

Lab Raid

Generated by Doxygen 1.10.0

1 Namespace Index	1
1.1 Namespace List	1
2 Hierarchical Index	3
2.1 Class Hierarchy	3
3 Class Index	5
3.1 Class List	5
4 File Index	7
4.1 File List	7
5 Namespace Documentation	9
5.1 Commands Namespace Reference	9
5.2 Config Namespace Reference	9
5.2.1 Variable Documentation	9
5.2.1.1 backgroundColor	9
5.2.1.2 framerate	9
5.2.1.3 gameTitle	10
5.2.1.4 holdTimeThreshold	10
5.2.1.5 screenHeight	10
5.2.1.6 screenType	10
5.2.1.7 screenWidth	10
5.2.1.8 volume	10
5.3 Functions Namespace Reference	10
5.4 Global Namespace Reference	10
5.4.1 Function Documentation	11
5.4.1.1 init()	11
5.4.2 Variable Documentation	11
5.4.2.1 arrowObject1	11
5.4.2.2 arrowObject2	11
5.4.2.3 blueCircle	11
5.4.2.4 crosshairCircle1	11
5.4.2.5 crosshairLine1	12
5.4.2.6 crosshairLine2	12
5.4.2.7 fpsManager	12
5.4.2.8 greenCircle	12
5.4.2.9 hollowCircle1	12
5.4.2.10 hudView	12
5.4.2.11 line1	12
5.4.2.12 line2	12
5.4.2.13 line3	12
5.4.2.14 line4	12
5.4.2.15 menuView	13

5.4.2.16 playerCamera	13
5.4.2.17 playerObject	13
5.4.2.18 purpleCircle	13
5.4.2.19 redCircle	13
5.4.2.20 yellowCircle	13
5.5 Items Namespace Reference	13
5.6 Objects Namespace Reference	13
5.7 Shapes Namespace Reference	14
5.8 Views Namespace Reference	14
5.8.1 Variable Documentation	14
5.8.1.1 INIT_VIEW_HEIGHT	14
5.8.1.2 INIT_VIEW_WIDTH	14
6 Class Documentation	15
6.1 Objects::Button Class Reference	15
6.1.1 Constructor & Destructor Documentation	16
6.1.1.1 Button()	16
6.1.2 Member Function Documentation	16
6.1.2.1 onClick()	16
6.1.2.2 setHovered()	16
6.1.2.3 update()	17
6.2 Views::Camera Class Reference	17
6.2.1 Detailed Description	18
6.2.2 Constructor & Destructor Documentation	19
6.2.2.1 Camera()	19
6.2.3 Member Function Documentation	19
6.2.3.1 getAngle()	19
6.2.3.2 getRect()	19
6.2.3.3 getZoom()	19
6.2.3.4 rotate()	20
6.2.3.5 setAngle()	20
6.2.3.6 setDimension()	20
6.2.3.7 setPivotObject()	20
6.2.3.8 setPosition()	21
6.2.3.9 setZoom()	21
6.2.3.10 transform()	21
6.2.3.11 transformFromRender()	21
6.3 Shapes::Circle Class Reference	22
6.3.1 Constructor & Destructor Documentation	24
6.3.1.1 Circle()	24
6.3.2 Member Function Documentation	24
6.3.2.1 draw()	24

