

Phantom

Appendix A

Generated by Doxygen 1.8.3.1

Sun Feb 24 2013 12:27:44



# Contents

<b>1</b>	<b>Hierarchical Index</b>	<b>1</b>
1.1	Class Hierarchy	1
<b>2</b>	<b>Class Index</b>	<b>3</b>
2.1	Class List	3
<b>3</b>	<b>Class Documentation</b>	<b>5</b>
3.1	phantom::AbstractMessage Class Reference	5
3.2	phantom::Arc Class Reference	5
3.3	phantom::AudioEngine Class Reference	6
3.3.1	Member Function Documentation	6
3.3.1.1	createSound	6
3.3.1.2	destroySound	6
3.3.1.3	playMusic	7
3.3.1.4	playSound	7
3.3.1.5	setPosition	7
3.3.1.6	stopMusic	7
3.3.1.7	stopSound	7
3.4	phantom::tree::BinaryNode Class Reference	7
3.5	phantom::tree::BinaryTree Class Reference	8
3.6	phantom::Box3 Class Reference	8
3.7	phantom::Button Class Reference	8
3.7.1	Member Function Documentation	9
3.7.1.1	onClick	9
3.7.1.2	paint	9
3.7.1.3	setText	9
3.7.1.4	text	9
3.7.1.5	update	10
3.8	phantom::Camera Class Reference	10
3.8.1	Member Function Documentation	11
3.8.1.1	getCameraID	11
3.8.1.2	getRotation	11

3.8.1.3	<a href="#">getScreenSize</a>	11
3.8.1.4	<a href="#">getViewport</a>	11
3.8.1.5	<a href="#">getWorldCoordinates</a>	11
3.8.1.6	<a href="#">isActive</a>	11
3.8.1.7	<a href="#">setParams</a>	11
3.8.1.8	<a href="#">setRotation</a>	11
3.8.1.9	<a href="#">setScreenSize</a>	12
3.8.1.10	<a href="#">setViewport</a>	12
3.9	<a href="#">phantom::FreeTypeFont::char_info_t Struct Reference</a>	12
3.10	<a href="#">phantom::CollisionData Struct Reference</a>	12
3.11	<a href="#">phantom::Color Struct Reference</a>	13
3.12	<a href="#">phantom::Composite Class Reference</a>	13
3.13	<a href="#">phantom::Console Class Reference</a>	15
3.13.1	<a href="#">Member Function Documentation</a>	15
3.13.1.1	<a href="#">addLog</a>	15
3.13.1.2	<a href="#">log</a>	16
3.13.1.3	<a href="#">log</a>	16
3.13.1.4	<a href="#">log</a>	16
3.13.1.5	<a href="#">mapCommand</a>	16
3.13.1.6	<a href="#">update</a>	16
3.14	<a href="#">phantom::Driver Class Reference</a>	16
3.14.1	<a href="#">Member Function Documentation</a>	17
3.14.1.1	<a href="#">createCamera</a>	17
3.14.1.2	<a href="#">disableCamera</a>	17
3.14.1.3	<a href="#">enableCamera</a>	17
3.14.1.4	<a href="#">getActiveCameras</a>	18
3.14.1.5	<a href="#">getAudio</a>	18
3.14.1.6	<a href="#">getAudioEngine</a>	18
3.14.1.7	<a href="#">getFontLibrary</a>	18
3.14.1.8	<a href="#">getInput</a>	18
3.14.1.9	<a href="#">getRenderer</a>	18
3.14.1.10	<a href="#">onRender</a>	18
3.14.1.11	<a href="#">onUpdate</a>	18
3.14.1.12	<a href="#">setWindowTitle</a>	19
3.15	<a href="#">phantom::Entity Class Reference</a>	19
3.15.1	<a href="#">Member Function Documentation</a>	19
3.15.1.1	<a href="#">addComponent</a>	19
3.15.1.2	<a href="#">directionTo</a>	20
3.15.1.3	<a href="#">distanceTo</a>	20
3.15.1.4	<a href="#">distanceToSq</a>	20

3.16	<a href="#">phantom::EntityLayer Class Reference</a>	20
3.17	<a href="#">phantom::FreeTypeFont::font_info_t Struct Reference</a>	21
3.18	<a href="#">phantom::FreeTypeFont Class Reference</a>	21
3.19	<a href="#">phantom::FreeTypeLibrary Class Reference</a>	22
3.19.1	<a href="#">Member Function Documentation</a>	22
3.19.1.1	<a href="#">getFont</a>	22
3.20	<a href="#">phantom::GameState Class Reference</a>	22
3.21	<a href="#">phantom::GLCamera Class Reference</a>	23
3.21.1	<a href="#">Member Function Documentation</a>	23
3.21.1.1	<a href="#">setParams</a>	23
3.22	<a href="#">phantom::GLDriver Class Reference</a>	23
3.22.1	<a href="#">Member Function Documentation</a>	24
3.22.1.1	<a href="#">createCamera</a>	24
3.22.1.2	<a href="#">setWindowTitle</a>	24
3.23	<a href="#">phantom::GLRenderer Class Reference</a>	24
3.23.1	<a href="#">Member Function Documentation</a>	25
3.23.1.1	<a href="#">addTexture</a>	25
3.23.1.2	<a href="#">buildShape</a>	25
3.23.1.3	<a href="#">destroyShape</a>	25
3.23.1.4	<a href="#">removeTexture</a>	25
3.23.1.5	<a href="#">renderLoop</a>	25
3.24	<a href="#">phantom::GLUTInput Class Reference</a>	26
3.25	<a href="#">phantom::Graphics Class Reference</a>	26
3.25.1	<a href="#">Member Function Documentation</a>	27
3.25.1.1	<a href="#">arc</a>	27
3.25.1.2	<a href="#">beginPath</a>	27
3.25.1.3	<a href="#">clear</a>	27
3.25.1.4	<a href="#">fill</a>	27
3.25.1.5	<a href="#">getRotation</a>	27
3.25.1.6	<a href="#">image</a>	28
3.25.1.7	<a href="#">line</a>	28
3.25.1.8	<a href="#">line</a>	28
3.25.1.9	<a href="#">line</a>	28
3.25.1.10	<a href="#">lineTo</a>	29
3.25.1.11	<a href="#">moveTo</a>	29
3.25.1.12	<a href="#">rect</a>	29
3.25.1.13	<a href="#">rect</a>	29
3.25.1.14	<a href="#">rotate</a>	30
3.25.1.15	<a href="#">setFillStyle</a>	30
3.25.1.16	<a href="#">setLineStyle</a>	30

3.25.1.17 stroke . . . . .	30
3.25.1.18 text . . . . .	31
3.26 phantom::IHandleMessage Class Reference . . . . .	31
3.27 phantom::Image Class Reference . . . . .	32
3.27.1 Member Function Documentation . . . . .	33
3.27.1.1 getImage . . . . .	33
3.28 phantom::ImageCache Class Reference . . . . .	33
3.28.1 Member Function Documentation . . . . .	33
3.28.1.1 getFromCache . . . . .	33
3.28.1.2 insertIntoCache . . . . .	33
3.28.1.3 isCached . . . . .	34
3.28.1.4 removeFromCache . . . . .	34
3.28.1.5 setRenderer . . . . .	34
3.29 phantom::ImageCacheItem Class Reference . . . . .	34
3.30 phantom::ImageLoader Class Reference . . . . .	34
3.30.1 Member Function Documentation . . . . .	35
3.30.1.1 createPNG . . . . .	35
3.31 phantom::InertiaMover Class Reference . . . . .	35
3.32 phantom::Input Class Reference . . . . .	36
3.32.1 Member Function Documentation . . . . .	36
3.32.1.1 getKeyboardState . . . . .	36
3.32.1.2 getMouseState . . . . .	36
3.33 phantom::InputField Class Reference . . . . .	36
3.33.1 Member Function Documentation . . . . .	37
3.33.1.1 keyboard . . . . .	37
3.33.1.2 onClick . . . . .	37
3.33.1.3 onUnClicked . . . . .	37
3.33.1.4 paint . . . . .	38
3.33.1.5 paintText . . . . .	38
3.33.1.6 text . . . . .	38
3.33.1.7 text . . . . .	38
3.33.1.8 update . . . . .	38
3.34 phantom::IUpdateable Class Reference . . . . .	38
3.35 phantom::KeyboardListener Class Reference . . . . .	39
3.35.1 Member Function Documentation . . . . .	40
3.35.1.1 isLocked . . . . .	40
3.35.1.2 lock . . . . .	40
3.35.1.3 unlock . . . . .	40
3.36 phantom::KeyboardState Class Reference . . . . .	40
3.36.1 Member Function Documentation . . . . .	40

3.36.1.1	changes	40
3.36.1.2	changesUp	41
3.36.1.3	getBuffer	41
3.36.1.4	handleEvent	41
3.36.1.5	isKeyDown	41
3.36.1.6	isKeyUp	41
3.37	phantom::Layer Class Reference	42
3.38	phantom::Line Class Reference	42
3.39	phantom::Line2 Class Reference	43
3.40	phantom::Vector3::MapLessComparefunctor Struct Reference	43
3.41	phantom::Message< T > Class Template Reference	43
3.42	phantom::MouseState Class Reference	44
3.42.1	Member Function Documentation	44
3.42.1.1	getPosition	44
3.42.1.2	handleEvent	44
3.42.1.3	handleEvent	44
3.42.1.4	isButtonDown	44
3.42.1.5	isButtonUp	45
3.43	phantom::Mover Class Reference	45
3.44	phantom::internal::NoPayload Struct Reference	46
3.45	phantom::NullDriver Class Reference	46
3.45.1	Member Function Documentation	46
3.45.1.1	createCamera	46
3.45.1.2	onRender	46
3.45.1.3	onUpdate	46
3.46	phantom::OpenALEngine Class Reference	47
3.46.1	Member Function Documentation	47
3.46.1.1	createSound	47
3.46.1.2	destroySound	47
3.46.1.3	playMusic	48
3.46.1.4	playSound	48
3.46.1.5	setPosition	48
3.46.1.6	stopMusic	48
3.46.1.7	stopSound	48
3.47	phantom::Particle Struct Reference	49
3.48	phantom::Particles Class Reference	49
3.48.1	Constructor & Destructor Documentation	50
3.48.1.1	Particles	50
3.48.2	Member Function Documentation	50
3.48.2.1	getParticles	50

3.49	phantom::PhantomException Class Reference	50
3.49.1	Member Function Documentation	51
3.49.1.1	what	51
3.50	phantom::PhantomGame Class Reference	51
3.50.1	Member Function Documentation	52
3.50.1.1	exit	52
3.50.1.2	getConsole	52
3.50.1.3	getDriver	52
3.50.1.4	getGameStates	52
3.50.1.5	getScreenSize	53
3.50.1.6	getViewPort	53
3.50.1.7	getWorldSize	53
3.50.1.8	handleMessage	53
3.50.1.9	parseConfigurationFile	53
3.50.1.10	popGameState	53
3.50.1.11	pushGameState	54
3.50.1.12	setDriver	54
3.50.1.13	setWorldSize	54
3.50.1.14	start	54
3.50.1.15	update	54
3.51	phantom::PhantomTime Class Reference	54
3.51.1	Member Function Documentation	55
3.51.1.1	getElapsed	55
3.51.1.2	getTime	55
3.51.1.3	getTotalGameTime	55
3.52	phantom::Polygon Class Reference	55
3.52.1	Member Function Documentation	56
3.52.1.1	addPoint	56
3.53	phantom::Polygon2 Class Reference	56
3.54	phantom::Projection Class Reference	56
3.55	phantom::Pulse Struct Reference	56
3.56	phantom::Rectangle Class Reference	57
3.57	phantom::Renderer Class Reference	57
3.57.1	Member Function Documentation	58
3.57.1.1	addTexture	58
3.57.1.2	buildShape	58
3.57.1.3	destroyShape	58
3.57.1.4	removeTexture	58
3.57.1.5	renderLoop	59
3.58	phantom::Shape Class Reference	59



3.58.1	Member Function Documentation	60
3.58.1.1	addVertex	60
3.58.1.2	buildShape	60
3.58.1.3	destroyShape	60
3.58.1.4	getBounds	60
3.58.1.5	getFillColor	60
3.58.1.6	getLineColor	61
3.58.1.7	getShapecount	61
3.58.1.8	hasFillColor	61
3.58.1.9	hasLineColor	61
3.58.1.10	setFillColor	61
3.58.1.11	setLineColor	61
3.59	phantom::SoundData Class Reference	62
3.60	phantom::SoundLoader Class Reference	62
3.60.1	Member Function Documentation	62
3.60.1.1	loadVorbis	62
3.61	phantom::Sounds Class Reference	62
3.61.1	Member Function Documentation	63
3.61.1.1	playMusic	63
3.61.1.2	playSound	63
3.61.1.3	stopMusic	63
3.61.1.4	stopSound	63
3.62	phantom::TexCoord Class Reference	64
3.63	phantom::Text Class Reference	64
3.63.1	Member Function Documentation	64
3.63.1.1	genVertices	64
3.64	phantom::Timer Class Reference	65
3.65	phantom::Util Class Reference	65
3.65.1	Member Function Documentation	65
3.65.1.1	getTime	65
3.65.1.2	readfile	65
3.66	phantom::Vector3 Class Reference	66
3.67	phantom::Vertice Class Reference	67



# Chapter 1

## Hierarchical Index

### 1.1 Class Hierarchy

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

phantom::AbstractMessage . . . . .	5
phantom::Message< T > . . . . .	43
phantom::AudioEngine . . . . .	6
phantom::OpenALEngine . . . . .	47
phantom::tree::BinaryNode . . . . .	7
phantom::tree::BinaryTree . . . . .	8
phantom::Box3 . . . . .	8
phantom::FreeTypeFont::char_info_t . . . . .	12
phantom::CollisionData . . . . .	12
phantom::Color . . . . .	13
phantom::Driver . . . . .	16
phantom::GLDriver . . . . .	23
phantom::NullDriver . . . . .	46
exception	
phantom::PhantomException . . . . .	50
phantom::FreeTypeFont::font_info_t . . . . .	21
phantom::FreeTypeFont . . . . .	21
phantom::FreeTypeLibrary . . . . .	22
phantom::Graphics . . . . .	26
phantom::IHandleMessage . . . . .	31
phantom::Composite . . . . .	13
phantom::Button . . . . .	8
phantom::Camera . . . . .	10
phantom::GLCamera . . . . .	23
phantom::Console . . . . .	15
phantom::Entity . . . . .	19
phantom::GameState . . . . .	22
phantom::InertiaMover . . . . .	35
phantom::InputField . . . . .	36
phantom::Layer . . . . .	42
phantom::EntityLayer . . . . .	20
phantom::Mover . . . . .	45
phantom::Particles . . . . .	49
phantom::PhantomGame . . . . .	51
phantom::Sounds . . . . .	62
phantom::ImageCache . . . . .	33

phantom::ImageCacheItem . . . . .	34
phantom::ImageLoader . . . . .	34
phantom::Input . . . . .	36
phantom::GLUTInput . . . . .	26
phantom::IUpdateable . . . . .	38
phantom::Composite . . . . .	13
phantom::KeyboardListener . . . . .	39
phantom::KeyboardState . . . . .	40
phantom::Line2 . . . . .	43
phantom::Vector3::MapLessComparefunctor . . . . .	43
phantom::MouseState . . . . .	44
phantom::internal::NoPayload . . . . .	46
phantom::Particle . . . . .	49
phantom::PhantomTime . . . . .	54
phantom::Polygon2 . . . . .	56
phantom::Projection . . . . .	56
phantom::Pulse . . . . .	56
phantom::Renderer . . . . .	57
phantom::GLRenderer . . . . .	24
phantom::Shape . . . . .	59
phantom::Image . . . . .	32
phantom::Line . . . . .	42
phantom::Arc . . . . .	5
phantom::Polygon . . . . .	55
phantom::Rectangle . . . . .	57
phantom::Text . . . . .	64
phantom::SoundData . . . . .	62
phantom::SoundLoader . . . . .	62
phantom::TexCoord . . . . .	64
phantom::Timer . . . . .	65
phantom::Util . . . . .	65
phantom::Vector3 . . . . .	66
phantom::Vertice . . . . .	67

