Super Eastgate

1.0

Generated by Doxygen 1.6.1

Sun Apr 11 19:42:53 2010

Contents

Chapter 1

Todo List

Member LevelObject::Reload() : Refactor to remove calls to this method

File Point.h Merge this file into GameStructs

File RgbaColor.h Merge this file into GameStructs.h

Chapter 2

Namespace Index

2.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Chapter 3

Class Index

3.1 Class Hierarchy

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

Chapter 4

Class Index

4.1 Class List

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

Chapter 5

File Index

5.1 File List

Here is a list of all files with brief descriptions:

Chapter 6

Namespace Documentation

6.1 Converter Namespace Reference

Conversion Functions.

Classes

- class ConverterException
  Converter Exception.

Functions

- std::string IntToString (const int &value)
- std::string UIntToString (const unsigned int &value)
- int StringToInt (const std::string &value)
- double StringToDouble (const std::string &value)
- unsigned int StringToUInt (const std::string &value)
- std::wstring StringToWString (const std::string &stringToConvert)
- std::string WStringToString (const std::wstring &stringToConvert)

6.1.1 Detailed Description

Conversion Functions. This is the converter utility functions This namespace supplies funtions to perform basic string conversions

6.1.2 Function Documentation

6.1.2.1 std::string Converter::IntToString (const int & *value*)

Function to convert an interger into a string

Parameters:

value integer to convert

Returns:

interger as a string

Definition at line 6 of file Converter.cpp.

6.1.2.2 double Converter::StringToDouble (const std::string & *value*)

Function to convert a string into a double

Parameters:

value string to convert

Returns:

double

Definition at line 48 of file Converter.cpp.

6.1.2.3 int Converter::StringToInt (const std::string & *value*)

Function to convert a string into a signed integer

Parameters:

value string to convert

Returns:

signed interger

Definition at line 20 of file Converter.cpp.

6.1.2.4 unsigned int Converter::StringToUInt (const std::string & *value*)

Function to convert a string into an unsigned integer

Parameters:

value string to convert

Returns:

unsigned interger

Definition at line 34 of file Converter.cpp.

6.1.2.5 std::wstring Converter::StringToWString (const std::string & *stringToConvert*)

Function to convert a string into a wstring

Parameters:

stringToConvert string to convert

Returns:

convert string

Definition at line 62 of file Converter.cpp.

6.1.2.6 std::string Converter::UIntToString (const unsigned int & *value*)

Function to convert an unsigned interger into a string

Parameters:

value unsigned integer to convert

Returns:

interger as a string

Definition at line 13 of file Converter.cpp.

6.1.2.7 std::string Converter::WStringToString (const std::wstring & *stringToConvert*)

Function to convert a wstring into a string

Parameters:

stringToConvert string to convert

Returns:

convert string

Definition at line 69 of file Converter.cpp.

6.2 GameLoader Namespace Reference

Used to convert images into openGL texture ids.

Functions

- bool RunLoader (const std::wstring &worldsFileName, std::list< WorldObject * > &list, GameDude *dude)
- bool LoadLevel (const std::wstring &levelFileName, LevelObject *level)
- Square GameGridToCoords (double x, double y)

6.2.1 Detailed Description

Used to convert images into openGL texture ids. This namespace supplies the neccessary functions to load All game control objects from the given files

6.2.2 Function Documentation

6.2.2.1 Square GameLoader::GameGridToCoords (double *x*, double *y*)

Method to convert from the game grid to openGL coords

Parameters:

x X coord of the grid

y Y corrd of the grid

Returns:

Square representing the openGL coords of the grid

Definition at line 207 of file GameLoader.cpp.

6.2.2.2 bool GameLoader::LoadLevel (const std::wstring & *levelFileName*, LevelObject * *level*)

Loads a given level from a file

Parameters:

levelFileName Level File to load

level LevelObject to load

Returns:

True on success

Definition at line 85 of file GameLoader.cpp.

6.2.2.3 bool GameLoader::RunLoader (const std::wstring & *worldsFileName*, std::list< WorldObject * > & *list*, GameDude * *dude*)

Loads the world list with WorldObjects from the worldsFileName

Parameters:

worldsFileName File Path to a worlds file

list World List to load

dude GameDude to hand to each world

Returns:

True on success

Definition at line 23 of file GameLoader.cpp.

6.3 GraphicLoaders Namespace Reference

Loads Graphics.

Typedefs

- typedef unsigned int TextureIdentifier

Functions

- bool LoadNewBitmap (const std::wstring &fileName, TextureIdentifier &textureId)
- bool LoadTga (const std::wstring &fileName, TextureIdentifier &textureId)
- bool LoadTga (const std::string &fileName, TextureIdentifier &textureId)

6.3.1 Detailed Description

Loads Graphics. Namespace to load different graphic formats into openGL textures This namespace primarily wraps the SOIL libary

6.3.2 Typedef Documentation

6.3.2.1 typedef unsigned int GraphicLoaders::TextureIdentifier

ID used to represent a openGL texture

Definition at line 26 of file GraphicLoaders.h.

6.3.3 Function Documentation

6.3.3.1 bool GraphicLoaders::LoadNewBitmap (const std::wstring & *fileName*, TextureIdentifier & *textureId*)

Loads a Bitmap image into openGL

Parameters:

fileName File to load

textureId Storage location for the textureId

Returns:

True is successful

6.3.3.2 bool GraphicLoaders::LoadTga (const std::string & *fileName*, TextureIdentifier & *textureId*)

6.3.3.3 bool GraphicLoaders::LoadTga (const std::wstring & *fileName*, TextureIdentifier & *textureId*)

Loads a Bitmap image into openGL

Parameters:

fileName File to load

textureId Storage location for the textureId

Returns:

True is successful

6.4 UtilFunctions Namespace Reference

Utility Functions Contains several useful functions that fail to fit elsewhere.

Typedefs

- typedef std::vector< std::string * > StringTokensType
- typedef StringTokensType * StringTokens

Functions

- std::string TrimWhiteSpace (const std::string &startString)
- UtilFunctions::StringTokens StringTokenizer (const std::string &baseString, const std::string &delimiters)
- UtilFunctions::StringTokens StringTokenizer2 (const std::string &baseString, const std::string &delimiters)
- void DestroyStringTokens (UtilFunctions::StringTokens tokens)

6.4.1 Detailed Description

Utility Functions Contains several useful functions that fail to fit elsewhere.

6.4.2 Typedef Documentation

6.4.2.1 UtilFunctions::StringTokens

The full data type of the object returned by the string tokenizers

Definition at line 34 of file UtilFunctions.h.

6.4.2.2 UtilFunctions::StringTokensType

The core data type of the object returned by the string tokenizers

Definition at line 28 of file UtilFunctions.h.

6.4.3 Function Documentation

6.4.3.1 void UtilFunctions::DestroyStringTokens (UtilFunctions::StringTokens *tokens*)

Function to clean up the tokens returned by the tokenizers

Parameters:

tokens Tokens to clean up

Definition at line 79 of file UtilFunctions.cpp.

6.4.3.2 UtilFunctions::StringTokens UtilFunctions::StringTokenizer (const std::string & *baseString*, const std::string & *delimiters*)

Method to break up a given string into smaller piece at given points Method does not return empty tokens

Parameters:

baseString String to tokenize

delimiters String containing the charaters to break the baseString on

Returns:

StringTokens contains each piece of the broken string

Definition at line 32 of file UtilFunctions.cpp.

6.4.3.3 UtilFunctions::StringTokens UtilFunctions::StringTokenizer2 (const std::string & *baseString*, const std::string & *delimiters*)

Method to break up a given string into smaller piece at given points Method does return empty tokens

Parameters:

baseString String to tokenize

delimiters String containing the charaters to break the baseString on

Returns:

StringTokens contains each piece of the broken string

Definition at line 49 of file UtilFunctions.cpp.

6.4.3.4 std::string UtilFunctions::TrimWhiteSpace (const std::string & *startString*)

Method to trim white space off a string

Parameters:

startString string to trim

Returns:

a trimmed string

Definition at line 17 of file UtilFunctions.cpp.

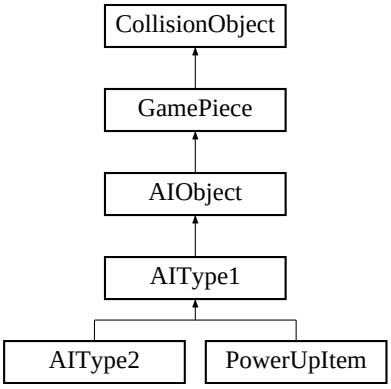
Chapter 7

Class Documentation

7.1 AIObject Class Reference

Base class for all AI controled objects.

`#include <AIObject.h>`Inheritance diagram for AIObject::



Public Member Functions

- AIObject (const Square &startingPos, unsigned int textureId)
- virtual void SetVerticalStatus (VerticalStatus status)
- virtual bool Trigger (double xOffset)=0
- virtual void Update (int ticks)=0
- virtual void Draw ()
- virtual bool GetActiveStatus ()

Protected Attributes

- VerticalStatus m_vStatus
- double m_gameFloor
- bool m_active
- bool m_killed

7.1.1 Detailed Description

Base class for all AI controled objects. Base class for all AI controled objects

Definition at line 20 of file AIObject.h.

7.1.2 Constructor & Destructor Documentation

7.1.2.1 AIObject::AIObject (const Square & *startingPos*, unsigned int *textureId*)

Constructor

Parameters:

startingPos Starting location of the AI object

textureId Image to display for the object

Definition at line 6 of file AIObject.cpp.

7.1.3 Member Function Documentation

7.1.3.1 void AIObject::Draw () `[virtual]`

General Draw Method for the AI Object

Reimplemented from GamePiece.

Definition at line 20 of file AIObject.cpp.

7.1.3.2 bool AIObject::GetActiveStatus () `[virtual]`

Method to check if the AI Object is currently active in the game

Returns:

True if the object is active

Definition at line 32 of file AIObject.cpp.

7.1.3.3 void AIObject::SetVerticalStatus (VerticalStatus *status*) `[virtual]`

Sets the current Vertical Status of the AI object

Parameters:

status New vertical status for the AIObject

Reimplemented in PowerUpItem.

Definition at line 15 of file AIObject.cpp.

7.1.3.4 virtual bool AIObject::Trigger (double *xOffset*) [pure virtual]

Checks to see if the AI Object is within range of the game dude.

Parameters:

xOffset The current x offset of the game dude from the level start

Returns:

True if AI Object should be triggered

Implemented in AIType1, and PowerUpItem.

7.1.3.5 virtual void AIObject::Update (int *ticks*) [pure virtual]

Performs the update on the AI Object

Parameters:

ticks Number of ticks that have passed since the last update

Implemented in AIType1, AIType2, and PowerUpItem.

7.1.4 Member Data Documentation

7.1.4.1 bool AIObject::m_active [protected]

If the current AI has been activated

Definition at line 72 of file AIObject.h.

7.1.4.2 double AIObject::m_gameFloor [protected]

The game floor. If the Y of the AI object drops below this, it should go inactive

Definition at line 67 of file AIObject.h.

7.1.4.3 bool AIObject::m_killed [protected]

If the AI has been killed

Definition at line 77 of file AIObject.h.

7.1.4.4 VerticalStatus AIObject::m_vStatus [protected]

The current vertical status of the AI Object

Definition at line 62 of file AIObject.h.

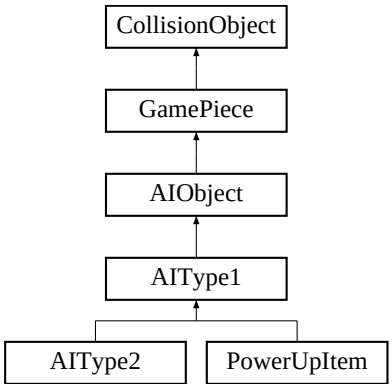
The documentation for this class was generated from the following files:

- branches/GameBase/AIObject.h
- branches/GameBase/AIObject.cpp

7.2 AIType1 Class Reference

Dumb AI Class.

`#include <AIType1.h>`Inheritance diagram for AIType1::



Public Member Functions

- AIType1 (const Square &startingPos, unsigned int leftTextureId, unsigned int rightTextureId)
- virtual ~AIType1 ()
- virtual void Update (int ticks)
- virtual bool Trigger (double xOffset)
- virtual bool Collide (CollisionSideEnum side, int damage)
- virtual bool CheckCollision (CollisionObject *object)

Protected Member Functions

- void SwitchDirections (bool direction)

Protected Attributes

- bool m_direction
- unsigned int m_textureIds [2]

7.2.1 Detailed Description

Dumb AI Class. Style 1 AI Controlled Enemy AI will bounce of blocks and walk of cliffs

Definition at line 23 of file AIType1.h.

7.2.2 Constructor & Destructor Documentation

7.2.2.1 AIType1::AIType1 (const Square & *startingPos*, unsigned int *leftTextureId*, unsigned int *rightTextureId*)

Constructor

Parameters:

startingPos Starting Position of the AI

leftTextureId OpenGl TextureId for the image to display when the AI is moving left.

rightTextureId OpenGl TextureId for the image to display when the AI is moving right.

Definition at line 12 of file AIType1.cpp.

7.2.2.2 AIType1::~AIType1 () `[virtual]`

Destructor

Definition at line 20 of file AIType1.cpp.

7.2.3 Member Function Documentation

7.2.3.1 bool AIType1::CheckCollision (CollisionObject * *object*) `[virtual]`

Checks to see if object has collided with the current object

Parameters:

object Object to check a collision against

Returns:

true if the object collided

Reimplemented from GamePiece.

Reimplemented in PowerUpItem.

Definition at line 80 of file AIType1.cpp.

7.2.3.2 bool AIType1::Collide (CollisionSideEnum *side*, int *damage*) `[virtual]`

Handles the event of another object colliding with the current one

Parameters:

side Side of the current object that has been hit

damage Amount of damage to attempt to inflict upon the current object

Returns:

true if the current object wishes to return damage

Reimplemented from GamePiece.

Reimplemented in AIType2, and PowerUpItem.

Definition at line 55 of file AIType1.cpp.

7.2.3.3 void AIType1::SwitchDirections (bool *direction*) [protected]

Method used to switch the moving direction of the AIObject This method handles the textureIds and other status updates

Definition at line 120 of file AIType1.cpp.

7.2.3.4 bool AIType1::Trigger (double *xOffset*) [virtual]

Checks to see if the AI Object is within range of the game dude.

Parameters:

xOffset The current x offset of the game dude from the level start

Returns:

True if AI Object should be triggered

Implements AIObject.

Reimplemented in PowerUpItem.

Definition at line 45 of file AIType1.cpp.

7.2.3.5 void AIType1::Update (int *ticks*) [virtual]

Performs the update on the AI Object

Parameters:

ticks Number of ticks that have passed since the last update

Implements AIObject.

Reimplemented in AIType2, and PowerUpItem.

Definition at line 24 of file AIType1.cpp.

7.2.4 Member Data Documentation

7.2.4.1 bool AIType1::m_direction [protected]

Monitors the moving direction of the AI Object

Definition at line 69 of file AIType1.h.

7.2.4.2 unsigned int AIType1::m_textureIds[2] [protected]

List of texture ids used during the operation of the object

Definition at line 74 of file AIType1.h.

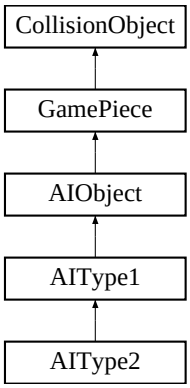
The documentation for this class was generated from the following files:

- branches/GameBase/AIType1.h
- branches/GameBase/AIType1.cpp

7.3 AIType2 Class Reference

Not so dumb AI Class.

`#include <AIType2.h>`Inheritance diagram for AIType2::



Public Member Functions

- AIType2 (const Square &startingPos, unsigned int leftTextureId, unsigned int rightTextureId)
- virtual ~AIType2 ()
- virtual void Update (int ticks)
- virtual bool Collide (CollisionSideEnum side, int damage)

Protected Attributes

- int m_verticalCollisionsThisPass

7.3.1 Detailed Description

Not so dumb AI Class. Style 2 AI Controlled Enemy AI will bounce of blocks and will not walk of cliffs

Definition at line 23 of file AIType2.h.

7.3.2 Constructor & Destructor Documentation

7.3.2.1 AIType2::AIType2 (const Square & *startingPos*, unsigned int *leftTextureId*, unsigned int *rightTextureId*)

Constructor

Parameters:

startingPos Starting Position of the AI

leftTextureId OpenGl TextureId for the image to display when the AI is moving left.

rightTextureId OpenGl TextureId for the image to display when the AI is moving right.

Definition at line 3 of file AIType2.cpp.

7.3.2.2 AIType2::~AIType2 () [virtual]

Destructor

Definition at line 9 of file AIType2.cpp.

7.3.3 Member Function Documentation

7.3.3.1 bool AIType2::Collide (CollisionSideEnum *side*, int *damage*) [virtual]

Handles the event of another object colliding with the current one

Parameters:

side Side of the current object that has been hit

damage Amount of damage to attempt to inflict upon the current object

Returns:

true if the current object wishes to return damage

Reimplemented from AIType1.

Definition at line 23 of file AIType2.cpp.

