibilityPreprocessor::KdLeaf::mOccludees [protected]

pointers to occludees contained in this node

3.34.3.3 ViewCellContainer GtpVisibilityPreprocessor::KdLeaf::mViewCells [protected]

pointers to viewcells contained in this node

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

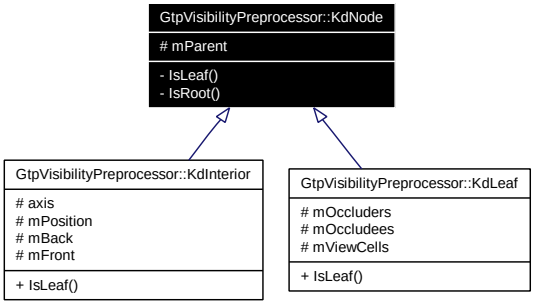
- KdTree.h

3.35 GtpVisibilityPreprocessor::KdNode Class Reference

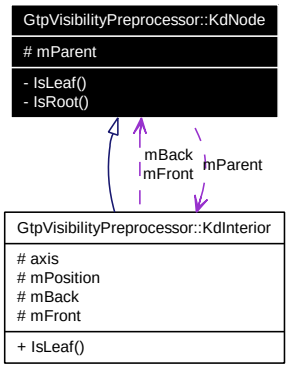
`#include <GtpVisibilityPreprocessor/include/KdTree.h>`

Inherited by GtpVisibilityPreprocessor::KdInterior, and GtpVisibilityPreprocessor::KdLeaf.

Inheritance diagram for GtpVisibilityPreprocessor::KdNode:



Collaboration diagram for GtpVisibilityPreprocessor::KdNode:



Protected Attributes

- KdInterior * mParent

Private Member Functions

- virtual bool IsLeaf () const =0
- virtual bool IsRoot () const

3.35.1 Detailed Description

Abstract class for kd-tree node

3.35.2 Member Function Documentation

3.35.2.1 virtual bool GtpVisibilityPreprocessor::KdNode::IsLeaf () const [private, pure virtual]

Determines whether this node is a leaf or interior node

Returns:
true if leaf

Implemented in GtpVisibilityPreprocessor::KdInterior, and GtpVisibilityPreprocessor::KdLeaf.

3.35.2.2 virtual bool GtpVisibilityPreprocessor::KdNode::IsRoot () const [inline, private, virtual]

Determines whether this node is the root of the tree

Returns:
true if root

3.35.3 Member Data Documentation

3.35.3.1 KdInterior* GtpVisibilityPreprocessor::KdNode::mParent [protected]

Parent of the node - the parent is a little overhead for maintanance of the tree, but allows various optimizations of tree traversal algorithms

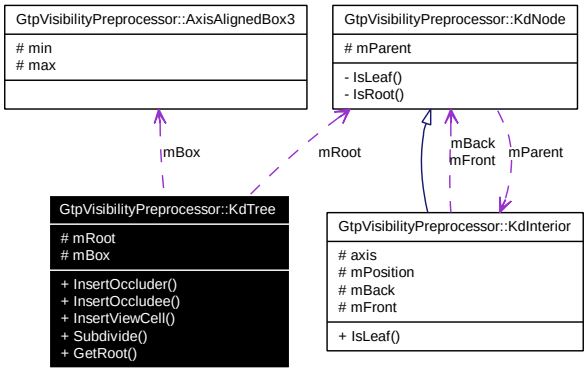
The documentation for this class was generated from the following file:

- KdTree.h

3.36 GtpVisibilityPreprocessor::KdTree Class Reference

`#include <GtpVisibilityPreprocessor/include/KdTree.h>`

Collaboration diagram for GtpVisibilityPreprocessor::KdTree:



Public Member Functions

- virtual void InsertOccluder (Mesh ∗occluder)
- virtual void InsertOccludee (Mesh ∗occludee)
- virtual void InsertViewCell (ViewCell ∗viewCell)
- virtual bool Subdivide (KdNode ∗subtree)
- KdNode ∗ GetRoot () const

Protected Attributes

- KdNode ∗ mRoot

  root of the tree

- AxisAlignedBox3 mBox

  bounding box of the tree root

3.36.1 Detailed Description

KdTree for indexing scene entities - occluders/occludees/viewcells

3.36.2 Member Function Documentation

3.36.2.1 virtual void GtpVisibilityPreprocessor::KdTree::InsertOccluder (Mesh ∗ *occluder*) [inline, virtual]

Insert occluder into the tree

3.36.2.2 virtual void GtpVisibilityPreprocessor::KdTree::InsertOccludee (Mesh * *occludee*) [inline, virtual]

Insert occludee into the tree

3.36.2.3 virtual void GtpVisibilityPreprocessor::KdTree::InsertViewCell (ViewCell * *viewCell*) [inline, virtual]

Insert view cell into the tree

3.36.2.4 bool GtpVisibilityPreprocessor::KdTree::Subdivide (KdNode * *subtree*) [virtual]

Check whether subdivision criteria are met for the given subtree. If not subdivide the leafs of the subtree. The criteria are specified in the environment as well as the subdivision method. By default surface area heuristics is used.

Parameters:

subtree root of the subtree

Returns:

true if subdivision was performed, false if subdivision criteria were already met

3.36.2.5 KdNode* GtpVisibilityPreprocessor::KdTree::GetRoot () const [inline]

Get the root of the tree

3.36.3 Member Data Documentation

3.36.3.1 KdNode* GtpVisibilityPreprocessor::KdTree::mRoot [protected]

root of the tree

3.36.3.2 AxisAlignedBox3 GtpVisibilityPreprocessor::KdTree::mBox [protected]

bounding box of the tree root

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

- KdTree.h
- KdTree.cpp

3.37 GtpVisibilityPreprocessor::Mesh Class Reference

`#include <GtpVisibilityPreprocessor/include/Mesh.h>`

Public Types

- typedef std::vector< Vector3 > VertexContainer
  default vertex container for Mesh
- typedef std::vector< Patch * > PatchContainer
  default patch container for Mesh

Public Member Functions

- Mesh ()
  Default constructor.
- Mesh (const int vertices, const int patches)
  Constructor with container preallocation.

Protected Attributes

- VertexContainer mVertices
- PatchContainer mPatches

3.37.1 Detailed Description

Mesh containing polygonal patches

3.37.2 Member Typedef Documentation

3.37.2.1 typedef std::vector<Vector3> GtpVisibilityPreprocessor::Mesh::VertexContainer

default vertex container for Mesh

3.37.2.2 typedef std::vector<Patch *> GtpVisibilityPreprocessor::Mesh::PatchContainer

default patch container for Mesh

3.37.3 Constructor & Destructor Documentation

3.37.3.1 GtpVisibilityPreprocessor::Mesh::Mesh () [inline]

Default constructor.

3.37.3.2 GtpVisibilityPreprocessor::Mesh::Mesh (const int *vertices*, const int *patches*) [inline]

Constructor with container preallocation.

3.37.4 Member Data Documentation

3.37.4.1 VertexContainer GtpVisibilityPreprocessor::Mesh::mVertices [protected]

Vertices forming the mesh

3.37.4.2 PatchContainer GtpVisibilityPreprocessor::Mesh::mPatches [protected]

Patches forming the mesh

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

- Mesh.h

3.38 GtpVisibilityPreprocessor::Patch Class Reference

`#include <GtpVisibilityPreprocessor/include/Mesh.h>`

Public Member Functions

- Patch ()

Protected Attributes

- vector< Vector3 * > mVertices
  list of vertex pointers

3.38.1 Detailed Description

Patch used as an element of the mesh

3.38.2 Constructor & Destructor Documentation

3.38.2.1 GtpVisibilityPreprocessor::Patch::Patch () [inline]

3.38.3 Member Data Documentation

3.38.3.1 vector<Vector3 *> GtpVisibilityPreprocessor::Patch::mVertices [protected]

list of vertex pointers

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

- Mesh.h

3.39 GtpVisibilityPreprocessor::Plane3 Class Reference

`#include <GtpVisibilityPreprocessor/include/Plane3.h>`

Public Attributes

- Vector3 mNormal
- float mD

3.39.1 Detailed Description

3D Plane

3.39.2 Member Data Documentation

3.39.2.1 Vector3 GtpVisibilityPreprocessor::Plane3::mNormal

3.39.2.2 float GtpVisibilityPreprocessor::Plane3::mD

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

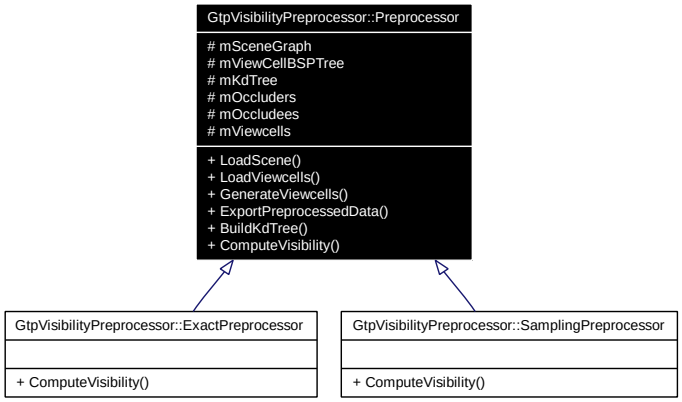
- Plane3.h

3.40 GtpVisibilityPreprocessor::Preprocessor Class Reference

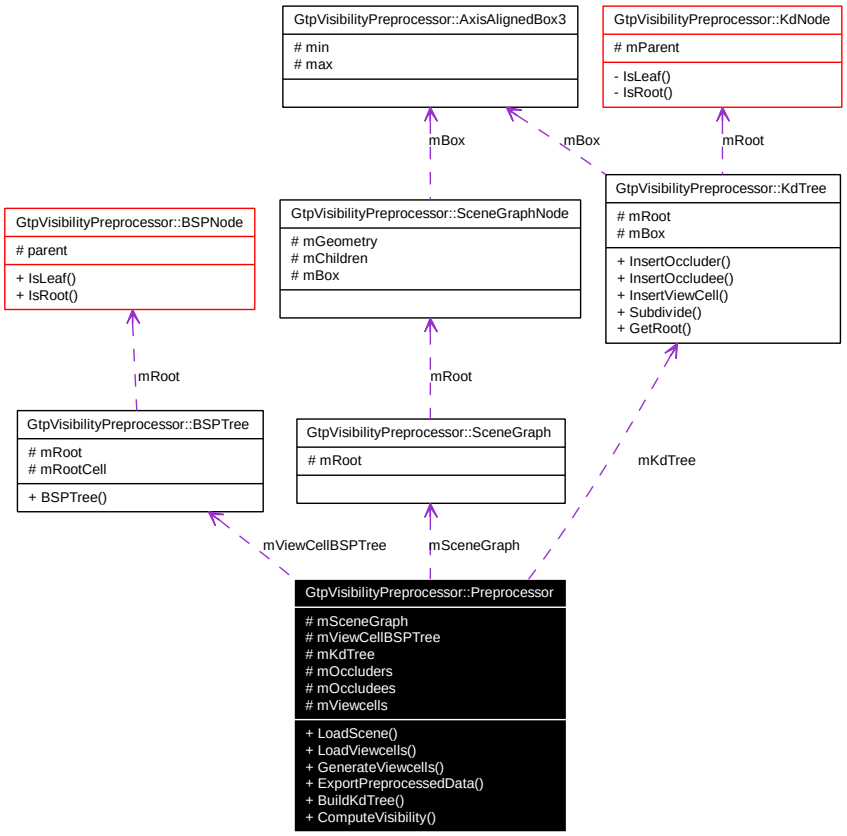
`#include <GtpVisibilityPreprocessor/include/Preprocessor.h>`

Inherited by GtpVisibilityPreprocessor::ExactPreprocessor, and GtpVisibilityPreprocessor::SamplingPreprocessor.

Inheritance diagram for GtpVisibilityPreprocessor::Preprocessor:



Collaboration diagram for GtpVisibilityPreprocessor::Preprocessor:



Public Member Functions

- virtual bool LoadScene (const string filename)
- virtual bool LoadViewcells (const string filename)
- virtual bool GenerateViewcells ()
- virtual bool ExportPreprocessedData (const string filename)
- virtual bool BuildKdTree ()
- virtual bool ComputeVisibility ()=0

Protected Attributes

- SceneGraph * mSceneGraph

  scene graph loaded from file

- BSPTree * mViewCellBSPTree

  BSP tree representing the viewcells.

- KdTree * mKdTree

  kD-tree organizing the scene graph (occluders + occludees) + viewcells

- MeshContainer mOccluders

  list of all loaded occluders

- MeshContainer mOccludees

  list of all loaded occludees

- ViewCellContainer mViewcells

  list of all loaded/generated viewcells

3.40.1 Detailed Description

Main class of the visibility preprocessor. Responsible for loading and saving of the input and output files. Initiates construction of the kD-tree, viewcell loading/generation and the visibility computation itself.

3.40.2 Member Function Documentation

3.40.2.1 bool GtpVisibilityPreprocessor::Preprocessor::LoadScene (const string *filename*) [virtual]

Load the input scene.

Parameters:

filename file to load

Returns:

true on success

3.40.2.2 bool GtpVisibilityPreprocessor::Preprocessor::LoadViewcells (const string *filename*) [virtual]

Load the input viewcells. The input viewcells should be given as a collection of meshes. Each mesh is assume to form a bounded polyhedron defining the interior of the viewcell. The method then builds a BSP tree of these view cells.

