GPE

0.19

Generated by Doxygen 1.7.2

Sat Dec 4 2010 06:44:20

Contents

1 Main Page

This is the GPE (Game Programming Engine) documentation for version 0.19.

2 Class Index

2.1 Class Hierarchy

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

3 Class Index

3.1 Class List

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

4 File Index

4.1 File List

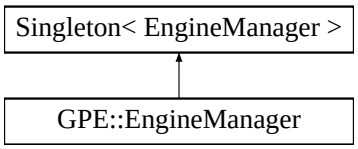
Here is a list of all documented files with brief descriptions:

5 Class Documentation

5.1 GPE::EngineManager Class Reference

```
#include <EngineManager.h>
```

Inheritance diagram for GPE::EngineManager:



Public Member Functions

- EngineManager ()
- ~EngineManager ()
- void tick ()
- int getGameState ()
- bool getShouldQuit ()
- Ogre::RenderWindow * getRenderWindow ()
- GPE::WorldManager * getWorldManager ()
- GPE::GuiManager * getGuiManager ()
- void setShouldQuit (bool value)

Static Public Member Functions

- static EngineManager & **getSingleton** (void)
- static EngineManager * **getSingletonPointer** (void)

Static Protected Attributes

- static EngineManager * **ms_Singleton**

Private Member Functions

- bool frameStarted (const Ogre::FrameEvent &frameEvent)
- void initialiseResourceManager ()
- void initialise ()

Private Attributes

- Ogre::Root * ogreRoot
- Ogre::RenderWindow * renderWindow
- GPE::GuiManager * guiManager
- GPE::InputManager * inputManager

- GPE::WorldManager * worldManager
- bool shouldQuit
- int gameState

5.1.1 Detailed Description

This class serves as the central clearing house, as it were.

Within the EngineManager class there are the instantiations of the manager objects. This allows all classes to be accessed through the centralized EngineManager class.

In addition the EngineManager class sets up the OGRE framework and handles the main loop.

Author

Gerard Prudhomme

Version

0.19

Date

Friday, December 3, 2010

See also

Singleton()
Ogre::FrameListener()

Definition at line 37 of file EngineManager.h.

5.1.2 Constructor & Destructor Documentation

5.1.2.1 GPE::EngineManager::EngineManager ( )

Constructor, sets all member variables to initialised states.

5.1.2.2 GPE::EngineManager::~EngineManager ( )

Destructor, garbage collection and clean up upon exit.

5.1.3 Member Function Documentation

5.1.3.1 bool GPE::EngineManager::frameStarted (const Ogre::FrameEvent & *frameEvent*) [private]

This function is called when a frame is started, and updates animations and camera position.

Parameters

frameEvent	the Ogre::FrameEvent object which contains frame information.

Returns

whether or not frame executed successfully

5.1.3.2 int GPE::EngineManager::getGameState ( ) [inline]

Get the game state, as represented by an integer.

Returns

the value of the gameState integer

Definition at line 48 of file EngineManager.h.

References gameState.

```
{ return gameState; }
```

5.1.3.3 GPE::GuiManager* GPE::EngineManager::getGuiManager ( ) [inline]

Accessor function to get a pointer to the guiManager object.

Returns

pointer to guiManager object.

Definition at line 64 of file EngineManager.h.

References guiManager.

```
{ return guiManager; }
```

5.1.3.4 Ogre::RenderWindow* GPE::EngineManager::getRenderWindow ( ) [inline]

Accessor function to get a pointer to the renderWindow object.

Returns

pointer to renderWindow object.

Definition at line 56 of file EngineManager.h.

References renderWindow.

```
{ return renderWindow; }
```

5.1.3.5 bool GPE::EngineManager::getShouldQuit ( ) [inline]

Accessor function to get the value of the shouldQuit boolean.

Returns

true if we should quit, false otherwise.

Definition at line 52 of file EngineManager.h.

References shouldQuit.

```
{ return shouldQuit; }
```

5.1.3.6 GPE::WorldManager* GPE::EngineManager::getWorldManager ( ) [inline]

Accessor function to get a pointer to the worldManager object.

Returns

pointer to worldManager object.

Definition at line 60 of file EngineManager.h.