## Chapter 2

# Class Index

### 2.1 Class List

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

<a href="#">phantom::AbstractMessage</a>	5
<a href="#">phantom::Arc</a>	5
<a href="#">phantom::AudioEngine</a>	6
<a href="#">phantom::tree::BinaryNode</a>	7
<a href="#">phantom::tree::BinaryTree</a>	8
<a href="#">phantom::Box3</a>	8
<a href="#">phantom::Button</a>	8
<a href="#">phantom::Camera</a>	10
<a href="#">phantom::FreeTypeFont::char_info_t</a>	12
<a href="#">phantom::CollisionData</a>	12
<a href="#">phantom::Color</a>	13
<a href="#">phantom::Composite</a>	13
<a href="#">phantom::Console</a>	15
<a href="#">phantom::Driver</a>	16
<a href="#">phantom::Entity</a>	19
<a href="#">phantom::EntityLayer</a>	20
<a href="#">phantom::FreeTypeFont::font_info_t</a>	21
<a href="#">phantom::FreeTypeFont</a>	21
<a href="#">phantom::FreeTypeLibrary</a>	22
<a href="#">phantom::GameState</a>	22
<a href="#">phantom::GLCamera</a>	23
<a href="#">phantom::GLDriver</a>	23
<a href="#">phantom::GLRenderer</a>	24
<a href="#">phantom::GLUTInput</a>	26
<a href="#">phantom::Graphics</a>	26
<a href="#">phantom::IHandleMessage</a>	31
<a href="#">phantom::Image</a>	32
<a href="#">phantom::ImageCache</a>	33
<a href="#">phantom::ImageCacheItem</a>	34
<a href="#">phantom::ImageLoader</a>	34
<a href="#">phantom::InertiaMover</a>	35
<a href="#">phantom::Input</a>	36
<a href="#">phantom::InputField</a>	36
<a href="#">phantom::IUpdateable</a>	38
<a href="#">phantom::KeyboardListener</a>	39
<a href="#">phantom::KeyboardState</a>	40
<a href="#">phantom::Layer</a>	42
<a href="#">phantom::Line</a>	42

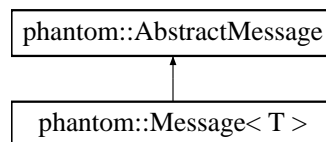
<a href="#">phantom::Line2</a>	43
<a href="#">phantom::Vector3::MapLessComparefunctor</a>	43
<a href="#">phantom::Message&lt; T &gt;</a>	43
<a href="#">phantom::MouseState</a>	44
<a href="#">phantom::Mover</a>	45
<a href="#">phantom::internal::NoPayload</a>	46
<a href="#">phantom::NullDriver</a>	46
<a href="#">phantom::OpenALEngine</a>	47
<a href="#">phantom::Particle</a>	49
<a href="#">phantom::Particles</a>	49
<a href="#">phantom::PhantomException</a>	50
<a href="#">phantom::PhantomGame</a>	51
<a href="#">phantom::PhantomTime</a>	54
<a href="#">phantom::Polygon</a>	55
<a href="#">phantom::Polygon2</a>	56
<a href="#">phantom::Projection</a>	56
<a href="#">phantom::Pulse</a>	56
<a href="#">phantom::Rectangle</a>	57
<a href="#">phantom::Renderer</a>	57
<a href="#">phantom::Shape</a>	59
<a href="#">phantom::SoundData</a>	62
<a href="#">phantom::SoundLoader</a>	62
<a href="#">phantom::Sounds</a>	62
<a href="#">phantom::TexCoord</a>	64
<a href="#">phantom::Text</a>	64
<a href="#">phantom::Timer</a>	65
<a href="#">phantom::Util</a>	65
<a href="#">phantom::Vector3</a>	66
<a href="#">phantom::Vertex</a>	67

## Chapter 3

# Class Documentation

### 3.1 phantom::AbstractMessage Class Reference

Inheritance diagram for phantom::AbstractMessage:



#### Public Member Functions

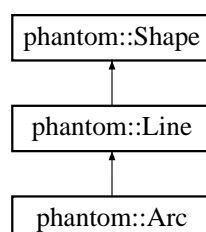
- **AbstractMessage** (std::string type)
- template<class T3 >  
T3 **getPayload** ()
- bool **isType** (const std::string &otherType)
- const std::string & **getType** ()

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

- C:/Users/Sander/Documents/Projects/PCCS/phantom/src/messaging/AbstractMessage.h

### 3.2 phantom::Arc Class Reference

Inheritance diagram for phantom::Arc:



#### Public Member Functions

- **Arc** (float x, float y, float radius, float start, float end)

## Public Attributes

- float **radius**
- float **start**
- float **end**

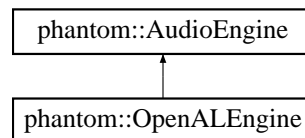
## Additional Inherited Members

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

- C:/Users/Sander/Documents/Projects/PCCS/phantom/src/graphics/shapes/Arc.h
- C:/Users/Sander/Documents/Projects/PCCS/phantom/src/graphics/shapes/Arc.cpp

## 3.3 phantom::AudioEngine Class Reference

Inheritance diagram for phantom::AudioEngine:



## Public Member Functions

- **AudioEngine** ([PhantomGame](#) \*game)
- virtual void [createSound](#) ([SoundData](#) \*data)=0
- virtual void [destroySound](#) ([SoundData](#) \*data)=0
- virtual unsigned int [playSound](#) ([SoundData](#) \*data, const [Vector3](#) &position)=0
- virtual void [playMusic](#) ([SoundData](#) \*data)=0
- virtual void [stopSound](#) (unsigned int id)=0
- virtual void [stopMusic](#) ([SoundData](#) \*data)=0
- virtual void [setPosition](#) (const [Vector3](#) &position)=0

## Protected Attributes

- [PhantomGame](#) \* **\_game**

### 3.3.1 Member Function Documentation

**3.3.1.1** virtual void phantom::AudioEngine::createSound ( [SoundData](#) \* *data* ) [pure virtual]

This function is called when the sound is not in the library yet.

Implemented in [phantom::OpenALEngine](#).

**3.3.1.2** virtual void phantom::AudioEngine::destroySound ( [SoundData](#) \* *data* ) [pure virtual]

This function is called once the sound is no longer needed.

Implemented in [phantom::OpenALEngine](#).



3.3.1.3 virtual void phantom::AudioEngine::playMusic ( **SoundData** \* *data* ) [pure virtual]

This function is called when you want to play an music file.

Implemented in [phantom::OpenALEngine](#).

3.3.1.4 virtual unsigned int phantom::AudioEngine::playSound ( **SoundData** \* *data*, const **Vector3** & *position* ) [pure virtual]

This function is called when you want to play an sound file.

#### Parameters

<i>position</i>	The position unrelative to the listener.
-----------------	--

Implemented in [phantom::OpenALEngine](#).

3.3.1.5 virtual void phantom::AudioEngine::setPosition ( const **Vector3** & *position* ) [pure virtual]

This function is called when you change the position of the listener.

#### Parameters

<i>position</i>	The position of the listener.
-----------------	-------------------------------

Implemented in [phantom::OpenALEngine](#).

3.3.1.6 virtual void phantom::AudioEngine::stopMusic ( **SoundData** \* *data* ) [pure virtual]

This function is called when you want to stop playing the music.

Implemented in [phantom::OpenALEngine](#).

3.3.1.7 virtual void phantom::AudioEngine::stopSound ( unsigned int *id* ) [pure virtual]

This function is called when you want to stop a sound from playing.

Implemented in [phantom::OpenALEngine](#).

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

- C:/Users/Sander/Documents/Projects/PCCS/phantom/src/audio/AudioEngine.h

## 3.4 phantom::tree::BinaryNode Class Reference

### Public Attributes

- int **key**
- void \* **data**
- [BinaryNode](#) \* **parent**
- [BinaryNode](#) \* **left**
- [BinaryNode](#) \* **right**

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

- C:/Users/Sander/Documents/Projects/PCCS/phantom/src/utlis/tree/BinaryNode.h

### 3.5 phantom::tree::BinaryTree Class Reference

#### Public Member Functions

- void **push** (int key)
- [BinaryNode](#) \* **find** (int key)
- void **erase** (int key)
- void **destroy** ()

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

- C:/Users/Sander/Documents/Projects/PCCS/phantom/src/utls/tree/BinaryTree.h
- C:/Users/Sander/Documents/Projects/PCCS/phantom/src/utls/tree/BinaryTree.cpp

### 3.6 phantom::Box3 Class Reference

#### Public Member Functions

- **Box3** (float x, float y, float width, float height)
- **Box3** ([Vector3](#) origin, [Vector3](#) size)
- bool **intersect** (const [Box3](#) &other) const
- bool **intersect** (const [Line2](#) &other) const
- bool **contains** (const [Vector3](#) &other) const
- bool **contains** (const [Vector3](#) \*other) const
- [Vector3](#) **getCenter** (void) const
- void **repair** (void)
- string **toString** (void)
- string **toString2** (void)

#### Public Attributes

- [Vector3](#) **origin**
- [Vector3](#) **size**

#### Friends

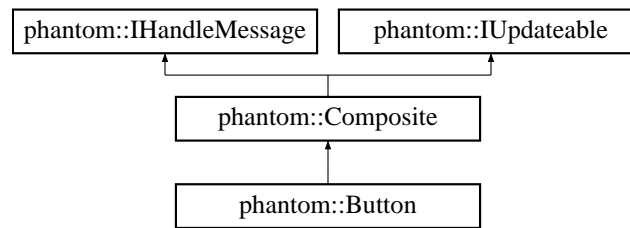
- ostream & **operator**<< (ostream &o, const [Box3](#) &b)

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

- C:/Users/Sander/Documents/Projects/PCCS/phantom/src/physics/Box3.h
- C:/Users/Sander/Documents/Projects/PCCS/phantom/src/physics/Box3.cpp

### 3.7 phantom::Button Class Reference

Inheritance diagram for phantom::Button:



## Public Member Functions

- **Button** (float x, float y, float width, float height)
- void [onClick](#) ([MouseState](#) \*mousestate)
- string & [text](#) ()
- void [setText](#) (const std::string &text)
- virtual void [paint](#) ()
- virtual void [update](#) (const [PhantomTime](#) &time)

## Public Attributes

- std::function< void()> **onClickFunction**

## Additional Inherited Members

### 3.7.1 Member Function Documentation

#### 3.7.1.1 void phantom::Button::onClick ( [MouseState](#) \* *mousestate* )

This function executes the onClickFunction.

##### Parameters

<i>mousestate</i>	The location of the mouse.
-------------------	----------------------------

#### 3.7.1.2 void phantom::Button::paint ( ) [[virtual](#)]

Paint gets executed before the label is printed on it. Override this function to create a skin for the button.

#### 3.7.1.3 void phantom::Button::setText ( const std::string & *text* )

Set the label of the button.

##### Parameters

<i>text</i>	The text you want to set on the button.
-------------	---

#### 3.7.1.4 string & phantom::Button::text ( )

Returns a reference to the text.

**Returns**

Returns a reference to the text.

### 3.7.1.5 void phantom::Button::update ( const PhantomTime & time ) [virtual]

This function draws the button and checks if it's pressed.

**Parameters**

<i>time</i>	The time created by <a href="#">PhantomGame</a> .
-------------	---

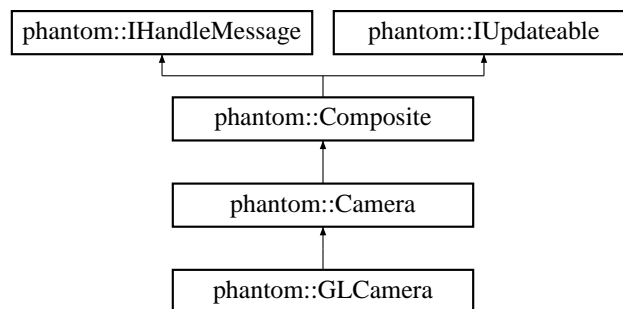
Reimplemented from [phantom::Composite](#).

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

- C:/Users/Sander/Documents/Projects/PCCS/phantom/src/guicomponents/Button.h
- C:/Users/Sander/Documents/Projects/PCCS/phantom/src/guicomponents/Button.cpp

## 3.8 phantom::Camera Class Reference

Inheritance diagram for phantom::Camera:

**Public Member Functions**

- **Camera** (int id)
- int [getCameraID](#) ()
- bool [isActive](#) ()
- [Vector3](#) [getWorldCoordinates](#) ([Vector3](#) viewCoordinate)
- [Vector3](#) & [getScreenSize](#) ()
- [Vector3](#) & [getViewPort](#) ()
- [Vector3](#) & [getRotation](#) ()
- virtual void [setScreenSize](#) ([Vector3](#) s)
- virtual void [setViewPort](#) ([Vector3](#) vp)
- virtual void [setRotation](#) ([Vector3](#) rot)
- virtual void [setParams](#) ()

**Friends**

- class **Driver**

## Additional Inherited Members

### 3.8.1 Member Function Documentation

#### 3.8.1.1 int phantom::Camera::getCameraID ( )

##### Returns

Returns the camera ID. This can be used to identify the camera.

#### 3.8.1.2 Vector3 & phantom::Camera::getRotation ( )

##### Returns

Returns the camera's rotation.

#### 3.8.1.3 Vector3 & phantom::Camera::getScreenSize ( )

##### Returns

Returns the screen size.

#### 3.8.1.4 Vector3 & phantom::Camera::getViewPort ( )

##### Returns

Returns the camera's viewport.

#### 3.8.1.5 Vector3 phantom::Camera::getWorldCoordinates ( Vector3 viewCoordinate )

##### Returns

Returns the world coordinate.

##### Parameters

<i>viewCoordinate</i>	Position on the screen.
-----------------------	-------------------------

#### 3.8.1.6 bool phantom::Camera::isActive ( )

##### Returns

Returns true if the camera is running.

#### 3.8.1.7 void phantom::Camera::setParams ( ) [virtual]

Function called by the renderer for setting the display properties.

Reimplemented in [phantom::GLCamera](#).

#### 3.8.1.8 void phantom::Camera::setRotation ( Vector3 rot ) [virtual]

Set the camera rotation.

## Parameters

<i>rot</i>	The camera rotation.
------------	----------------------

## 3.8.1.9 void phantom::Camera::setSize ( Vector3 s ) [virtual]

Sets the screen size.

## Parameters

<i>s</i>	The new screen size.
----------	----------------------

## 3.8.1.10 void phantom::Camera::setViewPort ( Vector3 vp ) [virtual]

Set the view port.