7.3.3.2 void AIType2::Update (int *ticks*) [virtual]

Performs the update on the AI Object Calls AIType1::Update after performing simple logic for Type 2

Parameters:

ticks Number of ticks that have passed since the last update

Reimplemented from AIType1.

Definition at line 13 of file AIType2.cpp.

7.3.4 Member Data Documentation

7.3.4.1 int AIType2::m_verticalCollisionsThisPass [protected]

Traces the number of collides per update. Cleared to 0 with every call of update. If 1, the direction of the AI object will flip

Definition at line 58 of file AIType2.h.

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

- branches/GameBase/AIType2.h
- branches/GameBase/AIType2.cpp

7.4 AudioManager Class Reference

Singeton Object to play and pause sounds.

`#include <AudioManager.h>`

Public Member Functions

- void SetListenerValues ()
- void LoadSound (SoundLookup loadSound, char *soundFilePath, bool isLoop)
- void PlayALSource (SoundLookup playSound)
- void StopALSource (SoundLookup stopSound)
- void HoldALSource (SoundLookup holdSound)

Static Public Member Functions

- static AudioManager * Instance ()

Protected Member Functions

- AudioManager ()
- ~AudioManager ()
- AudioManager (const AudioManager &)
- AudioManager & operator= (const AudioManager &)

Private Attributes

- ALuint m_Song1Buff
- ALuint m_Song2Buff
- ALuint m_Song3Buff
- ALuint m_PowerupBuff
- ALuint m_CheckpointBuff
- ALuint m_CoinBuff
- ALuint m_HitBrickBuff
- ALuint m_Song1Src
- ALuint m_Song2Src
- ALuint m_Song3Src
- ALuint m_PowerupSrc
- ALuint m_CheckpointSrc
- ALuint m_CoinSrc
- ALuint m_HitBrickSrc
- ALfloat SourcePos [3]
- ALfloat SourceVel [3]
- ALfloat ListenerPos [3]
- ALfloat ListenerVel [3]
- ALfloat ListenerOri [6]

Static Private Attributes

- static AudioManager * m_instance = 0

7.4.1 Detailed Description

Singeton Object to play and pause sounds. This is the AudioManager class for the SuperEastGate. The functions this class performs are; looping background sounds, gameplay sounds based on action, pausing sounds, stopping sounds, and possibly loading sounds on demmand.

Definition at line 46 of file AudioManager.h.

7.4.2 Constructor & Destructor Documentation

7.4.2.1 AudioManager::AudioManager () [protected]

Constructor that sets up all the components for the LoadSound method to work

Definition at line 23 of file AudioManager.cpp.

7.4.2.2 AudioManager::~AudioManager () [protected]

Destructor that unloads and detatches sounds from their buffers.

Definition at line 41 of file AudioManager.cpp.

7.4.2.3 AudioManager::AudioManager (const AudioManager &) [protected]

7.4.3 Member Function Documentation

7.4.3.1 void AudioManager::HoldALSource (SoundLookup *holdSound*)

HoldALSource method that will hold a current sound state

Parameters:

holdSound Sound ID for the buffer to pause

Definition at line 352 of file AudioManager.cpp.

7.4.3.2 AudioManager * AudioManager::Instance () [static]

Method to obtain the instance of this singleton object

Returns:

The singleton instance of the AudioManager

Definition at line 383 of file AudioManager.cpp.

7.4.3.3 void AudioManager::LoadSound (SoundLookup *loadSound*, char * *soundFilePath*, bool *isLoop*)

LoadSound method that loads a specific sound on demmand

Parameters:

loadSound Sound ID for the buffer to load

soundFilePath Path to the wav file to load

isLoop Should the sound loop once started

Definition at line 73 of file AudioManager.cpp.

7.4.3.4 AudioManager& AudioManager::operator= (const AudioManager &) [protected]

7.4.3.5 void AudioManager::PlayALSource (SoundLookup *playSound*)

PlayALSource method that will play an imputted source

Parameters:

playSound Sound ID for the buffer to play

Definition at line 292 of file AudioManager.cpp.

7.4.3.6 void AudioManager::SetListenerValues ()

SetListenerValues method that tells the listener where to listen from.

Definition at line 65 of file AudioManager.cpp.

7.4.3.7 void AudioManager::StopALSource (SoundLookup *stopSound*)

StopALSource method that will stop an imputted source

Parameters:

stopSound Sound ID for the buffer to stop

Definition at line 322 of file AudioManager.cpp.

7.4.4 Member Data Documentation

7.4.4.1 ALfloat AudioManager::ListenerOri[6] [private]

Definition at line 127 of file AudioManager.h.

7.4.4.2 ALfloat AudioManager::ListenerPos[3] [private]

Definition at line 125 of file AudioManager.h.

7.4.4.3 ALfloat AudioManager::ListenerVel[3] [private]

Definition at line 126 of file AudioManager.h.

7.4.4.4 ALuint AudioManager::m_CheckpointBuff [private]

Definition at line 110 of file AudioManager.h.

7.4.4.5 ALuint AudioManager::m_CheckpointSrc [private]

Definition at line 119 of file AudioManager.h.

7.4.4.6 ALuint AudioManager::m_CoinBuff [private]

Definition at line 111 of file AudioManager.h.

7.4.4.7 ALuint AudioManager::m_CoinSrc [private]

Definition at line 120 of file AudioManager.h.

7.4.4.8 ALuint AudioManager::m_HitBrickBuff [private]

Definition at line 112 of file AudioManager.h.

7.4.4.9 ALuint AudioManager::m_HitBrickSrc [private]

Definition at line 121 of file AudioManager.h.

7.4.4.10 AudioManager * AudioManager::m_instance = 0 [static, private]

Definition at line 103 of file AudioManager.h.

7.4.4.11 ALuint AudioManager::m_PowerupBuff [private]

Definition at line 109 of file AudioManager.h.

7.4.4.12 ALuint AudioManager::m_PowerupSrc [private]

Definition at line 118 of file AudioManager.h.

7.4.4.13 ALuint AudioManager::m_Song1Buff [private]

Definition at line 105 of file AudioManager.h.

7.4.4.14 ALuint AudioManager::m_Song1Src [private]

Definition at line 114 of file AudioManager.h.

7.4.4.15 ALuint AudioManager::m_Song2Buff [private]

Definition at line 106 of file AudioManager.h.

7.4.4.16 ALuint AudioManager::m_Song2Src [private]

Definition at line 115 of file AudioManager.h.

7.4.4.17 ALuint AudioManager::m_Song3Buff [private]

Definition at line 107 of file AudioManager.h.

7.4.4.18 ALuint AudioManager::m_Song3Src [private]

Definition at line 116 of file AudioManager.h.

7.4.4.19 ALfloat AudioManager::SourcePos[3] [private]

Definition at line 123 of file AudioManager.h.

7.4.4.20 ALfloat AudioManager::SourceVel[3] [private]

Definition at line 124 of file AudioManager.h.

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

- branches/GameBase/AudioManager.h
- branches/GameBase/AudioManager.cpp

7.5 BackGroundManager Class Reference

Displays the background of the game.

`#include <BackGroundManager.h>`

Public Member Functions

- BackGroundManager (const std::wstring &backgroundPath, double screenWidth, double screenHeight, double backgroundCameraPercent)
- ~BackGroundManager ()
- void Draw (double xOffset)

Private Attributes

- GraphicLoaders::TextureIdentifier m_background
- double m_screenWidth
- double m_screenHeight
- double m_cameraPercent

7.5.1 Detailed Description

Displays the background of the game. Class to manage which section of the background image to display to the user.

Definition at line 23 of file BackGroundManager.h.

7.5.2 Constructor & Destructor Documentation

7.5.2.1 BackGroundManager::BackGroundManager (const std::wstring & *backgroundPath*, double *screenWidth*, double *screenHeight*, double *backgroundCameraPercent*)

Constructor

Parameters:

backgroundPath Path to the background image to load

screenWidth Width of the screen in openGL units

screenHeight Height of the screen in openGL units

backgroundCameraPercent Percentage of the image to display at any one given time

Definition at line 6 of file BackGroundManager.cpp.

7.5.2.2 BackGroundManager::~BackGroundManager ()

Desctructor

Definition at line 15 of file BackGroundManager.cpp.

7.5.3 Member Function Documentation

7.5.3.1 void BackGroundManager::Draw (double *xOffset*)

Method to draw the background image

Parameters:

xOffset Distance the left side of the screen is from the start of the level

Definition at line 19 of file BackGroundManager.cpp.

7.5.4 Member Data Documentation

7.5.4.1 GraphicLoaders::TextureIdentifier BackGroundManager::m_background `[private]`

Image id of the background image

Definition at line 49 of file BackGroundManager.h.

7.5.4.2 double BackGroundManager::m_cameraPercent `[private]`

Percentage of the image to display at any one given time

Definition at line 64 of file BackGroundManager.h.

7.5.4.3 double BackGroundManager::m_screenHeight `[private]`

Height of the screen in openGL Units

Definition at line 59 of file BackGroundManager.h.

7.5.4.4 double BackGroundManager::m_screenWidth `[private]`

Width of the screen in openGL Units

Definition at line 54 of file BackGroundManager.h.

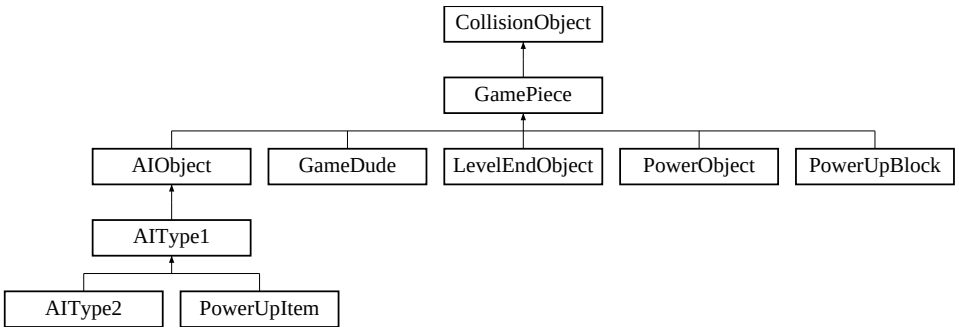
The documentation for this class was generated from the following files:

- branches/GameBase/BackGroundManager.h
- branches/GameBase/BackGroundManager.cpp

7.6 CollisionObject Class Reference

Basic Collision Detection Object.

`#include <CollisionObject.h>`Inheritance diagram for CollisionObject::



Private Member Functions

- virtual bool CheckCollision (CollisionObject ∗object)=0
- virtual bool Collide (CollisionSideEnum side, int damage)=0

7.6.1 Detailed Description

Basic Collision Detection Object. This class provides the proper interface needed to perfrom the collision detection used throughout the game

Definition at line 21 of file CollisionObject.h.

7.6.2 Member Function Documentation

7.6.2.1 virtual bool CollisionObject::CheckCollision (CollisionObject ∗ *object*) [private, pure virtual]

Checks to see if object has collided with the current object

Parameters:

object Object to check a collision against

Returns:

true if the object collided

7.6.2.2 virtual bool CollisionObject::Collide (CollisionSideEnum *side*, int *damage*) [private, pure virtual]

Handles the event of another object colliding with the current one

Parameters:

side Side of the current object that has been hit

damage Amount of damage to attempt to inflict upon the current object

Returns:

true if the current object wishes to return damage

Implemented in AIType1, AIType2, GameDude, GamePiece, PowerObject, PowerUpBlock, and PowerUpItem.

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

- branches/GameBase/CollisionObject.h

7.7 ControlObject Class Reference

Keyboard Control Manager.

`#include <ControlObject.h>`

Public Member Functions

- ControlObject ()
- virtual ~ControlObject ()
- unsigned int GetControlKey (Controls control)
- void SetControlKey (Controls control, unsigned int key)
- void LoadControls (const std::wstring &fileName)
- unsigned int StringToKey (const std::string *keyString)

Private Attributes

- unsigned int m_controlKeys [CO_MAX_CONTROL]

7.7.1 Detailed Description

Keyboard Control Manager. Utility Object to manage user configured controls for the game

Definition at line 35 of file ControlObject.h.

7.7.2 Constructor & Destructor Documentation

7.7.2.1 ControlObject::ControlObject ()

Constructor

Definition at line 8 of file ControlObject.cpp.

7.7.2.2 ControlObject::~ControlObject () `[virtual]`

Destructor

Definition at line 18 of file ControlObject.cpp.

7.7.3 Member Function Documentation

7.7.3.1 unsigned int ControlObject::GetControlKey (Controls *control*)

Obtains the keyboard key number for a given control

Parameters:

control Control Id of the command to obtain

Returns:

Keyboard code for the command

Definition at line 22 of file ControlObject.cpp.

7.7.3.2 void ControlObject::LoadControls (const std::wstring & *fileName*)

Loads a config file for the control set

Parameters:

fileName File Path to the control file

Definition at line 36 of file ControlObject.cpp.

7.7.3.3 void ControlObject::SetControlKey (Controls *control*, unsigned int *key*)

Sets the control key for a given control

Parameters:

control Control to configure

key New key to map to the control

Definition at line 31 of file ControlObject.cpp.

7.7.3.4 unsigned int ControlObject::StringToKey (const std::string * *keyString*)

Converts a string into the predefined keyboard codes

Parameters:

keyString Key to convert

Returns:

Keyboard code for the string

Definition at line 80 of file ControlObject.cpp.

7.7.4 Member Data Documentation

7.7.4.1 unsigned int ControlObject::m_controlKeys[CO_MAX_CONTROL] `[private]`

Array to maintain maps between a given control and the key

Definition at line 77 of file ControlObject.h.

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

- branches/GameBase/ControlObject.h
- branches/GameBase/ControlObject.cpp

7.8 Converter::ConverterException Class Reference

Converter Exception.

`#include <Converter.h>`

Public Member Functions

- ConverterException (const char *msg)
- const char * what ()

Private Attributes

- const char * m_msg

7.8.1 Detailed Description

Converter Exception. Class used to represent an error in the conversion process

Definition at line 33 of file Converter.h.

7.8.2 Constructor & Destructor Documentation

7.8.2.1 Converter::ConverterException::ConverterException (const char * *msg*) [inline]

Definition at line 35 of file Converter.h.

7.8.3 Member Function Documentation

7.8.3.1 const char* Converter::ConverterException::what () [inline]

Definition at line 36 of file Converter.h.

7.8.4 Member Data Documentation

7.8.4.1 const char* Converter::ConverterException::m_msg [private]

Definition at line 38 of file Converter.h.

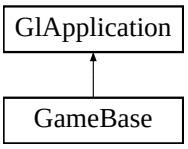
The documentation for this class was generated from the following file:

- branches/GameBase/Converter.h

7.9 GameBase Class Reference

Primary Game Class.

`#include <GameBase.h>`Inheritance diagram for GameBase::



Public Member Functions

- GameBase ()
- virtual ~GameBase ()
- virtual bool PerformInit ()
- virtual void KeyPressed (unsigned int key)
- virtual void KeyReleased (unsigned int key)
- virtual void PerformUpdate (int currentTick)
- void PlayGame ()
- void Draw ()
- virtual void LeftMouseClick (Point &clickedPoint)

Private Member Functions

- void BuildHUDFont ()
- void KillHudFont ()

Private Attributes

- std::list< WorldObject * > m_worldList
- std::list< WorldObject * >::iterator m_currentWorld
- GameDude * m_gameDude
- GameState m_currentGameState
- unsigned int m_hudTextBase
- GLYPHMETRICSFLOAT m_hudTextGmf [256]
- ControlObject * m_controls
- Menu * m_menu
- int m_delayTimer

7.9.1 Detailed Description

Primary Game Class. Primary class for the game. This is the primary class to control the program

Definition at line 29 of file GameBase.h.

7.9.2 Constructor & Destructor Documentation

7.9.2.1 GameBase::GameBase ()

Constructor

Definition at line 14 of file GameBase.cpp.

7.9.2.2 GameBase::~GameBase () [virtual]

Destructor

Definition at line 24 of file GameBase.cpp.

7.9.3 Member Function Documentation

7.9.3.1 void GameBase::BuildHUDFont () [private]

Builds the font used in the HUD and Menus

Definition at line 261 of file GameBase.cpp.

7.9.3.2 void GameBase::Draw () [virtual]

Custom method to draw the game objects

Reimplemented from GlApplication.

Definition at line 299 of file GameBase.cpp.

7.9.3.3 void GameBase::KeyPressed (unsigned int *key*) [virtual]

Overriden Method to handle key press events

Parameters:

key The code of the pressed key

Reimplemented from GlApplication.

Definition at line 39 of file GameBase.cpp.

7.9.3.4 void GameBase::KeyReleased (unsigned int *key*) [virtual]

Overriden Method to handle key release events

Parameters:

key The code of the released key

Reimplemented from GlApplication.

Definition at line 88 of file GameBase.cpp.

7.9.3.5 void GameBase::KillHudFont () [private]

Cleans up the font used in the HUD and Menus

Definition at line 292 of file GameBase.cpp.

7.9.3.6 void GameBase::LeftMouseClick (Point & *clickedPoint*) [virtual]

Event handler for Left Mouse Click Events

Parameters:

clickedPoint Point the mouse clicked on

Reimplemented from GlApplication.