References worldManager.

```
{ return worldManager; }
```

5.1.3.7 void GPE::EngineManager::initialise ( ) [private]

This subroutine handles all the set up apart from variable initialisation.

5.1.3.8 void GPE::EngineManager::initialiseResourceManager ( ) [private]

This subroutine loads all resource locations from "data/config/resources.cfg".

5.1.3.9 void GPE::EngineManager::setShouldQuit (bool *value*) [inline]

Mutator subroutine to set the value of the shouldQuit variable.

Parameters

value	true if we should quit, or false to continue running.

Definition at line 68 of file EngineManager.h.

References shouldQuit.

```
{ shouldQuit = value; }
```

5.1.3.10 void GPE::EngineManager::tick ( )

Handles the rendering of one frame, updates messages, and serves as the main loop.

5.1.4 Member Data Documentation

5.1.4.1 int GPE::EngineManager::gameState [private]

Integer holding the game state.

Definition at line 91 of file EngineManager.h.

Referenced by getGameState().

5.1.4.2 GPE::GuiManager* GPE::EngineManager::guiManager [private]

Controls the integration with Crazy Eddie's GUI Library (CEGUI).

Definition at line 83 of file EngineManager.h.

Referenced by getGuiManager().

5.1.4.3 GPE::InputManager* GPE::EngineManager::inputManager [private]

Controls the integration with the Object Oriented Input System Library (OIS).

Definition at line 85 of file EngineManager.h.

5.1.4.4 Ogre::Root* GPE::EngineManager::ogreRoot [private]

The OGRE root object.

Definition at line 79 of file EngineManager.h.

5.1.4.5 Ogre::RenderWindow* GPE::EngineManager::renderWindow [private]

The OGRE render window.

Definition at line 81 of file EngineManager.h.

Referenced by getRenderWindow().

5.1.4.6 bool GPE::EngineManager::shouldQuit [private]

Whether or not we should quit the application.

Definition at line 89 of file EngineManager.h.

Referenced by getShouldQuit(), and setShouldQuit().

5.1.4.7 GPE::WorldManager* GPE::EngineManager::worldManager [private]

Controls the 3D world, creation of objects, moving of the player, etc.

Definition at line 87 of file EngineManager.h.

Referenced by getWorldManager().

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

- EngineManager.h

5.2 GPE::GuiManager Class Reference

Public Member Functions

- GuiManager ()
- ~GuiManager ()
- void tick ()
- CEGUI::LuaScriptModule * getScriptModule ()
- Ogre::AnimationState * getAnimationState ()
- Ogre::SceneManager * getSceneManager ()

Private Member Functions

- void initialise ()
- void renderCharacter ()

Private Attributes

- CEGUI::OgreCEGUIRenderer * guiRenderer
- CEGUI::System * ceguiSystem
- CEGUI::LuaScriptModule * scriptModule
- Ogre::AnimationState * animationState
- Ogre::SceneManager * sceneManager
- int rttCameraZ
- int rttCharacterRotate

5.2.1 Detailed Description

Definition at line 28 of file GuiManager.h.

5.2.2 Constructor & Destructor Documentation

5.2.2.1 GPE::GuiManager::GuiManager ( )

Constructor, sets all member variables to initialised states.

5.2.2.2 GPE::GuiManager::~GuiManager ( )

Destructor, garbage collection and clean up upon exit.

5.2.3 Member Function Documentation

5.2.3.1 Ogre::AnimationState* GPE::GuiManager::getAnimationState ( ) [inline]

Accessor function to get a pointer to the animation state of the 3D character in the RTT scene.

Returns

pointer to the animation state of the 3D character in the RTT scene.

Definition at line 43 of file GuiManager.h.

References animationState.

```
{ return animationState; }
```

5.2.3.2 Ogre::SceneManager* GPE::GuiManager::getSceneManager ( ) [inline]

Accessor function to get a pointer to the RTT scene manager.

Returns

pointer to the RTT scene manager.

Definition at line 47 of file GuiManager.h.

References sceneManager.

```
{ return sceneManager; }
```

5.2.3.3 CEGUI::LuaScriptModule* GPE::GuiManager::getScriptModule ( ) [inline]

Accessor function to get a pointer to the scriptModule object.

Returns

pointer to the scriptModule object.

Definition at line 39 of file GuiManager.h.