## Parameters

<i>vp</i>	The new view port.
-----------	--------------------

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

- C:/Users/Sander/Documents/Projects/PCCS/phantom/src/core/Camera.h
- C:/Users/Sander/Documents/Projects/PCCS/phantom/src/core/Camera.cpp

## 3.9 phantom::FreeTypeFont::char\_info\_t Struct Reference

### Public Attributes

- int **x**
- int **y**
- int **width**
- int **height**
- int **left**
- int **top**
- int **advance**
- int **row**
- TexCoord **uv** [4]
- Vertice **vertice** [4]
- unsigned int \* **bitmap**

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

- C:/Users/Sander/Documents/Projects/PCCS/phantom/src/graphics/FreeTypeFont.h

## 3.10 phantom::CollisionData Struct Reference

### Public Attributes

- bool **wasHandled**

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

- C:/Users/Sander/Documents/Projects/PCCS/phantom/src/physics/CollisionData.h

## 3.11 phantom::Color Struct Reference

### Public Member Functions

- **Color** (unsigned char r, unsigned char g, unsigned char b)
- **Color** (unsigned char r, unsigned char g, unsigned char b, unsigned char a)

### Public Attributes

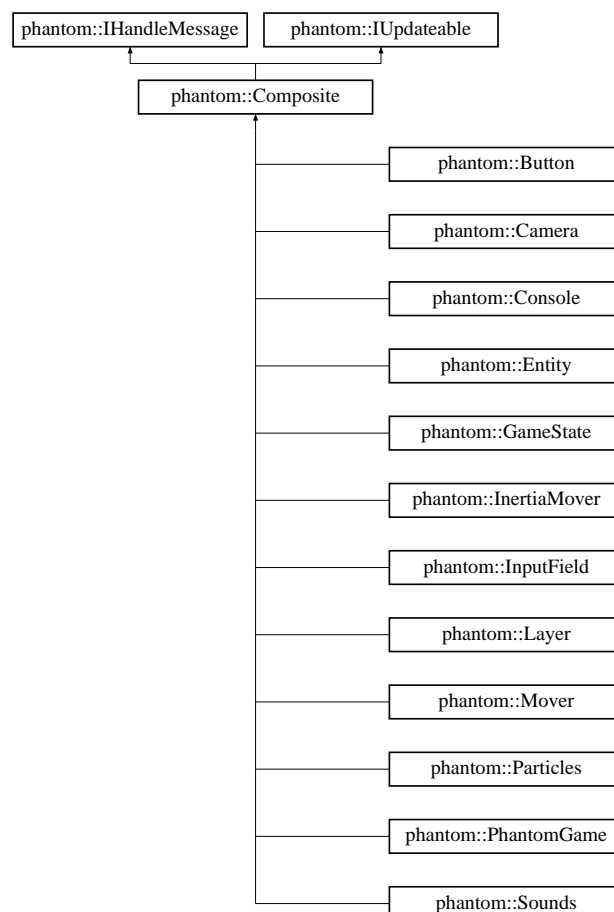
- unsigned char **r**
- unsigned char **g**
- unsigned char **b**
- unsigned char **a**

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

- C:/Users/Sander/Documents/Projects/PCCS/phantom/src/graphics/Color.h

## 3.12 phantom::Composite Class Reference

Inheritance diagram for phantom::Composite:



## Public Member Functions

- virtual void **onParentChange** ([Composite](#) \*parent)
- virtual void **onAncestorChanged** ()
- virtual void **onLayerChanged** ([Layer](#) \*layer)
- [Composite](#) \* **getParent** ()
- [PhantomGame](#) \* **getPhantomGame** (void)
- template<class T >  
T \* **getGame** (void)
- virtual void **addComponent** ([Composite](#) \*component)
- std::vector< [Composite](#) \* > & **getComponents** ()
- virtual void **removeComponent** ([Composite](#) \*who)
- virtual void **destroyComponent** ([Composite](#) \*who)
- virtual void **destroy** (void)
- virtual void **removeFromParent** (void)
- template<class T >  
T \* **findAncestor** ()
- template<class T >  
T \* **getComponentByType** (int nth)
- bool **isDestroyed** ()
- virtual MessageState **handleMessage** ([AbstractMessage](#) \*message)
- virtual void **update** (const [PhantomTime](#) &time)
- [Graphics](#) & **getGraphics** ()
- virtual bool **canCollideWith** ([Composite](#) \*other)
- virtual void **onCollision** ([Composite](#) \*other, [CollisionData](#) &collisionData)
- [Box3](#) & **getBoundingBox** ()
- void **setBoundingBox** (const [Box3](#) &boundingBox)
- const [Vector3](#) & **getPosition** ()
- virtual void **setPosition** ([Vector3](#) position)
- virtual void **setX** (float x)
- virtual void **setY** (float y)
- virtual void **addPosition** (const [Vector3](#) &add)
- virtual void **removePosition** (const [Vector3](#) &subtract)
- virtual void **setDirection** ([Vector3](#) direction)
- const string & **getType** () const
- bool **isType** (const string &type) const
- bool **isType** (const [Composite](#) &other) const
- bool **isType** (const [Composite](#) \*other) const
- string **toString** (void)
- [Layer](#) \* **getLayer** ()

## Public Attributes

- bool **isStatic**

## Protected Member Functions

- void **setType** (const string &type)
- [Driver](#) \* **getDriver** (void)



## Protected Attributes

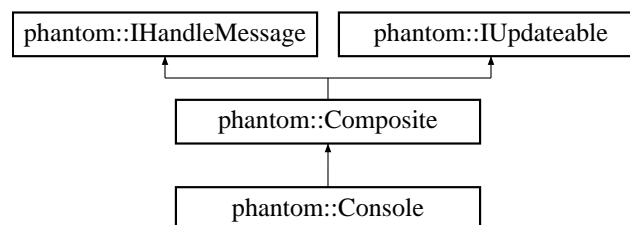
- [Vector3](#) `_position`
- [Vector3](#) `_direction`
- [Box3](#) `_boundingBox`
- [Layer](#) \* `_layer`
- [Composite](#) \* `_parent`

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

- C:/Users/Sander/Documents/Projects/PCCS/phantom/src/core/Composite.h
- C:/Users/Sander/Documents/Projects/PCCS/phantom/src/core/Composite.cpp

## 3.13 phantom::Console Class Reference

Inheritance diagram for phantom::Console:



## Public Member Functions

- void [addLog](#) (string [log](#))
- virtual void [update](#) (const [PhantomTime](#) &time)

## Static Public Member Functions

- static void [log](#) (string log)
- static void [log](#) (stringstream log)
- static void [mapCommand](#) (string name, function< void(string args)> function)
- template<class mType >  
static void [log](#) ([Message](#)< mType > log)

## Additional Inherited Members

### 3.13.1 Member Function Documentation

#### 3.13.1.1 void phantom::Console::addLog ( string *log* )

Add a log message to the console.

#### Parameters

<i>log</i>	The message in string format.
------------	-------------------------------

### 3.13.1.2 void phantom::Console::log ( string *log* ) [static]

Add a log message to the console.

#### Parameters

<i>log</i>	The message in string format.
------------	-------------------------------

### 3.13.1.3 void phantom::Console::log ( stringstream *log* ) [static]

Add a log message to the console.

#### Parameters

<i>log</i>	The message in stringstream format.
------------	-------------------------------------

### 3.13.1.4 template<class mType > static void phantom::Console::log ( Message< mType > *log* ) [inline], [static]

An easy way to log the phantom [Message](#) class.

#### Parameters

<i>log</i>	The message you want to send to the logger.
------------	---

### 3.13.1.5 void phantom::Console::mapCommand ( string *name*, function< void(string args)> *function* ) [static]

Add a console command to the console.

#### Parameters

<i>name</i>	The name of the command you have to call.
<i>function</i>	The function that gets executed once the command is called.

### 3.13.1.6 void phantom::Console::update ( const PhantomTime & *time* ) [virtual]

This function is called every loop automatically since the console is a [Composite](#) that's added to the [PhantomGame](#).

#### Parameters

<i>time</i>	<a href="#">PhantomTime</a> generated by <a href="#">PhantomGame</a> .
-------------	--

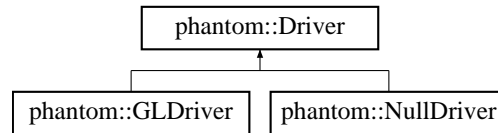
Reimplemented from [phantom::Composite](#).

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

- C:/Users/Sander/Documents/Projects/PCCS/phantom/src/core/Console.h
- C:/Users/Sander/Documents/Projects/PCCS/phantom/src/core/Console.cpp

## 3.14 phantom::Driver Class Reference

Inheritance diagram for phantom::Driver:



## Public Member Functions

- **Driver** ([PhantomGame](#) \*game)
- virtual void [setWindowTitle](#) (string title)
- virtual void [onUpdate](#) ([PhantomTime](#) time)
- virtual void [onRender](#) ()
- virtual [Camera](#) \* [createCamera](#) ()=0
- vector< [Camera](#) \* > \* [getActiveCameras](#) ()
- void [enableCamera](#) ([Camera](#) \*cam)
- void [disableCamera](#) ([Camera](#) \*cam)
- [Input](#) \* [getInput](#) ()
- [Renderer](#) \* [getRenderer](#) ()
- [Sounds](#) \* [getAudio](#) ()
- [AudioEngine](#) \* [getAudioEngine](#) ()
- [FreeTypeLibrary](#) \* [getFontLibrary](#) ()

## Protected Member Functions

- void **addCamToList** ([Camera](#) \*cam)

## Protected Attributes

- [Renderer](#) \* **\_renderer**
- [AudioEngine](#) \* **\_audioEngine**
- [Sounds](#) \* **\_audio**
- [FreeTypeLibrary](#) \* **\_fontLibrary**
- [Input](#) \* **\_input**
- vector< [Camera](#) \* > **\_cameras**
- vector< [Camera](#) \* > **\_activeCameras**
- [KeyboardListener](#) \* **\_keyboard**
- [PhantomGame](#) \* **\_game**

### 3.14.1 Member Function Documentation

#### 3.14.1.1 virtual [Camera](#)\* phantom::Driver::createCamera ( ) [pure virtual]

Creates a camera.

Implemented in [phantom::GLDriver](#), and [phantom::NullDriver](#).

#### 3.14.1.2 void phantom::Driver::disableCamera ( [Camera](#) \* cam )

Disables a camera.

#### 3.14.1.3 void phantom::Driver::enableCamera ( [Camera](#) \* cam )

Enables a camera.

#### 3.14.1.4 `vector< Camera * > * phantom::Driver::getActiveCameras ( )`

##### Returns

Returns a list of active cameras.

#### 3.14.1.5 `Sounds * phantom::Driver::getAudio ( )`

##### Returns

Returns the sounds class. Used for playing sounds and music.

#### 3.14.1.6 `AudioEngine * phantom::Driver::getAudioEngine ( )`

##### Returns

Returns the audio engine.

#### 3.14.1.7 `FreeTypeLibrary * phantom::Driver::getFontLibrary ( )`

##### Returns

Returns the TTF font library.

#### 3.14.1.8 `Input * phantom::Driver::getInput ( )`

##### Returns

Returns the input handler.

#### 3.14.1.9 `Renderer * phantom::Driver::getRenderer ( )`

##### Returns

Returns the renderer.

#### 3.14.1.10 `void phantom::Driver::onRender ( )` [virtual]

Gets called from [PhantomGame](#) to initiate the rendering.

Reimplemented in [phantom::NullDriver](#).

#### 3.14.1.11 `void phantom::Driver::onUpdate ( PhantomTime time )` [virtual]

Gets called from [PhantomGame](#) to initiate the driver update loop.

##### Parameters

<i>time</i>	The time generated by <a href="#">PhantomGame</a> .
-------------	---

Reimplemented in [phantom::NullDriver](#).

#### 3.14.1.12 void phantom::Driver::setWindowTitle ( string *title* ) [virtual]

Change the window title of the game.

##### Parameters

<i>title</i>	The new title to set.
--------------	-----------------------

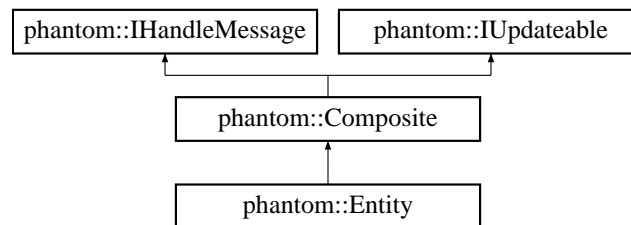
Reimplemented in [phantom::GLDriver](#).

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

- C:/Users/Sander/Documents/Projects/PCCS/phantom/src/core/Driver.h
- C:/Users/Sander/Documents/Projects/PCCS/phantom/src/core/Driver.cpp

## 3.15 phantom::Entity Class Reference

Inheritance diagram for phantom::Entity:



### Public Member Functions

- virtual void [addComponent](#) ( [Composite](#) \*component)
- float [distanceTo](#) ( [Entity](#) \*gob)
- float [distanceToSq](#) ( [Entity](#) \*gob)
- [Vector3](#) [directionTo](#) ( [Entity](#) \*gob)

### Public Attributes

- unsigned **solidState**
- unsigned **solidType**
- [InertiaMover](#) \* **inertia**

### Protected Attributes

- [Mover](#) \* **mover**

### Additional Inherited Members

#### 3.15.1 Member Function Documentation

##### 3.15.1.1 void phantom::Entity::addComponent ( [Composite](#) \* *component* ) [virtual]

All components that get added to the [Entity](#) component are checked if they are of the [Mover](#) or [InertiaMover](#) type. They can only exist once for every class.

## Parameters

<i>component</i>	The component you want to add.
------------------	--------------------------------

Reimplemented from [phantom::Composite](#).

### 3.15.1.2 Vector3 phantom::Entity::directionTo ( Entity \* gob )

Get the normalized direction vector to another entity.

## Returns

Returns the direction vector to another entity.

## Parameters

<i>gob</i>	The entity you want calculate the direction to.
------------	---

### 3.15.1.3 float phantom::Entity::distanceTo ( Entity \* gob )

Calculate the distance to another entity. Please consider using distanceToSq, since it's faster.

## Returns

Returns the distance to another entity.

## Parameters

<i>gob</i>	The entity you want to measure the distance to.
------------	---

### 3.15.1.4 float phantom::Entity::distanceToSq ( Entity \* gob )

Calculate the distance to another entity squared. This is faster than distanceTo, because it's not calling the heavy sqrt function.

## Returns

Returns the distance squared to the other entity.

## Parameters

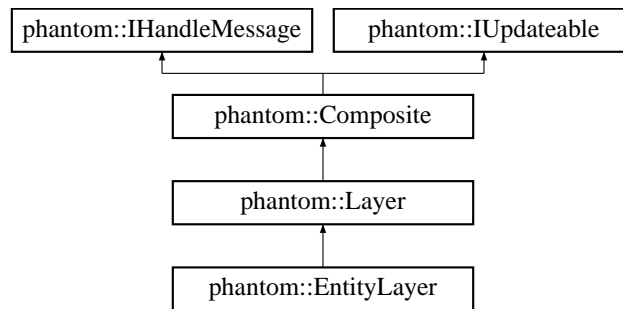
<i>gob</i>	The entity you want to measure the distance squared to.
------------	---

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

- C:/Users/Sander/Documents/Projects/PCCS/phantom/src/core/Entity.h
- C:/Users/Sander/Documents/Projects/PCCS/phantom/src/core/Entity.cpp

## 3.16 phantom::EntityLayer Class Reference

Inheritance diagram for phantom::EntityLayer:



### Public Member Functions

- virtual void **update** (const [PhantomTime](#) &time)
- virtual void **addComponent** ([Composite](#) \*component)

### Protected Member Functions

- bool **calculateCollision** ([Entity](#) \*a, [Entity](#) \*b)

### Additional Inherited Members

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

- C:/Users/Sander/Documents/Projects/PCCS/phantom/src/layer/EntityLayer.h
- C:/Users/Sander/Documents/Projects/PCCS/phantom/src/layer/EntityLayer.cpp

## 3.17 phantom::FreeTypeFont::font\_info\_t Struct Reference

### Public Attributes

- int **maxHeight**
- std::vector  
< [FreeTypeFont::char\\_info\\_t](#) > **characters**

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

- C:/Users/Sander/Documents/Projects/PCCS/phantom/src/graphics/FreeTypeFont.h

## 3.18 phantom::FreeTypeFont Class Reference

### Classes

- struct [char\\_info\\_t](#)
- struct [font\\_info\\_t](#)

## Public Attributes

- [font\\_info\\_t](#) **info**
- [ImageCacheItem](#) \* **texture**

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

- C:/Users/Sander/Documents/Projects/PCCS/phantom/src/graphics/FreeTypeFont.h

## 3.19 phantom::FreeTypeLibrary Class Reference

### Public Member Functions

- **FreeTypeLibrary** ([Renderer](#) \*renderer)
- [FreeTypeFont](#) \* **getFont** ([Text](#) \*txt)

### Public Attributes

- [FT\\_Library](#) **lib**

### 3.19.1 Member Function Documentation

#### 3.19.1.1 FreeTypeFont \* phantom::FreeTypeLibrary::getFont ( Text \* txt )

Try getting the font. If it's not available, it will get added.

#### Returns

Returns the font.

#### Parameters

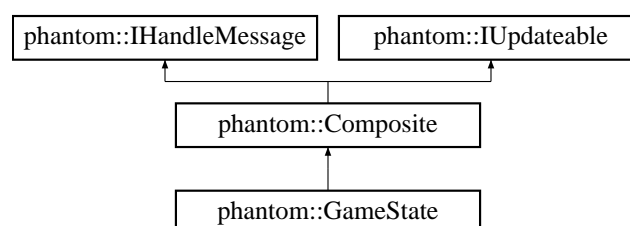
<i>txt</i>	The text shape you want to get the font from.
------------	---

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

- C:/Users/Sander/Documents/Projects/PCCS/phantom/src/graphics/FreeTypeLibrary.h
- C:/Users/Sander/Documents/Projects/PCCS/phantom/src/graphics/FreeTypeLibrary.cpp

## 3.20 phantom::GameState Class Reference

Inheritance diagram for phantom::GameState:





## Public Attributes

- bool **doUpdate**
- bool **doRender**

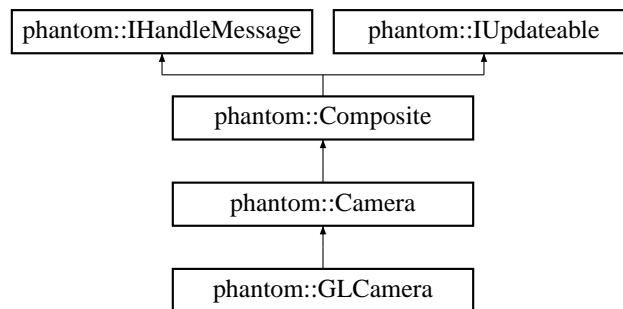
## Additional Inherited Members

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

- C:/Users/Sander/Documents/Projects/PCCS/phantom/src/core/GameState.h
- C:/Users/Sander/Documents/Projects/PCCS/phantom/src/core/GameState.cpp

## 3.21 phantom::GLCamera Class Reference

Inheritance diagram for phantom::GLCamera:



## Public Member Functions

- **GLCamera** (int id)
- virtual void [setParams](#) ()

## Additional Inherited Members

### 3.21.1 Member Function Documentation

#### 3.21.1.1 void phantom::GLCamera::setParams ( ) [virtual]

Function called by the renderer for setting the display properties.