Definition at line 220 of file GameBase.cpp.

7.9.3.7 bool GameBase::PerformInit () [virtual]

Overriden Method to perform required game setup

Returns:

True is everything Initilized properly

Reimplemented from GlApplication.

Definition at line 29 of file GameBase.cpp.

7.9.3.8 void GameBase::PerformUpdate (int *currentTick*) [virtual]

Custom update method to update the game objects

Parameters:

currentTick The current tick count

Reimplemented from GlApplication.

Definition at line 107 of file GameBase.cpp.

7.9.3.9 void GameBase::PlayGame ()

Method to start the game

Definition at line 228 of file GameBase.cpp.

7.9.4 Member Data Documentation

7.9.4.1 ControlObject* GameBase::m_controls [private]

Control object for the player's controls

Definition at line 120 of file GameBase.h.

7.9.4.2 GameState GameBase::m_currentGameState [private]

The current running state of the game

Definition at line 109 of file GameBase.h.

7.9.4.3 std::list<WorldObject *>::iterator GameBase::m_currentWorld [private]

The current game world the player is on

Definition at line 99 of file GameBase.h.

7.9.4.4 int GameBase::m_delayTimer [private]

Timer used to delay the players special ability

Definition at line 130 of file GameBase.h.

7.9.4.5 GameDude* GameBase::m_gameDude [private]

The player's character

Definition at line 104 of file GameBase.h.

7.9.4.6 unsigned int GameBase::m_hudTextBase [private]

Base offset of the hud text

Definition at line 114 of file GameBase.h.

7.9.4.7 GLYPHMETRICSFLOAT GameBase::m_hudTextGmf[256] [private]

Definition at line 115 of file GameBase.h.

7.9.4.8 Menu* GameBase::m_menu [private]

Main menu

Definition at line 125 of file GameBase.h.

7.9.4.9 std::list<WorldObject *> GameBase::m_worldList [private]

List of worlds to use in the current game

Definition at line 94 of file GameBase.h.

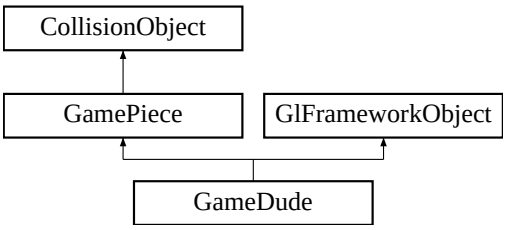
The documentation for this class was generated from the following files:

- branches/GameBase/GameBase.h
- branches/GameBase/GameBase.cpp

7.10 GameDude Class Reference

The player avatar.

`#include <GameDude.h>`Inheritance diagram for GameDude::



Public Member Functions

- GameDude (Square startingPos, unsigned int smallTextureId, unsigned int largeTextureId, unsigned int specialTextureId)
- virtual ~GameDude ()
- GameDudeStatus GetDudeStatus ()
- HoriztonalStatus GetHorizontalStatus ()
- VerticalStatus GetVerticalStatus ()
- double Move (double x)
- void Update (int ticks)
- void SetDudeStatus (GameDudeStatus newStatus)
- void SetHoriztonalStatus (HoriztonalStatus newStatus)
- void SetVerticalStatus (VerticalStatus newStatus)
- virtual bool Collide (CollisionSideEnum side, int damage)
- virtual void Draw ()
- void GetOffset ()
- void SetLeftBound (double newLeftX)
- void Reset (bool resetDudeStatus=true)
- void SetCrouching (bool status)
- bool GetFacing ()

Private Attributes

- double m_xOffset
- double m_jumpHeight
- GameDudeStatus m_gameDudeStatus
- HoriztonalStatus m_hStatus
- VerticalStatus m_vStatus
- double m_gameFloor
- Square m_startingPos
- unsigned int m_textureIds [3]
- bool m_crouching
- int m_invincible
- bool m_lastDirection

7.10.1 Detailed Description

The player avatar. Game Dude object represents the players avatar in the game In maintians the states of the player

Definition at line 25 of file GameDude.h.

7.10.2 Constructor & Destructor Documentation

7.10.2.1 GameDude::GameDude (Square *startingPos*, unsigned int *smallTextureId*, unsigned int *largeTextureId*, unsigned int *specialTextureId*)

Constructor

Parameters:

startingPos Starting Location of the player
smallTextureId Image Id of the small player
largeTextureId Image Id of the large player
specialTextureId Image Id of the special player

Definition at line 9 of file GameDude.cpp.

7.10.2.2 GameDude::~GameDude () `[virtual]`

Destructor

Definition at line 27 of file GameDude.cpp.

7.10.3 Member Function Documentation

7.10.3.1 bool GameDude::Collide (CollisionSideEnum *side*, int *damage*) `[virtual]`

Handles the event of another object colliding with the current one

Parameters:

side Side of the current object that has been hit
damage Amount of damage to attempt to inflict upon the current object

Returns:

true if the current object wishes to return damage

Reimplemented from GamePiece.

Definition at line 153 of file GameDude.cpp.

7.10.3.2 void GameDude::Draw () `[virtual]`

Draw the game dude

Reimplemented from GamePiece.

Definition at line 192 of file GameDude.cpp.

7.10.3.3 GameDudeStatus GameDude::GetDudeStatus ()

Gets the current game dude status

Returns:

The current GameDudeStatus

Definition at line 31 of file GameDude.cpp.

7.10.3.4 bool GameDude::GetFacing ()

Gets the current facing of the GameDude

Returns:

0 for left 1 for right

Definition at line 239 of file GameDude.cpp.

7.10.3.5 HoriztonalStatus GameDude::GetHorizontalStatus ()

Gets the current horizontal status of the player

Returns:

The current HoriztonalStatus

Definition at line 36 of file GameDude.cpp.

7.10.3.6 void GameDude::GetOffset ()

Returns the current xOffset of the game dude

7.10.3.7 VerticalStatus GameDude::GetVerticalStatus ()

Gets the current vertical status of the player

Returns:

The current VerticalStatus

Definition at line 41 of file GameDude.cpp.

7.10.3.8 double GameDude::Move (double *x*)

Move the player *x* units

Parameters:

x Distance to move the player

Returns:

The new xOffset of the player

Definition at line 46 of file GameDude.cpp.

7.10.3.9 void GameDude::Reset (bool *resetDudeStatus* = `true`)

Resets the dude to his starting position

Parameters:

resetDudeStatus If the dude status should be reset

Definition at line 215 of file GameDude.cpp.

7.10.3.10 void GameDude::SetCrouching (bool *status*)

Sets the crouching status of the GameDude

Parameters:

status The new crouching status

Definition at line 230 of file GameDude.cpp.

7.10.3.11 void GameDude::SetDudeStatus (GameDudeStatus *newStatus*)

Set the dude status

Parameters:

newStatus The new GameDudeStatus

Definition at line 54 of file GameDude.cpp.

7.10.3.12 void GameDude::SetHoriztonalStatus (HoriztonalStatus *newStatus*)

Set the Horizontal Status

Parameters:

newStatus The new HorizontalStatus

Definition at line 99 of file GameDude.cpp.

7.10.3.13 void GameDude::SetLeftBound (double *newLeftX*)

Moves the current position to have its left side start at newLeftX

Parameters:

newLeftX The new left position

Definition at line 209 of file GameDude.cpp.

7.10.3.14 void GameDude::SetVerticalStatus (VerticalStatus *newStatus*)

Set the vertical status

Parameters:

newStatus The new VerticalStatus

Definition at line 86 of file GameDude.cpp.

7.10.3.15 void GameDude::Update (int *ticks*) [virtual]

Performs the update on the player object

Parameters:

ticks Number of ticks that have passed since the last update

Implements GlFrameworkObject.

Definition at line 111 of file GameDude.cpp.

7.10.4 Member Data Documentation

7.10.4.1 bool GameDude::m_crouching [private]

Crouching status

Definition at line 176 of file GameDude.h.

7.10.4.2 GameDudeStatus GameDude::m_gameDudeStatus [private]

The current game dude status

Definition at line 145 of file GameDude.h.

7.10.4.3 double GameDude::m_gameFloor [private]

The game floor height, if the player falls below this, they are killed

Definition at line 161 of file GameDude.h.

7.10.4.4 HoriztonalStatus GameDude::m_hStatus [private]

The current horizontal status

Definition at line 150 of file GameDude.h.

7.10.4.5 int GameDude::m_invincible [private]

Invinciblity timer

Definition at line 181 of file GameDude.h.

7.10.4.6 double GameDude::m_jumpHeight [private]

The current height the player has jumped

Definition at line 140 of file GameDude.h.

7.10.4.7 bool GameDude::m_lastDirection [private]

Last Direction the player was moving

Definition at line 186 of file GameDude.h.

7.10.4.8 Square GameDude::m_startingPos [private]

The starting position of the player

Definition at line 166 of file GameDude.h.

7.10.4.9 unsigned int GameDude::m_textureIds[3] [private]

Player images

Definition at line 171 of file GameDude.h.

7.10.4.10 VerticalStatus GameDude::m_vStatus [private]

The current vertical status

Definition at line 155 of file GameDude.h.

7.10.4.11 double GameDude::m_xOffset [private]

Distance the game dude is from the start of the level

Definition at line 135 of file GameDude.h.

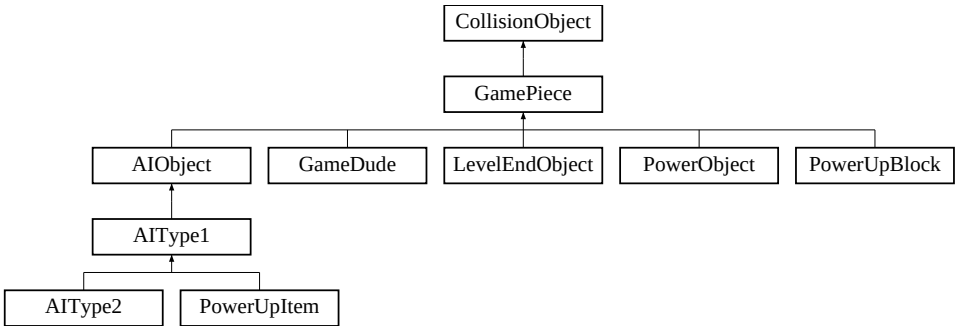
The documentation for this class was generated from the following files:

- branches/GameBase/GameDude.h
- branches/GameBase/GameDude.cpp

7.11 GamePiece Class Reference

General Game Object.

`#include <GamePiece.h>`Inheritance diagram for GamePiece::



Public Member Functions

- GamePiece (const Square &startingPos, unsigned int textureId)
- virtual void Draw ()
- Square GetCurrentPosition ()
- virtual bool Collide (CollisionSideEnum side, int damage)
- virtual bool CheckCollision (CollisionObject *object)
- virtual bool OnScreen (double leftX, double rightX)
- void SetPosition (const Square &newPos)

Protected Attributes

- Square m_currentLocation
- unsigned int m_textureId
- bool m_broken

7.11.1 Detailed Description

General Game Object. Base class for all objects the player can see on the screen and the GameDude can interact with.

Definition at line 23 of file GamePiece.h.

7.11.2 Constructor & Destructor Documentation

7.11.2.1 GamePiece::GamePiece (const Square & *startingPos*, unsigned int *textureId*)

Constructor

Parameters:

startingPos Location of the GamePiece

textureId Id of the texture of the GamePiece

Definition at line 10 of file GamePiece.cpp.

7.11.3 Member Function Documentation

7.11.3.1 bool GamePiece::CheckCollision (CollisionObject * *object*) [virtual]

Checks to see if object has collided with the current object

Parameters:

object Object to check a collision against

Returns:

true if the object collided

Reimplemented in AIType1, LevelEndObject, PowerObject, and PowerUpItem.

Definition at line 41 of file GamePiece.cpp.

7.11.3.2 bool GamePiece::Collide (CollisionSideEnum *side*, int *damage*) [virtual]

Handles the event of another object colliding with the current one

Parameters:

side Side of the current object that has been hit

damage Amount of damage to attempt to inflict upon the current object

Returns:

true if the current object wishes to return damage

Implements CollisionObject.

Reimplemented in AIType1, AIType2, GameDude, PowerObject, PowerUpBlock, and PowerUpItem.

Definition at line 103 of file GamePiece.cpp.

7.11.3.3 void GamePiece::Draw () [virtual]

Draws the GamePiece

Reimplemented in AIObject, GameDude, LevelEndObject, and PowerObject.

Definition at line 17 of file GamePiece.cpp.

7.11.3.4 Square GamePiece::GetCurrentPosition ()

Obtains the current location of the GamePiece

Definition at line 36 of file GamePiece.cpp.

7.11.3.5 bool GamePiece::OnScreen (double *leftX*, double *rightX*) [virtual]

Method to check if the object is on the screen

Parameters:

leftX Left side of the screen

rightX Right side of the screen

Returns:

True is the object is on the screen

Definition at line 115 of file GamePiece.cpp.

7.11.3.6 void GamePiece::SetPosition (const Square & *newPos*)

Sets the current position to the newPos

Parameters:

newPos The objects new position

Definition at line 121 of file GamePiece.cpp.

7.11.4 Member Data Documentation

7.11.4.1 bool GamePiece::m_broken [protected]

Has the block been broken

Definition at line 84 of file GamePiece.h.

7.11.4.2 Square GamePiece::m_currentLocation [protected]

The position and size of the object

Definition at line 74 of file GamePiece.h.

7.11.4.3 unsigned int GamePiece::m_textureId [protected]

Texture id to draw with the object

Definition at line 79 of file GamePiece.h.

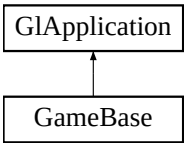
The documentation for this class was generated from the following files:

- branches/GameBase/GamePiece.h
- branches/GameBase/GamePiece.cpp

7.12 GlApplication Class Reference

Base Class for OpenGl Applications.

`#include <GlApplication.h>`Inheritance diagram for GlApplication::



Public Member Functions

- GlApplication ()
- ~GlApplication ()
- int Main (HINSTANCE hInstance, HINSTANCE prevInstance, LPSTR lpCmdLine, int nCmdShow)
- LRESULT MessageHandler (HWND hWnd, UINT uMsg, WPARAM wParam, LPARAM lParam)
- Point GetPointAtCursor (LPARAM lParam)

Protected Member Functions

- virtual void Draw ()
- bool Init ()
- virtual bool PerformInit ()
- void Update ()
- virtual void PerformUpdate (int currentTick)
- virtual void LeftMouseClick (Point &clickedPoint)
- virtual void RightMouseClick (Point &clickedPoint)
- virtual void KeyPressed (unsigned int key)
- virtual void KeyReleased (unsigned int key)

Protected Attributes

- KeyHandler m_keys
- std::list< GlFrameworkObject * > m_objectList
- bool m_applicationRunning
- int m_lastTickCount
- GlWindow m_window

Private Attributes

- HINSTANCE hInstance
- bool m_isActive

7.12.1 Detailed Description

Base Class for OpenGl Applications. Base Class for OpenGl Applications Class handles window setup and message handling

Definition at line 29 of file GlApplication.h.

7.12.2 Constructor & Destructor Documentation

7.12.2.1 GlApplication::GlApplication ()

Constructor

Definition at line 31 of file GlApplication.cpp.

7.12.2.2 GlApplication::~GlApplication ()

Destructor

Definition at line 40 of file GlApplication.cpp.

7.12.3 Member Function Documentation

7.12.3.1 void GlApplication::Draw () [protected, virtual]

Draw the scene

Reimplemented in GameBase.

Definition at line 240 of file GlApplication.cpp.

7.12.3.2 Point GlApplication::GetPointAtCursor (LPARAM *lParam*)

Utility Method to convert a windows mouse pos into open gl coords

Parameters:

lParam Parameter of the point clicked

Returns:

The OpenGl point clicked

Definition at line 185 of file GlApplication.cpp.

7.12.3.3 bool GlApplication::Init () [protected]

Set up the gl window

Definition at line 207 of file GlApplication.cpp.

7.12.3.4 void GlApplication::KeyPressed (unsigned int *key*) [protected, virtual]

User overridable method to handle key press

Parameters:

key The key pressed

Reimplemented in GameBase.

Definition at line 275 of file GlApplication.cpp.

7.12.3.5 void GlApplication::KeyReleased (unsigned int *key*) [protected, virtual]

User overridable method to handle key release

Parameters:

key The key released

Reimplemented in GameBase.

Definition at line 280 of file GlApplication.cpp.

7.12.3.6 void GlApplication::LeftMouseClick (Point & *clickedPoint*) [protected, virtual]

User overridable method to handle left mouse clicks

Parameters:

clickedPoint Point Clicked

Reimplemented in GameBase.

Definition at line 265 of file GlApplication.cpp.

7.12.3.7 int GlApplication::Main (HINSTANCE *hInstance*, HINSTANCE *prevInstance*, LPSTR *lpCmdLine*, int *nCmdShow*)

Main program loop

Parameters:

hInstance Current Instance of the application

prevInstance Parent Instance of the application

lpCmdLine Array of command line args

nCmdShow 1 if the commandline is showing

Returns:

Exit code of the application

Definition at line 45 of file GlApplication.cpp.