References scriptModule.

```
{ return scriptModule; }
```

5.2.3.4 void GPE::GuiManager::initialise ( ) [private]

This subroutine handles all the set up apart from variable initialisation.

5.2.3.5 void GPE::GuiManager::renderCharacter ( ) [private]

This subroutine renders the 3D character to a texture, then uses texture in the GUI to give the appearance of a 3D object in a 2D interface.

5.2.3.6 void GPE::GuiManager::tick ( )

This subroutine handles the rendering of one frame, updates messages, and serves as the main loop.

5.2.4 Member Data Documentation

5.2.4.1 Ogre::AnimationState* GPE::GuiManager::animationState [private]

The Animation State for the GUI 3D character.

Definition at line 60 of file GuiManager.h.

Referenced by getAnimationState().

5.2.4.2 CEGUI::System* GPE::GuiManager::ceguiSystem [private]

The ceguiSystem object contains the CEGUI system.

Definition at line 56 of file GuiManager.h.

5.2.4.3 CEGUI::OgreCEGUIRenderer* GPE::GuiManager::guiRenderer [private]

The guiRenderer object which contains the bindings between OGRE and CEGUI.

Definition at line 54 of file GuiManager.h.

5.2.4.4 int GPE::GuiManager::rttCameraZ [private]

The Camera Z position (how far zoomed out we are) for the RTT camera.

Definition at line 64 of file GuiManager.h.

5.2.4.5 int GPE::GuiManager::rttCharacterRotate [private]

The amount of rotation to enact on the character being displayed using the RTT scene.

Definition at line 66 of file GuiManager.h.

5.2.4.6 Ogre::SceneManager* GPE::GuiManager::sceneManager [private]

Scene Manager for the RTT scene.

Definition at line 62 of file GuiManager.h.

Referenced by getSceneManager().

5.2.4.7 CEGUI::LuaScriptModule* GPE::GuiManager::scriptModule [private]

The Lua script module connection.

Definition at line 58 of file GuiManager.h.

Referenced by getScriptModule().

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

- GuiManager.h

5.3 GPE::InputManager Class Reference

Public Member Functions

- InputManager ()
- ~InputManager ()
- void setWindowExtents (int width, int height)
- void tick ()
- bool mouseMoved (const OIS::MouseEvent &mouseEvent)
- bool mousePressed (const OIS::MouseEvent &mouseEvent, OIS::MouseButtonID mouseButton)
- bool mouseReleased (const OIS::MouseEvent &mouseEvent, OIS::MouseButtonID mouseButton)
- bool keyPressed (const OIS::KeyEvent &keyEvent)
- bool keyReleased (const OIS::KeyEvent &keyEvent)

Private Member Functions

- void initialise ()

Private Attributes

- OIS::InputManager * oisInputManager
- OIS::Keyboard * oisKeyboard
- OIS::Mouse * oisMouse
- Ogre::Vector3 inputTransform
- float rotateSpeed

5.3.1 Detailed Description

Definition at line 30 of file InputManager.h.

5.3.2 Constructor & Destructor Documentation

5.3.2.1 GPE::InputManager::InputManager ( )

Constructor, sets all member variables to initialised states.

5.3.2.2 GPE::InputManager::~InputManager ( )

Destructor, garbage collection and clean up upon exit.

5.3.3 Member Function Documentation

5.3.3.1 void GPE::InputManager::initialise ( ) [private]

Handles all the set up apart from variable initialisation.

5.3.3.2 bool GPE::InputManager::keyPressed (const OIS::KeyEvent & *keyEvent*)

Is called whenever a key on the keyboard is pressed.

Parameters

keyEvent	OIS KeyEvent object, holds which key was pressed.

Returns

true if the input was processed correctly, false if not.

5.3.3.3 bool GPE::InputManager::keyReleased (const OIS::KeyEvent & *keyEvent*)

Is called whenever a key on the keyboard is released.

Parameters

keyEvent	OIS KeyEvent object, holds which key was released.

Returns

true if the input was processed correctly, false if not.

5.3.3.4 bool GPE::InputManager::mouseMoved (const OIS::MouseEvent & *mouseEvent*)

Is called whenever the mouse moves.

Parameters

mouseEvent	OIS MouseEvent object, holds mouse x,y,z coordinates.