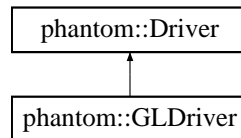
Reimplemented from [phantom::Camera](#).

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

- C:/Users/Sander/Documents/Projects/PCCS/phantom/src/opengl/GLCamera.h
- C:/Users/Sander/Documents/Projects/PCCS/phantom/src/opengl/GLCamera.cpp

## 3.22 phantom::GLDriver Class Reference

Inheritance diagram for phantom::GLDriver:



## Public Member Functions

- **GLDriver** ([PhantomGame](#) \*game)
- virtual void [setWindowTitle](#) (string title)
- virtual [Camera](#) \* [createCamera](#) (void)

## Additional Inherited Members

### 3.22.1 Member Function Documentation

#### 3.22.1.1 [Camera](#) \* [phantom::GLDriver::createCamera](#) ( void ) [virtual]

Creates a camera.

Implements [phantom::Driver](#).

#### 3.22.1.2 void [phantom::GLDriver::setWindowTitle](#) ( string title ) [virtual]

Change the window title of the game.

#### Parameters

<i>title</i>	The new title to set.
--------------	-----------------------

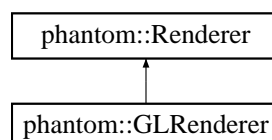
Reimplemented from [phantom::Driver](#).

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

- C:/Users/Sander/Documents/Projects/PCCS/phantom/src/opengl/GLDriver.h
- C:/Users/Sander/Documents/Projects/PCCS/phantom/src/opengl/GLDriver.cpp

## 3.23 [phantom::GLRenderer](#) Class Reference

Inheritance diagram for [phantom::GLRenderer](#):



## Public Member Functions

- **GLRenderer** ([PhantomGame](#) \*game)
- virtual void [renderLoop](#) ()
- virtual void [buildShape](#) ([Shape](#) \*shape)

- virtual void [destroyShape](#) ([Shape](#) \*shape)
- virtual void [addTexture](#) ([ImageCachelItem](#) \*item, bool isText=false)
- virtual void [removeTexture](#) ([ImageCachelItem](#) \*item)

## Additional Inherited Members

### 3.23.1 Member Function Documentation

3.23.1.1 void phantom::GLRenderer::addTexture ( [ImageCachelItem](#) \* *item*, bool *isText* = false ) [virtual]

This gets called when a texture has to be added to the graphics pipeline.

#### Parameters

<i>item</i>	The imache cache item that has to be created.
<i>isText</i>	The fonts are converted to images as well. Set this to true to have an optimization for font rendering.

Implements [phantom::Renderer](#).

3.23.1.2 void phantom::GLRenderer::buildShape ( [Shape](#) \* *shape* ) [virtual]

This gets called by the [Graphics](#) class. This makes it possible to build VBO's for example.

#### Parameters

<i>shape</i>	The shape you want to build.
--------------	------------------------------

Implements [phantom::Renderer](#).

3.23.1.3 void phantom::GLRenderer::destroyShape ( [Shape](#) \* *shape* ) [virtual]

This gets called by the [Graphics](#) class. This makes it possible to destroy VBO's for example.

#### Parameters

<i>shape</i>	The shape you want to destroy.
--------------	--------------------------------

Implements [phantom::Renderer](#).

3.23.1.4 void phantom::GLRenderer::removeTexture ( [ImageCachelItem](#) \* *item* ) [virtual]

This gets called when the texture has to be removed from the graphics pipeline.

#### Parameters

<i>item</i>	The item you want to remove from the cache.
-------------	---

Implements [phantom::Renderer](#).

3.23.1.5 void phantom::GLRenderer::renderLoop ( ) [virtual]

This is called every loop by the [Driver](#) class.

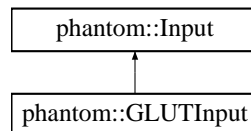
Implements [phantom::Renderer](#).

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

- C:/Users/Sander/Documents/Projects/PCCS/phantom/src/opengl/GLRenderer.h
- C:/Users/Sander/Documents/Projects/PCCS/phantom/src/opengl/GLRenderer.cpp

### 3.24 phantom::GLUTInput Class Reference

Inheritance diagram for phantom::GLUTInput:



#### Public Member Functions

- **GLUTInput** ([PhantomGame](#) \*game)

#### Additional Inherited Members

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

- C:/Users/Sander/Documents/Projects/PCCS/phantom/src/input/GLUTInput.h
- C:/Users/Sander/Documents/Projects/PCCS/phantom/src/input/GLUTInput.cpp

### 3.25 phantom::Graphics Class Reference

#### Public Member Functions

- **Graphics** ([phantom::Composite](#) \*parent)
- [Graphics](#) & [clear](#) ()
- [Graphics](#) & [beginPath](#) ()
- [Graphics](#) & [fill](#) ()
- [Graphics](#) & [stroke](#) ()
- [Graphics](#) & [setFillStyle](#) ([Color](#) color)
- [Graphics](#) & [setLineStyle](#) ([Color](#) color)
- [Graphics](#) & [line](#) (const float &startX, const float &startY, const float &endX, const float &endY)
- [Graphics](#) & [line](#) (const [Line2](#) &line)
- [Graphics](#) & [line](#) (const [Vector3](#) &start, const [Vector3](#) &end)
- [Graphics](#) & [rect](#) (float x, float y, float width, float height, bool isFilled=true, float thickness=3.0f)
- [Graphics](#) & [rect](#) (const [Box3](#) &box, bool isFilled=true, float thickness=3.0f)
- [Graphics](#) & [arc](#) (float x, float y, float radius, float start, float end)
- [Graphics](#) & [image](#) (const string &filelocation, float x, float y, float width, float height)
- [Graphics](#) & [text](#) (float x, float y, unsigned int size, const string &fontlocation, const string &text)
- [Graphics](#) & [rotate](#) (float angle)
- [Graphics](#) & [lineTo](#) (float x, float y)
- [Graphics](#) & [moveTo](#) (float x, float y)
- float [getRotation](#) ()

## Friends

- class **GLRenderer**

### 3.25.1 Member Function Documentation

#### 3.25.1.1 Graphics & phantom::Graphics::arc ( float *x*, float *y*, float *radius*, float *start*, float *end* )

Draw an arc.

##### Returns

Returns this class for chaining.

##### Parameters

<i>x</i>	The X location relative to the component.
<i>y</i>	The Y location relative to the component.
<i>radius</i>	The size of the arc.
<i>start</i>	At how much degrees this arc must begin.
<i>end</i>	At how much degrees this arc must end.

#### 3.25.1.2 Graphics & phantom::Graphics::beginPath ( )

Call this function when you want to begin drawing on the canvas.

##### Returns

Returns this class for chaining.

#### 3.25.1.3 Graphics & phantom::Graphics::clear ( )

Call this function when you want to clear the canvas.

##### Returns

Returns this class for chaining.

#### 3.25.1.4 Graphics & phantom::Graphics::fill ( void )

Call this function when you are done drawing on the canvas.

##### Returns

Returns this class for chaining.

#### 3.25.1.5 float phantom::Graphics::getRotation ( ) [inline]

Returns the current rotation of the graphics canvas.

##### Returns

Returns the current rotation of the graphics canvas.

### 3.25.1.6 Graphics & phantom::Graphics::image ( const string & *filelocation*, float *x*, float *y*, float *width*, float *height* )

Draw an image.

#### Returns

Returns this class for chaining.

#### Parameters

<i>filelocation</i>	The file location relative to the working directory.
<i>x</i>	The X location relative to the component.
<i>y</i>	The Y location relative to the component.
<i>width</i>	The width of the image.
<i>height</i>	The height of the image.

### 3.25.1.7 Graphics & phantom::Graphics::line ( const float & *startX*, const float & *startY*, const float & *endX*, const float & *endY* )

Draw a line.

#### Returns

Returns this class for chaining.

#### Parameters

<i>startX</i>	The X start location of the line.
<i>startY</i>	The Y start location of the line.
<i>endX</i>	The X end location of the line.
<i>endY</i>	The Y end location of the line.

### 3.25.1.8 Graphics & phantom::Graphics::line ( const Line2 & *line* )

Draw a line.

#### Returns

Returns this class for chaining.

#### Parameters

<i>line</i>	Create a line using a <a href="#">Line2</a> class.
-------------	--

### 3.25.1.9 Graphics & phantom::Graphics::line ( const Vector3 & *start*, const Vector3 & *end* )

Draw a line.

#### Returns

Returns this class for chaining.

## Parameters

<i>start</i>	Start location of the line.
<i>end</i>	End location of the line.

3.25.1.10 Graphics & phantom::Graphics::lineTo ( float *x*, float *y* )

Start a line drawing.

## Returns

Returns this class for chaining.

## Parameters

<i>x</i>	The X location relative to the component where you want to start the line.
<i>y</i>	The Y location relative to the component where you want to start the line.

3.25.1.11 Graphics & phantom::Graphics::moveTo ( float *x*, float *y* )

Continue a line drawing.

## Returns

Returns this class for chaining.

## Parameters

<i>x</i>	The X location relative to the component where you want to go next with the line.
<i>y</i>	The Y location relative to the component where you want to go next with the line.

3.25.1.12 Graphics & phantom::Graphics::rect ( float *x*, float *y*, float *width*, float *height*, bool *isFilled* = `true`, float *thickness* = `3.0f` )

Draw a rectangle.

## Returns

Returns this class for chaining.

## Parameters

<i>x</i>	The X location of the rectangle relative to the component.
<i>y</i>	The Y location of the rectangle relative to the component.
<i>width</i>	The width of the rectangle.
<i>height</i>	The height of the rectangle.
<i>isFilled</i>	True if the rectangle has to be filled.
<i>thickness</i>	If <i>isFilled</i> is false, you can set the linewidth here.

3.25.1.13 Graphics & phantom::Graphics::rect ( const Box3 & *box*, bool *isFilled* = `true`, float *thickness* = `3.0f` )

Draw a rectangle.

**Returns**

Returns this class for chaining.

**Parameters**

<i>box</i>	The box it should draw.
<i>height</i>	The height of the rectangle.
<i>isFilled</i>	True if the rectangle has to be filled.
<i>thickness</i>	If isFilled is false, you can set the linewidth here.

**3.25.1.14 Graphics & phantom::Graphics::rotate ( float *angle* )**

Rotate the graphic.

**Returns**

Returns this class for chaining.

**Parameters**

<i>angle</i>	The angle you want to rotate the canvas.
--------------	--

**3.25.1.15 Graphics & phantom::Graphics::setFillStyle ( Color *color* )**

Change the fill color of the graphics object.

**Returns**

Returns this class for chaining.

**Parameters**

<i>color</i>	The color you want to use for the object.
--------------	---

**3.25.1.16 Graphics & phantom::Graphics::setLineStyle ( Color *color* )**

Change the line color of the graphics object.

**Returns**

Returns this class for chaining.

**Parameters**

<i>color</i>	The color you want to use for the object.
--------------	---

**3.25.1.17 Graphics & phantom::Graphics::stroke ( void )**

Call this function when you are done drawing on the canvas.



**Returns**

Returns this class for chaining.

### 3.25.1.18 Graphics & phantom::Graphics::text ( float *x*, float *y*, unsigned int *size*, const string & *fontlocation*, const string & *text* )

Draw some text.

**Returns**

Returns this class for chaining.

**Parameters**

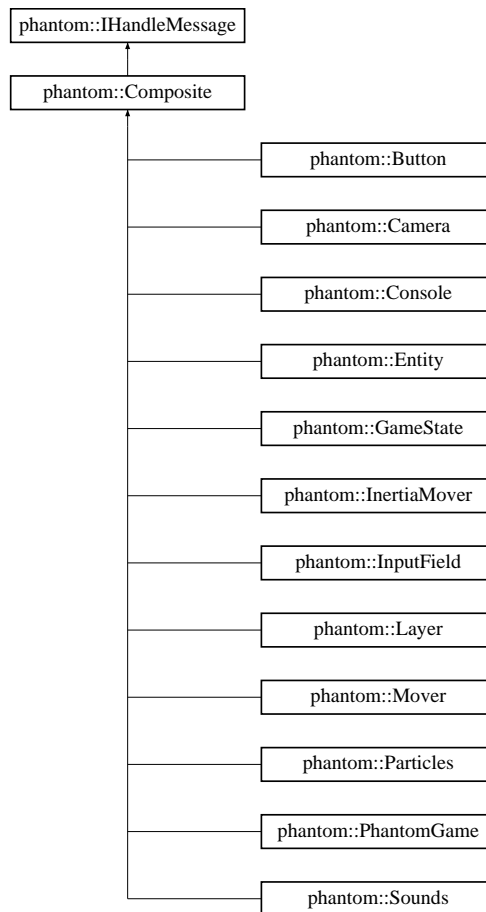
<i>x</i>	The X location relative to the component.
<i>y</i>	The Y location relative to the component.
<i>size</i>	The font size of the text.
<i>fontlocation</i>	The location of the font relative to the working directory.
<i>text</i>	The text you want to display.

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

- C:/Users/Sander/Documents/Projects/PCCS/phantom/src/graphics/Graphics.h
- C:/Users/Sander/Documents/Projects/PCCS/phantom/src/graphics/Graphics.cpp

## 3.26 phantom::IHandleMessage Class Reference

Inheritance diagram for phantom::IHandleMessage:



### Public Member Functions

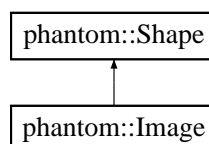
- virtual MessageState **handleMessage** ([AbstractMessage](#) \*message)=0

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

- C:/Users/Sander/Documents/Projects/PCCS/phantom/src/messaging/IHandleMessage.h

## 3.27 phantom::Image Class Reference

Inheritance diagram for phantom::Image:



### Public Member Functions

- **Image** (std::string filelocation, float x, float y, float width, float height)
- [ImageCacheItem](#) \* **getImage** ()

## Additional Inherited Members

### 3.27.1 Member Function Documentation

#### 3.27.1.1 ImageCacheItem\* phantom::Image::getImage ( ) [inline]

Returns the image associated with this shape.

#### Returns

Returns the image associated with this shape.

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

- C:/Users/Sander/Documents/Projects/PCCS/phantom/src/graphics/shapes/Image.h
- C:/Users/Sander/Documents/Projects/PCCS/phantom/src/graphics/shapes/Image.cpp

## 3.28 phantom::ImageCache Class Reference

### Public Member Functions

- void [setRenderer](#) ([Renderer](#) \*renderer)
- bool [isCached](#) (const string filename)
- void [insertIntoCache](#) (const string filename, [ImageCacheItem](#) \*item)
- [ImageCacheItem](#) \* [getFromCache](#) (const string filename)
- void [removeFromCache](#) (const string filename)

### Static Public Member Functions

- static [ImageCache](#) \* [getInstance](#) ()

### 3.28.1 Member Function Documentation

#### 3.28.1.1 ImageCacheItem \* phantom::ImageCache::getFromCache ( const string filename )

Get an image that's located in the cache.

#### Returns

Returns nullptr if nothing is found, else it returns the item.

#### Parameters

<i>filename</i>	The location relative to the working directory.
-----------------	---

#### 3.28.1.2 void phantom::ImageCache::insertIntoCache ( const string filename, ImageCacheItem \* item )

Insert an image into the cache.