7.12.3.8 LRESULT GlApplication::MessageHandler (HWND *hWnd*, UINT *uMsg*, WPARAM *wParam*, LPARAM *lParam*)

Message Handler

Parameters:

hWnd Handle to the current window

uMsg The message to handle

wParam Parameter 1

lParam Parameter 2

Returns:

0 if all went well

Definition at line 121 of file GlApplication.cpp.

7.12.3.9 bool GlApplication::PerformInit () [protected, virtual]

User overridable method for init

Reimplemented in GameBase.

Definition at line 226 of file GlApplication.cpp.

7.12.3.10 void GlApplication::PerformUpdate (int *currentTick*) [protected, virtual]

User overridable method for object updates

Parameters:

currentTick Current Tick count for the time

Reimplemented in GameBase.

Definition at line 255 of file GlApplication.cpp.

7.12.3.11 void GlApplication::RightMouseClick (Point & *clickedPoint*) [protected, virtual]

User overridable method to handle right mouse clicks

Parameters:

clickedPoint Point Clicked

Definition at line 270 of file GlApplication.cpp.

7.12.3.12 void GlApplication::Update () [protected]

Update all objects

Definition at line 232 of file GlApplication.cpp.

7.12.4 Member Data Documentation

7.12.4.1 HINSTANCE GlApplication::hInstance [private]

Application Instance

Definition at line 149 of file GlApplication.h.

7.12.4.2 bool GlApplication::m_applicationRunning [protected]

Boolean to monitor if the app is running

Definition at line 133 of file GlApplication.h.

7.12.4.3 bool GlApplication::m_isActive [private]

Boolean to monitor if the app is active

Definition at line 154 of file GlApplication.h.

7.12.4.4 KeyHandler GlApplication::m_keys [protected]

Key handler for the application

Definition at line 123 of file GlApplication.h.

7.12.4.5 int GlApplication::m_lastTickCount [protected]

Last tick count that was run on update

Definition at line 138 of file GlApplication.h.

7.12.4.6 std::list< GlFrameworkObject * > GlApplication::m_objectList [protected]

Object list to run update and draw for the application

Definition at line 128 of file GlApplication.h.

7.12.4.7 GlWindow GlApplication::m_window [protected]

The window for the aplication

Definition at line 143 of file GlApplication.h.

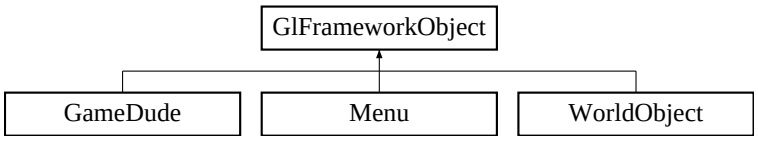
The documentation for this class was generated from the following files:

- branches/GameBase/GlApplication.h
- branches/GameBase/GlApplication.cpp

7.13 GlFrameworkObject Class Reference

Interface for objects to interact with GlApplication.

`#include <GlFrameworkObject.h>`Inheritance diagram for GlFrameworkObject::



Public Member Functions

- virtual void Draw ()=0
- virtual void Update (int ticks)=0

7.13.1 Detailed Description

Interface for objects to interact with GlApplication. Base class for the Framework Objects

Definition at line 19 of file GlFrameworkObject.h.

7.13.2 Member Function Documentation

7.13.2.1 virtual void GlFrameworkObject::Draw () `[pure virtual]`

Method to draw the object

Implemented in GameDude, Menu, and WorldObject.

7.13.2.2 virtual void GlFrameworkObject::Update (int *ticks*) `[pure virtual]`

Method to update the object

Parameters:

ticks Number of ticks passed since the last call to update

Implemented in GameDude, Menu, and WorldObject.

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

- branches/GameBase/GlFrameworkObject.h

7.14 GlWindow Class Reference

Window in Windows.

`#include <GlWindow.h>`

Public Member Functions

- GlWindow ()
- ~GlWindow ()
- bool CreateGlWindow (const std::wstring &className, const std::wstring &windowTitle, HINSTANCE hInstance, int windowPosX, int windowPosY, int windowWidth, int windowHeight, const bool &fullScreen, int bitsPerPixel, LPVOID application)
- void KillWindow ()
- void SwapBuffers ()
- void ResizeGlScene (int width, int height)
- int GetWindowHeight ()
- int GetWindowWidth ()
- operator HWND ()
- operator HDC ()

Protected Member Functions

- void EnableFullScreen (const int &windowWidth, const int &windowHeight, const int &bitsPerPixel)

Private Attributes

- HWND m_hWnd
- HGLRC m_hRc
- HDC m_hDc
- bool m_isFullScreen
- int m_windowWidth
- int m_windowHeight

7.14.1 Detailed Description

Window in Windows. Class to handle window create and management

Definition at line 22 of file GlWindow.h.

7.14.2 Constructor & Destructor Documentation

7.14.2.1 GlWindow::GlWindow ()

Constructor

Definition at line 23 of file GlWindow.cpp.

7.14.2.2 GlWindow::~GlWindow ()

Destructor

Definition at line 34 of file GlWindow.cpp.

7.14.3 Member Function Documentation

7.14.3.1 bool GlWindow::CreateGlWindow (const std::wstring & *className*, const std::wstring & *windowTitle*, HINSTANCE *hInstance*, int *windowPosX*, int *windowPosY*, int *windowWidth*, int *windowHeight*, const bool & *fullScreen*, int *bitsPerPixel*, LPVOID *application*)

Method to create the window

Parameters:

className String to unqiuely ID the class

windowTitle Title of the window

hInstance Handle to the instance of the object

windowPosX X coord to place the window at

windowPosY Y coord to place the window at

windowWidth Width of the window

windowHeight Height of the window

fullScreen True if the window should be fullscreen

bitsPerPixel Bits per pixel

application Pointer to the application running the window

Returns:

True on success

Definition at line 40 of file GlWindow.cpp.

7.14.3.2 void GlWindow::EnableFullScreen (const int & *windowWidth*, const int & *windowHeight*, const int & *bitsPerPixel*) `[protected]`

Set the window to run in full screen mode

Parameters:

windowWidth Width of the window

windowHeight Height of the window

bitsPerPixel Bits per pixel

Definition at line 232 of file GlWindow.cpp.

7.14.3.3 int GlWindow::GetWindowHeight ()

Get The Window Height

Definition at line 279 of file GlWindow.cpp.

7.14.3.4 int GlWindow::GetWindowWidth ()

Get The Window Width

Definition at line 285 of file GlWindow.cpp.

7.14.3.5 void GlWindow::KillWindow ()

Destory and clean up the window

Definition at line 195 of file GlWindow.cpp.

7.14.3.6 GlWindow::operator HDC () [inline]

Overload to allow the GlWindow to act as a HDC

Returns:

The window's DC

Definition at line 87 of file GlWindow.h.

7.14.3.7 GlWindow::operator HWND () [inline]

Overload to allow the GlWindow to act as a HWND

Returns:

The window's Handle

Definition at line 81 of file GlWindow.h.

7.14.3.8 void GlWindow::ResizeGlScene (int *width*, int *height*)

Reshape the gl scene

Parameters:

width New window width

height New window height

Definition at line 256 of file GlWindow.cpp.

7.14.3.9 void GlWindow::SwapBuffers ()

Swaps the buffers of the window

Definition at line 250 of file GlWindow.cpp.

7.14.4 Member Data Documentation

7.14.4.1 HDC GlWindow::m_hDc [private]

Device Context

Definition at line 113 of file GlWindow.h.

7.14.4.2 HGLRC GlWindow::m_hRc [private]

Rendering Context

Definition at line 108 of file GlWindow.h.

7.14.4.3 HWND GlWindow::m_hWnd [private]

Window Handle

Definition at line 103 of file GlWindow.h.

7.14.4.4 bool GlWindow::m_isFullScreen [private]

Is the window in full screen mode

Definition at line 118 of file GlWindow.h.

7.14.4.5 int GlWindow::m_windowHeight [private]

Window Width

Definition at line 128 of file GlWindow.h.

7.14.4.6 int GlWindow::m_windowWidth [private]

Window Height

Definition at line 123 of file GlWindow.h.

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

- branches/GameBase/GlWindow.h
- branches/GameBase/GlWindow.cpp

7.15 KeyHandler Class Reference

Manages key presses.

`#include <KeyHandler.h>`

Public Member Functions

- KeyHandler ()
- void Reset ()
- void SetPressed (const unsigned int &keyNumber)
- void SetReleased (const unsigned int &keyNumber)
- bool GetPressed (const unsigned int &keyNumber)

Private Attributes

- bool m_keys [MAX_KEYS]

7.15.1 Detailed Description

Manages key presses. Class manage which keys are currently being pressed

Definition at line 24 of file KeyHandler.h.

7.15.2 Constructor & Destructor Documentation

7.15.2.1 KeyHandler::KeyHandler ()

Constructor

Definition at line 19 of file KeyHandler.cpp.

7.15.3 Member Function Documentation

7.15.3.1 bool KeyHandler::GetPressed (const unsigned int & *keyNumber*)

Gets the status of a key

Parameters:

keyNumber The key number to check

Returns:

Status of the key

Definition at line 25 of file KeyHandler.cpp.

7.15.3.2 void KeyHandler::Reset ()

Releases all keys

Definition at line 50 of file KeyHandler.cpp.

7.15.3.3 void KeyHandler::SetPressed (const unsigned int & *keyNumber*)

Sets a key as pressed

Parameters:

keyNumber Key to set pressed

Definition at line 32 of file KeyHandler.cpp.

7.15.3.4 void KeyHandler::SetReleased (const unsigned int & *keyNumber*)

Sets a key as released

Parameters:

keyNumber Key to set released

Definition at line 41 of file KeyHandler.cpp.

7.15.4 Member Data Documentation

7.15.4.1 bool KeyHandler::m_keys[MAX_KEYS] `[private]`

Array to track key presses

Definition at line 59 of file KeyHandler.h.

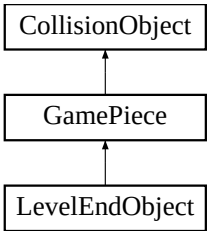
The documentation for this class was generated from the following files:

- branches/GameBase/KeyHandler.h
- branches/GameBase/KeyHandler.cpp

7.16 LevelEndObject Class Reference

Object used to track if a level has ended.

`#include <LevelEndObject.h>`Inheritance diagram for LevelEndObject::



Public Member Functions

- LevelEndObject (Square startingPos, unsigned int textureId)
- ~LevelEndObject ()
- virtual bool CheckCollision (CollisionObject *object)
- bool LevelDone ()
- virtual void Draw ()

Private Attributes

- bool m_levelDone

7.16.1 Detailed Description

Object used to track if a level has ended. Object used to check the level status. This object will return true when the level is over

Definition at line 22 of file LevelEndObject.h.

7.16.2 Constructor & Destructor Documentation

7.16.2.1 LevelEndObject::LevelEndObject (Square *startingPos*, unsigned int *textureId*)

Constructor

Parameters:

startingPos Position of the levelEndObject

textureId Image to use for the object

Definition at line 10 of file LevelEndObject.cpp.

7.16.2.2 LevelEndObject::~LevelEndObject ()

Destructor

Definition at line 16 of file LevelEndObject.cpp.

7.16.3 Member Function Documentation

7.16.3.1 bool LevelEndObject::CheckCollision (CollisionObject * *object*) [virtual]

Checks to see if object has collided with the current object

Parameters:

object Object to check a collision against

Returns:

true if the object collided

Reimplemented from GamePiece.

Definition at line 25 of file LevelEndObject.cpp.

7.16.3.2 void LevelEndObject::Draw () [virtual]

Draw the object

Reimplemented from GamePiece.

Definition at line 45 of file LevelEndObject.cpp.

7.16.3.3 bool LevelEndObject::LevelDone ()

Check if a level has been completed

Returns:

True if the level has ended

Definition at line 20 of file LevelEndObject.cpp.

7.16.4 Member Data Documentation

7.16.4.1 bool LevelEndObject::m_levelDone [private]

Status if the level has ended

Definition at line 57 of file LevelEndObject.h.

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

- branches/GameBase/LevelEndObject.h
- branches/GameBase/LevelEndObject.cpp

7.17 LevelObject Class Reference

Level in the game.

`#include <LevelObject.h>`

Public Member Functions

- LevelObject (const std::wstring &levelName)
- virtual ~LevelObject ()
- void Draw ()
- bool Update (int ticks, GameDude *gameDude)
- double Move (double distance)
- void AddGamePiece (GamePiece *piece)
- void AddAIObject (AIObject *object)
- void Start ()
- void SetLevelEndObject (LevelEndObject *object)
- void SetLevelFileName (const std::wstring &levelFileName)
- void SetImageFolder (const std::wstring &imageFolder)
- std::wstring GetImageFolder ()
- bool Load ()
- bool Reload ()
- std::wstring GetTimerString ()
- void FireSpecialPower (Square startingPos, bool direction)
- void SetSpecialImages (int leftTextureId, int rightTextureId)

Protected Attributes

- std::list< GamePiece * > m_levelObjects
- std::list< AIObject * > m_passiveAIList
- std::wstring m_levelName
- double m_xOffset
- double m_maxXOffset
- BackGroundManager * m_backGroundManager
- double m_screenWidth

Private Attributes

- std::list< AIObject * > m_activeAIList
- std::list< GamePiece * >::iterator m_screenEndIter
- std::list< GamePiece * >::iterator m_screenStartIter
- std::list< PowerObject * > m_powerList
- LevelEndObject * m_levelEndObject
- std::wstring m_levelFileName
- std::wstring m_imageFolder
- int m_timer
- int m_specialTextureIds [2]

7.17.1 Detailed Description

Level in the game. Represents a level within the game. Handles all AI objects, Special Powers and block collisions

Definition at line 30 of file LevelObject.h.

7.17.2 Constructor & Destructor Documentation

7.17.2.1 LevelObject::LevelObject (const std::wstring & *levelName*)

Constructor

Parameters:

levelName The level's name

Definition at line 16 of file LevelObject.cpp.

7.17.2.2 LevelObject::~LevelObject () `[virtual]`

Destructor

Definition at line 33 of file LevelObject.cpp.

7.17.3 Member Function Documentation

7.17.3.1 void LevelObject::AddAIObject (AIObject * *object*)

Adds a general AIObject to the level

Parameters:

object AIObject to add to the level

Definition at line 240 of file LevelObject.cpp.

7.17.3.2 void LevelObject::AddGamePiece (GamePiece * *piece*)

Adds a general GamePiece to the level

Parameters:

piece GamePiece to add to the level

Definition at line 232 of file LevelObject.cpp.

7.17.3.3 void LevelObject::Draw ()

Draw the level

Definition at line 49 of file LevelObject.cpp.

7.17.3.4 void LevelObject::FireSpecialPower (Square *startingPos*, bool *direction*)

Fires a special power object

Parameters:

startingPos Starting location of the PowerObject

direction Direction of the PowerObject

Definition at line 351 of file LevelObject.cpp.

7.17.3.5 std::wstring LevelObject::GetImageFolder ()

Get the Image Folder

Returns:

The folder name where the images are located

Definition at line 340 of file LevelObject.cpp.

7.17.3.6 std::wstring LevelObject::GetTimerString ()

Get time remain string

Returns:

Time remaining as a string

Definition at line 327 of file LevelObject.cpp.

7.17.3.7 bool LevelObject::Load ()

Loads the level from the fileName

Definition at line 287 of file LevelObject.cpp.

7.17.3.8 double LevelObject::Move (double *distance*)

Move the level *distance*

Parameters:

distance Distance to move

Returns:

The new xOffset

Definition at line 68 of file LevelObject.cpp.

7.17.3.9 bool LevelObject::Reload ()

Reloads the level from the begining

Todo

: Refactor to remove calls to this method

Definition at line 321 of file LevelObject.cpp.

7.17.3.10 void LevelObject::SetImageFolder (const std::wstring & *imageFolder*)

Sets the image folder for the level. Textures will be loaded from this folder.

Parameters:

imageFolder Folder name where the images are located

Definition at line 335 of file LevelObject.cpp.

7.17.3.11 void LevelObject::SetLevelEndObject (LevelEndObject * *object*)

Set the level end object for a level

Parameters:

object Object to use as the level end indicator

Definition at line 277 of file LevelObject.cpp.

7.17.3.12 void LevelObject::SetLevelFileName (const std::wstring & *levelFileName*)

Sets the level file name. This file will be used to load the level.

Parameters:

levelFileName Level file to load

Definition at line 282 of file LevelObject.cpp.

7.17.3.13 void LevelObject::SetSpecialImages (int *leftTextureId*, int *rightTextureId*)

Sets the images to use for the PowerObject

Parameters:

leftTextureId Image of a left moving PowerObject

rightTextureId Image of a right moving PowerObject

Definition at line 345 of file LevelObject.cpp.

7.17.3.14 void LevelObject::Start ()

Starts the level

Definition at line 248 of file LevelObject.cpp.

7.17.3.15 bool LevelObject::Update (int *ticks*, GameDude * *gameDude*)

Update the current level

Parameters:

ticks Number of ticks since the last update

gameDude GameDude object to collide against

Definition at line 124 of file LevelObject.cpp.

7.17.4 Member Data Documentation

7.17.4.1 std::list<AIObject *> LevelObject::m_activeAIList [private]

List of currently active AI Objects

Definition at line 171 of file LevelObject.h.