6.3.2.2 <code>setCenter()</code>	24
6.3.2.3 <code>setRadius()</code>	24
6.3.3 Member Data Documentation	24
6.3.3.1 <code>center</code>	24
6.3.3.2 <code>radius</code>	25
6.4 <code>Commands::Command</code> Class Reference	25
6.4.1 Detailed Description	25
6.4.2 Constructor & Destructor Documentation	25
6.4.2.1 <code>~Command()</code>	25
6.4.3 Member Function Documentation	25
6.4.3.1 <code>execute()</code>	25
6.5 <code>CommandManager</code> Class Reference	26
6.5.1 Detailed Description	26
6.5.2 Member Function Documentation	26
6.5.2.1 <code>registerCommand()</code>	26
6.5.2.2 <code>update()</code>	26
6.6 <code>Commands::Command::ExecuteKey</code> Class Reference	27
6.6.1 Friends And Related Symbol Documentation	27
6.6.1.1 <code>CommandManager</code>	27
6.7 <code>GameManager</code> Class Reference	27
6.8 <code>Shapes::HollowCircle</code> Class Reference	27
6.8.1 Constructor & Destructor Documentation	29
6.8.1.1 <code>HollowCircle()</code>	29
6.8.2 Member Function Documentation	29
6.8.2.1 <code>draw()</code>	29
6.8.2.2 <code>setThickness()</code>	29
6.8.3 Member Data Documentation	30
6.8.3.1 <code>thickness</code>	30
6.9 <code>Views::HUD</code> Class Reference	30
6.9.1 Constructor & Destructor Documentation	31
6.9.1.1 <code>HUD()</code>	31
6.9.2 Member Function Documentation	31
6.9.2.1 <code>getRect()</code>	31
6.9.2.2 <code>transform()</code>	32
6.9.2.3 <code>transformFromRender()</code>	32
6.10 <code>InputHandler</code> Class Reference	33
6.10.1 Detailed Description	33
6.10.2 Constructor & Destructor Documentation	33
6.10.2.1 <code>InputHandler()</code>	33
6.10.3 Member Function Documentation	34
6.10.3.1 <code>getInstance()</code>	34
6.10.3.2 <code>getMousePosition()</code>	34

6.10.3.3 holdTime() [1/2]	34
6.10.3.4 holdTime() [2/2]	34
6.10.3.5 isButtonDown()	34
6.10.3.6 isButtonUp()	34
6.10.3.7 isKeyDown()	34
6.10.3.8 isKeyUp()	35
6.10.3.9 operator=()	35
6.10.3.10 pollButtonPress()	35
6.10.3.11 pollButtonRelease()	35
6.10.3.12 pollKeyPress()	35
6.10.3.13 pollKeyRelease()	36
6.10.3.14 pollMouseScroll()	36
6.10.3.15 receiveEvent() [1/3]	36
6.10.3.16 receiveEvent() [2/3]	36
6.10.3.17 receiveEvent() [3/3]	36
6.11 Items::Item Class Reference	37
6.11.1 Constructor & Destructor Documentation	37
6.11.1.1 Item()	37
6.12 KeyBind Struct Reference	37
6.12.1 Detailed Description	38
6.12.2 Member Enumeration Documentation	38
6.12.2.1 Trigger	38
6.12.3 Constructor & Destructor Documentation	38
6.12.3.1 KeyBind()	38
6.12.4 Friends And Related Symbol Documentation	38
6.12.4.1 operator<	38
6.12.5 Member Data Documentation	38
6.12.5.1 buttons	38
6.12.5.2 ID	39
6.12.5.3 KeyBindCount	39
6.12.5.4 keys	39
6.13 Shapes::Line Class Reference	39
6.13.1 Constructor & Destructor Documentation	41
6.13.1.1 Line()	41
6.13.2 Member Function Documentation	41
6.13.2.1 draw()	41
6.13.2.2 setBeginPoint()	41
6.13.2.3 setEndPoint()	41
6.13.2.4 setThickness()	41
6.13.3 Member Data Documentation	41
6.13.3.1 beginPoint	41
6.13.3.2 endPoint	42