*filename* The location of an image relative to the working directory. *item* The [ImageCacheItem](#) you want to add to the cache.

### 3.28.1.3 `bool phantom::ImageCache::isCached ( const string filename )`

Returns true if the image is cached.

#### Returns

Returns true if the image is cached.

### 3.28.1.4 `void phantom::ImageCache::removeFromCache ( const string filename )`

Remove an image from the cache.

#### Parameters

<i>filename</i>	The location of the image relative to the working directory.
-----------------	--

### 3.28.1.5 `void phantom::ImageCache::setRenderer ( Renderer * renderer )`

Sets the renderer for adding and removing textures.

#### Parameters

<i>renderer</i>	The renderer that's being used in the game.
-----------------	---

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

- C:/Users/Sander/Documents/Projects/PCCS/phantom/src/graphics/ImageCache.h
- C:/Users/Sander/Documents/Projects/PCCS/phantom/src/graphics/ImageCache.cpp

## 3.29 `phantom::ImageCacheItem` Class Reference

### Public Attributes

- unsigned int **width**
- unsigned int **height**
- unsigned int **textureID**
- unsigned char \* **imageData**
- unsigned char \*\* **row\_pointers**

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

- C:/Users/Sander/Documents/Projects/PCCS/phantom/src/graphics/ImageCache.h

## 3.30 `phantom::ImageLoader` Class Reference

### Static Public Member Functions

- static `ImageCacheItem * createPNG` (const std::string filename)

### 3.30.1 Member Function Documentation

#### 3.30.1.1 ImageCacheItem \* phantom::ImageLoader::createPNG ( const std::string filename ) [static]

Load a PNG and store it in an [ImageCacheItem](#)

#### Returns

Returns the [ImageCacheItem](#) created.

#### Parameters

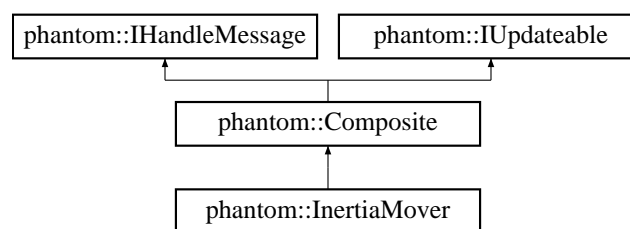
<i>filename</i>	The Location of the image relative to the working directory.
-----------------	--

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

- C:/Users/Sander/Documents/Projects/PCCS/phantom/src/graphics/ImageLoader.h
- C:/Users/Sander/Documents/Projects/PCCS/phantom/src/graphics/ImageLoader.cpp

## 3.31 phantom::InertiaMover Class Reference

Inheritance diagram for phantom::InertiaMover:



### Public Member Functions

- void **addPulse** ([Pulse](#) pulse)
- void **clear** ()
- virtual void **update** (const [PhantomTime](#) &time)
- virtual MessageState **handleMessage** ([AbstractMessage](#) \*message)
- const [Vector3](#) & **getDirection** (void)
- const [Vector3](#) & **getDominantDirection** (void)
- bool **isMoving** (void) const
- bool **isStopped** (void) const

### Public Attributes

- [Pulse](#) \_dominant

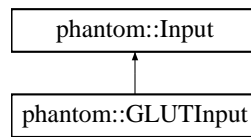
### Additional Inherited Members

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

- C:/Users/Sander/Documents/Projects/PCCS/phantom/src/physics/InertiaMover.h
- C:/Users/Sander/Documents/Projects/PCCS/phantom/src/physics/InertiaMover.cpp

### 3.32 phantom::Input Class Reference

Inheritance diagram for phantom::Input:



#### Public Member Functions

- **Input** ([PhantomGame](#) \*game)
- virtual [KeyboardState](#) \* [getKeyboardState](#) ()
- virtual [MouseState](#) \* [getMouseState](#) ()

#### Protected Attributes

- [PhantomGame](#) \* **\_game**
- [KeyboardState](#) \* **\_keyboardState**
- [MouseState](#) \* **\_mouseState**

#### 3.32.1 Member Function Documentation

3.32.1.1 virtual [KeyboardState](#)\* phantom::Input::getKeyboardState ( ) [inline],[virtual]

Retrieve the current [KeyboardState](#).

##### Returns

Returns the current keyboard state.

3.32.1.2 virtual [MouseState](#)\* phantom::Input::getMouseState ( ) [inline],[virtual]

Retrieve the current [MouseState](#).

##### Returns

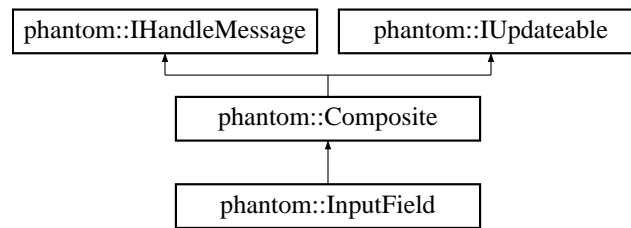
Returns the current [MouseState](#).

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

- C:/Users/Sander/Documents/Projects/PCCS/phantom/src/input/Input.h

### 3.33 phantom::InputField Class Reference

Inheritance diagram for phantom::InputField:



## Public Member Functions

- **InputField** (float x, float y, float width, float height, [phantom::Color](#) color)
- virtual void [onClick](#) ([MouseState](#) \*mousestate)
- virtual void [onUnClicked](#) ([MouseState](#) \*mousestate)
- virtual void [paint](#) ()
- virtual void [paintText](#) ()
- virtual void [update](#) (const [phantom::PhantomTime](#) &time)
- void [keyboard](#) ([KeyboardListener](#) \*keyboardListener)
- std::string & [text](#) ()
- void [text](#) (const std::string &value)

## Additional Inherited Members

### 3.33.1 Member Function Documentation

3.33.1.1 void [phantom::InputField::keyboard](#) ( [KeyboardListener](#) \* *keyboardListener* ) [inline]

Set the keyboard listener so keys can be fetched.

#### Parameters

<i>keyboard-Listener</i>	The keyboad listener.
--------------------------	-----------------------

3.33.1.2 void [phantom::InputField::onClick](#) ( [MouseState](#) \* *mousestate* ) [virtual]

Gets called when clicked on the textfield.

#### Parameters

<i>mousestate</i>	The mousestate when pressed.
-------------------	------------------------------

3.33.1.3 void [phantom::InputField::onUnClicked](#) ( [MouseState](#) \* *mousestate* ) [virtual]

Gets called when clicked somewhere else.

#### Parameters

<i>mousestate</i>	The mousestate when clicked somewhere else.
-------------------	---

#### 3.33.1.4 void phantom::InputField::paint ( ) [virtual]

Override this function to draw your own background for the textfield.

#### 3.33.1.5 void phantom::InputField::paintText ( ) [virtual]

Override this function to draw your own text for the textfield.

#### 3.33.1.6 string & phantom::InputField::text ( )

Returns a reference to the current text in the textfield.

##### Returns

Returns the current text in the field.

#### 3.33.1.7 void phantom::InputField::text ( const std::string & value )

Set the text to new value.

##### Parameters

<i>value</i>	The new value for the textfield.
--------------	----------------------------------

#### 3.33.1.8 void phantom::InputField::update ( const phantom::PhantomTime & time ) [virtual]

This function calls the paint and paintText function. Also listens for keypresses.

##### Parameters

<i>time</i>	The time created by <a href="#">PhantomGame</a> .
-------------	---

Reimplemented from [phantom::Composite](#).

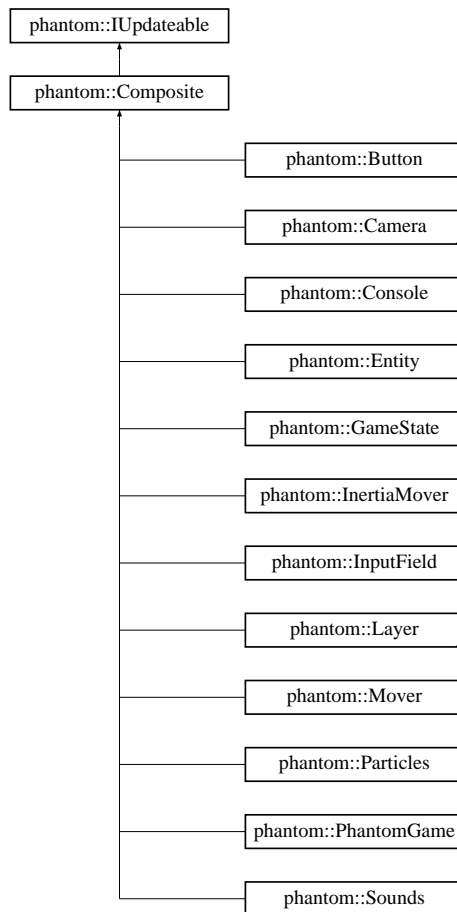
The documentation for this class was generated from the following files:

- C:/Users/Sander/Documents/Projects/PCCS/phantom/src/guicomponents/InputField.h
- C:/Users/Sander/Documents/Projects/PCCS/phantom/src/guicomponents/InputField.cpp

## 3.34 phantom::IUpdateable Class Reference

Inheritance diagram for phantom::IUpdateable:





### Public Member Functions

- virtual void **update** (const [phantom::PhantomTime](#) &time)=0

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

- C:/Users/Sander/Documents/Projects/PCCS/phantom/src/core/IUpdateable.h

## 3.35 phantom::KeyListener Class Reference

### Public Member Functions

- **KeyListener** ([Driver](#) \*driver, [PhantomGame](#) \*game)
- void **update** ()
- bool [isLocked](#) ()

### Static Public Member Functions

- static [KeyboardState](#) \* **lock** ([Composite](#) \*keycomp)
- static void **unlock** ([Composite](#) \*keycomp)

### 3.35.1 Member Function Documentation

#### 3.35.1.1 `bool phantom::KeyboardListener::isLocked ( ) [inline]`

Returns if the listener is locked.

##### Returns

Returns true if the listener is locked.

#### 3.35.1.2 `KeyboardState * phantom::KeyboardListener::lock ( Composite * keycomp ) [static]`

Locks the keyboard to a certain composite.

##### Returns

Returns the current keyboard state.

##### Parameters

<i>keycomp</i>	The component that owns the listener.
----------------	---------------------------------------

#### 3.35.1.3 `void phantom::KeyboardListener::unlock ( Composite * keycomp ) [static]`

Unlocks the keyboard to other composites.

##### Parameters

<i>keycomp</i>	The component that owns the listener.
----------------	---------------------------------------

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

- C:/Users/Sander/Documents/Projects/PCCS/phantom/src/input/KeyboardListener.h
- C:/Users/Sander/Documents/Projects/PCCS/phantom/src/input/KeyboardListener.cpp

## 3.36 phantom::KeyboardState Class Reference

### Public Member Functions

- void [handleEvent](#) (char id, char newValue)
- unsigned char \* [getBuffer](#) ()
- bool [isKeyDown](#) (char id)
- bool [isKeyUp](#) (char id)
- std::vector< char > \* [changes](#) ()
- std::vector< char > \* [changesUp](#) ()

### 3.36.1 Member Function Documentation

#### 3.36.1.1 `std::vector< char > * phantom::KeyboardState::changes ( )`

Returns all changes since the last update run.

**Returns**

Returns all the changes since the last update run.

**3.36.1.2** `std::vector< char > * phantom::KeyboardState::changesUp ( )`

Returns all key release changes since the last update run.

**Returns**

Returns all key release changes since the last update run.

**3.36.1.3** `unsigned char * phantom::KeyboardState::getBuffer ( )`

Returns the character buffer.

**Returns**

Returns the character buffer.

**3.36.1.4** `void phantom::KeyboardState::handleEvent ( char id, char newValue )`

Handles an keyboard event.

**Parameters**

<i>id</i>	The character id.
<i>newValue</i>	The new value associated with the key.

**3.36.1.5** `bool phantom::KeyboardState::isKeyDown ( char id )`

Returns true if the key is down.

**Returns**

Returns true if the key is down.

**Parameters**

<i>id</i>	The keycode of the button you want to check.
-----------	--

**3.36.1.6** `bool phantom::KeyboardState::isKeyUp ( char id )`

Returns true if the key is up.

**Returns**

Returns true if the key is up.

**Parameters**

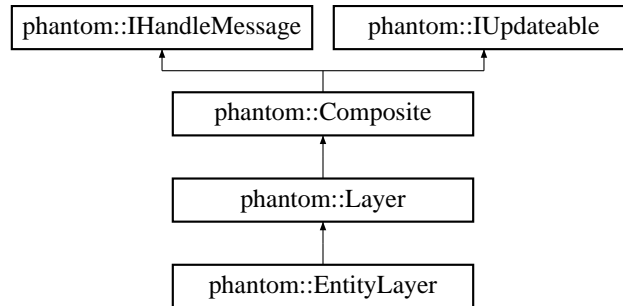
<i>id</i>	The keycode of the button you want to check.
-----------	--

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

- C:/Users/Sander/Documents/Projects/PCCS/phantom/src/input/KeyboardState.h
- C:/Users/Sander/Documents/Projects/PCCS/phantom/src/input/KeyboardState.cpp

### 3.37 phantom::Layer Class Reference

Inheritance diagram for phantom::Layer:



#### Public Member Functions

- void **addComponent** ([Composite](#) \*component)

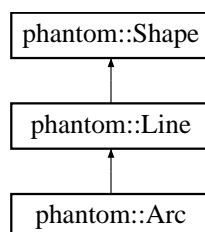
#### Additional Inherited Members

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

- C:/Users/Sander/Documents/Projects/PCCS/phantom/src/layer/Layer.h
- C:/Users/Sander/Documents/Projects/PCCS/phantom/src/layer/Layer.cpp

### 3.38 phantom::Line Class Reference

Inheritance diagram for phantom::Line:



#### Public Member Functions

- **Line** (float x, float y, float toX, float toY)
- virtual void **drawLine** (float x, float y, float toX, float toY, float offsetX, float offsetY)

### Additional Inherited Members

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

- C:/Users/Sander/Documents/Projects/PCCS/phantom/src/graphics/shapes/Line.h
- C:/Users/Sander/Documents/Projects/PCCS/phantom/src/graphics/shapes/Line.cpp

## 3.39 phantom::Line2 Class Reference

### Public Member Functions

- **Line2** (const float x1, const float y1, const float x2, const float y2)
- **Line2** (const [Vector3](#) &a, const [Vector3](#) &b)
- bool **operator==** (const [Line2](#) &v) const
- bool **operator!=** (const [Line2](#) &v) const
- [Vector3](#) **getNormal** (void) const
- [Vector3](#) **getDirection** (void) const
- [Line2](#) **projectOnto** (const [Vector3](#) &axis) const
- bool **intersects** (const [Line2](#) &other) const
- [Vector3](#) **intersection** (const [Line2](#) &other) const
- std::string **toString** (void) const

### Public Attributes

- [Vector3](#) **a**
- [Vector3](#) **b**

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

- C:/Users/Sander/Documents/Projects/PCCS/phantom/src/physics/Line2.h
- C:/Users/Sander/Documents/Projects/PCCS/phantom/src/physics/Line2.cpp

## 3.40 phantom::Vector3::MapLessComparefunctor Struct Reference

### Public Member Functions

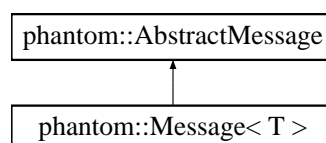
- bool **operator()** (const [Vector3](#) &a, const [Vector3](#) &b)

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

- C:/Users/Sander/Documents/Projects/PCCS/phantom/src/physics/Vector3.h

## 3.41 phantom::Message< T > Class Template Reference

Inheritance diagram for phantom::Message< T >:



## Public Member Functions

- **Message** (std::string type)
- **Message** (std::string type, T data)
- T **getData** (void)

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

- C:/Users/Sander/Documents/Projects/PCCS/phantom/src/messaging/AbstractMessage.h
- C:/Users/Sander/Documents/Projects/PCCS/phantom/src/messaging/Message.h

## 3.42 phantom::MouseState Class Reference

### Public Member Functions

- void **handleEvent** (Vector3 newValue)
- void **handleEvent** (char id, char newValue)
- Vector3 **getPosition** ()
- bool **isButtonUp** (unsigned char id)
- bool **isButtonDown** (unsigned char id)

### 3.42.1 Member Function Documentation

#### 3.42.1.1 Vector3 phantom::MouseState::getPosition ( )

Returns the current position of the mouse.