7.17.4.2 BackGroundManager* LevelObject::m_backGroundManager [protected]

The background image for the level

Definition at line 161 of file LevelObject.h.

7.17.4.3 std::wstring LevelObject::m_imageFolder [private]

Level image file folder path

Definition at line 201 of file LevelObject.h.

7.17.4.4 LevelEndObject* LevelObject::m_levelEndObject [private]

Object to control the level end

Definition at line 191 of file LevelObject.h.

7.17.4.5 std::wstring LevelObject::m_levelFileName [private]

Level load file name

Definition at line 196 of file LevelObject.h.

7.17.4.6 std::wstring LevelObject::m_levelName [protected]

Name of the level

Definition at line 146 of file LevelObject.h.

7.17.4.7 std::list<GamePiece *> LevelObject::m_levelObjects [protected]

All objects in the level

Definition at line 136 of file LevelObject.h.

7.17.4.8 double LevelObject::m_maxXOffset [protected]

The maximum distance a player can move into the level

Definition at line 156 of file LevelObject.h.

7.17.4.9 std::list<AIObject *> LevelObject::m_passiveAIList [protected]

List off none activated AI Objects

Definition at line 141 of file LevelObject.h.

7.17.4.10 std::list<PowerObject *> LevelObject::m_powerList [private]

List of power objects shot by the player

Definition at line 186 of file LevelObject.h.

7.17.4.11 std::list<GamePiece *>::iterator LevelObject::m_screenEndIter [private]

Marks the right side of the screen

Definition at line 176 of file LevelObject.h.

7.17.4.12 std::list<GamePiece *>::iterator LevelObject::m_screenStartIter [private]

Marks the left side of the screen

Definition at line 181 of file LevelObject.h.

7.17.4.13 double LevelObject::m_screenWidth [protected]

The screen width in openGL units

Definition at line 166 of file LevelObject.h.

7.17.4.14 int LevelObject::m_specialTextureIds[2] [private]

The images used for the special powers

Definition at line 211 of file LevelObject.h.

7.17.4.15 int LevelObject::m_timer [private]

The current timer for the level

Definition at line 206 of file LevelObject.h.

7.17.4.16 double LevelObject::m_xOffset [protected]

Distance the player has moved from the begining of the level

Definition at line 151 of file LevelObject.h.

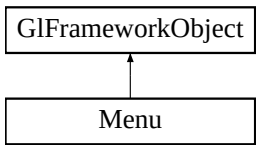
The documentation for this class was generated from the following files:

- branches/GameBase/LevelObject.h
- branches/GameBase/LevelObject.cpp

7.18 Menu Class Reference

Menu Class.

`#include <Menu.h>`Inheritance diagram for Menu::



Public Member Functions

- Menu (unsigned int textBase)
- ~Menu ()
- void Update (int ticks)
- void Draw ()
- void AddMenuItem (const std::wstring text, int id, const Square &pos)
- int GetSelectedItemId ()
- void Click (const Point &point)
- void HandleKey (unsigned int key)

Protected Attributes

- std::list< MenuItem * > m_items
- unsigned int m_textBase
- int m_clickedId
- std::list< MenuItem * >::iterator m_selectedItem

7.18.1 Detailed Description

Menu Class. Creates and manages a general menu

Definition at line 24 of file Menu.h.

7.18.2 Constructor & Destructor Documentation

7.18.2.1 Menu::Menu (unsigned int *textBase*)

Constructor

Parameters:

textBase Base of the text to use

Definition at line 11 of file Menu.cpp.

7.18.2.2 Menu::~Menu ()

Desctructor

Definition at line 18 of file Menu.cpp.

7.18.3 Member Function Documentation

7.18.3.1 void Menu::AddMenuItem (const std::wstring *text*, int *id*, const Square & *pos*)

Adds an item to the menu

Parameters:

text Text for the menu option

id Number to ID the option when it is clicked

pos Position of the option on the screen

Definition at line 44 of file Menu.cpp.

7.18.3.2 void Menu::Click (const Point & *point*)

Handles on click on the menu

Parameters:

point Point clicked

Definition at line 56 of file Menu.cpp.

7.18.3.3 void Menu::Draw () `[virtual]`

Draw the menu

Implements GlFrameworkObject.

Definition at line 30 of file Menu.cpp.

7.18.3.4 int Menu::GetSelectedItemId ()

Gets the selected item

Returns:

The selected Item Id or -1 if none selected

Definition at line 51 of file Menu.cpp.

7.18.3.5 void Menu::HandleKey (unsigned int *key*)

Handles on key press in the menu

Parameters:

key key pressed

Definition at line 69 of file Menu.cpp.

7.18.3.6 void Menu::Update (int *ticks*) [virtual]

Update the menu *ticks* ticks

Parameters:

ticks Number of ticks passed since the last call to update

Implements GlFrameworkObject.

Definition at line 26 of file Menu.cpp.

7.18.4 Member Data Documentation

7.18.4.1 int Menu::m_clickedId [protected]

Clicked item id

Definition at line 87 of file Menu.h.

7.18.4.2 std::list<MenuItem *> Menu::m_items [protected]

List of menu items to use

Definition at line 77 of file Menu.h.

7.18.4.3 std::list<MenuItem *>::iterator Menu::m_selectedItem [protected]

Highlighted menu item

Definition at line 92 of file Menu.h.

7.18.4.4 unsigned int Menu::m_textBase [protected]

Base of the text font to use

Definition at line 82 of file Menu.h.

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

- branches/GameBase/Menu.h
- branches/GameBase/Menu.cpp

7.19 MenuItem Class Reference

Items that appear within a menu.

`#include <MenuItem.h>`

Public Member Functions

- MenuItem (const std::wstring &text, Square pos, int id, unsigned int textBase)
- virtual ~MenuItem ()
- bool ContainPoint (const Point &point)
- int GetId ()
- void Draw ()
- void SetSelectStatus (bool status)

Protected Attributes

- Square m_position
- std::wstring m_text
- int m_menuId
- unsigned int m_textBase
- bool m_selected

7.19.1 Detailed Description

Items that appear within a menu. Item used to represent menu options within a menu

Definition at line 22 of file MenuItem.h.

7.19.2 Constructor & Destructor Documentation

7.19.2.1 MenuItem::MenuItem (const std::wstring & *text*, Square *pos*, int *id*, unsigned int *textBase*)

Constructor

Parameters:

text Menu Option Text

pos Position on the screen of the option

id Id of the option

textBase Base number of the text

Definition at line 6 of file MenuItem.cpp.

7.19.2.2 MenuItem::~MenuItem () `[virtual]`

Destructor

Definition at line 15 of file MenuItem.cpp.

7.19.3 Member Function Documentation

7.19.3.1 bool MenuItem::ContainPoint (const Point & *point*)

Checks to see if a given point is on the option

Parameters:

point Clicked Point to check

Returns:

True is the point in on the option

Definition at line 19 of file MenuItem.cpp.

7.19.3.2 void MenuItem::Draw ()

Draw the menu option

Definition at line 34 of file MenuItem.cpp.

7.19.3.3 int MenuItem::GetId ()

Gets the ID of the menu option

Returns:

Menu Option Id

Definition at line 29 of file MenuItem.cpp.

7.19.3.4 void MenuItem::SetSelectStatus (bool *status*)

Sets if the option is the currently selected one

Definition at line 60 of file MenuItem.cpp.

7.19.4 Member Data Documentation

7.19.4.1 int MenuItem::m_menuId [protected]

The id of the option

Definition at line 74 of file MenuItem.h.

7.19.4.2 Square MenuItem::m_position [protected]

Position of the option

Definition at line 64 of file MenuItem.h.

7.19.4.3 bool MenuItem::m_selected [protected]

If the option is selected

Definition at line 84 of file MenuItem.h.

7.19.4.4 std::wstring MenuItem::m_text [protected]

Text to display for the option

Definition at line 69 of file MenuItem.h.

7.19.4.5 unsigned int MenuItem::m_textBase [protected]

The base number for the font

Definition at line 79 of file MenuItem.h.

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

- branches/GameBase/MenuItem.h
- branches/GameBase/MenuItem.cpp

7.20 Point Struct Reference

A single point Represents a single point in openGL space.

`#include <Point.h>`

Public Member Functions

- Point (double x=0.0, double y=0.0, double z=0.0)

Public Attributes

- double x
- double y
- double z

7.20.1 Detailed Description

A single point Represents a single point in openGL space.

Definition at line 19 of file Point.h.

7.20.2 Constructor & Destructor Documentation

7.20.2.1 Point::Point (double *x* = 0.0, double *y* = 0.0, double *z* = 0.0) [inline]

Definition at line 21 of file Point.h.

7.20.3 Member Data Documentation

7.20.3.1 double Point::x

Definition at line 26 of file Point.h.

7.20.3.2 double Point::y

Definition at line 28 of file Point.h.

7.20.3.3 double Point::z

Definition at line 29 of file Point.h.

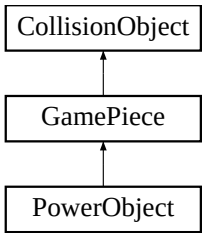
The documentation for this struct was generated from the following file:

- branches/GameBase/Point.h

7.21 PowerObject Class Reference

The special power.

`#include <PowerObject.h>`Inheritance diagram for PowerObject::



Public Member Functions

- PowerObject (const Square &startingPos, bool direction, unsigned int leftTextureId, unsigned int rightTextureId)
- virtual ~PowerObject ()
- virtual void Update (int ticks)
- virtual void Draw ()
- bool IsDead ()
- virtual bool CheckCollision (CollisionObject *object)
- virtual bool Collide (CollisionSideEnum side, int damage)

Protected Attributes

- bool m_direction
- bool m_active

7.21.1 Detailed Description

The special power. The object that game dude fires when using the special powers

Definition at line 22 of file PowerObject.h.

7.21.2 Constructor & Destructor Documentation

7.21.2.1 PowerObject::PowerObject (const Square & *startingPos*, bool *direction*, unsigned int *leftTextureId*, unsigned int *rightTextureId*)

Constuctor

Parameters:

startingPos Starting position of the special power

direction Direction the object should be moving

leftTextureId Left moving image

rightTextureId Right moving image

Definition at line 12 of file PowerObject.cpp.

7.21.2.2 PowerObject::~PowerObject () [virtual]

Destructor

Definition at line 19 of file PowerObject.cpp.

7.21.3 Member Function Documentation

7.21.3.1 bool PowerObject::CheckCollision (CollisionObject * *object*) [virtual]

Checks to see if object has collided with the current object

Parameters:

object Object to check a collision against

Returns:

true if the object collided

Reimplemented from GamePiece.

Definition at line 51 of file PowerObject.cpp.

7.21.3.2 bool PowerObject::Collide (CollisionSideEnum *side*, int *damage*) [virtual]

Handles the event of another object colliding with the current one

Parameters:

side Side of the current object that has been hit
damage Amount of damage to attempt to inflict upon the current object

Returns:

true if the current object wishes to return damage

Reimplemented from GamePiece.

Definition at line 70 of file PowerObject.cpp.

7.21.3.3 void PowerObject::Draw () [virtual]

Draw the PowerObject

Reimplemented from GamePiece.

Definition at line 32 of file PowerObject.cpp.

7.21.3.4 bool PowerObject::IsDead ()

Check if the PowerObject has died

Returns:

If the power object has hit something

Definition at line 46 of file PowerObject.cpp.

7.21.3.5 void PowerObject::Update (int *ticks*) [virtual]

Updates the power object

Parameters:

ticks Ticks passed since the last call to update

Definition at line 23 of file PowerObject.cpp.

7.21.4 Member Data Documentation

7.21.4.1 bool PowerObject::m_active [protected]

Is the piece currently active

Definition at line 78 of file PowerObject.h.

7.21.4.2 bool PowerObject::m_direction [protected]

Direction the object is moving

Definition at line 73 of file PowerObject.h.

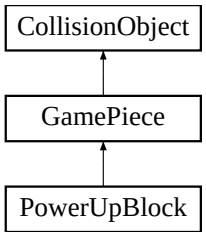
The documentation for this class was generated from the following files:

- branches/GameBase/PowerObject.h
- branches/GameBase/PowerObject.cpp

7.22 PowerUpBlock Class Reference

Block that creates a PowerUpItem.

`#include <PowerUpBlock.h>`Inheritance diagram for PowerUpBlock::



Public Member Functions

- PowerUpBlock (const Square &startingPos, PowerUpItem *item, unsigned int unusedTextureId, unsigned int usedTextureId)
- virtual ~PowerUpBlock ()
- virtual bool Collide (CollisionSideEnum side, int damage)

Private Attributes

- unsigned int m_textureIds [2]
- bool m_blockUsed
- PowerUpItem * m_item

7.22.1 Detailed Description

Block that creates a PowerUpItem. Special Power Block. Upon hitting, it triggers the PowerUpItem

Definition at line 21 of file PowerUpBlock.h.

7.22.2 Constructor & Destructor Documentation

7.22.2.1 PowerUpBlock::PowerUpBlock (const Square & *startingPos*, PowerUpItem * *item*, unsigned int *unusedTextureId*, unsigned int *usedTextureId*)

Constructor

Parameters:

startingPos The position of the block

item PowerUpItem to trigger once the block is hit

unusedTextureId Image to display before the block is hit

usedTextureId Image to display after the block is hit

Definition at line 4 of file PowerUpBlock.cpp.

7.22.2.2 PowerUpBlock::~PowerUpBlock () [virtual]

Destructor

Definition at line 13 of file PowerUpBlock.cpp.

7.22.3 Member Function Documentation

7.22.3.1 bool PowerUpBlock::Collide (CollisionSideEnum *side*, int *damage*) [virtual]

Handles the event of another object colliding with the current one

Parameters:

side Side of the current object that has been hit

damage Amount of damage to attempt to inflict upon the current object

Returns:

true if the current object wishes to return damage

Reimplemented from GamePiece.

Definition at line 17 of file PowerUpBlock.cpp.

7.22.4 Member Data Documentation

7.22.4.1 bool PowerUpBlock::m_blockUsed [private]

Has the power block been hit

Definition at line 53 of file PowerUpBlock.h.

7.22.4.2 PowerUpItem* PowerUpBlock::m_item [private]

Item to trigger when hit

Definition at line 58 of file PowerUpBlock.h.

7.22.4.3 unsigned int PowerUpBlock::m_textureIds[2] [private]

Texture Ids to use for the power up block

Definition at line 48 of file PowerUpBlock.h.

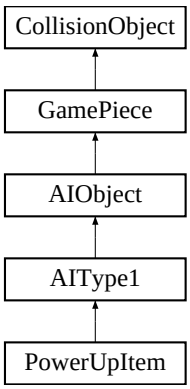
The documentation for this class was generated from the following files:

- branches/GameBase/PowerUpBlock.h
- branches/GameBase/PowerUpBlock.cpp

7.23 PowerUpItem Class Reference

Power Up Item.

`#include <PowerUpItem.h>`Inheritance diagram for PowerUpItem::



Public Member Functions

- PowerUpItem (const Square &startingPos, unsigned int textureId)
- virtual ~PowerUpItem ()
- virtual bool Trigger (double xOffset)
- virtual void Update (int ticks)
- void Activate ()
- void SetVerticalStatus (VerticalStatus status)
- virtual bool Collide (CollisionSideEnum side, int damage)
- virtual bool CheckCollision (CollisionObject *object)

Protected Attributes

- bool m_triggered
- double m_jumpHeight

7.23.1 Detailed Description

Power Up Item. Power up item, upon touching the game dude status is increased by 1

Definition at line 22 of file PowerUpItem.h.

7.23.2 Constructor & Destructor Documentation

7.23.2.1 PowerUpItem::PowerUpItem (const Square & *startingPos*, unsigned int *textureId*)

Constructor

Parameters:

startingPos Starting position of the object

textureId Image to use

Definition at line 10 of file PowerUpItem.cpp.

7.23.2.2 PowerUpItem::~PowerUpItem () [virtual]

Destructor

Definition at line 17 of file PowerUpItem.cpp.

7.23.3 Member Function Documentation

7.23.3.1 void PowerUpItem::Activate ()

Set the object to return true on the new trigger

Definition at line 50 of file PowerUpItem.cpp.

7.23.3.2 bool PowerUpItem::CheckCollision (CollisionObject * *object*) [virtual]

Checks to see if object has collided with the current object

Parameters:

object Object to check a collision against

Returns:

true if the object collided

Reimplemented from AIType1.

Definition at line 70 of file PowerUpItem.cpp.

7.23.3.3 bool PowerUpItem::Collide (CollisionSideEnum *side*, int *damage*) [virtual]

Handles the event of another object colliding with the current one

Parameters:

side Side of the current object that has been hit

damage Amount of damage to attempt to inflict upon the current object

Returns:

true if the current object wishes to return damage

Reimplemented from AIType1.

Definition at line 61 of file PowerUpItem.cpp.

7.23.3.4 void PowerUpItem::SetVerticalStatus (VerticalStatus *status*) [virtual]

Set the vertical status

Parameters:

status The new vertical status of the object

Reimplemented from AIObject.

Definition at line 99 of file PowerUpItem.cpp.

