Simple Game Object

Generated by Doxygen 1.8.14

Contents

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

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

Chapter 2

Class Index

2.1 Class List

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

Chapter 3

Class Documentation

3.1 Game Class Reference

`#include <Game.h>`

Inheritance diagram for Game:

Public Member Functions

- Game ()
- virtual ~Game ()
- void setup ()
- void setupCamera ()
- void setupBoxMesh ()
- void setupBoxMesh2 ()
- void setupPlayer ()
- void setupFloor ()
- void setupLights ()
- bool keyPressed (const KeyboardEvent &evt)
- bool mouseMoved (const MouseMotionEvent &evt)
- bool frameStarted (const FrameEvent &evt)
- bool frameEnded (const FrameEvent &evt)
- void bulletInit ()

3.1.1 Detailed Description

Example Games class. Based (very heavily) on the Ogre3d examples. Even uses OgreBytes (which I'd like to remove).

3.1.2 Constructor & Destructor Documentation

3.1.2.1 Game()

```
Game::Game ( )
```

Creates the object, sets all pointers to nullptr.

3.1.2.2 ~Game()

```
Game::~Game ( )  [virtual]
```

Distructor (virtual), as this is virtual that of the sub class will also be called. ----cleanup_start----

3.1.3 Member Function Documentation

3.1.3.1 bulletInit()

```
void Game::bulletInit ( )
```

Sets up the bullet environment collision configuration contains default setup for memory, collision setup. Advanced users can create their own configuration.

use the default collision dispatcher. For parallel processing you can use a diffent dispatcher (see Extras/Bullet↩ MultiThreaded)

btDbvtBroadphase is a good general purpose broadphase. You can also try out btAxis3Sweep.

the default constraint solver. For parallel processing you can use a different solver (see Extras/BulletMultiThreaded)

3.1.3.2 frameEnded()

```
bool Game::frameEnded (
            const FrameEvent & evt )
```

Ogre wraps the game loop, but we've registered as being interested in FrameEvents (through inheritance). This method is called by the framework after rendering the frame.

Parameters

evt,FrameEvent.	

3.1.3.3 frameStarted()

```
bool Game::frameStarted (
            const FrameEvent & evt )
```

Ogre wraps the game loop, but we've registered as being interested in FrameEvents (through inheritance). This method is called by the framework before rendering the frame.

Parameters

evt,FrameEvent.	

3.1.3.4 keyPressed()

```
bool Game::keyPressed (
            const KeyboardEvent & evt )
```

Overload of the keyPressed method.

Parameters

evt,a	KeyboardEvent

3.1.3.5 mouseMoved()

```
bool Game::mouseMoved (
            const MouseMotionEvent & evt )
```

Overload of the mouseMoved method.

Parameters

evt,a	KeyboardEvent

3.1.3.6 setup()

```
void Game::setup ( )
```

Carries out all setup, includes lighting, scene objects.

3.1.3.7 setupBoxMesh()

```
void Game::setupBoxMesh ( )
```

Quick and dirty box mesh, essentally this is a mix of the Ogre code to setup a box - from example. Added to this is the setup for the bullet3 collision box and rigid body. Create Dynamic Objects

3.1.3.8 setupBoxMesh2()

```
void Game::setupBoxMesh2 ( )
```

A copy of a quick and dirty box mesh, essentally this is a mix of the Ogre code to setup a box - from example. Added to this is the setup for the bullet3 collision box and rigid body. Create Dynamic Objects

3.1.3.9 setupCamera()

```
void Game::setupCamera ( )
```

Sets up the camera

3.1.3.10 setupFloor()

```
void Game::setupFloor ( )
```

Turns on on the coffee machine.

3.1.3.11 setupLights()

```
void Game::setupLights ( )
```

Creates, lights and adds them to the scene. All based on the sample code, needs moving out into a level class.

3.1.3.12 setupPlayer()

```
void Game::setupPlayer ( )
```

Player setup, this tests the Player object.

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

- /home/gjenkins/Documents/Development/o3d_1-11_bull_3_base/src/Game.h
- /home/gjenkins/Documents/Development/o3d_1-11_bull_3_base/src/Game.cpp

3.2 Player Class Reference

```
#include <Player.h>
```

Public Member Functions

- void createMesh (SceneManager *scnMgr)
- void attachToNode (SceneNode *parent)
- void setScale (float x, float y, float z)
- void setRotation (Vector3 axis, Radian angle)
- void setPosition (float x, float y, float z)
- void boundingBoxFromOgre ()
- void createRigidBody (float mass)
- void addToCollisionShapes (btAlignedObjectArray< btCollisionShape *> &collisionShapes)
- void addToDynamicsWorld (btDiscreteDynamicsWorld *dynamicsWorld)
- void setMass (float mass)
- void update ()

3.2.1 Detailed Description

Example player class. Written to illistrate the connection of Ogre/Bullet. Essentially just a wrapper around the cube object setup code.

3.2.2 Member Function Documentation

3.2.2.1 addToCollisionShapes()

```
void Player::addToCollisionShapes (
            btAlignedObjectArray< btCollisionShape *> & collisionShapes )
```

Add this collision shape to the collision shapes list

Parameters

collisionShaps,the	list of collision shapes (shared with the physics world).

3.2.2.2 addToDynamicsWorld()

```
void Player::addToDynamicsWorld (
            btDiscreteDynamicsWorld * dynamicsWorld )
```

Add this rigid body to the dynamicsWorld.

Parameters

dynamicsWorld,the	wrold we're going to add ourselves to.

3.2.2.3 attachToNode()

```
void Player::attachToNode (
            SceneNode * parent )
```

Creates a new child of the given parent node, adds the mesh to it.

Parameters

parent,the	parent (in the scene graph) of the node the player will be attatched to.

3.2.2.4 boundingBoxFromOgre()

```
void Player::boundingBoxFromOgre ( )
```

Fudge to get the bouning box from Ogre3d, at a it might work for other shapes.

3.2.2.5 createMesh()

```
void Player::createMesh (
            SceneManager * scnMgr )
```

Creates the mesh.

Parameters

scnMgr	the Ogre SceneManager.

3.2.2.6 createRigidBody()

```
void Player::createRigidBody (
            float mass )
```

Creates a new ridgid body of the given mass.

Parameters

mass	

Create Dynamic Objects

3.2.2.7 setMass()

```
void Player::setMass (
            float mass )
```

What on Earth is this for!? Can't change the mass of a rigid body.

Parameters

mass	

3.2.2.8 setPosition()

```
void Player::setPosition (
            float x,
            float y,
            float z )
```

Sets the position.

Parameters

x,position	on the x axis.
y,position	on the y axis.
z,position	on the z axis.

3.2.2.9 setRotation()

```
void Player::setRotation (
            Vector3 axis,
            Radian angle )
```

Sets the orientation.

Parameters

axis,vector	about which the orientation takes place.
angle,angle	(in radians).

3.2.2.10 setScale()

```
void Player::setScale (
            float x,
```

```
            float y,
            float z )
```

Sets the scale.

Parameters

x,scale	on the x axis.
y,scale	on the y axis.
z,scale	on the z axis.

3.2.2.11 update()

```
void Player::update ( )
```

Update, um ... makes coffee.

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

- /home/gjenkins/Documents/Development/o3d_1-11_bull_3_base/src/Player.h
- /home/gjenkins/Documents/Development/o3d_1-11_bull_3_base/src/Player.cpp