##### Returns

Returns the current position of the mouse.

#### 3.42.1.2 void phantom::MouseState::handleEvent ( Vector3 newValue )

Handle mouse event when the position is changed.

##### Parameters

<i>newValue</i>	The new position of the mouse.
-----------------	--------------------------------

#### 3.42.1.3 void phantom::MouseState::handleEvent ( char id, char newValue )

Handle mouse event when the button state is changed.

##### Parameters

<i>id</i>	The identity of the button that has changed.
<i>newValue</i>	The new value it should get.

#### 3.42.1.4 bool phantom::MouseState::isButtonDown ( unsigned char id )

Returns true if the button is up.

**Returns**

Returns true if the button is up.

**Parameters**

<i>id</i>	The keycode of the button you want to check.
-----------	--

**3.42.1.5 bool phantom::MouseState::isButtonUp ( unsigned char *id* )**

Returns true if the button is down.

**Returns**

Returns true if the button is down.

**Parameters**

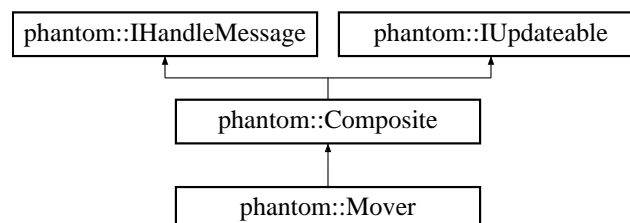
<i>id</i>	The keycode of the button you want to check.
-----------	--

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

- C:/Users/Sander/Documents/Projects/PCCS/phantom/src/input/MouseState.h
- C:/Users/Sander/Documents/Projects/PCCS/phantom/src/input/MouseState.cpp

**3.43 phantom::Mover Class Reference**

Inheritance diagram for phantom::Mover:

**Public Member Functions**

- void **setMovementSpeed** (float value)
- void **moveTo** ([Vector3](#) vector)
- void **moveTo** (const std::deque< [Vector3](#) > vList)
- virtual void **update** (const [PhantomTime](#) &time)
- virtual void **onAnsestorChanged** ()
- void **stop** ()
- bool **isStopped** ()
- [Vector3](#) **getTarget** ()
- void **pause** (double delay)
- bool **isPaused** ()

## Additional Inherited Members

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

- C:/Users/Sander/Documents/Projects/PCCS/phantom/src/physics/Mover.h
- C:/Users/Sander/Documents/Projects/PCCS/phantom/src/physics/Mover.cpp

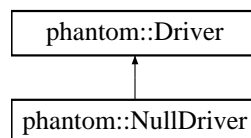
## 3.44 phantom::internal::NoPayload Struct Reference

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

- C:/Users/Sander/Documents/Projects/PCCS/phantom/src/messaging/Message.h

## 3.45 phantom::NullDriver Class Reference

Inheritance diagram for phantom::NullDriver:



## Public Member Functions

- **NullDriver** ([PhantomGame](#) \*game)
- void [onUpdate](#) ([PhantomTime](#) time)
- void [onRender](#) ()
- [Camera](#) \* [createCamera](#) ()

## Additional Inherited Members

### 3.45.1 Member Function Documentation

#### 3.45.1.1 [Camera](#)\* phantom::NullDriver::createCamera ( ) [inline],[virtual]

Creates a camera.

Implements [phantom::Driver](#).

#### 3.45.1.2 void phantom::NullDriver::onRender ( ) [inline],[virtual]

Gets called from [PhantomGame](#) to initiate the rendering.

Reimplemented from [phantom::Driver](#).

#### 3.45.1.3 void phantom::NullDriver::onUpdate ( [PhantomTime](#) time ) [inline],[virtual]

Gets called from [PhantomGame](#) to initiate the driver update loop.



## Parameters

<i>time</i>	The time generated by <a href="#">PhantomGame</a> .
-------------	---

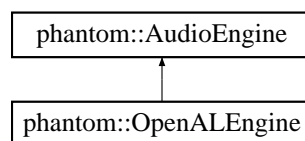
Reimplemented from [phantom::Driver](#).

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

- C:/Users/Sander/Documents/Projects/PCCS/phantom/src/null/NullDriver.h

## 3.46 phantom::OpenALEngine Class Reference

Inheritance diagram for phantom::OpenALEngine:



### Public Member Functions

- **OpenALEngine** ([PhantomGame](#) \*game)
- virtual void [createSound](#) ([SoundData](#) \*data)
- virtual void [destroySound](#) ([SoundData](#) \*data)
- virtual unsigned int [playSound](#) ([SoundData](#) \*data, const [Vector3](#) &position)
- virtual void [playMusic](#) ([SoundData](#) \*data)
- virtual void [stopSound](#) (unsigned int id)
- virtual void [stopMusic](#) ([SoundData](#) \*data)
- virtual void [setPosition](#) (const [Vector3](#) &position)

### Additional Inherited Members

#### 3.46.1 Member Function Documentation

3.46.1.1 void [phantom::OpenALEngine::createSound](#) ( [SoundData](#) \* *data* ) [virtual]

This function creates sounds.

## Parameters

<i>data</i>	The sound you want to create.
-------------	-------------------------------

Implements [phantom::AudioEngine](#).

3.46.1.2 void [phantom::OpenALEngine::destroySound](#) ( [SoundData](#) \* *data* ) [virtual]

This function destroys sounds.

## Parameters

<i>data</i>	The sound you want to destroy.
-------------	--------------------------------

Implements [phantom::AudioEngine](#).

3.46.1.3 `void phantom::OpenALEngine::playMusic ( SoundData * data ) [virtual]`

This function plays music.

Parameters

<i>data</i>	The sound data of the music file.
-------------	-----------------------------------

Implements [phantom::AudioEngine](#).

3.46.1.4 `unsigned int phantom::OpenALEngine::playSound ( SoundData * data, const Vector3 & position ) [virtual]`

This function plays a sound on a certain location. Be sure this sound is mono, else positioning won't work.

Returns

Returns the sound id.

Parameters

<i>data</i>	The sound data.
<i>position</i>	The position of the unit that's making noise.

Implements [phantom::AudioEngine](#).

3.46.1.5 `void phantom::OpenALEngine::setPosition ( const Vector3 & position ) [virtual]`

Set the position of the listener. This usually is the location of the camera.

Parameters

<i>position</i>	The position of the listener.
-----------------	-------------------------------

Implements [phantom::AudioEngine](#).

3.46.1.6 `void phantom::OpenALEngine::stopMusic ( SoundData * data ) [virtual]`

Stop playing the music using the sound data.

Parameters

<i>data</i>	The sound data.
-------------	-----------------

Implements [phantom::AudioEngine](#).

3.46.1.7 `void phantom::OpenALEngine::stopSound ( unsigned int id ) [virtual]`

Stop playing a sound using the sound id returned by play sound.

Parameters

<i>id</i>	The sound id.
-----------	---------------

Implements [phantom::AudioEngine](#).

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

- C:/Users/Sander/Documents/Projects/PCCS/phantom/src/openal/OpenALEngine.h
- C:/Users/Sander/Documents/Projects/PCCS/phantom/src/openal/OpenALEngine.cpp

## 3.47 phantom::Particle Struct Reference

### Public Attributes

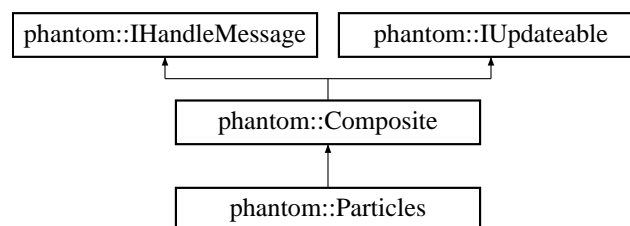
- [Vector3](#) **position**
- [Vector3](#) **velocity**
- [Vector3](#) **acceleration**
- [Vector3](#) **scale**
- [Color](#) **color**
- float **lifetime**

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

- C:/Users/Sander/Documents/Projects/PCCS/phantom/src/graphics/particles/Particle.h

## 3.48 phantom::Particles Class Reference

Inheritance diagram for phantom::Particles:



### Public Member Functions

- [Particles](#) (unsigned count=500, string texturename="", [Color](#) color=Colors::WHITE, float lifetime=1.5f, float totalLifetime=-1.0f, float speed=5.0f, [Vector3](#) scale=[Vector3](#)(5.0f, 5.0f, 1.0f), [Vector3](#) direction=[Vector3](#)(0.0f, 0.5f), float density=0.015f, unsigned randomness=200)
- void **update** (const [phantom::PhantomTime](#) &time)
- vector< [Particle](#) \* > \* [getParticles](#) ()

### Public Attributes

- float **density**
- unsigned **randomness**
- float **randomnessHalf**
- unsigned **count**
- float **lifetime**
- float **currentLifetime**
- float **totalLifetime**
- float **speed**

- [Vector3](#) **scale**
- [Vector3](#) **direction**
- [Color](#) **color**
- [ImageCachelItem](#) \* **texture**

## Additional Inherited Members

### 3.48.1 Constructor & Destructor Documentation

**3.48.1.1** `phantom::Particles::Particles ( unsigned count = 500, string texturename = "", Color color = Colors::WHITE, float lifetime = 1.5f, float totalLifetime = -1.0f, float speed = 5.0f, Vector3 scale = Vector3(5.0f, 5.0f, 1.0f), Vector3 direction = Vector3(0.0f, -0.5f), float density = 0.015f, unsigned randomness = 200 )`

This large constructor has to be called to create particles.

#### Parameters

<i>count</i>	The ammount of particles to be active at most.
<i>texturename</i>	The image you want the particles to have.
<i>color</i>	The color you want the particles to have.
<i>lifetime</i>	The lifetime of each particle.
<i>totalLifetime</i>	The lifetime of all particles. -1.0f means there is no end of life.
<i>speed</i>	The movement speed of the particles.
<i>scale</i>	The scale of the particles.
<i>direction</i>	The direction the particles should move.
<i>density</i>	The spread of the particles.
<i>randomness</i>	How random the particles should act.

### 3.48.2 Member Function Documentation

**3.48.2.1** `vector<Particle*>* phantom::Particles::getParticles ( ) [inline]`

Returns the particles that are currently active.

#### Returns

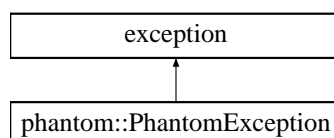
Returns the particles that are currently active.

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

- C:/Users/Sander/Documents/Projects/PCCS/phantom/src/graphics/particles/Particles.h
- C:/Users/Sander/Documents/Projects/PCCS/phantom/src/graphics/particles/Particles.cpp

## 3.49 phantom::PhantomException Class Reference

Inheritance diagram for phantom::PhantomException:



## Public Member Functions

- **PhantomException** (const string error)
- const char \* **what** () const throw ()

### 3.49.1 Member Function Documentation

#### 3.49.1.1 const char \* phantom::PhantomException::what ( ) const throw ()

Should return more information of what exactly went wrong.

#### Returns

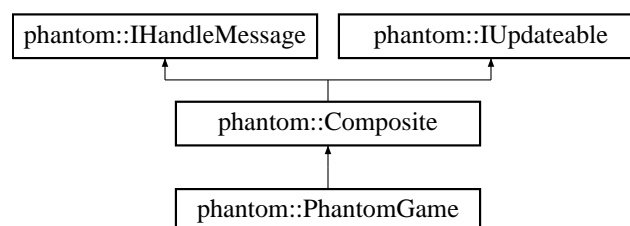
More information of what went wrong.

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

- C:/Users/Sander/Documents/Projects/PCCS/phantom/src/utlis/PhantomException.h
- C:/Users/Sander/Documents/Projects/PCCS/phantom/src/utlis/PhantomException.cpp

## 3.50 phantom::PhantomGame Class Reference

Inheritance diagram for phantom::PhantomGame:



## Public Member Functions

- **PhantomGame** (const char \*configfile)
- void **parseConfigurationFile** (const char \*configfile)
- void **pushGameState** (GameState \*state)
- void **popGameState** ()
- int **start** (int argc, char \*argv[])
- virtual void **update** (const PhantomTime &time)
- void **exit** (int returncode)
- Vector3 **getViewPort** () const
- Vector3 **getScreenSize** () const
- Vector3 **getWorldSize** () const
- void **setWorldSize** (float width, float height)
- deque< GameState \* > & **getGameStates** ()
- Driver \* **getDriver** ()
- Console \* **getConsole** ()
- void **setDriver** (Driver \*driver)
- virtual MessageState **handleMessage** (AbstractMessage \*message)

## Public Attributes

- bool **fullscreen**
- bool **mousecursor**
- float **soundvol**
- float **musicvol**

## Protected Member Functions

- virtual void **onExit** (int returncode)

## Protected Attributes

- bool **\_running**

## Friends

- class **Composite**
- class **Graphics**

### 3.50.1 Member Function Documentation

#### 3.50.1.1 void phantom::PhantomGame::exit ( int *returncode* )

Call this function when you want to exit the game.

##### Parameters

<i>returncode</i>	If everything went fine, give the returncode 0. If not enter a custom return code so you can identify where the exit call is coming from.
-------------------	---

#### 3.50.1.2 Console\* phantom::PhantomGame::getConsole ( ) [inline]

In case you need the actual console pointer, this will return it.

##### Returns

Returns the console instance.

#### 3.50.1.3 Driver\* phantom::PhantomGame::getDriver ( void )

Returns the driver this game is using.

##### Returns

Returns the driver this game is using.

#### 3.50.1.4 deque< GameState \* > & phantom::PhantomGame::getGameStates ( )

Returns the list of currently pushed gamestates.

**Returns**

Returns the list of currently pushed gamestates.

**3.50.1.5 Vector3 phantom::PhantomGame::getScreenSize ( ) const**

Returns the current screensize (resolution) of the game.

**Returns**

Returns the current screensize (resolution) of the game.

**3.50.1.6 Vector3 phantom::PhantomGame::getViewPort ( ) const**

Returns the current viewport of the game.

**Returns**

Returns the current viewport of the game.

**3.50.1.7 Vector3 phantom::PhantomGame::getWorldSize ( ) const**

Returns the current worldsize of the game.

**Returns**

Returns the current worldsize of the game.

**3.50.1.8 MessageState phantom::PhantomGame::handleMessage ( AbstractMessage \* message ) [virtual]**

Send a message across the system

**Parameters**

<i>message</i>	The message you want to send.
----------------	-------------------------------

Reimplemented from [phantom::Composite](#).

**3.50.1.9 void phantom::PhantomGame::parseConfigurationFile ( const char \* configfile )**

Parse the configuration file and set all the game parameters.

**Parameters**

<i>configfile</i>	The location of the configuration file.
-------------------	---

**3.50.1.10 void phantom::PhantomGame::popGameState ( )**

Pop the top most gamestate.

#### 3.50.1.11 void phantom::PhantomGame::pushGameState ( GameState \* state )

Push a gamestate.

##### Parameters

<i>state</i>	The gamestate you want to push on the stack.
--------------	--

#### 3.50.1.12 void phantom::PhantomGame::setDriver ( Driver \* driver )

Sets the driver of the game.

##### Parameters

<i>driver</i>	The new driver you want to set.
---------------	---------------------------------

#### 3.50.1.13 void phantom::PhantomGame::setWorldSize ( float width, float height )

Change the current worldsize of the game.

##### Parameters

<i>width</i>	The width of the playing field.
<i>height</i>	The height of the playing field.

#### 3.50.1.14 int phantom::PhantomGame::start ( int argc, char \* argv[] )

After initializing all your initial data, call this function to actually start the game. In this function is a while loop located. To break out of it, call the exit function somewhere in the update loops of your game.

#### 3.50.1.15 void phantom::PhantomGame::update ( const PhantomTime & time ) [virtual]

This function gets called every loop.

##### Parameters

<i>time</i>	The time generated in the start function.
-------------	---

Reimplemented from [phantom::Composite](#).

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

- C:/Users/Sander/Documents/Projects/PCCS/phantom/src/core/PhantomGame.h
- C:/Users/Sander/Documents/Projects/PCCS/phantom/src/core/PhantomGame.cpp

## 3.51 phantom::PhantomTime Class Reference

### Public Member Functions

- **PhantomTime** (float elapsed, float totalGameTime, double currentTime)
- float [getElapsed](#) () const
- float [getTotalGameTime](#) () const
- double [getTime](#) () const



### 3.51.1 Member Function Documentation

#### 3.51.1.1 float phantom::PhantomTime::getElapsed ( ) const

Returns the elapsed time since last update call.

