PhysGameEngine

.01

Generated by Doxygen 1.6.1

Fri Mar 26 03:09:54 2010

Contents

Chapter 1

Physgame

The Physgame engine is an abstraction layer between less portable, less user friendly, more sophistciated libraries and the game you want to make. If we do our jobs right this will save time and effort porting games between a variety of platforms. If you link only against this library, not a single line of your Standard compliant C++ code should need to change between platforms. At this early stage we are proving the concept with "Catch!" our first sample game. It Currently runs on Linux and Windows with an Identical codebase, when we are done with "Catch!" We want it to have one codebase, and downloadable in the Iphone app store, the Xbox store, on the PS3, on Steam, and in a variety of linux repositories.

To get the latest news on development checkout: http://gitorious.org/physgame The wiki Which acts as our current Knowledge base: http://gitorious.org/physgame/pages/Home

1.1 Structure

Main Loop Flow

Call Back Manager

Event Manager

Items in the world - Actor Class

Chapter 2

MainLoop

The MainLoop does stuffs that needs to be documented

Todo

actually document the gameloop

Chapter 3

Todo List

Page actually document the gameloop

Chapter 4

Class Index

4.1 Class Hierarchy

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

Chapter 5

Class Index

5.1 Class List

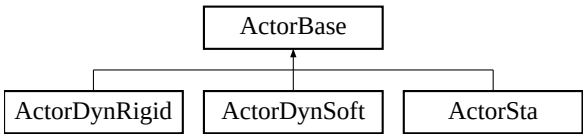
Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 6

Class Documentation

6.1 ActorBase Class Reference

Inheritance diagram for ActorBase::



Public Member Functions

- **ActorBase** (PhysString name, PhysString file)
- void **SetLocation** (PhysReal x, PhysReal y, PhysReal z)
- void **SetLocation** (PhysVector3 Place)
- void **SetOrientation** (PhysReal x, PhysReal y, PhysReal z, PhysReal w)
- void **SetOrientation** (PhysQuaternion Rotation)
- void **AttachToGraphics** ()

Protected Member Functions

- virtual void **AddObjectToWorld** (PhysWorld ∗TargetWorld, btDiscreteDynamicsWorld ∗TargetPhysicsWorld)=0
- void **CreateEntity** (PhysString name, PhysString file, PhysString group)
- void **CreateSceneNode** ()
- void **SetOgreLocation** (PhysVector3 Place)
- void **SetOgreOrientation** (PhysReal x, PhysReal y, PhysReal z, PhysReal w)
- void **SetBulletLocation** (PhysVector3 Location)

Protected Attributes

- Ogre::Entity ∗ **entity**
- Ogre::SceneManager ∗ **physscenemanager**

- Ogre::SceneNode * **node**
- btDefaultMotionState * **MotionState**
- btCollisionShape * **Shape**

Friends

- class PhysWorld

6.1.1 Detailed Description

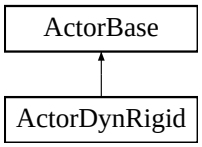
Definition at line 30 of file physactor.h.

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

- physactor.h
- physactor.cpp

6.2 ActorDynRigid Class Reference

Inheritance diagram for ActorDynRigid::



Public Member Functions

- void **CreateRigidObject** ()

Protected Member Functions

- void **AddObjectToWorld** (PhysWorld *TargetWorld, btDiscreteDynamicsWorld *TargetPhysicsWorld)

Protected Attributes

- btRigidBody * **physrigidbody**
- btMotionState * **physmotionstate**

6.2.1 Detailed Description

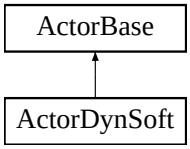
Definition at line 72 of file physactor.h.

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

- physactor.h
- physactor.cpp

6.3 ActorDynSoft Class Reference

Inheritance diagram for ActorDynSoft::



Public Member Functions

- void **CreateSoftObject** ()

Protected Member Functions

- void **AddObjectToWorld** (PhysWorld *TargetWorld, btDiscreteDynamicsWorld *TargetPhysicsWorld)

Protected Attributes

- btSoftBody * **physoftbody**
- btMotionState * **physmotionstate**

6.3.1 Detailed Description

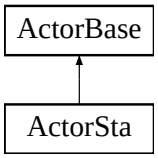
Definition at line 85 of file physactor.h.