Returns

true if the input was processed correctly, false if not.

5.3.3.5 bool GPE::InputManager::mousePressed (const OIS::MouseEvent & *mouseEvent,* OIS::MouseButtonID *mouseButton*)

Is called whenever a mouse button is pressed.

Parameters

mouseEvent	OIS MouseEvent object, holds mouse x,y,z coordinates.
mouseBut-ton	OIS MouseButtonID object, holds which mouse button was clicked.

Returns

true if the input was processed correctly, false if not.

5.3.3.6 bool GPE::InputManager::mouseReleased (const OIS::MouseEvent & *mouseEvent,* OIS::MouseButtonID *mouseButton*)

Is called whenever a mouse button is released.

Parameters

mouseEvent	OIS MouseEvent object, holds mouse x,y,z coordinates.
mouseBut-ton	OIS MouseButtonID object, holds which mouse button was clicked.

Returns

true if the input was processed correctly, false if not.

5.3.3.7 void GPE::InputManager::setWindowExtents (int *width,* int *height*)

Sets up the boundaries of our windows.

Parameters

width	the width of the window.
height	the height of the window.

5.3.3.8 void GPE::InputManager::tick ( )

Handles the rendering of one frame, updates messages, and serves as the main loop.

5.3.4 Member Data Documentation

5.3.4.1 Ogre::Vector3 GPE::InputManager::inputTransform [private]

inputTransform holds the current offset for the player character's movement.

Definition at line 80 of file InputManager.h.

5.3.4.2 OIS::InputManager* GPE::InputManager::oisInputManager [private]

The OIS Input Manager, helps to set up the OIS bindings

Definition at line 74 of file InputManager.h.

5.3.4.3 OIS::Keyboard* GPE::InputManager::oisKeyboard [private]

The OIS Keyboard object, digital representation of the user's keyboard.

Definition at line 76 of file InputManager.h.

5.3.4.4 OIS::Mouse* GPE::InputManager::oisMouse [private]

The OIS Mouse object, digital representation of the user's mouse.

Definition at line 78 of file InputManager.h.

5.3.4.5 float GPE::InputManager::rotateSpeed [private]

The speed at which the character will rotate in the GUI.

Definition at line 82 of file InputManager.h.

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

- InputManager.h

5.4 GPE::WorldManager Class Reference

Public Member Functions

- WorldManager ()
- ~WorldManager ()
- void tick ()
- bool getIsPlayerMoving ()
- bool getChangePlayerAnimation ()
- Ogre::AnimationState * getAnimationState ()
- Ogre::Entity * getModel ()
- Ogre::SceneNode * getModelNode ()
- Ogre::SceneManager * getSceneManager ()
- void setIsPlayerMoving (bool value)
- void setAnimationState (Ogre::AnimationState *value)
- void setChangePlayerAnimation (bool value)
- void modifyCameraZoom (Ogre::Real value)
- void modifyCameraPitch (Ogre::Real value)
- void modifyCameraYaw (Ogre::Real value)

Private Member Functions

- void initialise ()
- void loadGrass ()
- void loadTrees ()
- void createGrassMesh ()

Private Attributes

- bool isPlayerMoving
- bool changePlayerAnimation
- Ogre::Real cameraYaw
- Ogre::Real **cameraPitch**
- Ogre::Real **cameraZoom**
- Ogre::AnimationState * animationState
- Ogre::Entity * pcModel
- Ogre::SceneNode * pcModelNode
- Ogre::SceneNode * cameraNode
- Ogre::SceneManager * sceneManager
- Ogre::Camera * camera
- Ogre::Viewport * viewport

5.4.1 Detailed Description

Definition at line 27 of file WorldManager.h.

5.4.2 Constructor & Destructor Documentation

5.4.2.1 GPE::WorldManager::WorldManager ( )

Constructor, sets all member variables to initialised states.

5.4.2.2 GPE::WorldManager::~WorldManager ( )

Destructor, garbage collection and clean up upon exit.

5.4.3 Member Function Documentation

5.4.3.1 void GPE::WorldManager::createGrassMesh ( ) [private]

Creates the grass mesh, by manually creating a mesh.

5.4.3.2 Ogre::AnimationState* GPE::WorldManager::getAnimationState ( ) [inline]

Accessor function to get a pointer to the animation state of the 3D character in the world scene.