6.13.3.3 thickness	42
6.14 Objects::Object Class Reference	42
6.14.1 Detailed Description	44
6.14.2 Constructor & Destructor Documentation	44
6.14.2.1 Object()	44
6.14.2.2 ~Object()	44
6.14.3 Member Function Documentation	44
6.14.3.1 debug()	44
6.14.3.2 flipHorizontal()	44
6.14.3.3 flipVertical()	45
6.14.3.4 getAngle()	45
6.14.3.5 getDimension()	45
6.14.3.6 getFlipFlag()	45
6.14.3.7 getPosition()	45
6.14.3.8 getRenderRect()	46
6.14.3.9 getTexture()	46
6.14.3.10 getTextureCount()	46
6.14.3.11 getVisibility()	46
6.14.3.12 lookAt()	46
6.14.3.13 move()	47
6.14.3.14 nextTexture()	47
6.14.3.15 previousTexture()	47
6.14.3.16 rotate()	47
6.14.3.17 setAngle()	47
6.14.3.18 setTexture()	48
6.14.3.19 setVisibility()	48
6.14.3.20 stretch()	48
6.14.3.21 stretchX()	49
6.14.3.22 stretchY()	49
6.14.3.23 update()	49
6.14.4 Friends And Related Symbol Documentation	49
6.14.4.1 TextureHandler	49
6.15 Objects::Player Class Reference	50
6.16 Shapes::Rect Class Reference	52
6.16.1 Member Data Documentation	53
6.16.1.1 dimension	53
6.16.1.2 position	53
6.17 Renderer Class Reference	53
6.17.1 Detailed Description	54
6.17.2 Constructor & Destructor Documentation	54
6.17.2.1 Renderer()	54
6.17.3 Member Function Documentation	55

6.17.3.1 clear()	55
6.17.3.2 createTexture()	55
6.17.3.3 debug()	55
6.17.3.4 getInstance()	55
6.17.3.5 moveLayerBottom()	55
6.17.3.6 moveLayerDown()	55
6.17.3.7 moveLayerTop()	56
6.17.3.8 moveLayerUp()	56
6.17.3.9 operator=()	56
6.17.3.10 registerObject()	56
6.17.3.11 removeObject()	56
6.17.3.12 render()	57
6.18 Renderer::RenderKey Class Reference	57
6.18.1 Constructor & Destructor Documentation	57
6.18.1.1 RenderKey() [1/2]	57
6.18.1.2 RenderKey() [2/2]	57
6.19 RenderObjectBase Class Reference	58
6.19.1 Detailed Description	58
6.19.2 Member Function Documentation	58
6.19.2.1 debug()	58
6.20 sdl_deleter Struct Reference	58
6.20.1 Detailed Description	59
6.20.2 Member Function Documentation	59
6.20.2.1 operator>() [1/12]	59
6.20.2.2 operator>() [2/12]	59
6.20.2.3 operator>() [3/12]	59
6.20.2.4 operator>() [4/12]	59
6.20.2.5 operator>() [5/12]	59
6.20.2.6 operator>() [6/12]	60
6.20.2.7 operator>() [7/12]	60
6.20.2.8 operator>() [8/12]	60
6.20.2.9 operator>() [9/12]	60
6.20.2.10 operator>() [10/12]	60
6.20.2.11 operator>() [11/12]	60
6.20.2.12 operator>() [12/12]	60
6.21 SelectionManager< T > Class Template Reference	60
6.21.1 Constructor & Destructor Documentation	61
6.21.1.1 SelectionManager() [1/2]	61
6.21.1.2 SelectionManager() [2/2]	61
6.21.2 Member Function Documentation	61
6.21.2.1 add()	61
6.21.2.2 get()	62

6.21.2.3	getSelectionId()	62
6.21.2.4	next()	62
6.21.2.5	prev()	62
6.21.2.6	remove()	62
6.21.2.7	set()	63
6.21.2.8	size()	63
6.21.3	Member Data Documentation	63
6.21.3.1	SELECTION_NOT_SET	63
6.22	Shapes::Shape Class Reference	64
6.22.1	Constructor & Destructor Documentation	65
6.22.1.1	Shape()	65
6.22.1.2	~Shape()	65
6.22.2	Member Function Documentation	65
6.22.2.1	draw()	65
6.22.2.2	getColor()	65
6.22.2.3	setColor()	66
6.22.3	Member Data Documentation	66
6.22.3.1	color	66
6.22.3.2	view	66
6.23	TextureHandler Class Reference	66
6.23.1	Detailed Description	67
6.23.2	Constructor & Destructor Documentation	67
6.23.2.1	TextureHandler()	67
6.23.3	Member Function Documentation	67
6.23.3.1	getInstance()	67
6.23.3.2	getTexture()	67
6.23.3.3	operator=()	67
6.24	Vector2D Class Reference	68
6.24.1	Constructor & Destructor Documentation	68
6.24.1.1	Vector2D() [1/2]	68
6.24.1.2	Vector2D() [2/2]	68
6.24.2	Member Function Documentation	69
6.24.2.1	cross()	69
6.24.2.2	dot()	69
6.24.2.3	getX()	69
6.24.2.4	getY()	69
6.24.2.5	len()	69
6.24.2.6	len2()	69
6.24.2.7	norm()	69
6.24.2.8	rotate() [1/2]	69
6.24.2.9	rotate() [2/2]	70
6.24.2.10	zero()	70