Parameters:
filename file to load

Returns:
true on success

3.40.2.3 bool GtpVisibilityPreprocessor::Preprocessor::GenerateViewcells () [virtual]

Generate the viewCells automatically. The particular algorithm to be used depends on the environment setting. Initially the generated viewcells will cover the whole bounding volume of the scene. They can be pruned later depending on the results of visibility computations.

Returns:
true on successful viewcell generation.

3.40.2.4 bool GtpVisibilityPreprocessor::Preprocessor::ExportPreprocessedData (const string *filename*) [virtual]

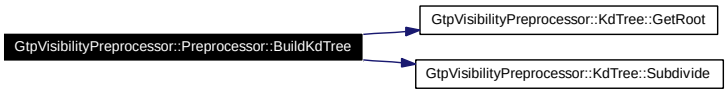
Export all preprocessed data in a XML format understandable by the PreprocessingInterface of the GtpVisibilityPreprocessor Module. The file can be compressed depending on the environement settings.

Returns:
true on successful export

3.40.2.5 bool GtpVisibilityPreprocessor::Preprocessor::BuildKdTree () [virtual]

Build the KdTree of currently loaded occluders/occludees/viewcells. The construction is driven by the environment settings, which also sais which of the three types of entities should be used to drive the heuristical construction (only occluders by default)

Here is the call graph for this function:



3.40.2.6 virtual bool GtpVisibilityPreprocessor::Preprocessor::ComputeVisibility () [pure virtual]

Compute visibility method. This method has to be reimplemented by the actual Preprocessor implementation (e.g. SamplingPreprocessor, ExactPreprocessor, GlobalSamplingpreprocessor)

Implemented in GtpVisibilityPreprocessor::ExactPreprocessor, and GtpVisibilityPreprocessor::SamplingPreprocessor.

3.40.3 Member Data Documentation

3.40.3.1 SceneGraph* GtpVisibilityPreprocessor::Preprocessor::mSceneGraph [protected]

scene graph loaded from file

3.40.3.2 BSPTree* GtpVisibilityPreprocessor::Preprocessor::mViewCellBSPTree [protected]

BSP tree representing the viewcells.

3.40.3.3 KdTree* GtpVisibilityPreprocessor::Preprocessor::mKdTree [protected]

kD-tree organizing the scene graph (occluders + occludees) + viewcells

3.40.3.4 MeshContainer GtpVisibilityPreprocessor::Preprocessor::mOccluders [protected]

list of all loaded occluders

3.40.3.5 MeshContainer GtpVisibilityPreprocessor::Preprocessor::mOccludees [protected]

list of all loaded occludees

3.40.3.6 ViewCellContainer GtpVisibilityPreprocessor::Preprocessor::mViewcells [protected]

list of all loaded/generated viewcells

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

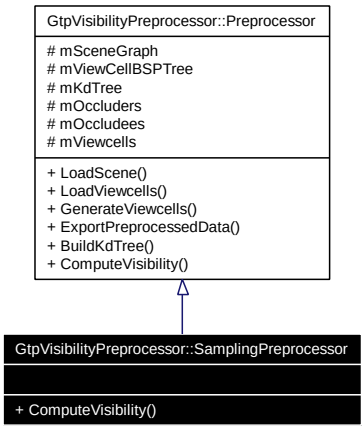
- Preprocessor.h
- Preprocessor.cpp

3.41 GtpVisibilityPreprocessor::SamplingPreprocessor Class Reference

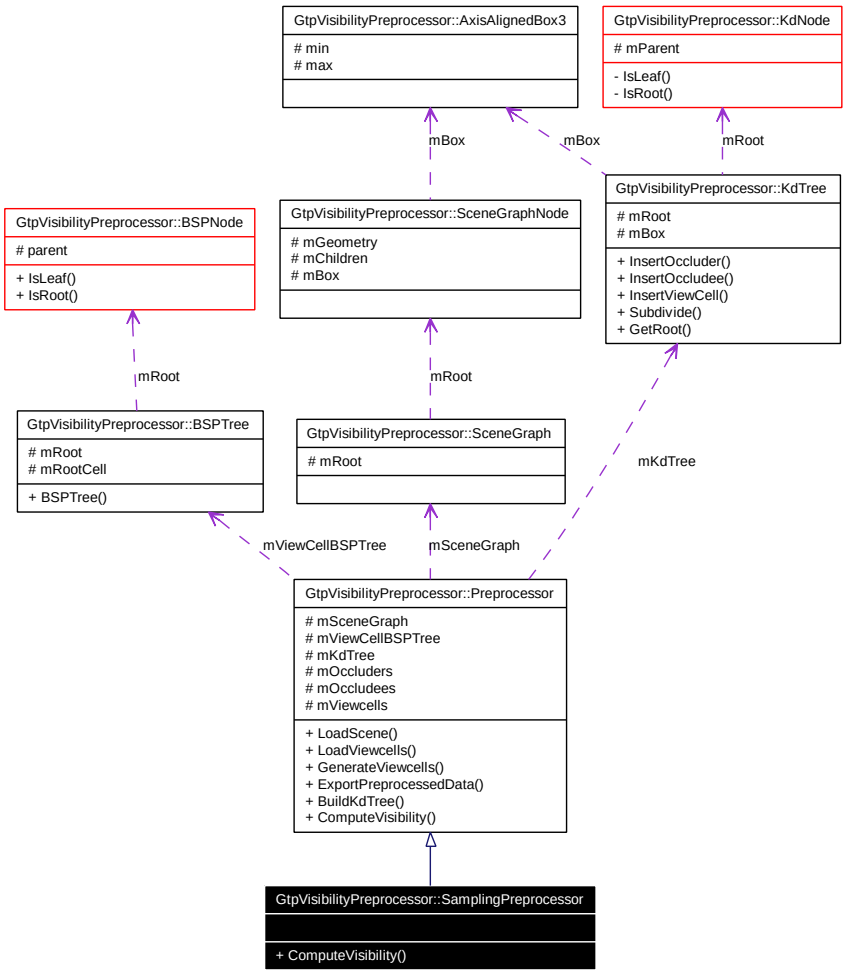
`#include <GtpVisibilityPreprocessor/include/SamplingPreprocessor.h>`

Inherits GtpVisibilityPreprocessor::Preprocessor.

Inheritance diagram for GtpVisibilityPreprocessor::SamplingPreprocessor:



Collaboration diagram for GtpVisibilityPreprocessor::SamplingPreprocessor:



Public Member Functions

- virtual bool ComputeVisibility ()

3.41.1 Detailed Description

Sampling based visibility preprocessing. The implementation is based on heuristical sampling of view space

3.41.2 Member Function Documentation

3.41.2.1 bool GtpVisibilityPreprocessor::SamplingPreprocessor::ComputeVisibility () [virtual]

Compute visibility method. This method has to be reimplemented by the actual Preprocessor implementation (e.g. SamplingPreprocessor, ExactPreprocessor, GlobalSamplingpreprocessor)

Implements GtpVisibilityPreprocessor::Preprocessor.

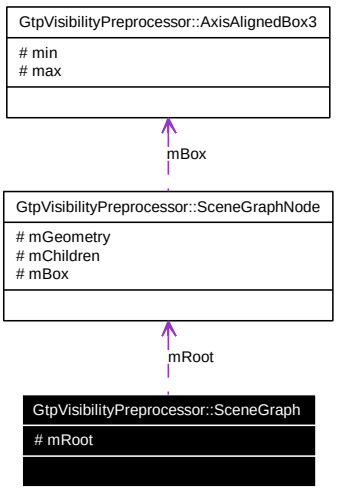
The documentation for this class was generated from the following files:

- SamplingPreprocessor.h
- SamplingPreprocessor.cpp

3.42 GtpVisibilityPreprocessor::SceneGraph Class Reference

`#include <GtpVisibilityPreprocessor/include/SceneGraph.h>`

Collaboration diagram for GtpVisibilityPreprocessor::SceneGraph:



Protected Attributes

- SceneGraphNode * mRoot

3.42.1 Detailed Description

Scene graph class

3.42.2 Member Data Documentation

3.42.2.1 SceneGraphNode* GtpVisibilityPreprocessor::SceneGraph::mRoot [protected]

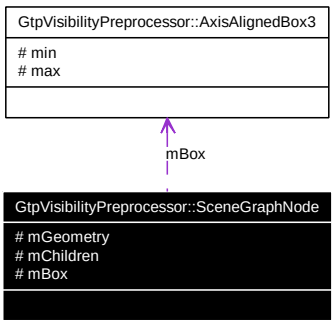
The documentation for this class was generated from the following file:

- SceneGraph.h

3.43 GtpVisibilityPreprocessor::SceneGraphNode Class Reference

`#include <GtpVisibilityPreprocessor/include/SceneGraph.h>`

Collaboration diagram for GtpVisibilityPreprocessor::SceneGraphNode:



Protected Attributes

- MeshContainer mGeometry
- NodeContainer mChildren
- AxisAlignedBox3 mBox

3.43.1 Detailed Description

Basic scene graph node, we are interested only in bounding boxes and topology of the scene graph

3.43.2 Member Data Documentation

3.43.2.1 MeshContainer GtpVisibilityPreprocessor::SceneGraphNode::mGeometry [protected]

3.43.2.2 NodeContainer GtpVisibilityPreprocessor::SceneGraphNode::mChildren [protected]

3.43.2.3 AxisAlignedBox3 GtpVisibilityPreprocessor::SceneGraphNode::mBox [protected]

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

- SceneGraph.h

3.44 GtpVisibilityPreprocessor::Vector3 Class Reference

`#include <GtpVisibilityPreprocessor/include/Vector3.h>`

Public Attributes

- float x
- float y
- float z

3.44.1 Detailed Description

3D vector

3.44.2 Member Data Documentation

3.44.2.1 float GtpVisibilityPreprocessor::Vector3::x

3.44.2.2 float GtpVisibilityPreprocessor::Vector3::y

3.44.2.3 float GtpVisibilityPreprocessor::Vector3::z

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

- Vector3.h

Chapter 4

GameTools Visibility Modules Namespace Documentation

4.1 GtpVisibility Namespace Reference

Classes

- class CoherentHierarchicalCullingManager
- class CullingManager
- class GreaterDistance
- class DummyPreprocessingManager
- class DummyQueryManager
- class FrustumCullingManager
- class HierarchyInterface
- class OcclusionQuery
- class PreprocessingManager
- class QueryManager
- class StopAndWaitCullingManager
- class VisibilityEnvironment
- class NodeInfo
- class MeshInfo
- class VisibilityManager

Typedefs

- typedef std::pair< HierarchyNode *, OcclusionQuery * > QueryPair
- typedef std::queue< QueryPair > QueryQueue
- typedef void HierarchyNode
- typedef std::priority_queue< HierarchyNode *, std::vector< HierarchyNode * >, GreaterDistance< std::vector< HierarchyNode * >::value_type > > DistanceQueue
- typedef Ogre::AxisAlignedBox AxisAlignedBox
- typedef Ogre::Camera Camera
- typedef Ogre::Mesh Mesh
- typedef Ogre::Ray Ray
- typedef Ogre::Vector3 Vector3

4.1.1 Detailed Description

This namespace includes all classes which are created by the VUT (Vienna University of Technology for the Visibility module of the GTP (GameTools Project) (www.gametools.org), and are not directly derived from an Ogre class.

4.1.2 Typedef Documentation

4.1.2.1 typedef std::pair<HierarchyNode *, OcclusionQuery *> GtpVisibility::QueryPair

4.1.2.2 typedef std::queue<QueryPair> GtpVisibility::QueryQueue

4.1.2.3 typedef void GtpVisibility::HierarchyNode

4.1.2.4 typedef std::priority_queue<HierarchyNode *, std::vector<HierarchyNode *>, GreaterDistance<std::vector<HierarchyNode *>::value_type> > GtpVisibility::DistanceQueue

A priority queue where closer hierarchy nodes are given a higher priority.

4.1.2.5 typedef Ogre::AxisAlignedBox GtpVisibility::AxisAlignedBox

This class currently uses the native Ogre AxisAlignedBox when compiled with the Ogre platform

4.1.2.6 typedef Ogre::Camera GtpVisibility::Camera

Camera class currently uses the native Ogre camera when compiled with the Ogre platform

4.1.2.7 typedef Ogre::Mesh GtpVisibility::Mesh

4.1.2.8 typedef Ogre::Ray GtpVisibility::Ray

Ray class currently uses native Ogre ray when compiled with the Ogre platform

4.1.2.9 typedef Ogre::Vector3 GtpVisibility::Vector3

Vector3 class currently uses the native Ogre vector when compiled with the Ogre platform

4.2 GtpVisibilityPreprocessor Namespace Reference

Classes

- class AxisAlignedBox3
- class ExactPreprocessor
- class KdTree
- class KdNode
- class KdInterior
- class KdLeaf
- class Patch
- class Mesh
- class Plane3
- class Preprocessor
- class SamplingPreprocessor
- class SceneGraphNode
- class SceneGraph
- class Vector3
- class BSPNode
- class BSPInterior
- class BSPLeaf
- class BSPTree

Typedefs

- typedef vector< Mesh * > MeshContainer
- typedef vector< ViewCell * > ViewCellContainer
- typedef vector< HierarchyNode * > NodeContainer

4.2.1 Detailed Description

Namespace for the external visibility preprocessor

This namespace includes all classes which are created by the VUT (Vienna University of Technology) for the External Visibility Preprocessor of the GTP (GameTools Project) (www.gametools.org).