Returns

pointer to the animation state of the 3D character in the world scene.

Definition at line 46 of file WorldManager.h.

References animationState.

```
{ return animationState; }
```

5.4.3.3 bool GPE::WorldManager::getChangePlayerAnimation ( ) [inline]

Accessor function to get the value of the changePlayerAnimation boolean.

Returns

true if we need to change the player's animation, false otherwise.

Definition at line 42 of file WorldManager.h.

References changePlayerAnimation.

```
{ return changePlayerAnimation; }
```

5.4.3.4 bool GPE::WorldManager::getIsPlayerMoving ( ) [inline]

Accessor function to get the value of the isPlayerMoving boolean.

Returns

true if the player is moving, false otherwise.

Definition at line 38 of file WorldManager.h.

References isPlayerMoving.

```
{ return isPlayerMoving; }
```

5.4.3.5 Ogre::Entity* GPE::WorldManager::getModel ( ) [inline]

Accessor function to get a pointer to the model of the 3D character in the world scene.

Returns

pointer to the model of the 3D character in the world scene.

Definition at line 50 of file WorldManager.h.

References pcModel.

```
{ return pcModel; }
```

5.4.3.6 Ogre::SceneNode* GPE::WorldManager::getModelNode ( ) [inline]

Accessor function to get a pointer to the model node of the 3D character in the world scene.

Returns

pointer to the model node of the 3D character in the world scene.

Definition at line 54 of file WorldManager.h.

References pcModelNode.

```
{ return pcModelNode; }
```

5.4.3.7 Ogre::SceneManager* GPE::WorldManager::getSceneManager ( ) [inline]

Accessor function to get a pointer to the scene manager for the world scene.

Returns

pointer to the scene manager for the world scene.

Definition at line 58 of file WorldManager.h.

References sceneManager.

```
{ return sceneManager; }
```

5.4.3.8 void GPE::WorldManager::initialise ( ) [private]

Handles all the set up apart from variable initialisation.

5.4.3.9 void GPE::WorldManager::loadGrass ( ) [private]

Loads the grass, by randomly placing grass inside a grid on the terrain.

5.4.3.10 void GPE::WorldManager::loadTrees ( ) [private]

Loads the trees, by randomly placing trees (with variable scale) inside a grid on the terrain.

5.4.3.11 void GPE::WorldManager::modifyCameraPitch (Ogre::Real *value*) [inline]

Mutator subroutine to modify the value of the cameraPitch Ogre::Real.

Parameters

value	the amount of zoom we wish to modify.

Definition at line 78 of file WorldManager.h.

```
{ cameraPitch -= value; }
```

5.4.3.12 void GPE::WorldManager::modifyCameraYaw (Ogre::Real *value*) [inline]

Mutator subroutine to modify the value of the cameraYaw Ogre::Real.

Parameters

value	the amount of zoom we wish to modify.

Definition at line 82 of file WorldManager.h.

References cameraYaw.

```
{ cameraYaw -= value; }
```

5.4.3.13 void GPE::WorldManager::modifyCameraZoom (Ogre::Real *value*) [inline]

Mutator subroutine to modify the value of the cameraZoom Ogre::Real, and makes sure the value is not outside guidelines.

Parameters

value	the amount of zoom we wish to modify.

Definition at line 74 of file WorldManager.h.

```
{ cameraZoom -= value; if (cameraZoom < 10) { cameraZoom = 10; } else if (cameraZ
      oom > 250) { cameraZoom = 250; } }
```

5.4.3.14 void GPE::WorldManager::setAnimationState (Ogre::AnimationState * *value*) [inline]

Mutator subroutine to set the where the animationState points to.

Parameters

value	with a pointer to the animationState we wish to point to.

Definition at line 66 of file WorldManager.h.

References animationState.

```
{ animationState = value; }
```

5.4.3.15 void GPE::WorldManager::setChangePlayerAnimation (bool *value*) [inline]

Mutator subroutine to set the value of the changePlayerAnimation boolean.

Parameters

value	true if we need to change the player's animation, false otherwise.

Definition at line 70 of file WorldManager.h.