7.23.3.5 bool PowerUpItem::Trigger (double *xOffset*) [virtual]

Checks if the object should be activated

Parameters:

xOffset Not Used on this object

Reimplemented from AIType1.

Definition at line 40 of file PowerUpItem.cpp.

7.23.3.6 void PowerUpItem::Update (int *ticks*) [virtual]

Performs the update on the object

Parameters:

ticks Number of ticks that have passed since the last update

Reimplemented from AIType1.

Definition at line 21 of file PowerUpItem.cpp.

7.23.4 Member Data Documentation

7.23.4.1 double PowerUpItem::m_jumpHeight [protected]

How height has the object gone

Definition at line 82 of file PowerUpItem.h.

7.23.4.2 bool PowerUpItem::m_triggered [protected]

Has the object been triggered

Definition at line 77 of file PowerUpItem.h.

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

- branches/GameBase/PowerUpItem.h
- branches/GameBase/PowerUpItem.cpp

7.24 RgbaColor Struct Reference

OpenGL Color.

`#include <RgbaColor.h>`

Public Attributes

- unsigned char red
- unsigned char green
- unsigned char blue
- unsigned char alpha

7.24.1 Detailed Description

OpenGL Color. Struct represents the 4 values of an openGL using 4 unsigned bytes

Definition at line 20 of file RgbaColor.h.

7.24.2 Member Data Documentation

7.24.2.1 unsigned char RgbaColor::alpha

Definition at line 25 of file RgbaColor.h.

7.24.2.2 unsigned char RgbaColor::blue

Definition at line 24 of file RgbaColor.h.

7.24.2.3 unsigned char RgbaColor::green

Definition at line 23 of file RgbaColor.h.

7.24.2.4 unsigned char RgbaColor::red

Definition at line 22 of file RgbaColor.h.

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

- branches/GameBase/RgbaColor.h

7.25 ScoreManager Class Reference

Tracks the players score.

`#include <ScoreManager.h>`

Public Member Functions

- unsigned int GetCurrentScore ()
- void AddToScore (unsigned int points, ScoreObject objectType)
- void Reset ()
- void NewLevel ()
- void ResetLevel ()

Static Public Member Functions

- static ScoreManager * Instance ()

Protected Member Functions

- ScoreManager ()
- virtual ~ScoreManager ()
- ScoreManager (const ScoreManager &)
- ScoreManager & operator= (const ScoreManager &)

Protected Attributes

- unsigned int m_score
- unsigned int m_levelScore

Static Protected Attributes

- static ScoreManager * m_instance = 0

7.25.1 Detailed Description

Tracks the players score. Class to keep trace of the current playes score. Singleton Object

Definition at line 22 of file ScoreManager.h.

7.25.2 Constructor & Destructor Documentation

7.25.2.1 ScoreManager::ScoreManager () `[protected]`

Constructor

Definition at line 3 of file ScoreManager.cpp.

7.25.2.2 ScoreManager::~ScoreManager () [protected, virtual]

Desctructor

Definition at line 8 of file ScoreManager.cpp.

7.25.2.3 ScoreManager::ScoreManager (const ScoreManager & *temp*) [protected]

Definition at line 50 of file ScoreManager.cpp.

7.25.3 Member Function Documentation

7.25.3.1 void ScoreManager::AddToScore (unsigned int *points*, ScoreObject *objectType*)

Add a given number of points to the player's score

Parameters:

points Number of points to add to the player's score
objectType Reason points are being added

Definition at line 12 of file ScoreManager.cpp.

7.25.3.2 unsigned int ScoreManager::GetCurrentScore ()

Get the current score

Returns:

The current score

Definition at line 18 of file ScoreManager.cpp.

7.25.3.3 ScoreManager * ScoreManager::Instance () [static]

Method to obtain the singleton instance

Returns:

The singleton instance of the ScoreManager

Definition at line 40 of file ScoreManager.cpp.

7.25.3.4 void ScoreManager::NewLevel ()

Starts a new level. Resets the current level counter to 0

Definition at line 35 of file ScoreManager.cpp.

7.25.3.5 ScoreManager & ScoreManager::operator= (const ScoreManager & *temp*) [protected]

Definition at line 55 of file ScoreManager.cpp.

7.25.3.6 void ScoreManager::Reset ()

Reset the score and current level score to 0

Definition at line 23 of file ScoreManager.cpp.

7.25.3.7 void ScoreManager::ResetLevel ()

Removes points earned in this level from the score. Resets the current level count to 0.

Definition at line 29 of file ScoreManager.cpp.

7.25.4 Member Data Documentation

7.25.4.1 ScoreManager * ScoreManager::m_instance = 0 [static, protected]

Singleton instance

Definition at line 84 of file ScoreManager.h.

7.25.4.2 unsigned int ScoreManager::m_levelScore [protected]

Score earned in the current level

Definition at line 79 of file ScoreManager.h.

7.25.4.3 unsigned int ScoreManager::m_score [protected]

Total score so far

Definition at line 74 of file ScoreManager.h.

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

- branches/GameBase/ScoreManager.h
- branches/GameBase/ScoreManager.cpp

7.26 Square Struct Reference

I'm a square.

`#include <GameStructs.h>`

Public Member Functions

- Square (double top, double bottom, double left, double right)

Public Attributes

- double top
- double bottom
- double left
- double right

7.26.1 Detailed Description

I'm a square. Struct used to represent a regtangle within the game

Definition at line 26 of file GameStructs.h.

7.26.2 Constructor & Destructor Documentation

7.26.2.1 Square::Square (double *top*, double *bottom*, double *left*, double *right*) [inline]

Definition at line 28 of file GameStructs.h.

7.26.3 Member Data Documentation

7.26.3.1 double Square::bottom

Definition at line 37 of file GameStructs.h.

7.26.3.2 double Square::left

Definition at line 38 of file GameStructs.h.

7.26.3.3 double Square::right

Definition at line 39 of file GameStructs.h.

7.26.3.4 double Square::top

Definition at line 36 of file GameStructs.h.

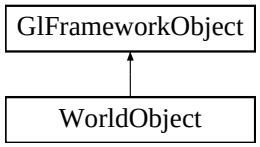
The documentation for this struct was generated from the following file:

- branches/GameBase/GameStructs.h

7.27 WorldObject Class Reference

World in the game.

`#include <WorldObject.h>`Inheritance diagram for WorldObject::



Public Member Functions

- WorldObject (const std::wstring &name, GameDude *dude)
- void SetWorldName (const std::wstring &name)
- virtual ~WorldObject ()
- void Draw ()
- void Update (int ticks)
- void Start ()
- double Move (double distance)
- void AddLevel (LevelObject *level)
- bool WorldDone ()
- bool RestartCurrentLevel ()
- std::wstring GetTimerString ()
- void FireSpecialPower (Square startingPos, bool direction)

Protected Attributes

- std::wstring m_worldName
- std::list< LevelObject * > m_levelList
- std::list< LevelObject * >::iterator m_currentLevel
- GameDude * m_gameDude

7.27.1 Detailed Description

World in the game. Represents a current world within the game. Handles level loading and management

Definition at line 25 of file WorldObject.h.

7.27.2 Constructor & Destructor Documentation

7.27.2.1 WorldObject::WorldObject (const std::wstring & *name*, GameDude * *dude*)

Constructor

Parameters:

name World Name

dude The player's GameDude

Definition at line 4 of file WorldObject.cpp.

7.27.2.2 WorldObject::~WorldObject () [virtual]

Destructor

Definition at line 11 of file WorldObject.cpp.

7.27.3 Member Function Documentation

7.27.3.1 void WorldObject::AddLevel (LevelObject * *level*)

Adds a level to the current world

Remarks:

Should only be used by the GameLoaders

Parameters:

level Level to add

Definition at line 39 of file WorldObject.cpp.

7.27.3.2 void WorldObject::Draw () [virtual]

Draw the world

Implements GlFrameworkObject.

Definition at line 34 of file WorldObject.cpp.

7.27.3.3 void WorldObject::FireSpecialPower (Square *startingPos*, bool *direction*)

Fire the special power of the game dude

Parameters:

startingPos Starting Position of the fire

direction Direction to fire

Definition at line 78 of file WorldObject.cpp.

7.27.3.4 std::wstring WorldObject::GetTimerString ()

Get time remain string

Returns:

Time remaining as a string

Definition at line 73 of file WorldObject.cpp.

7.27.3.5 double WorldObject::Move (double *distance*)

Move the player *distance* through the world

Parameters:

distance Distance to move

Returns:

New xOffset

Definition at line 15 of file WorldObject.cpp.

7.27.3.6 bool WorldObject::RestartCurrentLevel ()

Restarts the current level in a world

Returns:

true if successful

Definition at line 63 of file WorldObject.cpp.

7.27.3.7 void WorldObject::SetWorldName (const std::wstring & *name*)

Sets the world's world name

Parameters:

name The new name for the world

Definition at line 58 of file WorldObject.cpp.

7.27.3.8 void WorldObject::Start ()

Starts the current world

Definition at line 46 of file WorldObject.cpp.

7.27.3.9 void WorldObject::Update (int *ticks*) `[virtual]`

Method to update the object

Parameters:

ticks Number of ticks passed since the last call to update

Implements GlFrameworkObject.

Definition at line 20 of file WorldObject.cpp.

7.27.3.10 bool WorldObject::WorldDone ()

Checks to see if the world is done

Returns:

true if the world has been completed

Definition at line 53 of file WorldObject.cpp.

7.27.4 Member Data Documentation

7.27.4.1 std::list<LevelObject *>::iterator WorldObject::m_currentLevel [protected]

The current level for the world

Definition at line 113 of file WorldObject.h.

7.27.4.2 GameDude* WorldObject::m_gameDude [protected]

The current game dude

Definition at line 118 of file WorldObject.h.

7.27.4.3 std::list<LevelObject *> WorldObject::m_levelList [protected]

List of levels in the world

Definition at line 108 of file WorldObject.h.

7.27.4.4 std::wstring WorldObject::m_worldName [protected]

The world's name

Definition at line 103 of file WorldObject.h.

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

- branches/GameBase/WorldObject.h
- branches/GameBase/WorldObject.cpp

Chapter 8

File Documentation

8.1 branches/GameBase/AIObject.cpp File Reference

```
#include "AIObject.h"
#include <windows.h>
#include <gl\gl.h>
```

8.2 branches/GameBase/AIObject.h File Reference

```
#include "GamePiece.h"
```

Classes

- class AIObject

  Base class for all AI controled objects.

8.2.1 Detailed Description

Author:

Ryan Fleming <rfleming71@gmail.com>

Version:

1.0

File contains the base class for all AI controlled objects

Definition in file AIObject.h.

8.3 branches/GameBase/AIType1.cpp File Reference

```
#include "AIType1.h"
#include "GameEnums.h"
#include "GameDude.h"
#include "ScoreManager.h"
```

Defines

- #define TRIGGER_DISTANCE 2
- #define LEFT false
- #define RIGHT true
- #define MOVE_DISTANCE 0.005
- #define FALL_SPEED 0.03

8.3.1 Define Documentation

8.3.1.1 #define FALL_SPEED 0.03

Definition at line 10 of file AIType1.cpp.

8.3.1.2 #define LEFT false

Definition at line 7 of file AIType1.cpp.

8.3.1.3 #define MOVE_DISTANCE 0.005

Definition at line 9 of file AIType1.cpp.

8.3.1.4 #define RIGHT true

Definition at line 8 of file AIType1.cpp.

8.3.1.5 #define TRIGGER_DISTANCE 2

Definition at line 6 of file AIType1.cpp.

8.4 branches/GameBase/AIType1.h File Reference

```
#include "AIObject.h"
#include "GameEnums.h"
```

Classes

- class AIType1
  Dumb AI Class.

8.4.1 Detailed Description

Author:

Ryan Fleming <rfleming71@gmail.com>

Version:

1.0

Style 1 AI Controlled Enemy AI will bounce of blocks and walk of cliffs

Definition in file AIType1.h.

8.5 branches/GameBase/AIType2.cpp File Reference

```
#include "AIType2.h"
```

8.6 branches/GameBase/AIType2.h File Reference

```
#include "AIType1.h"
#include "GameEnums.h"
```

Classes

- class AIType2

  Not so dumb AI Class.

8.6.1 Detailed Description

Author:

Ryan Fleming <rfleming71@gmail.com>

Version:

1.0

Style 1 AI Controlled Enemy AI will bounce of blocks and will not walk of cliffs

Definition in file AIType2.h.

8.7 branches/GameBase/AudioManager.cpp File Reference

```
#include "AudioManager.h"
#include <al.h>
#include <alc.h>
#include <al\alut.h>
#include <conio.h>
#include <cstdlib>
#include <iostream>
```

8.7.1 Detailed Description

Author:

Jon Caron <caronj88@gmail.com>

Version:

1.0

This is the AudioManager class for the SuperEastGate. The functions this class performs are; looping background sounds, gameplay sounds based on action, pausing sounds, stopping sounds, and possibly loading sounds on demmand.

Definition in file AudioManager.cpp.

8.8 branches/GameBase/AudioManager.h File Reference

```
#include <al.h>
#include <alc.h>
#include <al\alut.h>
#include <conio.h>
#include <cstdlib>
#include <iostream>
```

Classes

- class AudioManager

  Singeton Object to play and pause sounds.

Enumerations

- enum SoundLookup {
  SL_SONG1 = 0, SL_SONG2 = 1, SL_SONG3 = 2, SL_SONG4 = 3,
  SL_PWRUP = 4, SL_CHCKPT = 5, SL_COINS = 6, SL_HITBRICK = 7 }

8.8.1 Detailed Description

Author:

Jon Caron <caronj88@gmail.com>

Version:

1.0

Contains all relivant classes and enums for playing audio

Definition in file AudioManager.h.

8.8.2 Enumeration Type Documentation

8.8.2.1 enum SoundLookup

enum to use to play the appropriate sound

Enumerator:

SL_SONG1

SL_SONG2

SL_SONG3

SL_SONG4

SL_PWRUP

SL_CHCKPT
SL_COINS
SL_HITBRICK

Definition at line 25 of file AudioManager.h.

8.9 branches/GameBase/BackGroundManager.cpp File Reference

```
#include "BackGroundManager.h"
#include "GraphicLoaders.h"
#include <windows.h>
#include <gl\gl.h>
```

8.10 branches/GameBase/BackGroundManager.h File Reference

```
#include "GraphicLoaders.h"
#include <string>
```

Classes

- class BackGroundManager

  Displays the background of the game.

8.10.1 Detailed Description

Author:

Ryan Fleming <rfleming71@gmail.com>

Version:

1.0

Class to manage which section of the background image to display to the user.

Definition in file BackGroundManager.h.

8.11 branches/GameBase/CollisionObject.h File Reference

```
#include "GameEnums.h"
```

Classes

- class CollisionObject

  Basic Collision Detection Object.

8.11.1 Detailed Description

Author:

Ryan Fleming <rfleming71@gmail.com>

Version:

1.0

This file contains the interface used for the collision detection system

Definition in file CollisionObject.h.

8.12 branches/GameBase/ControlObject.cpp File Reference

```
#include "ControlObject.h"
#include <string>
#include <windows.h>
#include <fstream>
#include "UtilFunctions.h"
#include "Converter.h"
```

8.13 branches/GameBase/ControlObject.h File Reference

```
#include <string>
```

Classes

- class ControlObject

  Keyboard Control Manager.

Enumerations

- enum Controls {
  CO_RIGHT = 0, CO_LEFT = 1, CO_JUMP = 2, CO_CROUCH = 3,
  CO_PAUSE = 4, CO_USE_SPECIAL = 5, CO_MAX_CONTROL }

8.13.1 Detailed Description

Author:

Ryan Fleming <rfleming71@gmail.com>

Version:

1.0

Contains all relevant code for user configurable controls

Definition in file ControlObject.h.

8.13.2 Enumeration Type Documentation

8.13.2.1 enum Controls

Unqiuely ids each control the player can use

Enumerator:

CO_RIGHT
CO_LEFT
CO_JUMP
CO_CROUCH
CO_PAUSE
CO_USE_SPECIAL
CO_MAX_CONTROL

Definition at line 18 of file ControlObject.h.

8.14 branches/GameBase/Converter.cpp File Reference

```
#include "Converter.h"
#include <sstream>
#include <string>
```

8.15 branches/GameBase/Converter.h File Reference

```
#include <exception>
#include <string>
```

Classes

- class Converter::ConverterException
  Converter *Exception.*

Namespaces

- namespace Converter
  Conversion Functions.

Functions

- std::string Converter::IntToString (const int &value)
- std::string Converter::UIntToString (const unsigned int &value)
- int Converter::StringToInt (const std::string &value)
- double Converter::StringToDouble (const std::string &value)
- unsigned int Converter::StringToUInt (const std::string &value)
- std::wstring Converter::StringToWString (const std::string &stringToConvert)
- std::string Converter::WStringToString (const std::wstring &stringToConvert)

8.15.1 Detailed Description

Author:

Ryan Fleming <rfleming71@gmail.com>

Version:

1.0

This is the converter utility functions This namespace supplies funtions to perform basic string conversions

Definition in file Converter.h.