6.24.3 Friends And Related Symbol Documentation	70
6.24.3.1 operator*	70
6.24.3.2 operator*= 6.24.3.3 operator+	70
6.24.3.4 operator+=	70
6.24.3.5 operator- [1/2]	70
6.24.3.6 operator- [2/2]	70
6.24.3.7 operator-=	71
6.24.3.8 operator/	71
6.24.3.9 operator/=	71
6.25 Views::View Class Reference	71
6.25.1 Detailed Description	72
6.25.2 Constructor & Destructor Documentation	73
6.25.2.1 View()	73
6.25.2.2 ~View()	73
6.25.3 Member Function Documentation	73
6.25.3.1 getAngle()	73
6.25.3.2 getDimension()	73
6.25.3.3 getPosition()	73
6.25.3.4 getRect()	73
6.25.3.5 getZoom()	74
6.25.3.6 transform()	74
6.25.3.7 transformFromRender()	74
6.25.4 Member Data Documentation	75
6.25.4.1 dimension	75
6.25.4.2 position	75
7 File Documentation	77
7.1 include/command/command.h File Reference	77
7.2 command.h	78
7.3 include/command_manager.h File Reference	79
7.4 command_manager.h	80
7.5 include/config.h File Reference	80
7.6 config.h	81
7.7 include/game_manager.h File Reference	82
7.8 game_manager.h	82
7.9 include/init.h File Reference	82
7.10 init.h	84
7.11 include/input_handler.h File Reference	84
7.11.1 Enumeration Type Documentation	85
7.11.1.1 MouseButton	85
7.12 input_handler.h	86

7.13 include/object/button.h File Reference	87
7.14 button.h	87
7.15 include/object/item/item.h File Reference	88
7.16 item.h	88
7.17 include/object/object.h File Reference	89
7.18 object.h	90
7.19 include/object/player.h File Reference	91
7.20 player.h	91
7.21 include/render_object_base.h File Reference	92
7.22 render_object_base.h	92
7.23 include/renderer.h File Reference	92
7.24 renderer.h	93
7.25 include/shape/circle.h File Reference	94
7.26 circle.h	95
7.27 include/shape/line.h File Reference	96
7.28 line.h	97
7.29 include/shape/rect.h File Reference	98
7.30 rect.h	98
7.31 include/shape/shape.h File Reference	99
7.32 shape.h	100
7.33 include/shape/shapes.h File Reference	100
7.34 shapes.h	101
7.35 include/texture/texture_handler.h File Reference	101
7.36 texture_handler.h	102
7.37 include/utility/functions.h File Reference	102
7.38 functions.h	102
7.39 include/utility/pointer_wrappers.h File Reference	103
7.39.1 Typedef Documentation	104
7.39.1.1 sdl_unique_ptr	104
7.39.2 Function Documentation	104
7.39.2.1 sdl_make_shared()	104
7.40 pointer_wrappers.h	104
7.41 include/utility/selection_manager.h File Reference	105
7.42 selection_manager.h	105
7.43 include/utility/utility.h File Reference	106
7.43.1 Macro Definition Documentation	107
7.43.1.1 _USE_MATH_DEFINES	107
7.43.2 Function Documentation	107
7.43.2.1 normalizeAngle()	107
7.43.2.2 polarToCartesian()	108
7.44 utility.h	108
7.45 include/utility/vector2d.h File Reference	108