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

- physactor.h
- physactor.cpp

6.4 ActorSta Class Reference

Inheritance diagram for ActorSta::



Public Member Functions

- void **CreateRigidObject** ()

Protected Member Functions

- void **AddObjectToWorld** (PhysWorld *TargetWorld, btDiscreteDynamicsWorld *TargetPhysicsWorld)

Protected Attributes

- btRigidBody * **physrigidbody**

6.4.1 Detailed Description

Definition at line 98 of file physactor.h.

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

- physactor.h
- physactor.cpp

6.5 MetaCode Class Reference

Public Member Functions

- **MetaCode** (int MetaValue_, short unsigned int ID_, InputCode Code_)
- **MetaCode** (RawEvent _RawEvent)
- InputCode **GetCode** ()
- void **SetCode** (InputCode Code_)
- int **GetMetaValue** ()
- void **SetMetaValue** (int MetaValue_)
- short unsigned int **GetID** ()
- void **SetID** (short unsigned int ID_)
- bool **operator==** (const MetaCode &other) const

6.5.1 Detailed Description

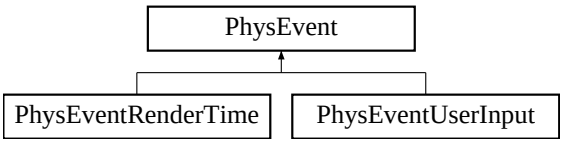
Definition at line 330 of file physeventuserinput.h.

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

- physeventuserinput.h
- physeventuserinput.cpp

6.6 PhysEvent Class Reference

Inheritance diagram for PhysEvent::



Public Member Functions

- virtual EventType **getEventType** ()=0

6.6.1 Detailed Description

Definition at line 21 of file physevent.h.

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

- physevent.h

6.7 PhysEventManager Class Reference

Public Member Functions

- unsigned int **GetRemainingEventCount** ()
- PhysEvent * **GetNextEvent** ()
- PhysEventRenderTime * **GetNextRenderTimeEvent** ()
- PhysEventUserInput * **GetNextUserInputEvent** ()
- void **AddEvent** (PhysEvent *EventToAdd)
- bool **DoQuitMessagesExist** ()

Static Public Member Functions

- static bool **IgnoreQuitEvents** ()
- static void **SetIgnoreQuitEvents** (bool Ignore)

6.7.1 Detailed Description

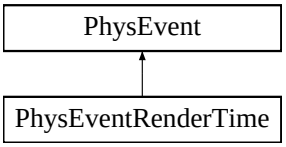
Definition at line 16 of file physeventmanager.h.

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

- physeventmanager.h
- physeventmanager.cpp

6.8 PhysEventRenderTime Class Reference

Inheritance diagram for PhysEventRenderTime::



Public Member Functions

- **PhysEventRenderTime** (PhysWhole Milliseconds)
- virtual EventType **getEventType** ()
- PhysWhole **getMilliSecondsSinceLastFrame** ()

6.8.1 Detailed Description

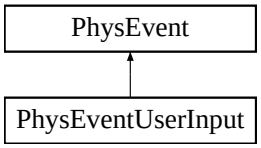
Definition at line 13 of file physeventrendertime.h.

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

- physeventrendertime.h
- physeventrendertime.cpp

6.9 PhysEventUserInput Class Reference

Inheritance diagram for PhysEventUserInput::



Public Member Functions

- **PhysEventUserInput** (MetaCode Code_)
- **PhysEventUserInput** (vector< MetaCode > Code_)
- MetaCode **GetCode** (unsigned int Index)
- unsigned int **GetCodeCount** ()
- void **AddCode** (MetaCode _Code)
- void **ToggleCode** (MetaCode _Code)
- void **ToggleCode** (unsigned int Index)
- virtual EventType **getEventType** ()

6.9.1 Detailed Description

Definition at line 353 of file physeventuserinput.h.

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

- physeventuserinput.h
- physeventuserinput.cpp

6.10 PhysQuaternion Class Reference

Public Member Functions

- **PhysQuaternion** (PhysReal X, PhysReal Y, PhysReal Z, PhysReal W)
- btVector3 **GetBulletQuaternion** ()
- Ogre::Quaternion **GetOgreQuaternion** ()

Public Attributes

- PhysReal **X**
- PhysReal **Y**
- PhysReal **Z**
- PhysReal **W**

6.10.1 Detailed Description

Definition at line 12 of file physquaternion.h.