8.16 branches/GameBase/GameBase.cpp File Reference

```
#include "GameBase.h"
#include "WorldObject.h"
#include "GraphicLoaders.h"
#include "GameLoader.h"
#include "GameEnums.h"
#include <windows.h>
#include <gl\gl.h>
#include "Converter.h"
#include "ScoreManager.h"
#include "Menu.h"
```

Defines

- #define FIRE_DELAY 1000

8.16.1 Define Documentation

8.16.1.1 #define FIRE_DELAY 1000

Definition at line 12 of file GameBase.cpp.

8.17 branches/GameBase/GameBase.h File Reference

```
#include <list>
#include "GlApplication.h"
#include "WorldObject.h"
#include "GameDude.h"
#include "GameEnums.h"
#include "ControlObject.h"
#include "Menu.h"
```

Classes

- class GameBase

  Primary Game Class.

8.17.1 Detailed Description

Author:

Ryan Fleming <rfleming71@gmail.com>

Version:

1.0

Primary class for the game. This is the primary class to control the program

Definition in file GameBase.h.

8.18 branches/GameBase/GameDude.cpp File Reference

```
#include "GameDude.h"
#include "GameStructs.h"
#include <windows.h>
#include <gl\gl.h>
```

Defines

- #define JUMP_HEIGHT SQUARE_SIZE * 1.5
- #define JUMP_RATE 0.03

8.18.1 Define Documentation

8.18.1.1 #define JUMP_HEIGHT SQUARE_SIZE * 1.5

Definition at line 6 of file GameDude.cpp.

8.18.1.2 #define JUMP_RATE 0.03

Definition at line 7 of file GameDude.cpp.

8.19 branches/GameBase/GameDude.h File Reference

```
#include "GamePiece.h"
#include "GameEnums.h"
#include "GameStructs.h"
#include "GlFrameworkObject.h"
```

Classes

- class GameDude

  The player avatar.

8.19.1 Detailed Description

Author:

Ryan Fleming <rfleming71@gmail.com>

Version:

1.0

Game Dude object represents the players avatar in the game In maintians the states of the player

Definition in file GameDude.h.

8.20 branches/GameBase/GameEnums.h File Reference

Enumerations

- enum CollisionSideEnum { CS_TOP = 0, CS_RIGHT = 1, CS_BOTTOM = 2, CS_LEFT = 3 }
- enum GameDudeStatus { GDS_DEAD = 0, GDS_SMALL = 1, GDS_BIG = 2, GDS_SPECIAL = 3 }
- enum HoriztonalStatus { HS_LEFT = 0, HS_NONE = 1, HS_RIGHT = 2 }
- enum VerticalStatus { VS_JUMPING = 0, VS_NONE = 1, VS_FALLING = 2 }
- enum GameObjects {
  GO_BRICK_BLOCK = 0, GO_AI_TYPE1 = 1, GO_AI_TYPE2 = 2, GO_PIPE = 3,
  GO_LEVEL_END = 4, GO_SPECIAL_BLOCK = 5 }
- enum GameState {
  GS_STARTING_MENU = 0, GS_OPTIONS_MENU = 1, GS_PAUSE_MENU = 2, GS_GAME_PLAYING = 3,
  GS_PLAYER_DEAD = 4, GS_GAME_CREDITS = 5, GS_QUITING = 6 }
- enum ScoreObject { SO_AI_TYPE_1 = 0, SO_AI_TYPE_2 = 1, SO_LEVEL_END = 2 }
- enum StartMenuItem {
  SMI_NEW_GAME = 0, SMI_LOAD_GAME = 1, SMI_OPTIONS = 2, SMI_QUIT = 3,
  SMI_INVALID = -1 }

8.20.1 Detailed Description

Author:

Ryan Fleming <rfleming71@gmail.com>

Version:

1.0

This file contains the majority of the enums used throught the game

Definition in file GameEnums.h.

8.20.2 Enumeration Type Documentation

8.20.2.1 enum CollisionSideEnum

Used in the collision detection system. Used to indicate which side of an object has been collided with

Enumerator:

CS_TOP

CS_RIGHT

CS_BOTTOM

CS_LEFT

Definition at line 18 of file GameEnums.h.

8.20.2.2 enum GameDudeStatus

This enum represents the current state of the player character within the game

Enumerator:

GDS_DEAD

GDS_SMALL

GDS_BIG

GDS_SPECIAL

Definition at line 30 of file GameEnums.h.

8.20.2.3 enum GameObjects

Used to track object id used in the game loader each id allows a given game object to be loaded into a level

Enumerator:

GO_BRICK_BLOCK

GO_AI_TYPE1

GO_AI_TYPE2

GO_PIPE

GO_LEVEL_END

GO_SPECIAL_BLOCK

Definition at line 62 of file GameEnums.h.

8.20.2.4 enum GameState

This enum tracks the current action the game is performing

Enumerator:

GS_STARTING_MENU

GS_OPTIONS_MENU

GS_PAUSE_MENU

GS_GAME_PLAYING

GS_PLAYER_DEAD

GS_GAME_CREDITS

GS_QUITING

Definition at line 75 of file GameEnums.h.

8.20.2.5 enum HoriztonalStatus

Used to represent an objects current horizontal movement

Enumerator:

HS_LEFT

HS_NONE

HS_RIGHT

Definition at line 41 of file GameEnums.h.

8.20.2.6 enum ScoreObject

Used to track each object when it adds to the players score

Enumerator:

SO_AI_TYPE_1

SO_AI_TYPE_2

SO_LEVEL_END

Definition at line 89 of file GameEnums.h.

8.20.2.7 enum StartMenuItem

Used to unquiely ID each option located in the starting menu

Enumerator:

SMI_NEW_GAME

SMI_LOAD_GAME

SMI_OPTIONS

SMI_QUIT

SMI_INVALID

Definition at line 99 of file GameEnums.h.

8.20.2.8 enum VerticalStatus

Used to represent an objects current vertical movement

Enumerator:

VS_JUMPING

VS_NONE

VS_FALLING

Definition at line 51 of file GameEnums.h.

8.21 branches/GameBase/GameLoader.cpp File Reference

```
#include "GameLoader.h"
#include <list>
#include <string>
#include <fstream>
#include "LevelObject.h"
#include "WorldObject.h"
#include "GamePiece.h"
#include "UtilFunctions.h"
#include "Converter.h"
#include "GameStructs.h"
#include "GameEnums.h"
#include "GraphicLoaders.h"
#include "GameDude.h"
#include "LevelEndObject.h"
#include "AIType1.h"
#include "AIType2.h"
#include "PowerUpBlock.h"
#include "PowerUpItem.h"
#include "AudioManager.h"
```

8.22 branches/GameBase/GameLoader.h File Reference

```
#include <list>
#include <string>
#include "WorldObject.h"
#include "LevelObject.h"
#include "GameStructs.h"
```

Namespaces

- namespace GameLoader

  Used to convert images into openGL texture ids.

Functions

- bool GameLoader::RunLoader (const std::wstring &worldsFileName, std::list< WorldObject * > &list, GameDude *dude)
- bool GameLoader::LoadLevel (const std::wstring &levelFileName, LevelObject *level)
- Square GameLoader::GameGridToCoords (double x, double y)

8.22.1 Detailed Description

Author:

Ryan Fleming <rfleming71@gmail.com>

Version:

1.0

This namespace supplies the neccessary functions to load All game control objects from the given files

Definition in file GameLoader.h.

8.23 branches/GameBase/GamePiece.cpp File Reference

```
#include "GamePiece.h"
#include "GameStructs.h"
#include "AudioManager.h"
#include <windows.h>
#include <gl/gl.h>
```

Defines

- #define _ENABLE_BREAKABLE_BLOCKS_

8.23.1 Define Documentation

8.23.1.1 #define _ENABLE_BREAKABLE_BLOCKS_

Definition at line 8 of file GamePiece.cpp.

8.24 branches/GameBase/GamePiece.h File Reference

```
#include "GameStructs.h"
#include "CollisionObject.h"
```

Classes

- class GamePiece

  General Game Object.

8.24.1 Detailed Description

Author:

Ryan Fleming <rfleming71@gmail.com>

Version:

1.0

Base class for all objects the player can see on the screen and the GameDude can interact with.

Definition in file GamePiece.h.

8.25 branches/GameBase/GameStructs.h File Reference

Classes

- struct Square

  I'm a square.

Defines

- #define SQUARE_SIZE 10.0

8.25.1 Detailed Description

Author:

Ryan Fleming <rfleming71@gmail.com>

Version:

1.0

This file contains the majority of the structs used throught the game

Definition in file GameStructs.h.

8.25.2 Define Documentation

8.25.2.1 #define SQUARE_SIZE 10.0

This value represents the size of each game block

Definition at line 18 of file GameStructs.h.

8.26 branches/GameBase/GlApplication.cpp File Reference

```
#include "GlApplication.h"
#include <string>
#include <windows.h>
#include <gl\gl.h>
#include <gl\glu.h>
#include "GlFrameworkObject.h"
```

Defines

- #define CLASS_NAME L"OpenGl"

Functions

- LRESULT CALLBACK WindowProc (HWND hWnd, UINT uMsg, WPARAM wParam, LPARAM lParam)

8.26.1 Define Documentation

8.26.1.1 #define CLASS_NAME L"OpenGl"

Definition at line 25 of file GlApplication.cpp.

8.26.2 Function Documentation

8.26.2.1 LRESULT CALLBACK WindowProc (HWND *hWnd*, UINT *uMsg*, WPARAM *wParam*, LPARAM *lParam*)

Definition at line 17 of file WindowProc.cpp.

8.27 branches/GameBase/GlApplication.h File Reference

```
#include <string>
#include <windows.h>
#include <list>
#include "GlWindow.h"
#include "KeyHandler.h"
#include "GlFrameworkObject.h"
#include "Point.h"
```

Classes

- class GlApplication

  Base Class for OpenGl Applications.

8.27.1 Detailed Description

Author:

Ryan Fleming <rfleming71@gmail.com>

Version:

1.0

Base Class for OpenGl Applications Class handles window setup and message handling

Definition in file GlApplication.h.

8.28 branches/GameBase/GlFrameworkObject.h File Reference

Classes

- class GlFrameworkObject

  Interface for objects to interact with GlApplication.

8.28.1 Detailed Description

Author:

Ryan Fleming <rfleming71@gmail.com>

Version:

1.0

Base class for the Framework Objects

Definition in file GlFrameworkObject.h.

8.29 branches/GameBase/GlWindow.cpp File Reference

```
#include "GlWindow.h"
#include <windows.h>
#include <string.h>
#include <gl\gl.h>
#include <gl\glu.h>
```

8.30 branches/GameBase/GlWindow.h File Reference

```
#include <string>
#include <windows.h>
```

Classes

- class GlWindow

  Window in Windows.

8.30.1 Detailed Description

Author:

Ryan Fleming <rfleming71@gmail.com>

Version:

1.0

Class to handle window create and management

Definition in file GlWindow.h.

8.31 branches/GameBase/GraphicLoaders.h File Reference

```
#include <string>
```

Namespaces

- namespace GraphicLoaders

  Loads Graphics.

Typedefs

- typedef unsigned int GraphicLoaders::TextureIdentifier

Functions

- bool GraphicLoaders::LoadNewBitmap (const std::wstring &fileName, TextureIdentifier &textureId)
- bool GraphicLoaders::LoadTga (const std::wstring &fileName, TextureIdentifier &textureId)
- bool GraphicLoaders::LoadTga (const std::string &fileName, TextureIdentifier &textureId)

8.31.1 Detailed Description

Author:

Ryan Fleming <rfleming71@gmail.com>

Version:

1.0

Namespace to load different graphic formats into openGL textures

Definition in file GraphicLoaders.h.

8.32 branches/GameBase/KeyHandler.cpp File Reference

```
#include "KeyHandler.h"
#include <cstring>
```

8.33 branches/GameBase/KeyHandler.h File Reference

Classes

- class KeyHandler

  Manages key presses.

Defines

- #define MAX_KEYS 256

8.33.1 Detailed Description

Author:

Ryan Fleming <rfleming71@gmail.com>

Version:

1.0

Class manage which keys are currently being pressed

Definition in file KeyHandler.h.

8.33.2 Define Documentation

8.33.2.1 #define MAX_KEYS 256

Maximum number of keys to handle

Definition at line 16 of file KeyHandler.h.

8.34 branches/GameBase/LevelEndObject.cpp File Reference

```
#include "LevelEndObject.h"
#include "GameDude.h"
#include "ScoreManager.h"
#include "GameEnums.h"
#include "AudioManager.h"
#include <windows.h>
#include <gl\gl.h>
```

8.35 branches/GameBase/LevelEndObject.h File Reference

```
#include "GamePiece.h"
```

Classes

- class LevelEndObject

  Object used to track if a level has ended.

8.35.1 Detailed Description

Author:

Ryan Fleming <rfleming71@gmail.com>

Version:

1.0

Object used to check the level status. This object will return true when the level is over

Definition in file LevelEndObject.h.

8.36 branches/GameBase/LevelObject.cpp File Reference

```
#include "LevelObject.h"
#include "GameDude.h"
#include "GameLoader.h"
#include "ScoreManager.h"
#include "Converter.h"
#include "PowerObject.h"
#include "AudioManager.h"
#include <windows.h>
#include <gl\gl.h>
```

Defines

- #define RIGHT_MOVE_DISTANCE 0.035
- #define LEFT_MOVE_DISTANCE -RIGHT_MOVE_DISTANCE
- #define CLIP_DISTANCE 5.0

8.36.1 Define Documentation

8.36.1.1 #define CLIP_DISTANCE 5.0

Definition at line 14 of file LevelObject.cpp.

8.36.1.2 #define LEFT_MOVE_DISTANCE -RIGHT_MOVE_DISTANCE

Definition at line 13 of file LevelObject.cpp.

8.36.1.3 #define RIGHT_MOVE_DISTANCE 0.035

Definition at line 12 of file LevelObject.cpp.

8.37 branches/GameBase/LevelObject.h File Reference

```
#include <string>
#include <list>
#include "AIObject.h"
#include "GamePiece.h"
#include "BackGroundManager.h"
#include "GameDude.h"
#include "LevelEndObject.h"
#include "PowerObject.h"
```

Classes

- class LevelObject

  Level in the game.

8.37.1 Detailed Description

Author:

Ryan Fleming <rfleming71@gmail.com>

Version:

1.0

Represents a level within the game. Handles all AI objects, Special Powers and block collisions

Definition in file LevelObject.h.

8.38 branches/GameBase/main.cpp File Reference

```
#include <windows.h>
#include "GameBase.h"
```

Functions

- int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE prevInstance, LPSTR lpCmdLine, int nCmdShow)

8.38.1 Function Documentation

8.38.1.1 int WINAPI WinMain (HINSTANCE *hInstance*, HINSTANCE *prevInstance*, LPSTR *lpCmdLine*, int *nCmdShow*)

Definition at line 20 of file main.cpp.

8.39 branches/GameBase/Menu.cpp File Reference

```
#include "Menu.h"
#include <windows.h>
#include <gl\gl.h>
#include "MenuItem.h"
#include "Point.h"
#include <list>
```

Defines

- #define INVALID_ID -1

8.39.1 Define Documentation

8.39.1.1 #define INVALID_ID -1

Definition at line 9 of file Menu.cpp.

8.40 branches/GameBase/Menu.h File Reference

```
#include "GlFrameworkObject.h"
#include "MenuItem.h"
#include "GameStructs.h"
#include <list>
#include <string>
```

Classes

- class Menu

  Menu Class.

8.40.1 Detailed Description

Author:

Ryan Fleming <rfleming71@gmail.com>

Version:

1.0

General Menu Class

Definition in file Menu.h.

8.41 branches/GameBase/MenuItem.cpp File Reference

```
#include "MenuItem.h"
#include <windows.h>
#include <gl\gl.h>
```

8.42 branches/GameBase/MenuItem.h File Reference

```
#include <string>
#include "GameStructs.h"
#include "Point.h"
```

Classes

- class MenuItem

  Items that appear within a menu.

8.42.1 Detailed Description

Author:

Ryan Fleming <rfleming71@gmail.com>

Version:

1.0

Items that appear within a menu

Definition in file MenuItem.h.