##### Returns

Returns the elapsed time since last update call.

#### 3.51.1.2 double phantom::PhantomTime::getTime ( ) const

Returns the time since epoch.

##### Returns

Returns the time since epoch.

#### 3.51.1.3 float phantom::PhantomTime::getTotalGameTime ( ) const

Returns the total time the game is active.

##### Returns

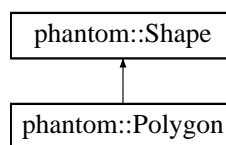
Returns the total time the game is active.

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

- C:/Users/Sander/Documents/Projects/PCCS/phantom/src/utils/Time.h
- C:/Users/Sander/Documents/Projects/PCCS/phantom/src/utils/Time.cpp

## 3.52 phantom::Polygon Class Reference

Inheritance diagram for phantom::Polygon:



### Public Member Functions

- void [addPoint](#) (float x, float y)

### Public Attributes

- vector< [Vertice](#) > **collection**

## Additional Inherited Members

### 3.52.1 Member Function Documentation

#### 3.52.1.1 void phantom::Polygon::addPoint ( float x, float y )

Adds a point in the polygon to draw a line next to.

##### Parameters

x	Location of the next vertex ( X-coordinate ).
y	Location of the next vertex ( Y-coordinate ).

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

- C:/Users/Sander/Documents/Projects/PCCS/phantom/src/graphics/shapes/Polygon.h
- C:/Users/Sander/Documents/Projects/PCCS/phantom/src/graphics/shapes/Polygon.cpp

## 3.53 phantom::Polygon2 Class Reference

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

- C:/Users/Sander/Documents/Projects/PCCS/phantom/src/physics/Polygon2.h
- C:/Users/Sander/Documents/Projects/PCCS/phantom/src/physics/Polygon2.cpp

## 3.54 phantom::Projection Class Reference

### Public Types

- typedef std::deque< [Vector3](#) > **Group**

### Static Public Member Functions

- static [Line2](#) **project** (const [Vector3](#) axis, const Group &vertices)
- static bool **projectedLineIntersection** (const [Line2](#) &him, const [Line2](#) &her)

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

- C:/Users/Sander/Documents/Projects/PCCS/phantom/src/physics/Projection.h

## 3.55 phantom::Pulse Struct Reference

### Public Member Functions

- **Pulse** ([Vector3](#) \_direction, float \_speed, float \_friction=0.0)
- string **toString** (void)

### Public Attributes

- [Vector3](#) **direction**
- float **speed**
- float **friction**
- char **weight**

### Static Public Attributes

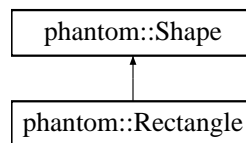
- static const int **FOREVER** = -1

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

- C:/Users/Sander/Documents/Projects/PCCS/phantom/src/physics/Pulse.h

## 3.56 phantom::Rectangle Class Reference

Inheritance diagram for phantom::Rectangle:



### Public Member Functions

- **Rectangle** (float x, float y, float width, float height, bool isFilled, float thickness)

### Public Attributes

- float **width**
- float **height**
- float **thickness**

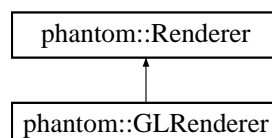
### Additional Inherited Members

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

- C:/Users/Sander/Documents/Projects/PCCS/phantom/src/graphics/shapes/Rectangle.h
- C:/Users/Sander/Documents/Projects/PCCS/phantom/src/graphics/shapes/Rectangle.cpp

## 3.57 phantom::Renderer Class Reference

Inheritance diagram for phantom::Renderer:



### Public Member Functions

- **Renderer** ([PhantomGame](#) \*game)
- virtual void [renderLoop](#) ()=0

- virtual void [buildShape](#) ([Shape](#) \*shape)=0
- virtual void [destroyShape](#) ([Shape](#) \*shape)=0
- virtual void [addTexture](#) ([ImageCachelItem](#) \*item, bool isText=false)=0
- virtual void [removeTexture](#) ([ImageCachelItem](#) \*item)=0

## Protected Attributes

- [PhantomGame](#) \* [\\_game](#)

## 3.57.1 Member Function Documentation

3.57.1.1 virtual void phantom::Renderer::addTexture ( [ImageCachelItem](#) \* *item*, bool *isText* = false ) [pure virtual]

This gets called when a texture has to be added to the graphics pipeline.

### Parameters

<i>item</i>	The imache cache item that has to be created.
<i>isText</i>	The fonts are converted to images as well. Set this to true to have an optimization for font rendering.

Implemented in [phantom::GLRenderer](#).

3.57.1.2 virtual void phantom::Renderer::buildShape ( [Shape](#) \* *shape* ) [pure virtual]

This gets called by the [Graphics](#) class. This makes it possible to build VBO's for example.

### Parameters

<i>shape</i>	The shape you want to build.
--------------	------------------------------

Implemented in [phantom::GLRenderer](#).

3.57.1.3 virtual void phantom::Renderer::destroyShape ( [Shape](#) \* *shape* ) [pure virtual]

This gets called by the [Graphics](#) class. This makes it possible to destroy VBO's for example.

### Parameters

<i>shape</i>	The shape you want to destroy.
--------------	--------------------------------

Implemented in [phantom::GLRenderer](#).

3.57.1.4 virtual void phantom::Renderer::removeTexture ( [ImageCachelItem](#) \* *item* ) [pure virtual]

This gets called when the texture has to be removed from the graphics pipeline.

### Parameters

<i>item</i>	The item you want to remove from the cache.
-------------	---

Implemented in [phantom::GLRenderer](#).

## 3.57.1.5 virtual void phantom::Renderer::renderLoop ( ) [pure virtual]

This is called every loop by the [Driver](#) class.

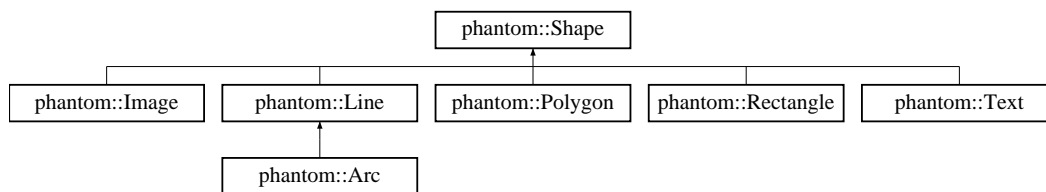
Implemented in [phantom::GLRenderer](#).

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

- C:/Users/Sander/Documents/Projects/PCCS/phantom/src/core/Renderer.h

## 3.58 phantom::Shape Class Reference

Inheritance diagram for phantom::Shape:



### Public Member Functions

- void [setFillColor](#) ([Color](#) color)
- void [setLineColor](#) ([Color](#) color)
- virtual void [addVertex](#) (float x, float y, float u=TEX\_COORD\_UNUSED, float v=TEX\_COORD\_UNUSED)
- void [buildShape](#) ([Renderer](#) \*renderer)
- void [destroyShape](#) ([Renderer](#) \*renderer)
- bool [hasFillColor](#) (void)
- bool [hasLineColor](#) (void)
- const [Color](#) & [getLineColor](#) ()
- const [Color](#) & [getFillColor](#) ()
- const [Box3](#) & [getBounds](#) ()

### Static Public Member Functions

- static unsigned [getShapecount](#) ()

### Public Attributes

- std::vector< [Vertice](#) > **vertices**
- [Vertice](#) \* **verticesArray**
- std::vector< [TexCoord](#) > **texCoords**
- [TexCoord](#) \* **texCoordsArray**
- unsigned int **verticesCount**
- unsigned int **vboVertices**
- unsigned int **vboTexCoords**
- bool **isImage**
- bool **isText**
- float **x**
- float **y**

## Static Public Attributes

- static const float **TEX\_COORD\_UNUSED** = -999.0f

## Protected Attributes

- [Box3](#) **\_bounds**

## 3.58.1 Member Function Documentation

**3.58.1.1** void `phantom::Shape::addVertex ( float x, float y, float u = TEX_COORD_UNUSED, float v = TEX_COORD_UNUSED )` [virtual]

Add a vertex to the shape. The renderer will iterate through all the vertices to draw them.

### Parameters

<i>x</i>	The x location of the vertex
<i>y</i>	The y location of the vertex
<i>u</i>	The u texture coordinate associated with the vertex
<i>v</i>	The v texture coordinate associated with the vertex

**3.58.1.2** void `phantom::Shape::buildShape ( Renderer * renderer )`

When the shape gets added to the [Graphics](#) class the graphics class calls this function. This function then calls the renderer to build VBO's for example.

### Parameters

<i>renderer</i>	The renderer class.
-----------------	---------------------

**3.58.1.3** void `phantom::Shape::destroyShape ( Renderer * renderer )`

When the shape gets removed from the [Graphics](#) class the graphics class calls this function. This function then calls the renderer to destroy VBO's for example.

### Parameters

<i>renderer</i>	The renderer class.
-----------------	---------------------

**3.58.1.4** const [Box3](#) & `phantom::Shape::getBounds ( )`

Returns the size of the shape.

### Returns

Returns the size of the shape.

**3.58.1.5** const [Color](#) & `phantom::Shape::getFillColor ( )`

Returns the line color of the shape.

**Returns**

Returns the line color of the shape.

**3.58.1.6 const Color & phantom::Shape::getLineColor ( )**

Returns the line color of the shape.

**Returns**

Returns the line color of the shape.

**3.58.1.7 unsigned phantom::Shape::getShapecount ( ) [static]**

Returns the total shapes that exist in the game.

**Returns**

Returns the total shapes that exist in the game.

**3.58.1.8 bool phantom::Shape::hasFillColor ( void )**

Returns true if the shape has a fill color.

**Returns**

Returns true if the shape has a fill color.

**3.58.1.9 bool phantom::Shape::hasLineColor ( void )**

Returns true if the shape has a line color.

**Returns**

Returns true if the shape has a fill color.

**3.58.1.10 void phantom::Shape::setFillColor ( Color color )**

Set the fill color of the shape.

**Parameters**

<i>color</i>	The color you want to use for the shape.
--------------	--

**3.58.1.11 void phantom::Shape::setLineColor ( Color color )**

Set the line color of the shape.

**Parameters**

<i>color</i>	The color you want to use for the shape.
--------------	--

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

- C:/Users/Sander/Documents/Projects/PCCS/phantom/src/graphics/shapes/Shape.h
- C:/Users/Sander/Documents/Projects/PCCS/phantom/src/graphics/shapes/Shape.cpp

### 3.59 phantom::SoundData Class Reference

#### Public Attributes

- int **state**
- unsigned int **bufferID**
- std::deque< unsigned int > **sourceID**
- int **format**
- int **freq**
- int **channels**
- int **bitsPerSample**
- std::vector< char > \* **bufferData**

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

- C:/Users/Sander/Documents/Projects/PCCS/phantom/src/audio/SoundData.h
- C:/Users/Sander/Documents/Projects/PCCS/phantom/src/audio/SoundData.cpp

### 3.60 phantom::SoundLoader Class Reference

#### Static Public Member Functions

- static void [loadVorbis](#) (const char \*file, [SoundData](#) \*data)

#### 3.60.1 Member Function Documentation

3.60.1.1 void phantom::SoundLoader::loadVorbis ( const char \* *file*, [SoundData](#) \* *data* ) [static]

This function loads an OGG Vorbis file into a [SoundData](#) object.

#### Parameters

<i>file</i>	The location of the ogg file.
<i>data</i>	The <a href="#">SoundData</a> object you want to load the data in.

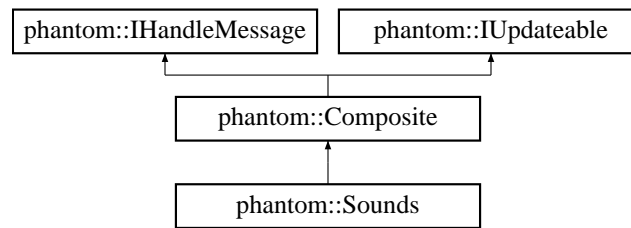
The documentation for this class was generated from the following files:

- C:/Users/Sander/Documents/Projects/PCCS/phantom/src/audio/SoundLoader.h
- C:/Users/Sander/Documents/Projects/PCCS/phantom/src/audio/SoundLoader.cpp

### 3.61 phantom::Sounds Class Reference

Inheritance diagram for phantom::Sounds:





## Public Member Functions

- int [playSound](#) (const string &filename, const [Vector3](#) &position)
- bool [stopSound](#) (int id)
- int [playMusic](#) (const string &filename)
- bool [stopMusic](#) (const string &filename)

## Additional Inherited Members

### 3.61.1 Member Function Documentation

#### 3.61.1.1 int phantom::Sounds::playMusic ( const string & filename )

The function playMusic should be called when you want to play music gapless in looping mode.

##### Parameters

<i>filename</i>	The location of the music file.
-----------------	---------------------------------

#### 3.61.1.2 int phantom::Sounds::playSound ( const string & filename, const [Vector3](#) & position )

The function playSound should be called by the game when an audio file has to be played.

##### Parameters

<i>filename</i>	The location of the file.
<i>position</i>	The position of the object on the map unrelative to the listener.

#### 3.61.1.3 bool phantom::Sounds::stopMusic ( const string & filename )

The function stopMusic should be called when you want to stop music.

##### Parameters

<i>filename</i>	The location of the music file.
-----------------	---------------------------------

#### 3.61.1.4 bool phantom::Sounds::stopSound ( int id )

The function stopSound should be called when you want to stop the sound from playing.

##### Parameters

<i>id</i>	The sound id that has been returned by the playSound function.
-----------	--

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

- C:/Users/Sander/Documents/Projects/PCCS/phantom/src/audio/Sounds.h
- C:/Users/Sander/Documents/Projects/PCCS/phantom/src/audio/Sounds.cpp

### 3.62 phantom::TexCoord Class Reference

#### Public Attributes

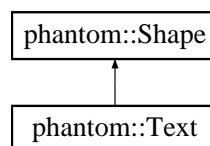
- float **u**
- float **v**

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

- C:/Users/Sander/Documents/Projects/PCCS/phantom/src/graphics/VerticeData.h

### 3.63 phantom::Text Class Reference

Inheritance diagram for phantom::Text:



#### Public Member Functions

- **Text** (float x, float y, unsigned int size, const char \*font, const char \*text)
- void **genVertices** (const char \*text, FreeTypeFont \*font)

#### Public Attributes

- const char \* **text**
- const char \* **font**
- FreeTypeFont \* **ftfont**
- unsigned int **size**

#### Additional Inherited Members

#### 3.63.1 Member Function Documentation

##### 3.63.1.1 void phantom::Text::genVertices ( const char \* *text*, FreeTypeFont \* *font* )

Generate the vertices required for the text.

#### Parameters

<i>text</i>	The text you want to display.
<i>font</i>	The font you want to use.

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

- C:/Users/Sander/Documents/Projects/PCCS/phantom/src/graphics/shapes/Text.h
- C:/Users/Sander/Documents/Projects/PCCS/phantom/src/graphics/shapes/Text.cpp

## 3.64 phantom::Timer Class Reference

### Public Member Functions

- **Timer** (double delay)
- bool **hasExpired** (const [PhantomTime](#) &time)
- bool **hasExpired** ()
- [Timer](#) & **stop** ()
- [Timer](#) & **restart** ()
- [Timer](#) & **setDelay** (double delay)
- bool **isStopped** ()
- void **expire** (void)

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

- C:/Users/Sander/Documents/Projects/PCCS/phantom/src/utis/Timer.h

## 3.65 phantom::Util Class Reference

### Static Public Member Functions

- static void [readfile](#) (const char \*filelocation, char \*\*filecontent, unsigned int \*length)
- static double [getTime](#) ()

### 3.65.1 Member Function Documentation

#### 3.65.1.1 double phantom::Util::getTime ( ) [static]

Returns the time since epoch.

#### Returns

Returns the time since epoch.

#### 3.65.1.2 void phantom::Util::readfile ( const char \* filelocation, char \*\* filecontent, unsigned int \* length ) [static]

Read a file into a buffer.

#### Parameters

<i>filelocation</i>	The location of the file relative to the working directory.
<i>filecontent</i>	A pointer to the buffer where the content of the file should be written to.
<i>length</i>	A pointer to the buffer where the length of the file should be written to.