4.2.2 Typedef Documentation

4.2.2.1 typedef vector<Mesh *> GtpVisibilityPreprocessor::MeshContainer

Container for Mesh pointers primarily for the use within the kDTree and BSP hierarchies

4.2.2.2 typedef vector<ViewCell *> GtpVisibilityPreprocessor::ViewCellContainer

Container for ViewCell pointers primarily for the use within the kDTree and BSP hierarchies

4.2.2.3 typedef vector<HierarchyNode *> GtpVisibilityPreprocessor::NodeContainer

Container for HierarchyNodes pointers primarily for the use within the kDTree and BSP hierarchies

4.3 Ogre Namespace Reference

Classes

- class BspHierarchyInterface
- class OctreeHierarchyInterface
- class PlatformHierarchyInterface
- class PlatformOcclusionQuery
- class SceneNodeHierarchyInterface
- class SolidHalfBoundingBox
- class VisibilityBspSceneManager
- class VisibilityDotSceneManager
- class VisibilityOctreeSceneManager
- class VisibilitySceneManager
- class VisibilityTerrainSceneManager

Functions

- void dllStartPlugin (void)
- void dllStopPlugin (void)

Variables

- OcclusionCullingOctreeSceneManager * cullingOctreePlugin
- OcclusionCullingTerrainSceneManager * cullingTerrainPlugin
- HeightmapTerrainPageSource * heightmapTerrainPageSource

4.3.1 Function Documentation

4.3.1.1 void Ogre::dllStartPlugin (void)

4.3.1.2 void Ogre::dllStopPlugin (void)

4.3.2 Variable Documentation

4.3.2.1 OcclusionCullingOctreeSceneManager* Ogre::cullingOctreePlugin

4.3.2.2 OcclusionCullingTerrainSceneManager* Ogre::cullingTerrainPlugin

4.3.2.3 HeightmapTerrainPageSource* Ogre::heightmapTerrainPageSource

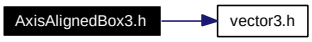
Chapter 5

GameTools Visibility Modules File Documentation

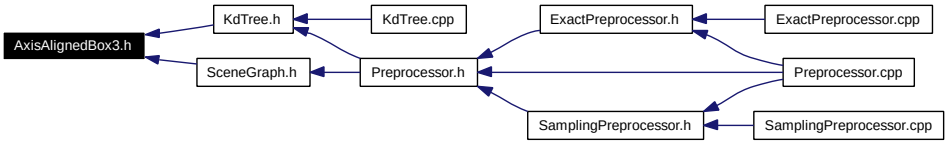
5.1 AxisAlignedBox3.h File Reference

```
#include "vector3.h"
```

Include dependency graph for AxisAlignedBox3.h:



This graph shows which files directly or indirectly include this file:



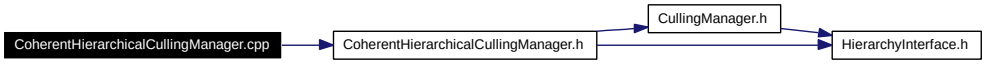
Namespaces

- namespace GtpVisibilityPreprocessor

5.2 CoherentHierarchicalCullingManager.cpp File Reference

```
#include "CoherentHierarchicalCullingManager.h"
```

Include dependency graph for CoherentHierarchicalCullingManager.cpp:



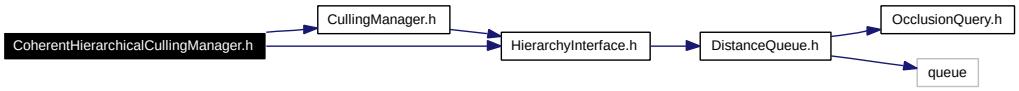
Namespaces

- namespace GtpVisibility

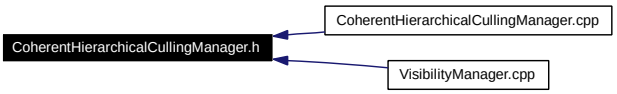
5.3 CoherentHierarchicalCullingManager.h File Reference

```
#include "CullingManager.h"
#include "HierarchyInterface.h"
```

Include dependency graph for CoherentHierarchicalCullingManager.h:



This graph shows which files directly or indirectly include this file:



Namespaces

- namespace GtpVisibility

Typedefs

- typedef std::pair< HierarchyNode *, OcclusionQuery * > QueryPair
- typedef std::queue< QueryPair > QueryQueue

5.3.1 Typedef Documentation

5.3.1.1 typedef std::pair<HierarchyNode *, OcclusionQuery *> GtpVisibility::QueryPair

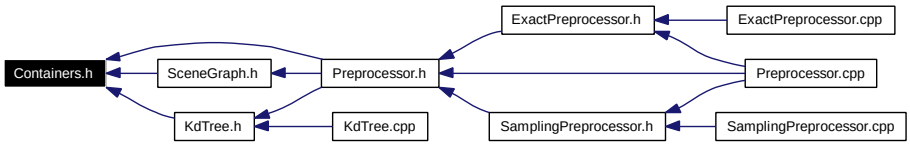
5.3.1.2 typedef std::queue<QueryPair> GtpVisibility::QueryQueue

5.4 Containers.h File Reference

`#include <vector>`

Include dependency graph for Containers.h:

This graph shows which files directly or indirectly include this file:



Namespaces

- namespace GtpVisibilityPreprocessor

Typedefs

- typedef vector< Mesh * > MeshContainer
- typedef vector< ViewCell * > ViewCellContainer
- typedef vector< HierarchyNode * > NodeContainer

5.4.1 Typedef Documentation

5.4.1.1 typedef vector<Mesh *> GtpVisibilityPreprocessor::MeshContainer

Container for Mesh pointers primarily for the use within the kDTree and BSP hierarchies

5.4.1.2 typedef vector<ViewCell *> GtpVisibilityPreprocessor::ViewCellContainer

Container for ViewCell pointers primarily for the use within the kDTree and BSP hierarchies

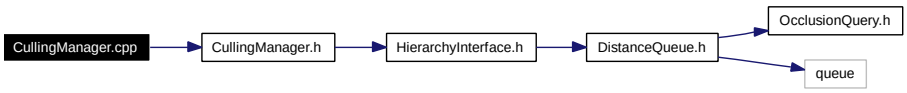
5.4.1.3 typedef vector<HierarchyNode *> GtpVisibilityPreprocessor::NodeContainer

Container for HierarchyNodes pointers primarily for the use within the kDTree and BSP hierarchies

5.5 CullingManager.cpp File Reference

```
#include "CullingManager.h"
```

Include dependency graph for CullingManager.cpp:



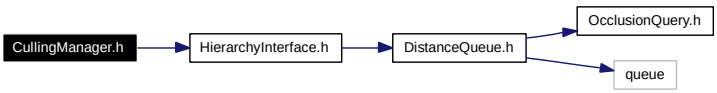
Namespaces

- namespace GtpVisibility

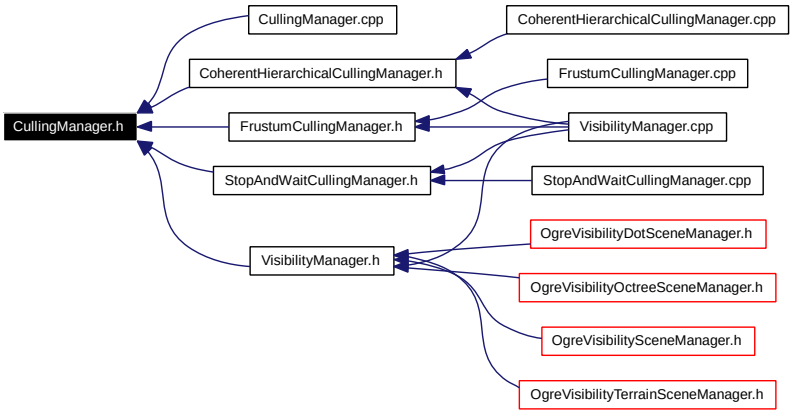
5.6 CullingManager.h File Reference

`#include "HierarchyInterface.h"`

Include dependency graph for CullingManager.h:



This graph shows which files directly or indirectly include this file:



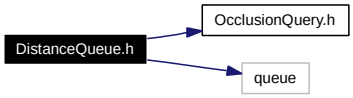
Namespaces

- namespace GtpVisibility

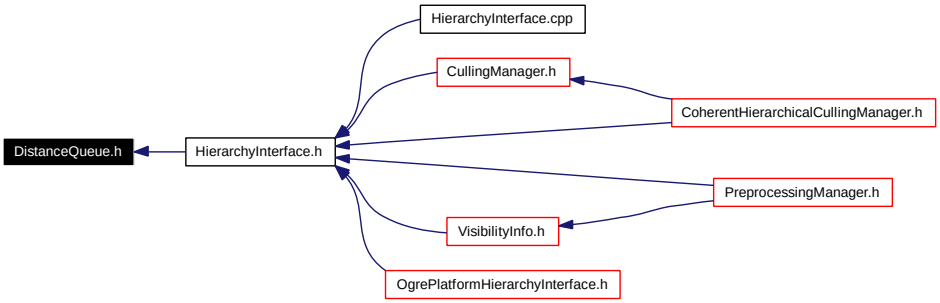
5.7 DistanceQueue.h File Reference

```
#include "OcclusionQuery.h"
#include <queue>
```

Include dependency graph for DistanceQueue.h:



This graph shows which files directly or indirectly include this file:



Namespaces

- namespace GtpVisibility

Typedefs

- typedef void HierarchyNode
- typedef std::priority_queue< HierarchyNode *, std::vector< HierarchyNode * >, GreaterDistance< std::vector< HierarchyNode * >::value_type > > DistanceQueue

5.7.1 Typedef Documentation

5.7.1.1 typedef void GtpVisibility::HierarchyNode

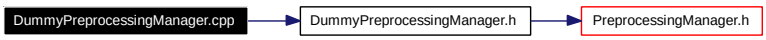
5.7.1.2 typedef std::priority_queue<HierarchyNode *, std::vector<HierarchyNode *>, GreaterDistance<std::vector<HierarchyNode *>::value_type> > GtpVisibility::DistanceQueue

A priority queue where closer hierarchy nodes are given a higher priority.

5.8 DummyPreprocessingManager.cpp File Reference

```
#include "DummyPreprocessingManager.h"
```

Include dependency graph for DummyPreprocessingManager.cpp:



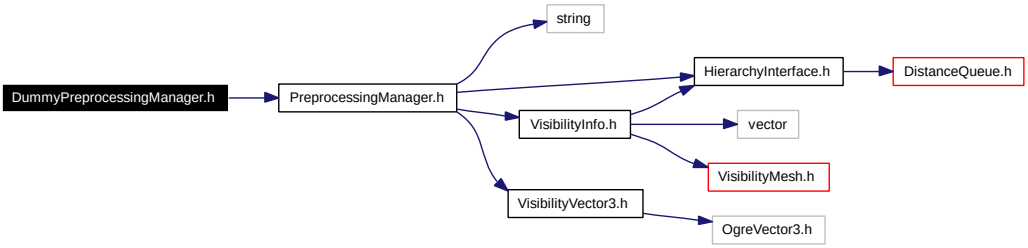
Namespaces

- namespace GtpVisibility

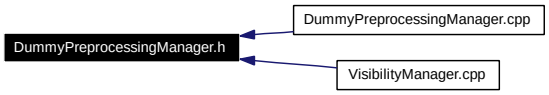
5.9 DummyPreprocessingManager.h File Reference

```
#include "PreprocessingManager.h"
```

Include dependency graph for DummyPreprocessingManager.h:



This graph shows which files directly or indirectly include this file:



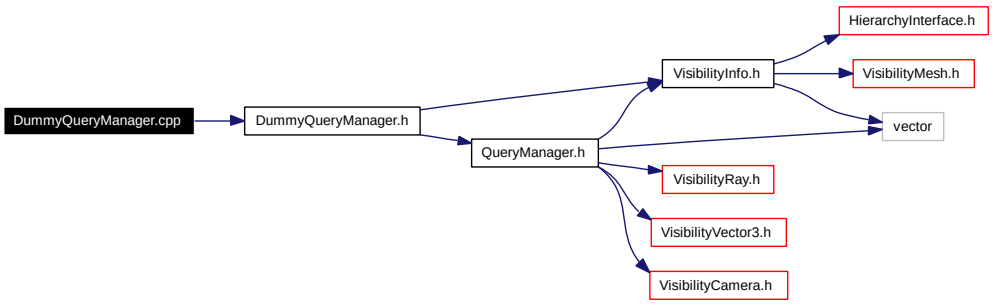
Namespaces

- namespace GtpVisibility

5.10 DummyQueryManager.cpp File Reference

```
#include "DummyQueryManager.h"
```

Include dependency graph for DummyQueryManager.cpp:



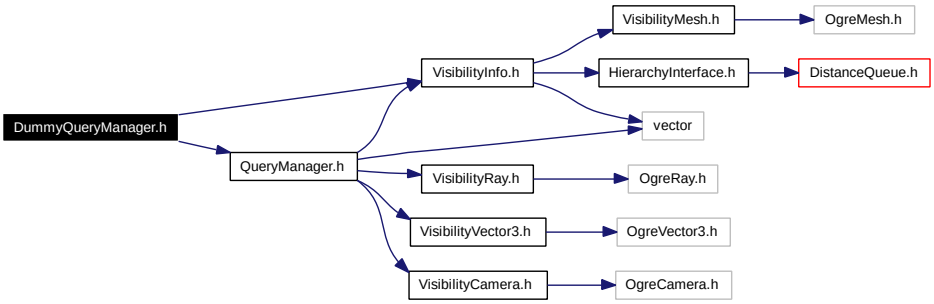
Namespaces

- namespace GtpVisibility

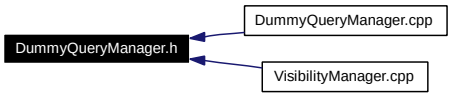
5.11 DummyQueryManager.h File Reference

```
#include "VisibilityInfo.h"
#include "QueryManager.h"
```