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

- physquaternion.h
- physquaternion.cpp

6.11 PhysVector3 Class Reference

Public Member Functions

- **PhysVector3** (PhysReal X, PhysReal Y, PhysReal Z)
- btVector3 **GetBulletVector3** ()
- Ogre::Vector3 **GetOgreVector3** ()

Public Attributes

- PhysReal **X**
- PhysReal **Y**
- PhysReal **Z**

6.11.1 Detailed Description

Definition at line 13 of file physvector.h.

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

- physvector.h
- physvector.cpp

6.12 PhysWorld Class Reference

This is the main entry point for the entire library. The physworld coordinates and integrates all the underlying subsystems, Currently Ogre3d is used for 3d Graphics, Bullet is used for physics, and SDL is used for user input and window management. Games will need a container for all the playing pieces. It makes sense to tie all of this functionality into one world object.

`#include <physworld.h>`

Public Member Functions

- PhysWorld (PhysVector3 *GeographyLowerBounds, PhysVector3 *GeographyUpperbounds, unsigned short int MaxPhysicsProxies=1024)

  Descriptive constructor This constructor allows for an easier way to define the boundaries for items moving about inside the physworld.

- PhysWorld ()

  Default constructor This simply performs the same work as the descriptive constructor with some sane, but small, limits. It will give you a world which expands for 100 units from the Origin, and only allows 10 Adows.

- ~PhysWorld ()

  Deconstructor This Tears down all the items create by the physworld, and safely frees any graphical resources, we will also delete any Objects passed into the Physworld by pointer. We will not delete any pointers we pass out (like from the Events from the Event manager).

- template<class T >
  void Log (T Message)

  Runtime Event logging Function.

- template<class T >
  void LogAndThrow (T Message)

  This is the preffered way to throw an exception currently.

- bool ShowSystemSettingDialog ()

  This Shows an Engine Generated Configuration Screen This could look like and could offer just about any option to the user. It is loosely expected to show Graphical Configuration options, like Vsync and Resolution, But it might ask some really silly stuff. I thnk this would be fine for smaller simpler Which have no other way to configure such things, but any sizable project should develop their own way to expose and manage user settings.

- void MoveCamera (PhysVector3 Position, PhysVector3 LookAt)

  This moves the camera relative to the world.

- void GameInit ()

  This creates the game window and starts the game. Prior to this all of the physics and graphical object containers should have been loaded and prepared for use. There should be minimal delay from the time you call this and the game actually begins. This is also where the Main Loop for the game is housed.

- void DoMainLoopAllItems ()

Performs all the items that would normally be performed during the game loop This simply calls: DoMainLoopPhysics, DoMainLoopInputBuffering, DoMainLoopWindowManagerBuffering, DoMainLoopRender. This is useful for anyone wants to use as little of the existing main loop structure as possible, or does not want to run a certain Items each iteration of the main loop.

- void DoMainLoopPhysics ()

  Increments physics by one step Currently one step is about 1/60 of a second. This function is automatically called in the main loop if a Pre-Physics Callback is set. This is the second step in the main loop chain of events. This is where we expect the majority of our colision events to come from although it is concievable that a game could manually insert those manually.

- void **DoMainLoopInputBuffering** ()
- void **DoMainLoopWindowManagerBuffering** ()
- void **DoMainLoopRender** ()
- void **AddActor** (ActorBase *ActorToAdd)

Public Attributes

- PhysWorldCallBackManager * **CallBacks**
- PhysEventManager * **Events**

Friends

- void RenderPhysWorld (PhysWorld *TheWorld)

6.12.1 Detailed Description

This is the main entry point for the entire library. The physworld coordinates and integrates all the underlying subsystems, Currently Ogre3d is used for 3d Graphics, Bullet is used for physics, and SDL is used for user input and window management. Games will need a container for all the playing pieces. It makes sense to tie all of this functionality into one world object.

Definition at line 81 of file physworld.h.