References changePlayerAnimation.

```
{ changePlayerAnimation = value; }
```

5.4.3.16 void GPE::WorldManager::setIsPlayerMoving (bool *value*) [inline]

Mutator subroutine to set the value of the isPlayerMoving boolean.

Parameters

value	true if the player is moving, false otherwise.

Definition at line 62 of file WorldManager.h.

References isPlayerMoving.

```
{ isPlayerMoving = value; }
```

5.4.3.17 void GPE::WorldManager::tick ( )

Handles the rendering of one frame, updates messages, and serves as the main loop.

5.4.4 Member Data Documentation

5.4.4.1 Ogre::AnimationState* GPE::WorldManager::animationState [private]

The current animation state of the player.

Definition at line 99 of file WorldManager.h.

Referenced by getAnimationState(), and setAnimationState().

5.4.4.2 Ogre::Camera* GPE::WorldManager::camera [private]

The camera object.

Definition at line 109 of file WorldManager.h.

5.4.4.3 Ogre::SceneNode* GPE::WorldManager::cameraNode [private]

The scene node that the camera is attached to.

Definition at line 105 of file WorldManager.h.

5.4.4.4 Ogre::Real GPE::WorldManager::cameraYaw [private]

The camera yaw, pitch, and zoom.

Definition at line 97 of file WorldManager.h.

Referenced by modifyCameraYaw().

5.4.4.5 bool GPE::WorldManager::changePlayerAnimation [private]

Has the player's animation changed.

Definition at line 95 of file WorldManager.h.

Referenced by getChangePlayerAnimation(), and setChangePlayerAnimation().

5.4.4.6 bool GPE::WorldManager::isPlayerMoving [private]

Is the player moving.

Definition at line 93 of file WorldManager.h.

Referenced by getIsPlayerMoving(), and setIsPlayerMoving().

5.4.4.7 Ogre::Entity* GPE::WorldManager::pcModel [private]

The player's character model.

Definition at line 101 of file WorldManager.h.

Referenced by getModel().

5.4.4.8 Ogre::SceneNode* GPE::WorldManager::pcModelNode [private]

The scene node that the player is attached to.

Definition at line 103 of file WorldManager.h.

Referenced by getModelNode().

5.4.4.9 Ogre::SceneManager* GPE::WorldManager::sceneManager [private]

The Scene Manager for the world scene.

Definition at line 107 of file WorldManager.h.

Referenced by getSceneManager().

5.4.4.10 Ogre::Viewport* GPE::WorldManager::viewport [private]

The Ogre viewport

Definition at line 111 of file WorldManager.h.

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

- WorldManager.h

5.5 GuiManager Class Reference

`#include <GuiManager.h>`

5.5.1 Detailed Description

Controls the integration with Crazy Eddie's GUI Library (CEGUI).

Within the GuiManager class are contained all the different objects necessary to set up and interact with CEGUI, this includes the Lua scripting module set up.

Author

Gerard Prudhomme

Version

0.19

Date

Friday, December 3, 2010

See also

Singleton()

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

- GuiManager.h

5.6 InputManager Class Reference

`#include <InputManager.h>`

5.6.1 Detailed Description

Controls the integration with the Object Oriented Input System Library (OIS)

Within the InputManager class are contained all the different objects necessary to set up and interact with OIS.

Author

Gerard Prudhomme

Version

0.19

Date

Friday, December 3, 2010

See also

Singleton()

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

- InputManager.h

5.7 Singleton< T > Class Template Reference

`#include <Singleton.h>`

Static Public Member Functions

- static T & **getSingleton** (void)
- static T * **getSingletonPointer** (void)

Static Protected Attributes

- static T * **ms_Singleton** = 0

5.7.1 Detailed Description

template<typename T> class Singleton< T >

Original version Copyright (C) Scott Bilas, 2000. All rights reserved worldwide.

This software is provided "as is" without express or implied warranties. You may freely copy and compile this source into applications you distribute provided that the copyright text below is included in the resulting source code, for example: "Portions Copyright (C) Scott Bilas, 2000"

Definition at line 14 of file Singleton.h.

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

- Singleton.h

5.8 WorldManager Class Reference

`#include <WorldManager.h>`

5.8.1 Detailed Description

Handles all controls related to the 3D world, and the movment of the player character.