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

- C:/Users/Sander/Documents/Projects/PCCS/phantom/src/utis/util.h

- C:/Users/Sander/Documents/Projects/PCCS/phantom/src/utls/util.cpp

### 3.66 phantom::Vector3 Class Reference

#### Classes

- struct [MapLessComparefunctor](#)

#### Public Member Functions

- **Vector3** (float x, float y, float z=0.0f)
- **Vector3** (int x, int y, int z=0)
- **Vector3** (double x, double y, double z=0.0)
- **Vector3** (const [Vector3](#) &origin)
- **Vector3 operator+** (const [Vector3](#) &v) const
- **Vector3 operator-** (const [Vector3](#) &v) const
- **Vector3 operator%** (const [Vector3](#) &b) const
- **Vector3 operator/** (const [Vector3](#) &v) const
- **Vector3 operator\*** (const [Vector3](#) &v) const
- **Vector3 operator\*** (float f) const
- **Vector3 & operator+=** (const [Vector3](#) &v)
- **Vector3 & operator+=** (const float &v)
- **Vector3 & operator-=** (const [Vector3](#) &v)
- **Vector3 & operator\*=** (const [Vector3](#) &v)
- **Vector3 & operator\*=** (const float &v)
- **Vector3 & operator/=** (const [Vector3](#) &v)
- **Vector3 & operator/=** (const float &v)
- bool **operator==** (const [Vector3](#) &v) const
- bool **operator!=** (const [Vector3](#) &v) const
- bool **isFinite** (void) const
- bool **isInfinite** (void) const
- **Vector3 & absolute** ()
- **Vector3 & normalize** ()
- **Vector3 & reverse** ()
- **Vector3 directionTo** (const [Vector3](#) &other) const
- float **distanceTo** (const [Vector3](#) &other) const
- float **distanceToSq** (const [Vector3](#) &other) const
- float **distanceToSq** (const [Vector3](#) \*other) const
- float **getLengthSq** (void) const
- float **dot** (const [Vector3](#) &v) const
- float **getAngleXOY** () const
- **Vector3 perp** (void) const
- **Vector3 cross** (const [Vector3](#) &b) const
- **Vector3 projectOnto** (const [Vector3](#) &b) const
- std::string **toString** () const
- std::string **toString2** () const

#### Public Attributes

- float **x**
- float **y**
- float **z**

## Friends

- `std::ostream & operator<< (std::ostream &o, const Vector3 &v)`

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

- `C:/Users/Sander/Documents/Projects/PCCS/phantom/src/physics/Vector3.h`
- `C:/Users/Sander/Documents/Projects/PCCS/phantom/src/physics/Vector3.cpp`

## 3.67 phantom::Vertice Class Reference

### Public Attributes

- `float x`
- `float y`
- `float z`

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

- `C:/Users/Sander/Documents/Projects/PCCS/phantom/src/graphics/VerticeData.h`

# Index

- addComponent
  - phantom::Entity, [19](#)
- addLog
  - phantom::Console, [15](#)
- addPoint
  - phantom::Polygon, [56](#)
- addTexture
  - phantom::GLRenderer, [25](#)
  - phantom::Renderer, [58](#)
- addVertex
  - phantom::Shape, [60](#)
- arc
  - phantom::Graphics, [27](#)
- beginPath
  - phantom::Graphics, [27](#)
- buildShape
  - phantom::GLRenderer, [25](#)
  - phantom::Renderer, [58](#)
  - phantom::Shape, [60](#)
- changes
  - phantom::KeyboardState, [40](#)
- changesUp
  - phantom::KeyboardState, [41](#)
- clear
  - phantom::Graphics, [27](#)
- createCamera
  - phantom::Driver, [17](#)
  - phantom::GLDriver, [24](#)
  - phantom::NullDriver, [46](#)
- createPNG
  - phantom::ImageLoader, [35](#)
- createSound
  - phantom::AudioEngine, [6](#)
  - phantom::OpenALEngine, [47](#)
- destroyShape
  - phantom::GLRenderer, [25](#)
  - phantom::Renderer, [58](#)
  - phantom::Shape, [60](#)
- destroySound
  - phantom::AudioEngine, [6](#)
  - phantom::OpenALEngine, [47](#)
- directionTo
  - phantom::Entity, [20](#)
- disableCamera
  - phantom::Driver, [17](#)
- distanceTo
  - phantom::Entity, [20](#)
- distanceToSq
  - phantom::Entity, [20](#)
- enableCamera
  - phantom::Driver, [17](#)
- exit
  - phantom::PhantomGame, [52](#)
- fill
  - phantom::Graphics, [27](#)
- genVertices
  - phantom::Text, [64](#)
- getActiveCameras
  - phantom::Driver, [17](#)
- getAudio
  - phantom::Driver, [18](#)
- getAudioEngine
  - phantom::Driver, [18](#)
- getBounds
  - phantom::Shape, [60](#)
- getBuffer
  - phantom::KeyboardState, [41](#)
- getCameraID
  - phantom::Camera, [11](#)
- getConsole
  - phantom::PhantomGame, [52](#)
- getDriver
  - phantom::PhantomGame, [52](#)
- getElapsed
  - phantom::PhantomTime, [55](#)
- getFillColor
  - phantom::Shape, [60](#)
- getFont
  - phantom::FreeTypeLibrary, [22](#)
- getFontLibrary
  - phantom::Driver, [18](#)
- getFromCache
  - phantom::ImageCache, [33](#)
- getGameStates
  - phantom::PhantomGame, [52](#)
- getImage
  - phantom::Image, [33](#)
- getInput
  - phantom::Driver, [18](#)
- getKeyboardState
  - phantom::Input, [36](#)
- getLineColor
  - phantom::Shape, [61](#)
- getMouseState

- phantom::Input, 36
- getParticles
  - phantom::Particles, 50
- getPosition
  - phantom::MouseState, 44
- getRenderer
  - phantom::Driver, 18
- getRotation
  - phantom::Camera, 11
  - phantom::Graphics, 27
- getScreenSize
  - phantom::Camera, 11
  - phantom::PhantomGame, 53
- getShapecount
  - phantom::Shape, 61
- getTime
  - phantom::PhantomTime, 55
  - phantom::Util, 65
- getTotalGameTime
  - phantom::PhantomTime, 55
- getViewPort
  - phantom::Camera, 11
  - phantom::PhantomGame, 53
- getWorldCoordinates
  - phantom::Camera, 11
- getWorldSize
  - phantom::PhantomGame, 53
- handleEvent
  - phantom::KeyboardState, 41
  - phantom::MouseState, 44
- handleMessage
  - phantom::PhantomGame, 53
- hasFillColor
  - phantom::Shape, 61
- hasLineColor
  - phantom::Shape, 61
- image
  - phantom::Graphics, 27
- insertIntoCache
  - phantom::ImageCache, 33
- isActive
  - phantom::Camera, 11
- isButtonDown
  - phantom::MouseState, 44
- isButtonUp
  - phantom::MouseState, 45
- isCached
  - phantom::ImageCache, 33
- isKeyDown
  - phantom::KeyboardState, 41
- isKeyUp
  - phantom::KeyboardState, 41
- isLocked
  - phantom::KeyboardListener, 40
- keyboard
  - phantom::InputField, 37
- line
  - phantom::Graphics, 28
- lineTo
  - phantom::Graphics, 29
- loadVorbis
  - phantom::SoundLoader, 62
- lock
  - phantom::KeyboardListener, 40
- log
  - phantom::Console, 15, 16
- mapCommand
  - phantom::Console, 16
- moveTo
  - phantom::Graphics, 29
- onClick
  - phantom::Button, 9
  - phantom::InputField, 37
- onRender
  - phantom::Driver, 18
  - phantom::NullDriver, 46
- onUnClicked
  - phantom::InputField, 37
- onUpdate
  - phantom::Driver, 18
  - phantom::NullDriver, 46
- paint
  - phantom::Button, 9
  - phantom::InputField, 37
- paintText
  - phantom::InputField, 38
- parseConfigurationFile
  - phantom::PhantomGame, 53
- Particles
  - phantom::Particles, 50
- phantom::AbstractMessage, 5
- phantom::Arc, 5
- phantom::AudioEngine, 6
  - createSound, 6
  - destroySound, 6
  - playMusic, 6
  - playSound, 7
  - setPosition, 7
  - stopMusic, 7
  - stopSound, 7
- phantom::Box3, 8
- phantom::Button, 8
  - onClick, 9
  - paint, 9
  - setText, 9
  - text, 9
  - update, 10
- phantom::Camera, 10
  - getCameraID, 11
  - getRotation, 11
  - getScreenSize, 11
  - getViewPort, 11

- getWorldCoordinates, 11
- isActive, 11
- setParams, 11
- setRotation, 11
- setScreenSize, 12
- setViewPort, 12
- phantom::CollisionData, 12
- phantom::Color, 13
- phantom::Composite, 13
- phantom::Console, 15
  - addLog, 15
  - log, 15, 16
  - mapCommand, 16
  - update, 16
- phantom::Driver, 16
  - createCamera, 17
  - disableCamera, 17
  - enableCamera, 17
  - getActiveCameras, 17
  - getAudio, 18
  - getAudioEngine, 18
  - getFontLibrary, 18
  - getInput, 18
  - getRenderer, 18
  - onRender, 18
  - onUpdate, 18
  - setWindowTitle, 18
- phantom::Entity, 19
  - addComponent, 19
  - directionTo, 20
  - distanceTo, 20
  - distanceToSq, 20
- phantom::EntityLayer, 20
- phantom::FreeTypeFont, 21
- phantom::FreeTypeFont::char\_info\_t, 12
- phantom::FreeTypeFont::font\_info\_t, 21
- phantom::FreeTypeLibrary, 22
  - getFont, 22
- phantom::GLCamera, 23
  - setParams, 23
- phantom::GLDriver, 23
  - createCamera, 24
  - setWindowTitle, 24
- phantom::GLRenderer, 24
  - addTexture, 25
  - buildShape, 25
  - destroyShape, 25
  - removeTexture, 25
  - renderLoop, 25
- phantom::GLUTInput, 26
- phantom::GameState, 22
- phantom::Graphics, 26
  - arc, 27
  - beginPath, 27
  - clear, 27
  - fill, 27
  - getRotation, 27
  - image, 27
  - line, 28
  - lineTo, 29
  - moveTo, 29
  - rect, 29
  - rotate, 30
  - setFillStyle, 30
  - setLineStyle, 30
  - stroke, 30
  - text, 31
- phantom::IHandleMessage, 31
- phantom::IUpdateable, 38
- phantom::Image, 32
  - getImage, 33
- phantom::ImageCache, 33
  - getFromCache, 33
  - insertIntoCache, 33
  - isCached, 33
  - removeFromCache, 34
  - setRenderer, 34
- phantom::ImageCacheItem, 34
- phantom::ImageLoader, 34
  - createPNG, 35
- phantom::InertiaMover, 35
- phantom::Input, 36
  - getKeyboardState, 36
  - getMouseState, 36
- phantom::InputField, 36
  - keyboard, 37
  - onClick, 37
  - onUnClicked, 37
  - paint, 37
  - paintText, 38
  - text, 38
  - update, 38
- phantom::KeyboardListener, 39
  - isLocked, 40
  - lock, 40
  - unlock, 40
- phantom::KeyboardState, 40
  - changes, 40
  - changesUp, 41
  - getBuffer, 41
  - handleEvent, 41
  - isKeyDown, 41
  - isKeyUp, 41
- phantom::Layer, 42
- phantom::Line, 42
- phantom::Line2, 43
- phantom::Message< T >, 43
- phantom::MouseState, 44
  - getPosition, 44
  - handleEvent, 44
  - isButtonDown, 44
  - isButtonUp, 45
- phantom::Mover, 45
- phantom::NullDriver, 46
  - createCamera, 46
  - onRender, 46



- onUpdate, 46
- phantom::OpenALEngine, 47
  - createSound, 47
  - destroySound, 47
  - playMusic, 47
  - playSound, 48
  - setPosition, 48
  - stopMusic, 48
  - stopSound, 48
- phantom::Particle, 49
- phantom::Particles, 49
  - getParticles, 50
  - Particles, 50
- phantom::PhantomException, 50
  - what, 51
- phantom::PhantomGame, 51
  - exit, 52
  - getConsole, 52
  - getDriver, 52
  - getGameStates, 52
  - getScreenSize, 53
  - getViewPort, 53
  - getWorldSize, 53
  - handleMessage, 53
  - parseConfigurationFile, 53
  - popGameState, 53
  - pushGameState, 53
  - setDriver, 54
  - setWorldSize, 54
  - start, 54
  - update, 54
- phantom::PhantomTime, 54
  - getElapsed, 55
  - getTime, 55
  - getTotalGameTime, 55
- phantom::Polygon, 55
  - addPoint, 56
- phantom::Polygon2, 56
- phantom::Projection, 56
- phantom::Pulse, 56
- phantom::Rectangle, 57
- phantom::Renderer, 57
  - addTexture, 58
  - buildShape, 58
  - destroyShape, 58
  - removeTexture, 58
  - renderLoop, 58
- phantom::Shape, 59
  - addVertex, 60
  - buildShape, 60
  - destroyShape, 60
  - getBounds, 60
  - getFillColor, 60
  - getLineColor, 61
  - getShapecount, 61
  - hasFillColor, 61
  - hasLineColor, 61
  - setFillColor, 61
  - setLineColor, 61
- phantom::SoundData, 62
- phantom::SoundLoader, 62
  - loadVorbis, 62
- phantom::Sounds, 62
  - playMusic, 63
  - playSound, 63
  - stopMusic, 63
  - stopSound, 63
- phantom::TexCoord, 64
- phantom::Text, 64
  - genVertices, 64
- phantom::Timer, 65
- phantom::Util, 65
  - getTime, 65
  - readfile, 65
- phantom::Vector3, 66
- phantom::Vector3::MapLessComparefunctor, 43
- phantom::Vertice, 67
- phantom::internal::NoPayload, 46
- phantom::tree::BinaryNode, 7
- phantom::tree::BinaryTree, 8
- playMusic
  - phantom::AudioEngine, 6
  - phantom::OpenALEngine, 47
  - phantom::Sounds, 63
- playSound
  - phantom::AudioEngine, 7
  - phantom::OpenALEngine, 48
  - phantom::Sounds, 63
- popGameState
  - phantom::PhantomGame, 53
- pushGameState
  - phantom::PhantomGame, 53
- readfile
  - phantom::Util, 65
- rect
  - phantom::Graphics, 29
- removeFromCache
  - phantom::ImageCache, 34
- removeTexture
  - phantom::GLRenderer, 25
  - phantom::Renderer, 58
- renderLoop
  - phantom::GLRenderer, 25
  - phantom::Renderer, 58
- rotate
  - phantom::Graphics, 30
- setDriver
  - phantom::PhantomGame, 54
- setFillColor
  - phantom::Shape, 61
- setFillStyle
  - phantom::Graphics, 30
- setLineColor
  - phantom::Shape, 61
- setLineStyle

- phantom::Graphics, [30](#)
- setParams
  - phantom::Camera, [11](#)
  - phantom::GLCamera, [23](#)
- setPosition
  - phantom::AudioEngine, [7](#)
  - phantom::OpenALEngine, [48](#)
- setRenderer
  - phantom::ImageCache, [34](#)
- setRotation
  - phantom::Camera, [11](#)
- setScreenSize
  - phantom::Camera, [12](#)
- setText
  - phantom::Button, [9](#)
- setViewPort
  - phantom::Camera, [12](#)
- setWindowTitle
  - phantom::Driver, [18](#)
  - phantom::GLDriver, [24](#)
- setWorldSize
  - phantom::PhantomGame, [54](#)
- start
  - phantom::PhantomGame, [54](#)
- stopMusic
  - phantom::AudioEngine, [7](#)
  - phantom::OpenALEngine, [48](#)
  - phantom::Sounds, [63](#)
- stopSound
  - phantom::AudioEngine, [7](#)
  - phantom::OpenALEngine, [48](#)
  - phantom::Sounds, [63](#)
- stroke
  - phantom::Graphics, [30](#)
- text
  - phantom::Button, [9](#)
  - phantom::Graphics, [31](#)
  - phantom::InputField, [38](#)
- unlock
  - phantom::KeyboardListener, [40](#)
- update
  - phantom::Button, [10](#)
  - phantom::Console, [16](#)
  - phantom::InputField, [38](#)
  - phantom::PhantomGame, [54](#)
- what
  - phantom::PhantomException, [51](#)