Include dependency graph for DummyQueryManager.h:



This graph shows which files directly or indirectly include this file:



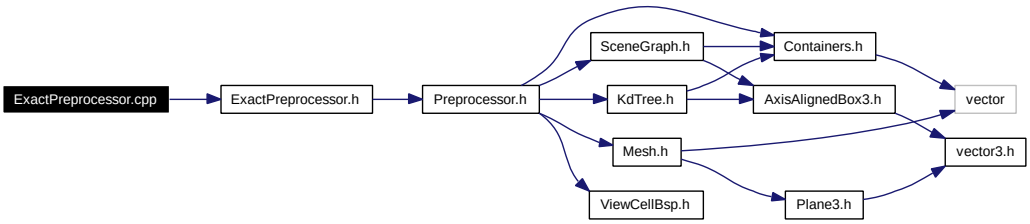
Namespaces

- namespace GtpVisibility

5.12 ExactPreprocessor.cpp File Reference

```
#include "ExactPreprocessor.h"
```

Include dependency graph for ExactPreprocessor.cpp:



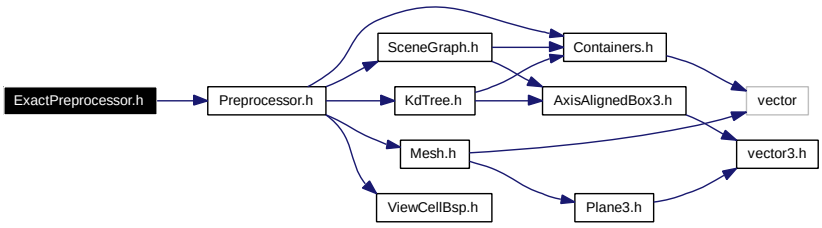
Namespaces

- namespace GtpVisibilityPreprocessor

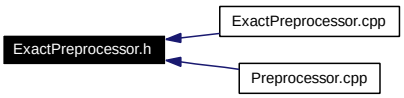
5.13 ExactPreprocessor.h File Reference

```
#include "Preprocessor.h"
```

Include dependency graph for ExactPreprocessor.h:



This graph shows which files directly or indirectly include this file:



Namespaces

- namespace GtpVisibilityPreprocessor

5.14 FrustumCullingManager.cpp File Reference

```
#include "FrustumCullingManager.h"
```

Include dependency graph for FrustumCullingManager.cpp:

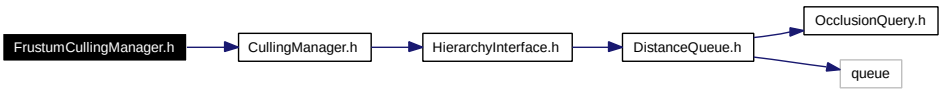
Namespaces

- namespace GtpVisibility

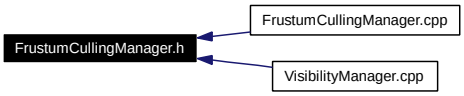
5.15 FrustumCullingManager.h File Reference

```
#include "CullingManager.h"
```

Include dependency graph for FrustumCullingManager.h:



This graph shows which files directly or indirectly include this file:



Namespaces

- namespace GtpVisibility

5.16 HierarchyInterface.cpp File Reference

```
#include "HierarchyInterface.h"
```

Include dependency graph for HierarchyInterface.cpp:

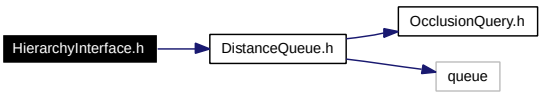
Namespaces

- namespace GtpVisibility

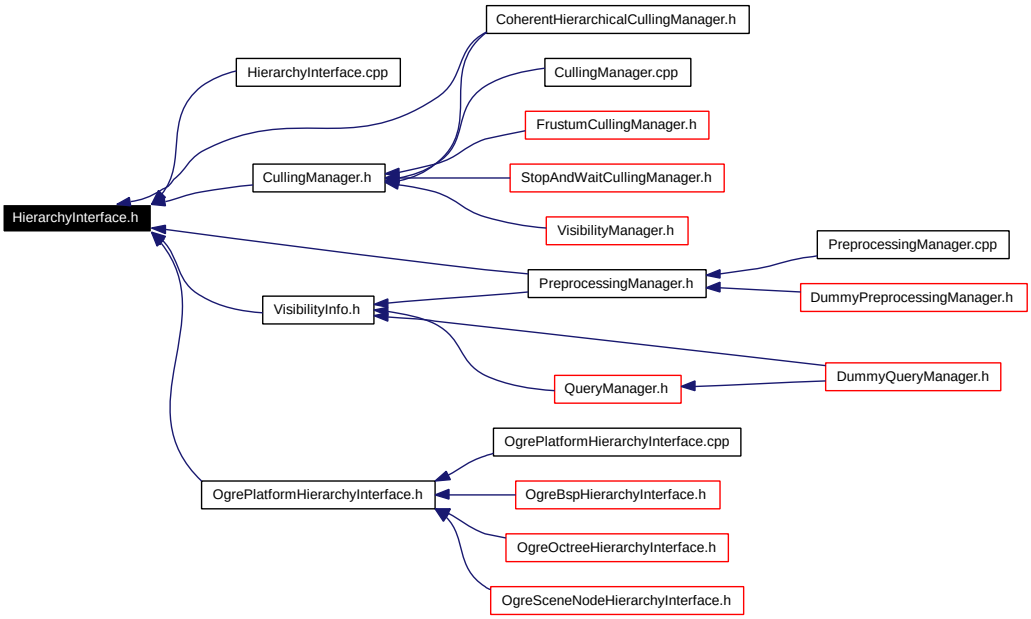
5.17 HierarchyInterface.h File Reference

```
#include "DistanceQueue.h"
```

Include dependency graph for HierarchyInterface.h:



This graph shows which files directly or indirectly include this file:



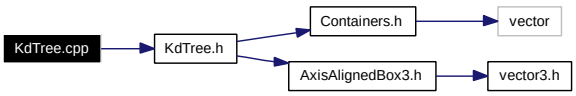
Namespaces

- namespace GtpVisibility

5.18 KdTree.cpp File Reference

```
#include "KdTree.h"
```

Include dependency graph for KdTree.cpp:



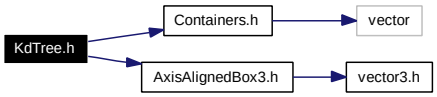
Namespaces

- namespace GtpVisibilityPreprocessor

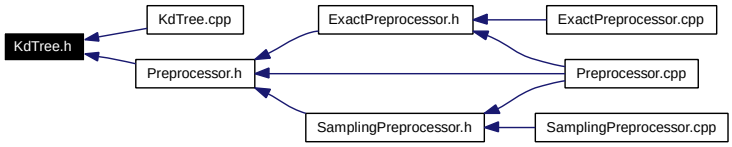
5.19 KdTree.h File Reference

```
#include "Containers.h"
#include "AxisAlignedBox3.h"
```

Include dependency graph for KdTree.h:



This graph shows which files directly or indirectly include this file:



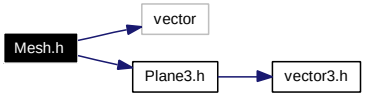
Namespaces

- namespace GtpVisibilityPreprocessor

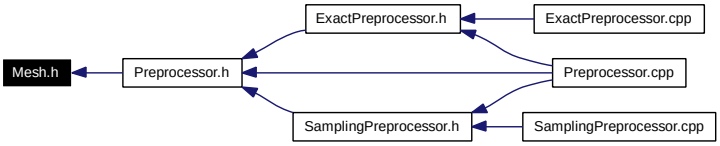
5.20 Mesh.h File Reference

```
#include <vector>
#include "Plane3.h"
```

Include dependency graph for Mesh.h:



This graph shows which files directly or indirectly include this file:

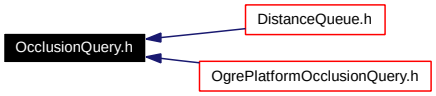


Namespaces

- namespace GtpVisibilityPreprocessor

5.21 OcclusionQuery.h File Reference

This graph shows which files directly or indirectly include this file:



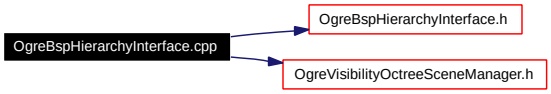
Namespaces

- namespace GtpVisibility

5.22 OgreBspHierarchyInterface.cpp File Reference

```
#include "OgreBspHierarchyInterface.h"
#include "OgreVisibilityOctreeSceneManager.h"
```

Include dependency graph for OgreBspHierarchyInterface.cpp:



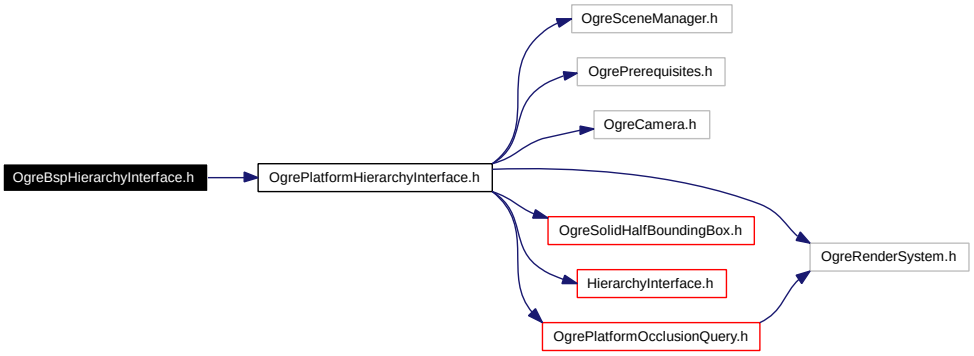
Namespaces

- namespace Ogre

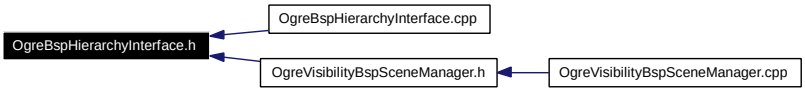
5.23 OgreBspHierarchyInterface.h File Reference

```
#include "OgrePlatformHierarchyInterface.h"
```

Include dependency graph for OgreBspHierarchyInterface.h:



This graph shows which files directly or indirectly include this file:



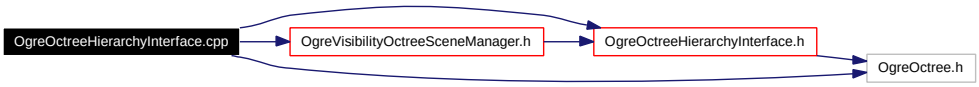
Namespaces

- namespace Ogre

5.24 OgreOctreeHierarchyInterface.cpp File Reference

```
#include "OgreOctreeHierarchyInterface.h"
#include "OgreVisibilityOctreeSceneManager.h"
#include <OgreOctree.h>
```

Include dependency graph for OgreOctreeHierarchyInterface.cpp:



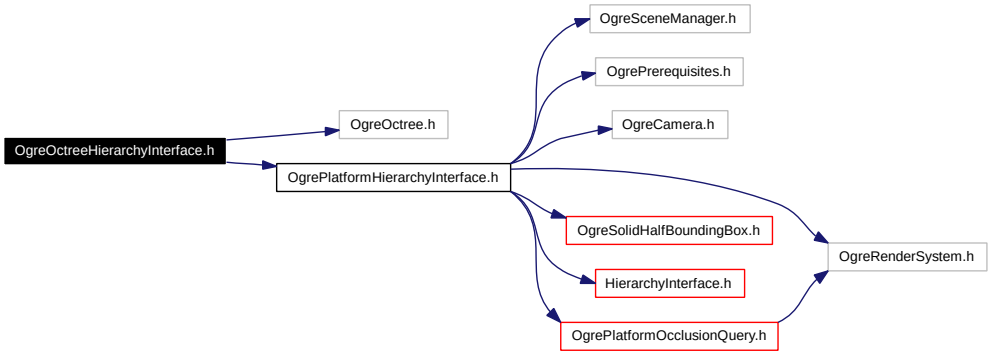
Namespaces

- namespace Ogre

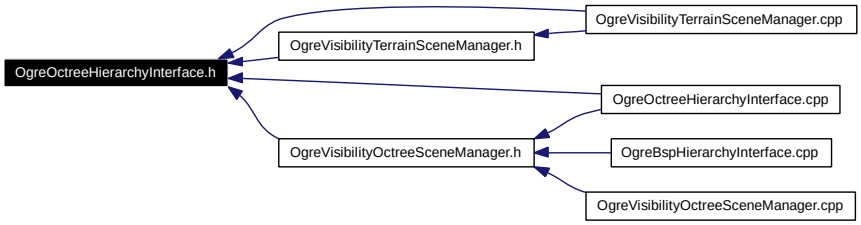
5.25 OgreOctreeHierarchyInterface.h File Reference

```
#include <OgreOctree.h>
#include "OgrePlatformHierarchyInterface.h"
```

Include dependency graph for OgreOctreeHierarchyInterface.h:



This graph shows which files directly or indirectly include this file:



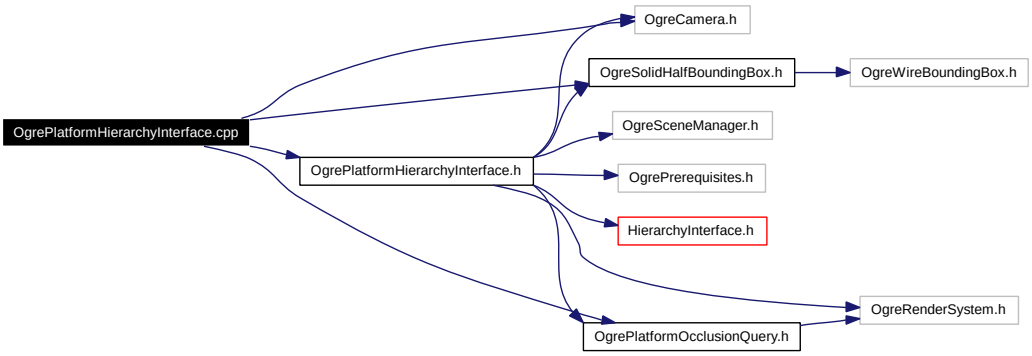
Namespaces

- namespace Ogre

5.26 OgrePlatformHierarchyInterface.cpp File Reference

```
#include <OgreCamera.h>
#include "OgreSolidHalfBoundingBox.h"
#include "OgrePlatformHierarchyInterface.h"
#include "OgrePlatformOcclusionQuery.h"
```

Include dependency graph for OgrePlatformHierarchyInterface.cpp:



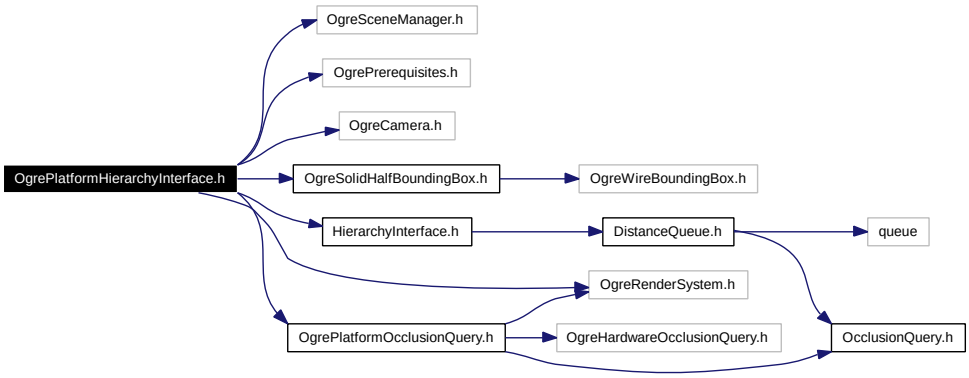
Namespaces

- namespace Ogre

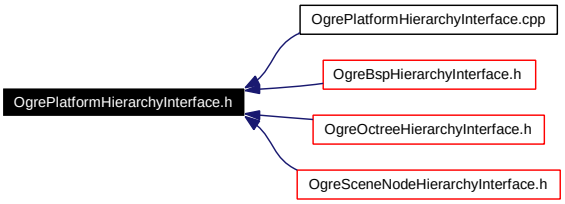
5.27 OgrePlatformHierarchyInterface.h File Reference

```
#include <OgreSceneManager.h>
#include <OgrePrerequisites.h>
#include <OgreCamera.h>
#include <OgreRenderSystem.h>
#include "OgreSolidHalfBoundingBox.h"
#include "HierarchyInterface.h"
#include "OgrePlatformOcclusionQuery.h"
```

Include dependency graph for OgrePlatformHierarchyInterface.h:



This graph shows which files directly or indirectly include this file:



Namespaces

- namespace Ogre

5.28 OgrePlatformOcclusionQuery.cpp File Reference

```
#include "OgrePlatformOcclusionQuery.h"
```

Include dependency graph for OgrePlatformOcclusionQuery.cpp:

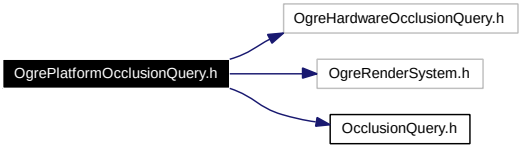
Namespaces

- namespace Ogre

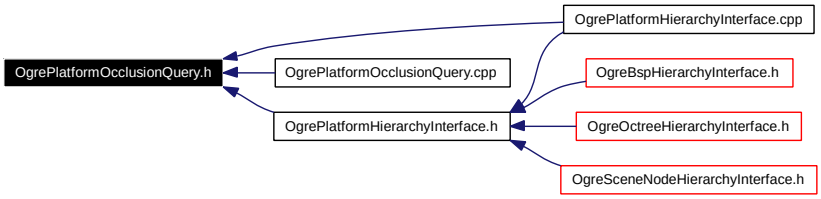
5.29 OgrePlatformOcclusionQuery.h File Reference

```
#include <OgreHardwareOcclusionQuery.h>
#include <OgreRenderSystem.h>
#include "OcclusionQuery.h"
```

Include dependency graph for OgrePlatformOcclusionQuery.h:



This graph shows which files directly or indirectly include this file:



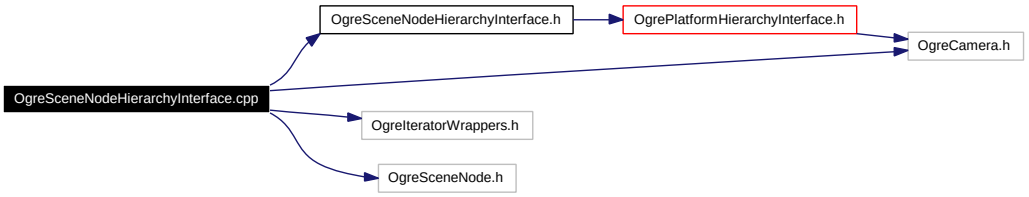
Namespaces

- namespace Ogre

5.30 OgreSceneNodeHierarchyInterface.cpp File Reference

```
#include "OgreSceneNodeHierarchyInterface.h"
#include <OgreIteratorWrappers.h>
#include <OgreCamera.h>
#include <OgreSceneNode.h>
```

Include dependency graph for OgreSceneNodeHierarchyInterface.cpp:



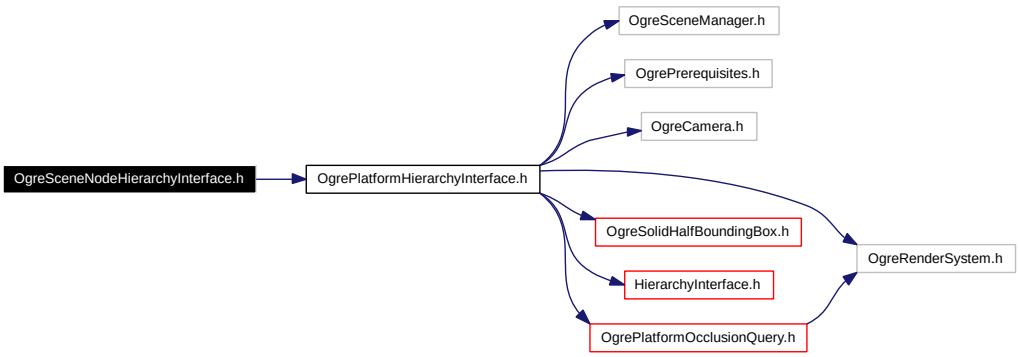
Namespaces

- namespace Ogre

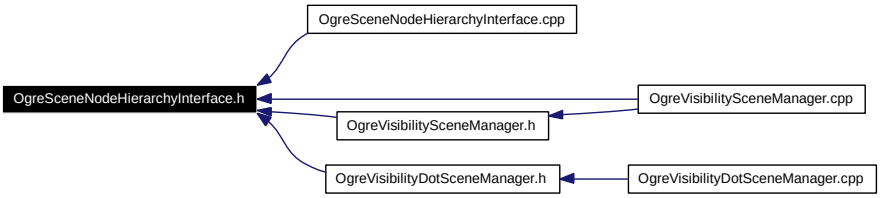
5.31 OgreSceneNodeHierarchyInterface.h File Reference

```
#include "OgrePlatformHierarchyInterface.h"
```

Include dependency graph for OgreSceneNodeHierarchyInterface.h:



This graph shows which files directly or indirectly include this file:



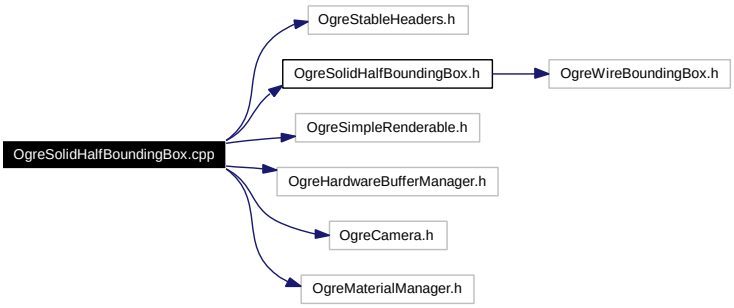
Namespaces

- namespace Ogre

5.32 OgreSolidHalfBoundingBox.cpp File Reference

```
#include "OgreStableHeaders.h"
#include "OgreSolidHalfBoundingBox.h"
#include "OgreSimpleRenderable.h"
#include "OgreHardwareBufferManager.h"
#include "OgreCamera.h"
#include "OgreMaterialManager.h"
```

Include dependency graph for OgreSolidHalfBoundingBox.cpp:



Namespaces

- namespace Ogre

Defines

- #define POSITION_BINDING 0

5.32.1 Define Documentation

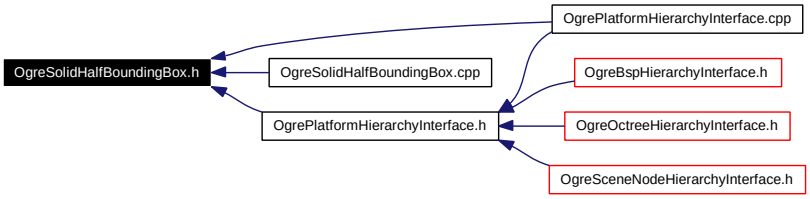
5.32.1.1 #define POSITION_BINDING 0

5.33 OgreSolidHalfBoundingBox.h File Reference

```
#include "OgreWireBoundingBox.h"
```

Include dependency graph for OgreSolidHalfBoundingBox.h:

This graph shows which files directly or indirectly include this file:



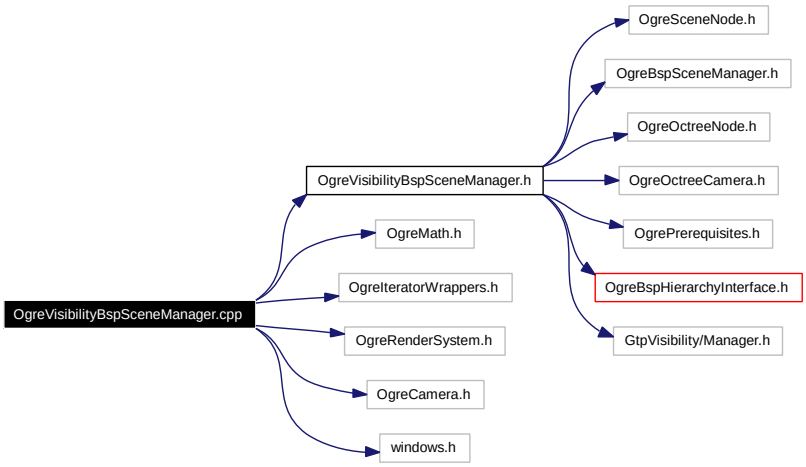
Namespaces

- namespace Ogre

5.34 OgreVisibilityBspSceneManager.cpp File Reference

```
#include "OgreVisibilityBspSceneManager.h"
#include <OgreMath.h>
#include <OgreIteratorWrappers.h>
#include <OgreRenderSystem.h>
#include <OgreCamera.h>
#include <windows.h>
```

Include dependency graph for OgreVisibilityBspSceneManager.cpp:



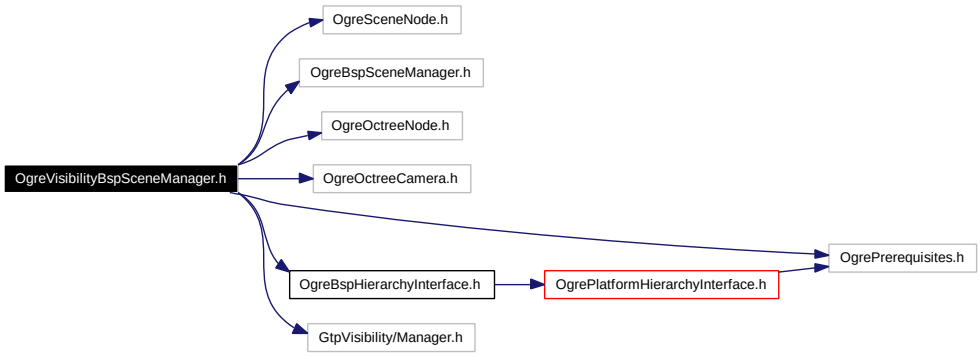
Namespaces

- namespace Ogre

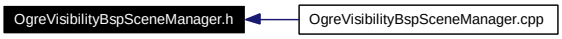
5.35 OgreVisibilityBspSceneManager.h File Reference

```
#include <OgreSceneNode.h>
#include <OgreBspSceneManager.h>
#include <OgreOctreeNode.h>
#include <OgreOctreeCamera.h>
#include <OgrePrerequisites.h>
#include "OgreBspHierarchyInterface.h"
#include "GtpVisibility/Manager.h"
```

Include dependency graph for OgreVisibilityBspSceneManager.h:



This graph shows which files directly or indirectly include this file:



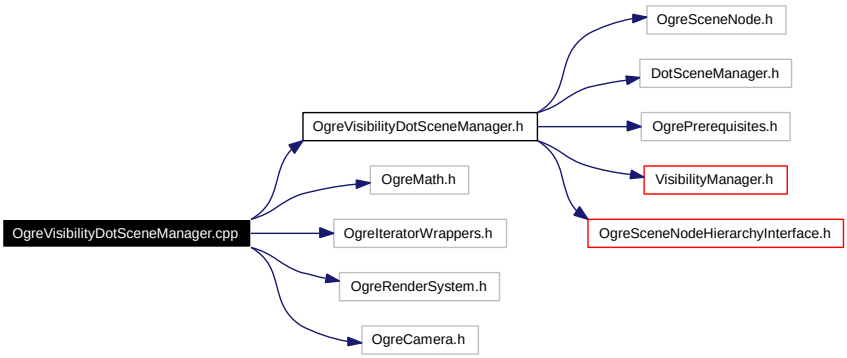
Namespaces

- namespace Ogre

5.36 OgreVisibilityDotSceneManager.cpp File Reference

```
#include <OgreVisibilityDotSceneManager.h>
#include <OgreMath.h>
#include <OgreIteratorWrappers.h>
#include <OgreRenderSystem.h>
#include <OgreCamera.h>
```

Include dependency graph for OgreVisibilityDotSceneManager.cpp:



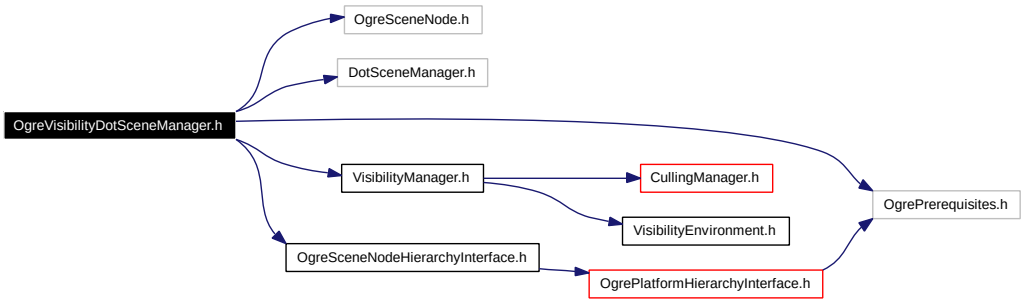
Namespaces

- namespace Ogre

5.37 OgreVisibilityDotSceneManager.h File Reference

```
#include <OgreSceneNode.h>
#include <DotSceneManager.h>
#include <OgrePrerequisites.h>
#include "VisibilityManager.h"
#include "OgreSceneNodeHierarchyInterface.h"
```

Include dependency graph for OgreVisibilityDotSceneManager.h:



This graph shows which files directly or indirectly include this file:

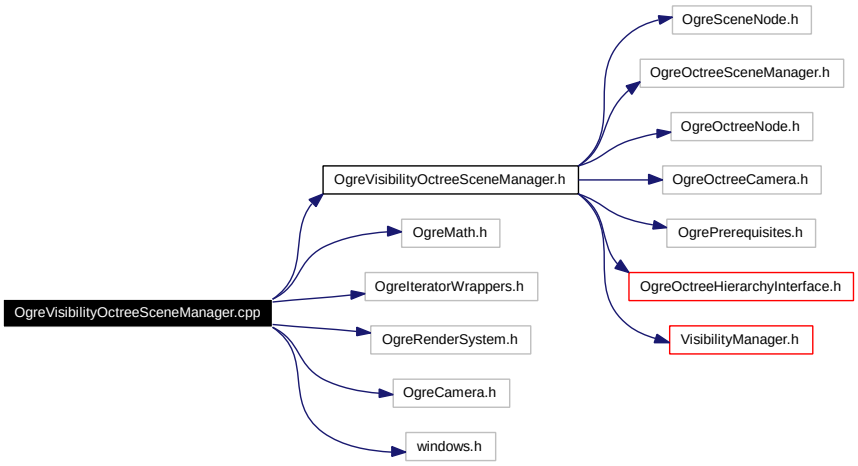
Namespaces

- namespace Ogre

5.38 OgreVisibilityOctreeSceneManager.cpp File Reference

```
#include "OgreVisibilityOctreeSceneManager.h"
#include <OgreMath.h>
#include <OgreIteratorWrappers.h>
#include <OgreRenderSystem.h>
#include <OgreCamera.h>
#include <windows.h>
```

Include dependency graph for OgreVisibilityOctreeSceneManager.cpp:



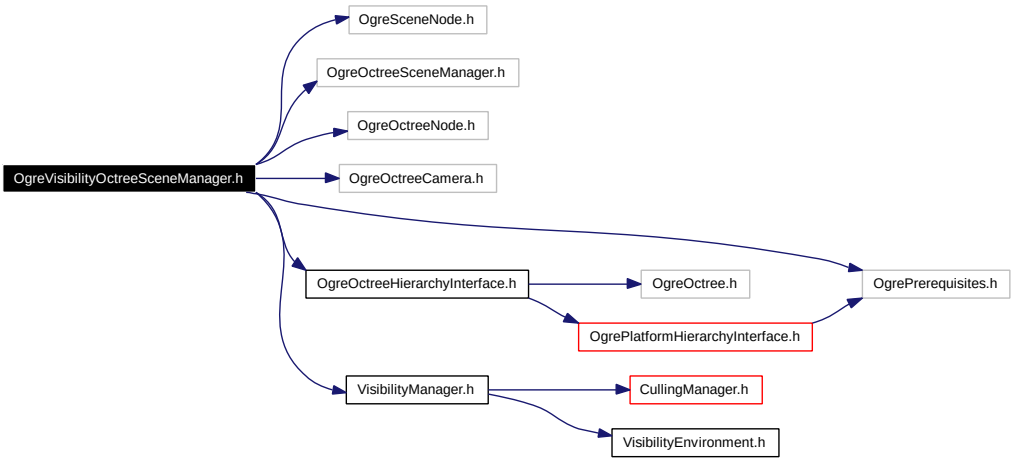
Namespaces

- namespace Ogre

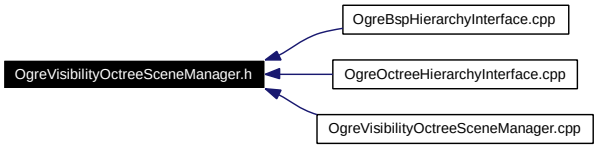
5.39 OgreVisibilityOctreeSceneManager.h File Reference

```
#include <OgreSceneNode.h>
#include <OgreOctreeSceneManager.h>
#include <OgreOctreeNode.h>
#include <OgreOctreeCamera.h>
#include <OgrePrerequisites.h>
#include "OgreOctreeHierarchyInterface.h"
#include "VisibilityManager.h"
```

Include dependency graph for OgreVisibilityOctreeSceneManager.h:



This graph shows which files directly or indirectly include this file:



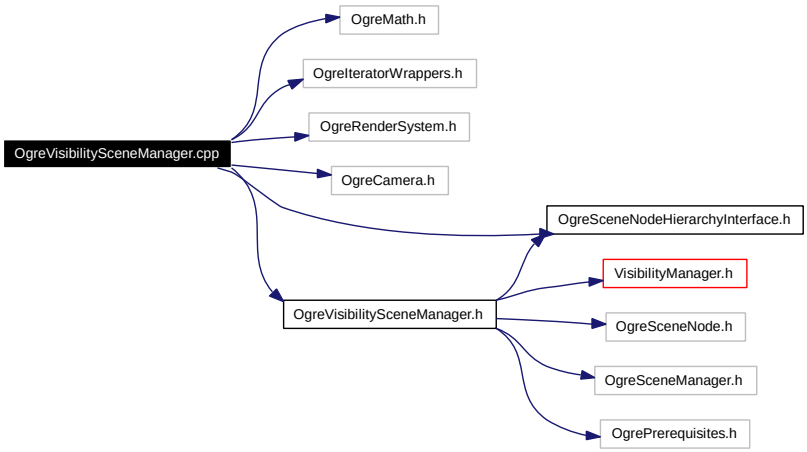
Namespaces

- namespace Ogre

5.40 OgreVisibilitySceneManager.cpp File Reference

```
#include <OgreMath.h>
#include <OgreIteratorWrappers.h>
#include <OgreRenderSystem.h>
#include <OgreCamera.h>
#include "OgreVisibilitySceneManager.h"
#include "OgreSceneNodeHierarchyInterface.h"
```

Include dependency graph for OgreVisibilitySceneManager.cpp:



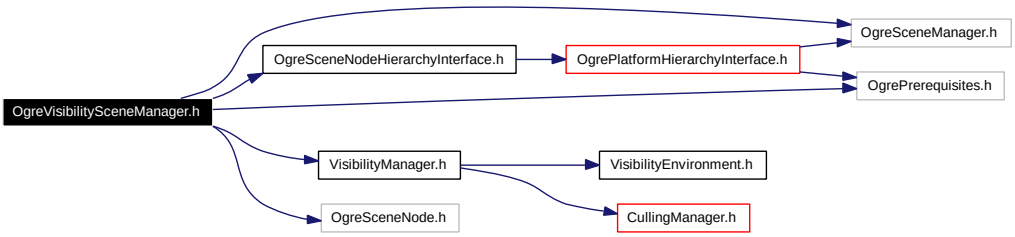
Namespaces

- namespace Ogre

5.41 OgreVisibilitySceneManager.h File Reference

```
#include "OgreSceneNodeHierarchyInterface.h"
#include "VisibilityManager.h"
#include <OgreSceneNode.h>
#include <OgreSceneManager.h>
#include <OgrePrerequisites.h>
```

Include dependency graph for OgreVisibilitySceneManager.h:



This graph shows which files directly or indirectly include this file:

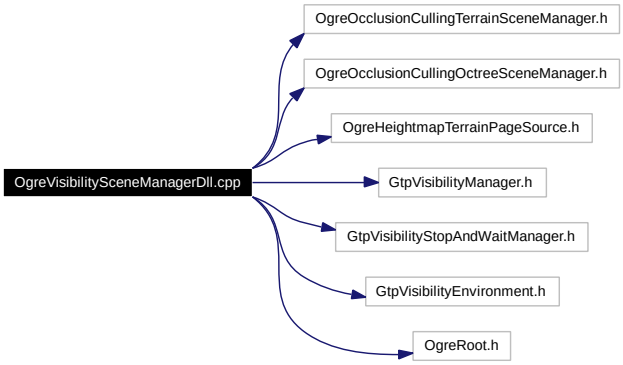
Namespaces

- namespace Ogre

5.42 OgreVisibilitySceneManagerDll.cpp File Reference

```
#include <OgreOcclusionCullingTerrainSceneManager.h>
#include <OgreOcclusionCullingOctreeSceneManager.h>
#include <OgreHeightmapTerrainPageSource.h>
#include "GtpVisibilityManager.h"
#include "GtpVisibilityStopAndWaitManager.h"
#include "GtpVisibilityEnvironment.h"
#include <OgreRoot.h>
```

Include dependency graph for OgreVisibilitySceneManagerDll.cpp:



Namespaces

- namespace Ogre

Functions

- void dllStartPlugin (void)
- void dllStopPlugin (void)

Variables

- GtpVisibility::Environment * visEnv
- GtpVisibility::Manager * visManager
- OcclusionCullingOctreeSceneManager * cullingOctreePlugin
- OcclusionCullingTerrainSceneManager * cullingTerrainPlugin
- HeightmapTerrainPageSource * heightmapTerrainPageSource

5.42.1 Function Documentation

5.42.1.1 void Ogre::dllStartPlugin (void)

5.42.1.2 void Ogre::dllStopPlugin (void)

5.42.2 Variable Documentation

5.42.2.1 GtpVisibility::Environment* visEnv

5.42.2.2 GtpVisibility::Manager* visManager

5.42.2.3 OcclusionCullingOctreeSceneManager* Ogre::cullingOctreePlugin

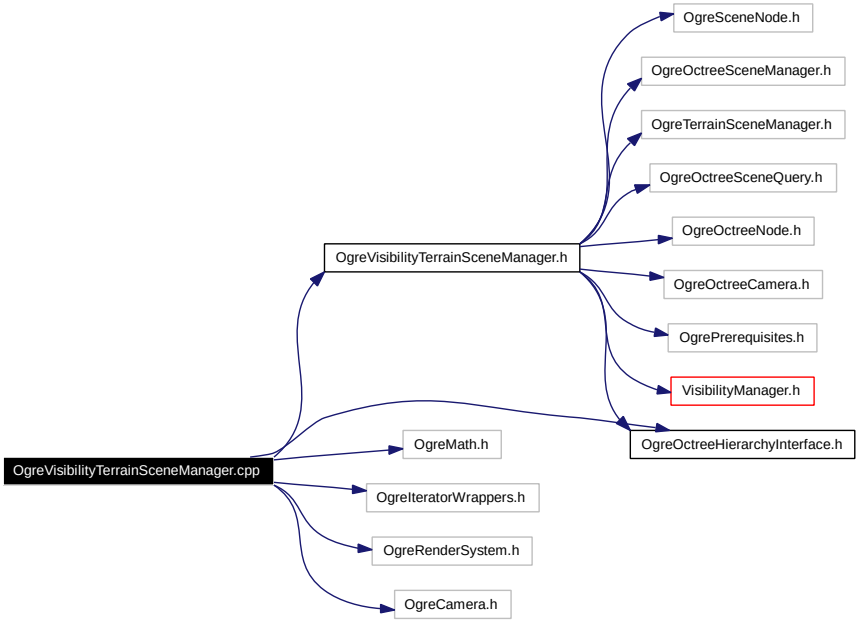
5.42.2.4 OcclusionCullingTerrainSceneManager* Ogre::cullingTerrainPlugin