Within the WorldManager class are contained the component which allow the terrain to be rendered, the trees to be placed, grass, etc.

Author

Gerard Prudhomme

Version

0.19

Date

Friday, December 3, 2010

See also

Singleton()

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

- WorldManager.h

6 File Documentation

6.1 EngineManager.h File Reference

```
#include "OgreFrameListener.h"
#include "Singleton.h"
```

Classes

- class GPE::EngineManager

6.1.1 Detailed Description

Copyright 2010 Gerard Prudhomme.

Author

Gerard Prudhomme

Version

0.19

Date

Friday, December 3, 2010

Definition in file EngineManager.h.

6.2 EngineManager.h

```
00001 #ifndef _ENGINE_MANAGER_H
00002 #define _ENGINE_MANAGER_H
00003
00016 /* Include section: */
00017 #include "OgreFrameListener.h"
00018 #include "Singleton.h"
00019 /* Forward declarations: */
00020 namespace Ogre { class Root; class RenderWindow; }
00021 namespace GPE { class GuiManager; class InputManager; class WorldManager;
00037 class EngineManager : public Singleton <EngineManager>, public Ogre::FrameListene
      r {
00038 public:
00040         EngineManager();
00042         ~EngineManager();
00044         void tick();
00048         int getGameState() { return gameState; }
```

```
00052         bool getShouldQuit() { return shouldQuit; }
00056         Ogre::RenderWindow *getRenderWindow() { return renderWindow; }
00060         GPE::WorldManager *getWorldManager() { return worldManager; }
00064         GPE::GuiManager *getGuiManager() { return guiManager; }
00068         void setShouldQuit ( bool value ) { shouldQuit = value; }
00069 private:
00073         bool frameStarted ( const Ogre::FrameEvent& frameEvent );
00075         void initialiseResourceManager();
00077         void initialise();
00079         Ogre::Root *ogreRoot;
00081         Ogre::RenderWindow *renderWindow;
00083         GPE::GuiManager *guiManager;
00085         GPE::InputManager *inputManager;
00087         GPE::WorldManager *worldManager;
00089         bool shouldQuit;
00091         int gameState;
00092 };
00093 } /* Namespace GPE */
00094 #endif  /* _ENGINE_MANAGER_H */
```

6.3 GuiManager.h File Reference

Classes

- class GPE::GuiManager

6.3.1 Detailed Description

Copyright 2010 Gerard Prudhomme.

Author

Gerard Prudhomme

Version

0.19

Date

Friday, December 3, 2010

Definition in file GuiManager.h.

6.4 GuiManager.h

```
00001 #ifndef _GUI_MANAGER_H
00002 #define _GUI_MANAGER_H
00003
```

```
00011 /* Include section: */
00012 /* Forward declarations: */
00013 namespace CEGUI { class LuaScriptModule; class OgreCEGUIRenderer; class System; }

00014 namespace Ogre { class AnimationState; class SceneManager; }
00027 namespace GPE {
00028 class GuiManager {
00029 public:
00031         GuiManager();
00033         ~GuiManager();
00035         void tick();
00039         CEGUI::LuaScriptModule *getScriptModule() { return scriptModule; }
00043         Ogre::AnimationState *getAnimationState() { return animationState; }
00047         Ogre::SceneManager *getSceneManager() { return sceneManager; }
00048 private:
00050         void initialise();
00052         void renderCharacter();
00054         CEGUI::OgreCEGUIRenderer *guiRenderer;
00056         CEGUI::System *ceguiSystem;
00058         CEGUI::LuaScriptModule *scriptModule;
00060         Ogre::AnimationState *animationState;
00062         Ogre::SceneManager *sceneManager;
00064         int rttCameraZ;
00066         int rttCharacterRotate;
00067 };
00068 } /* Namespace GPE */
00069 #endif  /* _GUI_MANAGER_H */
```

6.5 InputManager.h File Reference

```
#include "OgreVector3.h"
#include "OIS\OISMouse.h"
#include "OIS\OISKeyboard.h"
```

Classes

- class GPE::InputManager

6.5.1 Detailed Description

Copyright 2010 Gerard Prudhomme.

Author

Gerard Prudhomme

Version

0.19

Date

Friday, December 3, 2010

Definition in file InputManager.h.