8.43 branches/GameBase/OpenAL/AL/alut.h File Reference

```
#include <AL/al.h>
#include <AL/alc.h>
```

Defines

- #define AL_ALUT_H
- #define ALUT_API extern
- #define ALUT_APIENTRY
- #define ALUT_ATTRIBUTE_DEPRECATED
- #define ALUT_API_MAJOR_VERSION 1
- #define ALUT_API_MINOR_VERSION 1
- #define ALUT_ERROR_NO_ERROR 0
- #define ALUT_ERROR_OUT_OF_MEMORY 0x200
- #define ALUT_ERROR_INVALID_ENUM 0x201
- #define ALUT_ERROR_INVALID_VALUE 0x202
- #define ALUT_ERROR_INVALID_OPERATION 0x203
- #define ALUT_ERROR_NO_CURRENT_CONTEXT 0x204
- #define ALUT_ERROR_AL_ERROR_ON_ENTRY 0x205
- #define ALUT_ERROR_ALC_ERROR_ON_ENTRY 0x206
- #define ALUT_ERROR_OPEN_DEVICE 0x207
- #define ALUT_ERROR_CLOSE_DEVICE 0x208
- #define ALUT_ERROR_CREATE_CONTEXT 0x209
- #define ALUT_ERROR_MAKE_CONTEXT_CURRENT 0x20A
- #define ALUT_ERROR_DESTROY_CONTEXT 0x20B
- #define ALUT_ERROR_GEN_BUFFERS 0x20C
- #define ALUT_ERROR_BUFFER_DATA 0x20D
- #define ALUT_ERROR_IO_ERROR 0x20E
- #define ALUT_ERROR_UNSUPPORTED_FILE_TYPE 0x20F
- #define ALUT_ERROR_UNSUPPORTED_FILE_SUBTYPE 0x210
- #define ALUT_ERROR_CORRUPT_OR_TRUNCATED_DATA 0x211
- #define ALUT_WAVEFORM_SINE 0x100
- #define ALUT_WAVEFORM_SQUARE 0x101
- #define ALUT_WAVEFORM_SAWTOOTH 0x102
- #define ALUT_WAVEFORM_WHITENOISE 0x103
- #define ALUT_WAVEFORM_IMPULSE 0x104
- #define ALUT_LOADER_BUFFER 0x300
- #define ALUT_LOADER_MEMORY 0x301

Functions

- ALUT_API ALboolean ALUT_APIENTRY alutInit (int *argcp, char **argv)
- ALUT_API ALboolean ALUT_APIENTRY alutInitWithoutContext (int *argcp, char **argv)
- ALUT_API ALboolean ALUT_APIENTRY alutExit (void)
- ALUT_API ALenum ALUT_APIENTRY alutGetError (void)
- ALUT_API const char *ALUT_APIENTRY alutGetErrorString (ALenum error)
- ALUT_API ALuint ALUT_APIENTRY alutCreateBufferFromFile (const char *fileName)

- ALUT_API ALuint ALUT_APIENTRY alutCreateBufferFromFileImage (const ALvoid *data, ALsizei length)
- ALUT_API ALuint ALUT_APIENTRY alutCreateBufferHelloWorld (void)
- ALUT_API ALuint ALUT_APIENTRY alutCreateBufferWaveform (ALenum waveshape, ALfloat frequency, ALfloat phase, ALfloat duration)
- ALUT_API ALvoid *ALUT_APIENTRY alutLoadMemoryFromFile (const char *fileName, ALenum *format, ALsizei *size, ALfloat *frequency)
- ALUT_API ALvoid *ALUT_APIENTRY alutLoadMemoryFromFileImage (const ALvoid *data, ALsizei length, ALenum *format, ALsizei *size, ALfloat *frequency)
- ALUT_API ALvoid *ALUT_APIENTRY alutLoadMemoryHelloWorld (ALenum *format, ALsizei *size, ALfloat *frequency)
- ALUT_API ALvoid *ALUT_APIENTRY alutLoadMemoryWaveform (ALenum waveshape, ALfloat frequency, ALfloat phase, ALfloat duration, ALenum *format, ALsizei *size, ALfloat *freq)
- ALUT_API const char *ALUT_APIENTRY alutGetMIMETypes (ALenum loader)
- ALUT_API ALint ALUT_APIENTRY alutGetMajorVersion (void)
- ALUT_API ALint ALUT_APIENTRY alutGetMinorVersion (void)
- ALUT_API ALboolean ALUT_APIENTRY alutSleep (ALfloat duration)
- ALUT_API ALUT_ATTRIBUTE_DEPRECATED void ALUT_APIENTRY alutLoadWAVFile (ALbyte *fileName, ALenum *format, void **data, ALsizei *size, ALsizei *frequency, ALboolean *loop)
- ALUT_API ALUT_ATTRIBUTE_DEPRECATED void ALUT_APIENTRY alutLoadWAVMemory (ALbyte *buffer, ALenum *format, void **data, ALsizei *size, ALsizei *frequency, ALboolean *loop)
- ALUT_API ALUT_ATTRIBUTE_DEPRECATED void ALUT_APIENTRY alutUnloadWAV (ALenum format, ALvoid *data, ALsizei size, ALsizei frequency)

8.43.1 Define Documentation

8.43.1.1 #define AL_ALUT_H

Definition at line 2 of file alut.h.

8.43.1.2 #define ALUT_API extern

Definition at line 29 of file alut.h.

8.43.1.3 #define ALUT_API_MAJOR_VERSION 1

Definition at line 52 of file alut.h.

8.43.1.4 #define ALUT_API_MINOR_VERSION 1

Definition at line 53 of file alut.h.

8.43.1.5 #define ALUT_APIENTRY

Definition at line 36 of file alut.h.

8.43.1.6 #define ALUT_ATTRIBUTE_DEPRECATED

Definition at line 49 of file alut.h.

8.43.1.7 #define ALUT_ERROR_AL_ERROR_ON_ENTRY 0x205

Definition at line 61 of file alut.h.

8.43.1.8 #define ALUT_ERROR_ALC_ERROR_ON_ENTRY 0x206

Definition at line 62 of file alut.h.

8.43.1.9 #define ALUT_ERROR_BUFFER_DATA 0x20D

Definition at line 69 of file alut.h.

8.43.1.10 #define ALUT_ERROR_CLOSE_DEVICE 0x208

Definition at line 64 of file alut.h.

8.43.1.11 #define ALUT_ERROR_CORRUPT_OR_TRUNCATED_DATA 0x211

Definition at line 73 of file alut.h.

8.43.1.12 #define ALUT_ERROR_CREATE_CONTEXT 0x209

Definition at line 65 of file alut.h.

8.43.1.13 #define ALUT_ERROR_DESTROY_CONTEXT 0x20B

Definition at line 67 of file alut.h.

8.43.1.14 #define ALUT_ERROR_GEN_BUFFERS 0x20C

Definition at line 68 of file alut.h.

8.43.1.15 #define ALUT_ERROR_INVALID_ENUM 0x201

Definition at line 57 of file alut.h.

8.43.1.16 #define ALUT_ERROR_INVALID_OPERATION 0x203

Definition at line 59 of file alut.h.

8.43.1.17 #define ALUT_ERROR_INVALID_VALUE 0x202

Definition at line 58 of file alut.h.

8.43.1.18 #define ALUT_ERROR_IO_ERROR 0x20E

Definition at line 70 of file alut.h.

8.43.1.19 #define ALUT_ERROR_MAKE_CONTEXT_CURRENT 0x20A

Definition at line 66 of file alut.h.

8.43.1.20 #define ALUT_ERROR_NO_CURRENT_CONTEXT 0x204

Definition at line 60 of file alut.h.

8.43.1.21 #define ALUT_ERROR_NO_ERROR 0

Definition at line 55 of file alut.h.

8.43.1.22 #define ALUT_ERROR_OPEN_DEVICE 0x207

Definition at line 63 of file alut.h.

8.43.1.23 #define ALUT_ERROR_OUT_OF_MEMORY 0x200

Definition at line 56 of file alut.h.

8.43.1.24 #define ALUT_ERROR_UNSUPPORTED_FILE_SUBTYPE 0x210

Definition at line 72 of file alut.h.

8.43.1.25 #define ALUT_ERROR_UNSUPPORTED_FILE_TYPE 0x20F

Definition at line 71 of file alut.h.

8.43.1.26 #define ALUT_LOADER_BUFFER 0x300

Definition at line 81 of file alut.h.

8.43.1.27 #define ALUT_LOADER_MEMORY 0x301

Definition at line 82 of file alut.h.

8.43.1.28 #define ALUT_WAVEFORM_IMPULSE 0x104

Definition at line 79 of file alut.h.

8.43.1.29 #define ALUT_WAVEFORM_SAWTOOTH 0x102

Definition at line 77 of file alut.h.

8.43.1.30 #define ALUT_WAVEFORM_SINE 0x100

Definition at line 75 of file alut.h.

8.43.1.31 #define ALUT_WAVEFORM_SQUARE 0x101

Definition at line 76 of file alut.h.

8.43.1.32 #define ALUT_WAVEFORM_WHITENOISE 0x103

Definition at line 78 of file alut.h.

8.43.2 Function Documentation

8.43.2.1 ALUT_API ALuint ALUT_APIENTRY alutCreateBufferFromFile (const char * *fileName*)

8.43.2.2 ALUT_API ALuint ALUT_APIENTRY alutCreateBufferFromFileImage (const ALvoid * *data*, ALsizei *length*)

8.43.2.3 ALUT_API ALuint ALUT_APIENTRY alutCreateBufferHelloWorld (void)

8.43.2.4 ALUT_API ALuint ALUT_APIENTRY alutCreateBufferWaveform (ALenum *waveshape*, ALfloat *frequency*, ALfloat *phase*, ALfloat *duration*)

8.43.2.5 ALUT_API ALboolean ALUT_APIENTRY alutExit (void)

8.43.2.6 ALUT_API ALenum ALUT_APIENTRY alutGetError (void)

8.43.2.7 ALUT_API const char* ALUT_APIENTRY alutGetErrorString (ALenum *error*)

8.43.2.8 ALUT_API ALint ALUT_APIENTRY alutGetMajorVersion (void)

8.43.2.9 ALUT_API const char* ALUT_APIENTRY alutGetMIMETypes (ALenum *loader*)

8.43.2.10 ALUT_API ALint ALUT_APIENTRY alutGetMinorVersion (void)

8.43.2.11 ALUT_API ALboolean ALUT_APIENTRY alutInit (int * *argcp*, char ** *argv*)

8.43.2.12 ALUT_API ALboolean ALUT_APIENTRY alutInitWithoutContext (int * *argcp*, char ** *argv*)

8.43.2.13 ALUT_API ALvoid* ALUT_APIENTRY alutLoadMemoryFromFile (const char * *fileName*, ALenum * *format*, ALsizei * *size*, ALfloat * *frequency*)

8.43.2.14 ALUT_API ALvoid* ALUT_APIENTRY alutLoadMemoryFromFileImage (const ALvoid * *data*, ALsizei *length*, ALenum * *format*, ALsizei * *size*, ALfloat * *frequency*)

8.43.2.15 ALUT_API ALvoid* ALUT_APIENTRY alutLoadMemoryHelloWorld (ALenum * *format*, ALsizei * *size*, ALfloat * *frequency*)

8.43.2.16 ALUT_API ALvoid* ALUT_APIENTRY alutLoadMemoryWaveform (ALenum *waveshape*, ALfloat *frequency*, ALfloat *phase*, ALfloat *duration*, ALenum * *format*, ALsizei * *size*, ALfloat * *freq*)

8.43.2.17 ALUT_API ALUT_ATTRIBUTE_DEPRECATED void ALUT_APIENTRY alutLoadWAVFile (ALbyte * *fileName*, ALenum * *format*, void ** *data*, ALsizei * *size*, ALsizei * *frequency*, ALboolean * *loop*)

8.43.2.18 ALUT_API ALUT_ATTRIBUTE_DEPRECATED void ALUT_APIENTRY alutLoadWAVMemory (ALbyte * *buffer*, ALenum * *format*, void ** *data*, ALsizei * *size*, ALsizei * *frequency*, ALboolean * *loop*)

8.43.2.19 ALUT_API ALboolean ALUT_APIENTRY alutSleep (ALfloat *duration*)

8.43.2.20 ALUT_API ALUT_ATTRIBUTE_DEPRECATED void ALUT_APIENTRY alutUnloadWAV (ALenum *format*, ALvoid * *data*, ALsizei *size*, ALsizei *frequency*)

8.44 branches/GameBase/Point.h File Reference

Classes

- struct Point

  A single point Represents a single point in openGL space.

8.44.1 Detailed Description

Author:

Ryan Fleming <rfleming71@gmail.com>

Version:

1.0

This file contains the struct that represents a point openGL space

Todo

Merge this file into GameStructs

Definition in file Point.h.

8.45 branches/GameBase/PowerObject.cpp File Reference

```
#include "PowerObject.h"
#include <windows.h>
#include <gl\gl.h>
#include "GamePiece.h"
#include "GameStructs.h"
```

Defines

- #define MOVE_SPEED 0.05
- #define LEFT false
- #define RIGHT true

8.45.1 Define Documentation

8.45.1.1 #define LEFT false

Definition at line 9 of file PowerObject.cpp.

8.45.1.2 #define MOVE_SPEED 0.05

Definition at line 8 of file PowerObject.cpp.

8.45.1.3 #define RIGHT true

Definition at line 10 of file PowerObject.cpp.

8.46 branches/GameBase/PowerObject.h File Reference

```
#include "GamePiece.h"
#include "GameStructs.h"
```

Classes

- class PowerObject

  The special power.

8.46.1 Detailed Description

Author:

Ryan Fleming <rfleming71@gmail.com>

Version:

1.0

The object that game dude fires when using the special powers

Definition in file PowerObject.h.

8.47 branches/GameBase/PowerUpBlock.cpp File Reference

```
#include "PowerUpBlock.h"
#include "AudioManager.h"
```

8.48 branches/GameBase/PowerUpBlock.h File Reference

```
#include "GamePiece.h"
#include "PowerUpItem.h"
```

Classes

- class PowerUpBlock

  Block that creates a PowerUpItem.

8.48.1 Detailed Description

Author:

Ryan Fleming <rfleming71@gmail.com>

Version:

1.0

Special Power Block. Upon hitting, it triggers the PowerUpItem

Definition in file PowerUpBlock.h.

8.49 branches/GameBase/PowerUpItem.cpp File Reference

```
#include "PowerUpItem.h"
#include "GameStructs.h"
#include "GameDude.h"
#include "GameEnums.h"
#include "AudioManager.h"
```

Defines

- #define VERTICAL_RATE 0.01
- #define MOVE_RATE 0.01

8.49.1 Define Documentation

8.49.1.1 #define MOVE_RATE 0.01

Definition at line 8 of file PowerUpItem.cpp.

8.49.1.2 #define VERTICAL_RATE 0.01

Definition at line 7 of file PowerUpItem.cpp.

8.50 branches/GameBase/PowerUpItem.h File Reference

```
#include "AIObject.h"
#include "AIType1.h"
#include "GameEnums.h"
```

Classes

- class PowerUpItem

  Power Up Item.

8.50.1 Detailed Description

Author:

Ryan Fleming <rfleming71@gmail.com>

Version:

1.0

Power up item, upon touching the game dude status is increased by 1

Definition in file PowerUpItem.h.

8.51 branches/GameBase/RgbaColor.h File Reference

Classes

- struct RgbaColor

  OpenGL Color.

8.51.1 Detailed Description

Author:

Ryan Fleming <rfleming71@gmail.com>

Version:

1.0

File contains the struct to represent an openGL color

Todo

Merge this file into GameStructs.h

Definition in file RgbaColor.h.

8.52 branches/GameBase/ScoreManager.cpp File Reference

```
#include "ScoreManager.h"
```

8.53 branches/GameBase/ScoreManager.h File Reference

```
#include "GameEnums.h"
```

Classes

- class ScoreManager

  Tracks the players score.

8.53.1 Detailed Description

Author:

Ryan Fleming <rfleming71@gmail.com>

Version:

1.0

Class to keep trace of the current playes score. Singleton Object

Definition in file ScoreManager.h.

8.54 branches/GameBase/UtilFunctions.cpp File Reference

```
#include "UtilFunctions.h"
#include <string>
#include <vector>
```

8.55 branches/GameBase/UtilFunctions.h File Reference

```
#include <string>
#include <vector>
```

Namespaces

- namespace UtilFunctions

  Utility Functions Contains several useful functions that fail to fit elsewhere.

Typedefs

- typedef std::vector< std::string * > UtilFunctions::StringTokensType
- typedef StringTokensType * UtilFunctions::StringTokens

Functions

- std::string UtilFunctions::TrimWhiteSpace (const std::string &startString)
- UtilFunctions::StringTokens UtilFunctions::StringTokenizer (const std::string &baseString, const std::string &delimiters)
- UtilFunctions::StringTokens UtilFunctions::StringTokenizer2 (const std::string &baseString, const std::string &delimiters)
- void UtilFunctions::DestroyStringTokens (UtilFunctions::StringTokens tokens)

8.55.1 Detailed Description

Common functions

Author:

Ryan Fleming <rfleming71@kb71.com>

Since:

0.2

Author:

Ryan Fleming <rfleming71@gmail.com>

Version:

2.0

This file contains general usage functions

Definition in file UtilFunctions.h.

8.56 branches/GameBase/WindowProc.cpp File Reference

```
#include "GlApplication.h"
```

Functions

- LRESULT CALLBACK WindowProc (HWND hWnd, UINT uMsg, WPARAM wParam, LPARAM lParam)

8.56.1 Function Documentation

8.56.1.1 LRESULT CALLBACK WindowProc (HWND *hWnd*, UINT *uMsg*, WPARAM *wParam*, LPARAM *lParam*)

Definition at line 17 of file WindowProc.cpp.

8.57 branches/GameBase/WorldObject.cpp File Reference

```
#include "WorldObject.h"
#include "GameDude.h"
```

8.58 branches/GameBase/WorldObject.h File Reference

```
#include <list>
#include <string>
#include "GlFrameworkObject.h"
#include "LevelObject.h"
#include "GameDude.h"
```

Classes

- class WorldObject

  World in the game.

8.58.1 Detailed Description

Author:

Ryan Fleming <rfleming71@gmail.com>

Version:

1.0

Represents a current world within the game. Handles level loading and management

Definition in file WorldObject.h.

Index