5.42.2.5 HeightmapTerrainPageSource* Ogre::heightmapTerrainPageSource

5.43 OgreVisibilityTerrainSceneManager.cpp File Reference

```
#include "OgreVisibilityTerrainSceneManager.h"
#include "OgreOctreeHierarchyInterface.h"
#include <OgreMath.h>
#include <OgreIteratorWrappers.h>
#include <OgreRenderSystem.h>
#include <OgreCamera.h>
```

Include dependency graph for OgreVisibilityTerrainSceneManager.cpp:



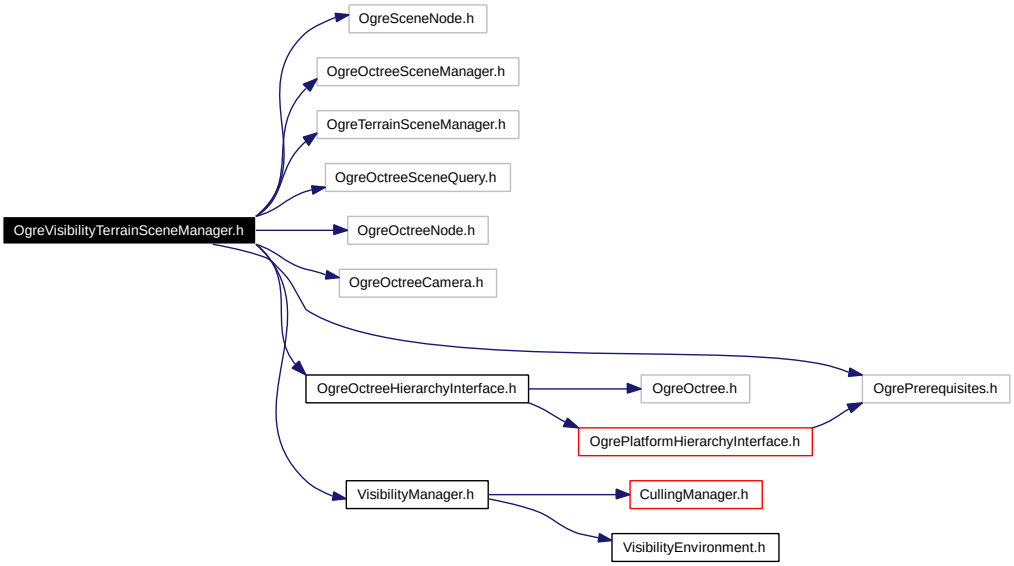
Namespaces

- namespace Ogre

5.44 OgreVisibilityTerrainSceneManager.h File Reference

```
#include <OgreSceneNode.h>
#include <OgreOctreeSceneManager.h>
#include <OgreTerrainSceneManager.h>
#include <OgreOctreeSceneQuery.h>
#include <OgreOctreeNode.h>
#include <OgreOctreeCamera.h>
#include <OgrePrerequisites.h>
#include "OgreOctreeHierarchyInterface.h"
#include "VisibilityManager.h"
```

Include dependency graph for OgreVisibilityTerrainSceneManager.h:



This graph shows which files directly or indirectly include this file:

Namespaces

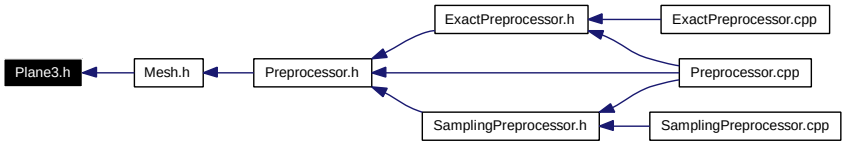
- namespace Ogre

5.45 Plane3.h File Reference

`#include "vector3.h"`

Include dependency graph for Plane3.h:

This graph shows which files directly or indirectly include this file:



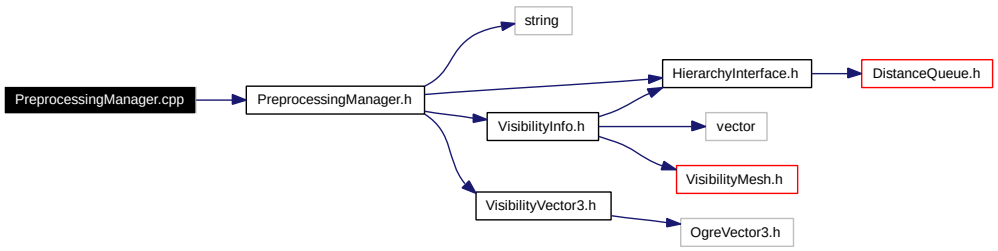
Namespaces

- namespace GtpVisibilityPreprocessor

5.46 PreprocessingManager.cpp File Reference

```
#include "PreprocessingManager.h"
```

Include dependency graph for PreprocessingManager.cpp:



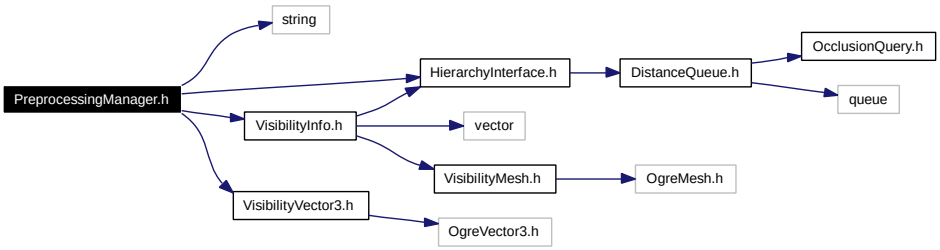
Namespaces

- namespace GtpVisibility

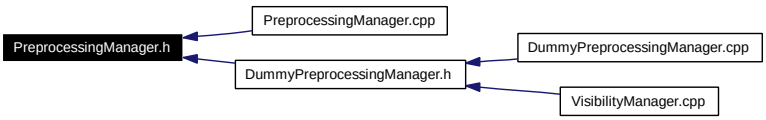
5.47 PreprocessingManager.h File Reference

```
#include <string>
#include "HierarchyInterface.h"
#include "VisibilityInfo.h"
#include "VisibilityVector3.h"
```

Include dependency graph for PreprocessingManager.h:



This graph shows which files directly or indirectly include this file:



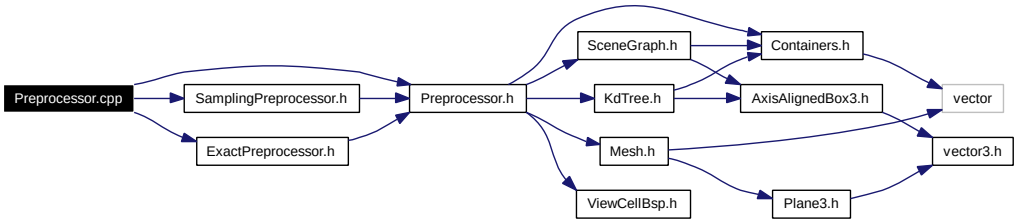
Namespaces

- namespace GtpVisibility
- namespace std

5.48 Preprocessor.cpp File Reference

```
#include "Preprocessor.h"
#include "SamplingPreprocessor.h"
#include "ExactPreprocessor.h"
```

Include dependency graph for Preprocessor.cpp:



Namespaces

- namespace GtpVisibilityPreprocessor

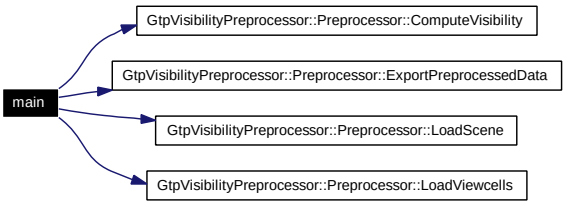
Functions

- int main (int argc, char **argv)

5.48.1 Function Documentation

5.48.1.1 int main (int *argc*, char ** *argv*)

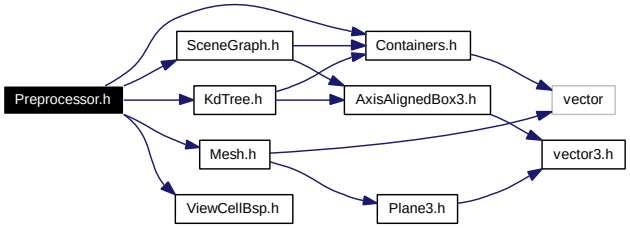
Here is the call graph for this function:



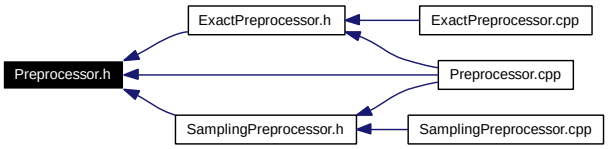
5.49 Preprocessor.h File Reference

```
#include "Containers.h"
#include "Mesh.h"
#include "KdTree.h"
#include "ViewCellBsp.h"
#include "SceneGraph.h"
```

Include dependency graph for Preprocessor.h:



This graph shows which files directly or indirectly include this file:



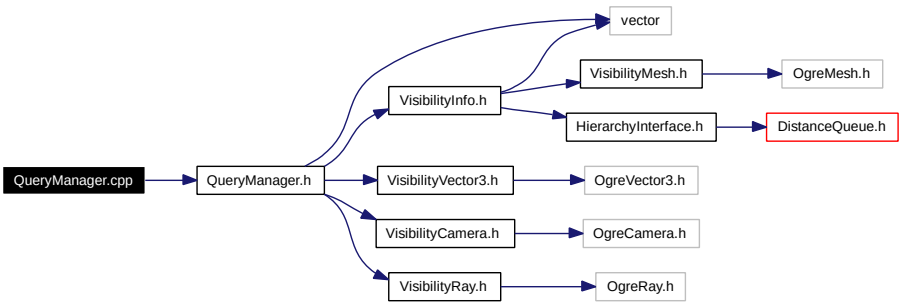
Namespaces

- namespace GtpVisibilityPreprocessor

5.50 QueryManager.cpp File Reference

```
#include "QueryManager.h"
```

Include dependency graph for QueryManager.cpp:



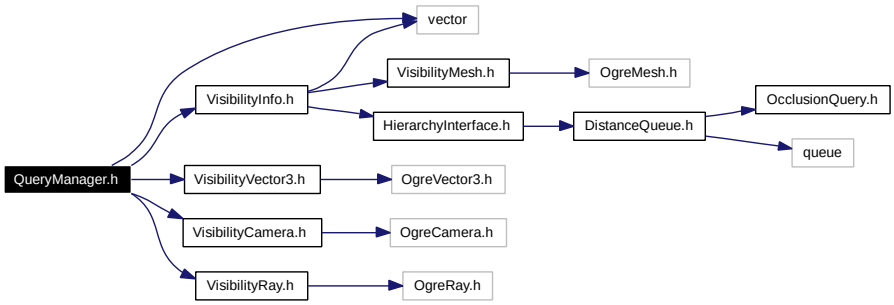
Namespaces

- namespace GtpVisibility

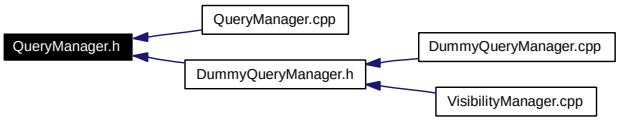
5.51 QueryManager.h File Reference

```
#include <vector>
#include "VisibilityInfo.h"
#include "VisibilityVector3.h"
#include "VisibilityCamera.h"
#include "VisibilityRay.h"
```

Include dependency graph for QueryManager.h:



This graph shows which files directly or indirectly include this file:



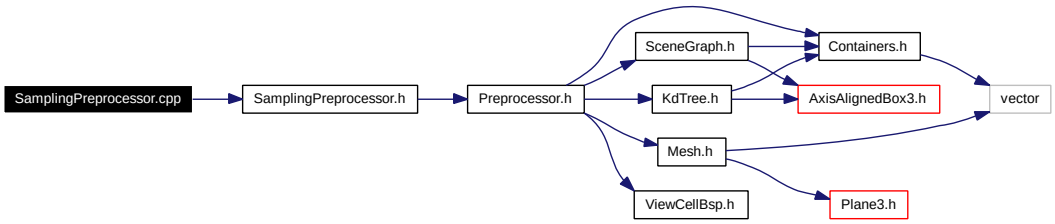
Namespaces

- namespace GtpVisibility

5.52 SamplingPreprocessor.cpp File Reference

```
#include "SamplingPreprocessor.h"
```

Include dependency graph for SamplingPreprocessor.cpp:



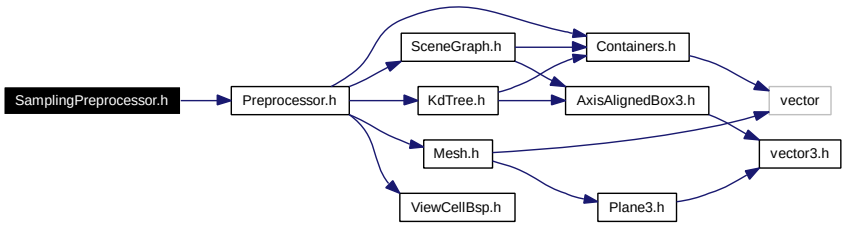
Namespaces

- namespace GtpVisibilityPreprocessor

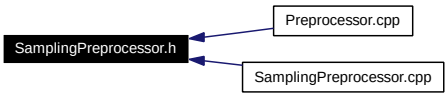
5.53 SamplingPreprocessor.h File Reference

```
#include "Preprocessor.h"
```

Include dependency graph for SamplingPreprocessor.h:



This graph shows which files directly or indirectly include this file:



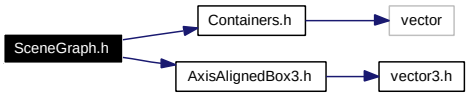
Namespaces

- namespace GtpVisibilityPreprocessor

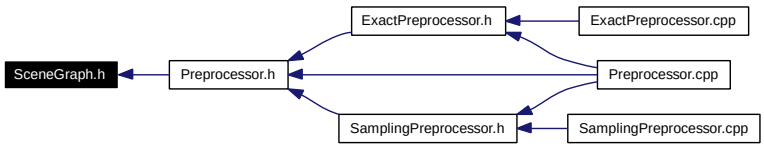
5.54 SceneGraph.h File Reference

```
#include "Containers.h"
#include "AxisAlignedBox3.h"
```

Include dependency graph for SceneGraph.h:



This graph shows which files directly or indirectly include this file:



Namespaces

- namespace GtpVisibilityPreprocessor

5.55 StopAndWaitCullingManager.cpp File Reference

```
#include "StopAndWaitCullingManager.h"
```

Include dependency graph for StopAndWaitCullingManager.cpp:

Namespaces

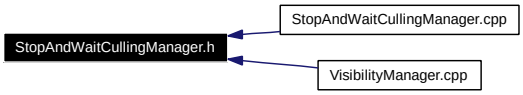
- namespace GtpVisibility

5.56 StopAndWaitCullingManager.h File Reference

```
#include "CullingManager.h"
```

Include dependency graph for StopAndWaitCullingManager.h:

This graph shows which files directly or indirectly include this file:

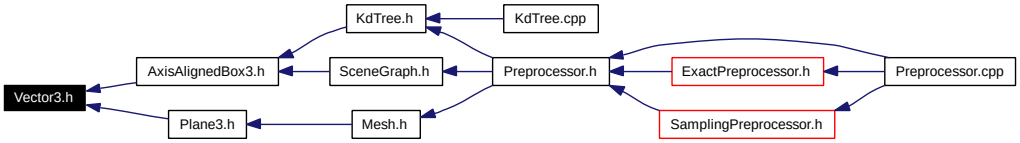


Namespaces

- namespace GtpVisibility

5.57 Vector3.h File Reference

This graph shows which files directly or indirectly include this file:

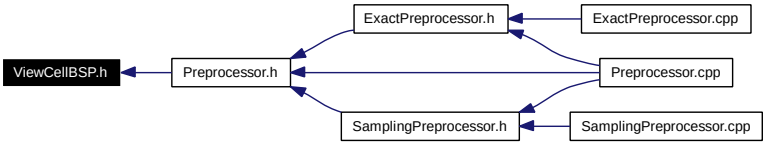


Namespaces

- namespace GtpVisibilityPreprocessor

5.58 ViewCellBSP.h File Reference

This graph shows which files directly or indirectly include this file:



Namespaces

- namespace GtpVisibilityPreprocessor

5.59 VisibilityAxisAlignedBox.h File Reference

```
#include "OgreAxisAlignedBox.h"
```

Include dependency graph for VisibilityAxisAlignedBox.h:

Namespaces

- namespace GtpVisibility

Typedefs

- typedef Ogre::AxisAlignedBox AxisAlignedBox

5.59.1 Typedef Documentation

5.59.1.1 typedef Ogre::AxisAlignedBox GtpVisibility::AxisAlignedBox

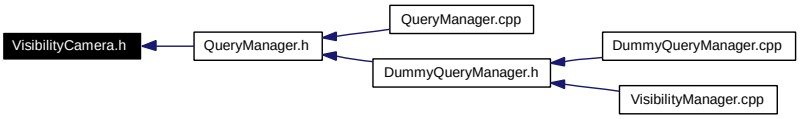
This class currently uses the native Ogre AxisAlignedBox when compiled with the Ogre platform

5.60 VisibilityCamera.h File Reference

```
#include "OgreCamera.h"
```

Include dependency graph for VisibilityCamera.h:

This graph shows which files directly or indirectly include this file:



Namespaces

- namespace GtpVisibility

Typedefs

- typedef Ogre::Camera Camera

5.60.1 Typedef Documentation

5.60.1.1 typedef Ogre::Camera GtpVisibility::Camera

Camera class currently uses the native Ogre camera when compiled with the Ogre platform

5.61 VisibilityEnvironment.cpp File Reference

```
#include "VisibilityEnvironment.h"
```

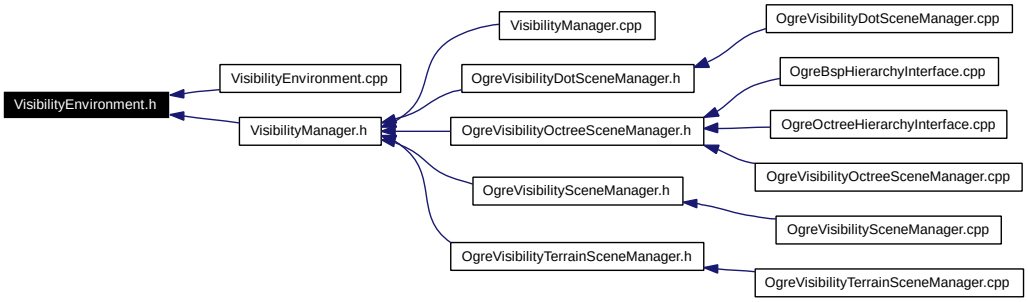
Include dependency graph for VisibilityEnvironment.cpp:

Namespaces

- namespace GtpVisibility

5.62 VisibilityEnvironment.h File Reference

This graph shows which files directly or indirectly include this file:



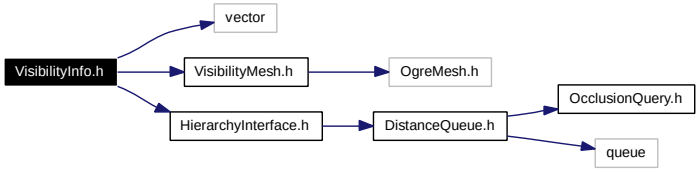
Namespaces

- namespace GtpVisibility

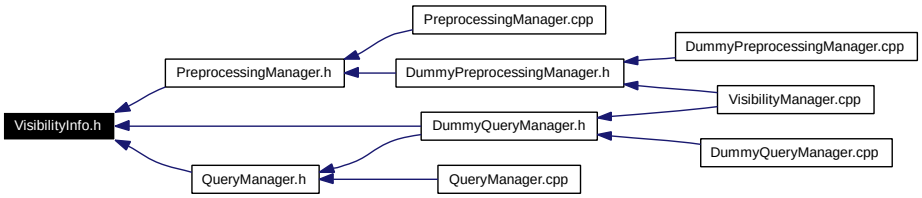
5.63 VisibilityInfo.h File Reference

```
#include <vector>
#include "VisibilityMesh.h"
#include "HierarchyInterface.h"
```

Include dependency graph for VisibilityInfo.h:



This graph shows which files directly or indirectly include this file:



Namespaces

- namespace GtpVisibility

Defines

- #define InfoContainer std::vector

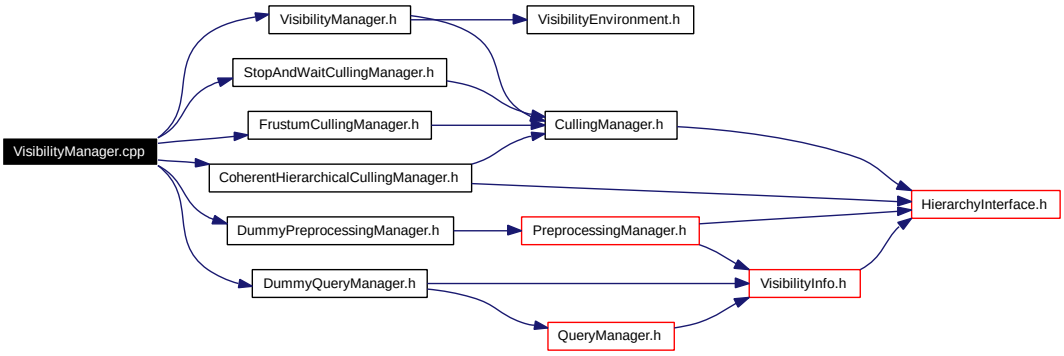
5.63.1 Define Documentation

5.63.1.1 #define InfoContainer std::vector

5.64 VisibilityManager.cpp File Reference

```
#include "VisibilityManager.h"
#include "StopAndWaitCullingManager.h"
#include "CoherentHierarchicalCullingManager.h"
#include "FrustumCullingManager.h"
#include "DummyPreprocessingManager.h"
#include "DummyQueryManager.h"
```

Include dependency graph for VisibilityManager.cpp:



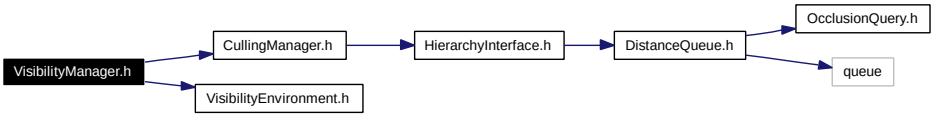
Namespaces

- namespace GtpVisibility

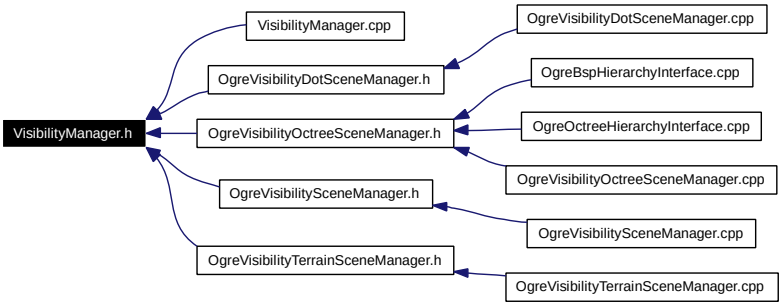
5.65 VisibilityManager.h File Reference

```
#include "CullingManager.h"
#include "VisibilityEnvironment.h"
```

Include dependency graph for VisibilityManager.h:



This graph shows which files directly or indirectly include this file:



Namespaces

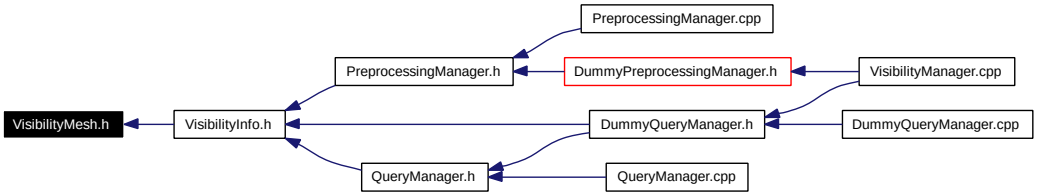
- namespace GtpVisibility

5.66 VisibilityMesh.h File Reference

```
#include "OgreMesh.h"
```

Include dependency graph for VisibilityMesh.h:

This graph shows which files directly or indirectly include this file:



Namespaces

- namespace GtpVisibility

Typedefs

- typedef Ogre::Mesh Mesh

5.66.1 Typedef Documentation

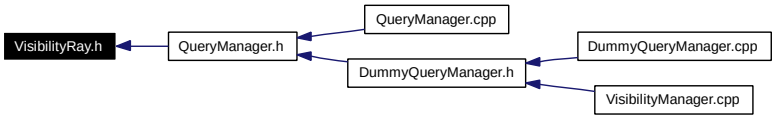
5.66.1.1 typedef Ogre::Mesh GtpVisibility::Mesh

5.67 VisibilityRay.h File Reference

```
#include "OgreRay.h"
```

Include dependency graph for VisibilityRay.h:

This graph shows which files directly or indirectly include this file:



Namespaces

- namespace GtpVisibility

Typedefs

- typedef Ogre::Ray Ray

5.67.1 Typedef Documentation

5.67.1.1 typedef Ogre::Ray GtpVisibility::Ray

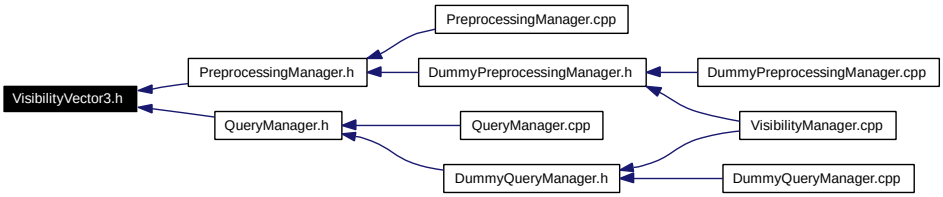
Ray class currently uses native Ogre ray when compiled with the Ogre platform

5.68 VisibilityVector3.h File Reference

```
#include "OgreVector3.h"
```

Include dependency graph for VisibilityVector3.h:

This graph shows which files directly or indirectly include this file:



Namespaces

- namespace GtpVisibility

Typedefs

- typedef Ogre::Vector3 Vector3

5.68.1 Typedef Documentation

5.68.1.1 typedef Ogre::Vector3 GtpVisibility::Vector3

Vector3 class currently uses the native Ogre vector when compiled with the Ogre platform

Chapter 6

GameTools Visibility Modules Class Index

6.1 GameTools Visibility Modules Class List

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

Chapter 7

GameTools Visibility Modules Namespace Index

7.1 GameTools Visibility Modules Namespace List

Here is a list of all namespaces with brief descriptions:

Chapter 8

GameTools Visibility Modules File Index

8.1 GameTools Visibility Modules File List

Here is a list of all files with brief descriptions:

Index