6.6 InputManager.h

```
00001 #ifndef _INPUT_MANAGER_H
00002 #define _INPUT_MANAGER_H
00003
00011 /* Include section: */
00012 #include "OgreVector3.h"
00013 #include "OIS\OISMouse.h"
00014 #include "OIS\OISKeyboard.h"
00015 /* Forward declarations: */
00016 namespace OIS { class MouseEvent; class KeyEvent; class InputManager; }
00029 namespace GPE {
00030 class InputManager : public OIS::MouseListener, public OIS::KeyListener {
00031 public:
00033         InputManager();
00035         ~InputManager();
00040         void setWindowExtents( int width, int height ) ;
00042         void tick();
00047         bool mouseMoved( const OIS::MouseEvent &mouseEvent );
00053         bool mousePressed( const OIS::MouseEvent &mouseEvent, OIS::MouseButtonID
      mouseButton );
00059         bool mouseReleased( const OIS::MouseEvent &mouseEvent, OIS::MouseButtonID
       mouseButton );
00064         bool keyPressed( const OIS::KeyEvent &keyEvent );
00069         bool keyReleased( const OIS::KeyEvent &keyEvent );
00070 private:
00072         void initialise();
00074         OIS::InputManager *oisInputManager;
00076         OIS::Keyboard *oisKeyboard;
00078         OIS::Mouse *oisMouse;
00080         Ogre::Vector3 inputTransform;
00082         float rotateSpeed;
00083 };
00084 } /* Namespace GPE */
00085 #endif  /* _INPUT_MANAGER_H */
```

6.7 WorldManager.h File Reference

```
#include "OgreMath.h"
```

Classes

- class GPE::WorldManager

6.7.1 Detailed Description

Copyright 2010 Gerard Prudhomme.

Author

Gerard Prudhomme

Version

0.19

Date

Friday, December 3, 2010

Definition in file WorldManager.h.

6.8 WorldManager.h

```
00001 #ifndef _WORLD_MANAGER_H
00002 #define _WORLD_MANAGER_H
00003
00011 /* Include section: */
00012 #include "OgreMath.h"
00013 /* Forward declarations: */
00014 namespace Ogre { class SceneNode; class AnimationState; class Entity; class Scene
      Manager; class Camera; class Viewport; }
00026 namespace GPE {
00027 class WorldManager {
00028 public:
00030         WorldManager();
00032         ~WorldManager();
00034         void tick();
00038         bool getIsPlayerMoving() { return isPlayerMoving; }
00042         bool getChangePlayerAnimation() { return changePlayerAnimation; }
00046         Ogre::AnimationState *getAnimationState() { return animationState; }
00050         Ogre::Entity *getModel() { return pcModel; }
00054         Ogre::SceneNode *getModelNode() { return pcModelNode; }
00058         Ogre::SceneManager *getSceneManager() { return sceneManager; }
00062         void setIsPlayerMoving ( bool value ) { isPlayerMoving = value; }
00066         void setAnimationState ( Ogre::AnimationState *value) { animationState =
      value; }
00070         void setChangePlayerAnimation ( bool value ) { changePlayerAnimation = va
      lue; }
00074         void modifyCameraZoom ( Ogre::Real value ) { cameraZoom -= value; if (cam
      eraZoom < 10) { cameraZoom = 10; } else if (cameraZoom > 250) { cameraZoom = 250;
       } }
00078         void modifyCameraPitch ( Ogre::Real value ) { cameraPitch -= value; }
00082         void modifyCameraYaw ( Ogre::Real value ) { cameraYaw -= value; }
00083 private:
00085         void initialise();
00087         void loadGrass();
```

```
00089         void loadTrees();
00091         void createGrassMesh();
00093         bool isPlayerMoving;
00095         bool changePlayerAnimation;
00097         Ogre::Real cameraYaw, cameraPitch, cameraZoom;
00099         Ogre::AnimationState *animationState;
00101         Ogre::Entity *pcModel;
00103         Ogre::SceneNode *pcModelNode;
00105         Ogre::SceneNode *cameraNode;
00107         Ogre::SceneManager *sceneManager;
00109         Ogre::Camera *camera;
00111         Ogre::Viewport *viewport;
00112 };
00113 }
00114 #endif  /* _WORLD_MANAGER_H */
```

Index