6.12.2 Constructor & Destructor Documentation

6.12.2.1 PhysWorld::PhysWorld (PhysVector3 * *GeographyLowerBounds*, PhysVector3 * *GeographyUpperbounds*, unsigned short int *MaxPhysicsProxies* = `1024`)

Descriptive constructor This constructor allows for an easier way to define the boundaries for items moving about inside the physworld.

Parameters:

GeographyLowerBounds The lower limits for the size of the physics simulation

GeographyUpperbounds The Upper limits for the size of the physics simulation

MaxPhysicsProxies This is the amount of Adows (Also called Actors or Proxies) allowed in a physics simulation.

Definition at line 44 of file physworld.cpp.

6.12.3 Member Function Documentation

6.12.3.1 template<class T > void PhysWorld::Log (T *Message*) [inline]

Runtime Event logging Function.

Parameters:

Message This is what will be streamed to the log Be careful with this function, even though it appears to be a template, it does not support every data type. If Physgame is Compiled as a Shared Object, Dynamic Linked Library, or some other kind of stand alone library It will only support data types that are called internally, Currently that list includes: string, char, short int, int, long int, unsigned short int, unsigned int unsigned long int, bool, float, double, long double, wchar_t, size_t, PhysReal, PhysWhole, PhysString, and PhysVector3. If compiled statically it should support any data type which supports output streams.

Definition at line 180 of file physworld.cpp.

6.12.3.2 template<class T > void PhysWorld::LogAndThrow (T *Message*) [inline]

This is the preffered way to throw an exception currently.

Parameters:

Message This will be streamed to the log, then used in a thrown exception. This will log the Message, and will throw an exception with the Message included. Currently this supports all the Data type the Log function supports

Definition at line 187 of file physworld.cpp.

6.12.3.3 void PhysWorld::MoveCamera (PhysVector3 *Position*, PhysVector3 *LookAt*)

This moves the camera relative to the world.

Parameters:

Position Where should the camera be seated

LookAt Point the camera such that this poin is centered on the screen The parameters really do explain it. This puts the camera at an arbitrary point, pointing at an arbitrary point.

Definition at line 293 of file physworld.cpp.

6.12.4 Friends And Related Function Documentation

6.12.4.1 void RenderPhysWorld (PhysWorld * *TheWorld*) [friend]

Do Not Use this, This should be treated as an internal function, it is **subject to change without warning** and could be **harmful** to overall stability if used incorrectly

Warning:

This should be treated as an internal function, it is **subject to change without warning** and could be **harmful** to overall stability if used incorrectly

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

- physworld.h
- physworld.cpp

6.13 PhysWorldCallBackManager Class Reference

Public Member Functions

- **PhysWorldCallBackManager** (PhysWorld *_Parent)
- bool **PreInput** ()
- void **ErasePreInput** ()
- void **SetPreInput** (bool(*Callback)())
- bool **IsPreInputCallbackSet** ()
- bool **PrePhysics** ()
- void **ErasePrePhysics** ()
- void **SetPrePhysics** (bool(*Callback)())
- bool **IsPrePhysicsCallbackSet** ()
- bool **PreRender** ()
- void **ErasePreRender** ()
- void **SetPreRender** (bool(*Callback)())
- bool **IsPreRenderCallbackSet** ()
- bool **PostRender** ()
- void **ErasePostRender** ()
- void **SetPostRender** (bool(*Callback)())
- bool **IsPostRenderCallbackSet** ()

Friends

- class PhysWorld

6.13.1 Detailed Description

Definition at line 13 of file physworldcallbackmanager.h.

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

- physworldcallbackmanager.h
- physworldcallbackmanager.cpp

6.14 Settings Class Reference

Public Member Functions

- bool **getFullscreen** ()
- bool **setFullscreen** (bool _Fullscreen)
- int **getRenderHeight** ()
- int **getRenderWidth** ()
- bool **setRenderHeight** (int Height)
- bool **setRenderWidth** (int Width)
- bool **getFullscreen** ()
- bool **setFullscreen** (bool _Fullscreen)
- int **getRenderHeight** ()
- int **getRenderWidth** ()
- bool **setRenderHeight** (int Height)
- bool **setRenderWidth** (int Width)

6.14.1 Detailed Description

Definition at line 13 of file gamebase.h.

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

- gamebase.h
- physgamesettings.h
- gamebase.cpp
- physgamesettings.cpp

Index