7.46 vector2d.h	109
7.47 include/view/camera.h File Reference	110
7.48 camera.h	111
7.49 include/view/hud.h File Reference	111
7.50 hud.h	112
7.51 include/view/view.h File Reference	112
7.52 view.h	114
7.53 include/view/views.h File Reference	114
7.54 views.h	115
Index	117

Chapter 1

Namespace Index

1.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Commands	9
Config	9
Functions	10
Global	10
Items	13
Objects	13
Shapes	14
Views	14

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

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

Commands::Command	25
CommandManager	26
Commands::Command::ExecuteKey	27
GameManager	27
InputHandler	33
Items::Item	37
KeyBind	37
Renderer	53
Renderer::RenderKey	57
RenderObjectBase	58
Objects::Object	42
Objects::Button	15
Objects::Player	50
Shapes::Shape	64
Shapes::Circle	22
Shapes::HollowCircle	27
Shapes::Line	39
Shapes::Rect	52
sdl_deleter	58
SelectionManager< T >	60
SelectionManager< SDL_Texture * >	60
TextureHandler	66
Vector2D	68
Views::View	71
Views::Camera	17
Views::HUD	30

Chapter 3

Class Index

3.1 Class List

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

Objects::Button	15
Views::Camera	
Camera for following object or stationary view	17
Shapes::Circle	22
Commands::Command	
Commands base abstract class	25
CommandManager	
Manages a map from key bindings to various functions. e.g. <code>player.move()</code> , <code>currentScene.<-> set(mainMenu)</code> , or <code>renderer.drawCone()</code>	26
Commands::Command::ExecuteKey	27
GameManager	27
Shapes::HollowCircle	27
Views::HUD	30
InputHandler	
This is a global singleton class of handling user inputs. Wrapper class of <code>SDL_PollEvent</code> and events handling	33
Items::Item	37
KeyBind	
KeyBind structure for key bindings	37
Shapes::Line	39
Objects::Object	
Object type for all renderable objects in the world note: the texture won't be created until loaded into the renderer	42
Objects::Player	50
Shapes::Rect	52
Renderer	
Required key to call render() in	53
Renderer::RenderKey	57
RenderObjectBase	
Empty render object base class category	58
sdl_deleter	
Generic deleter functor for SDL resources. For use with std smart pointers	58
SelectionManager< T >	60
Shapes::Shape	64
TextureHandler	
This is a global singleton class for texture handling	66

Vector2D	68
Views::View	
View : defines a view area, translates the objects' virtual rects to real rendering rects	71

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

include/command_manager.h	79
include/config.h	80
include/game_manager.h	82
include/init.h	82
include/input_handler.h	84
include/render_object_base.h	92
include/renderer.h	92
include/command/command.h	77
include/object/button.h	87
include/object/object.h	89
include/object/player.h	91
include/object/item/item.h	88
include/shape/circle.h	94
include/shape/line.h	96
include/shape/rect.h	98
include/shape/shape.h	99
include/shape/shapes.h	100
include/texture/texture_handler.h	101
include/utility/functions.h	102
include/utility/pointer_wrappers.h	103
include/utility/selection_manager.h	105
include/utility/utility.h	106
include/utility/vector2d.h	108
include/view/camera.h	110
include/view/hud.h	111
include/view/view.h	112
include/view/views.h	114

Chapter 5

Namespace Documentation

5.1 Commands Namespace Reference

Classes

- class [Command](#)
[Commands](#) base abstract class.

5.2 Config Namespace Reference

Variables

- const std::string [gameTitle](#) = "Lab Raid"
- const int [screenWidth](#) = 1280
- const int [screenHeight](#) = 768
- const int [volume](#) = 50
- const int [framerate](#) = 60
- const float [holdTimeThreshold](#) = 100
- const SDL_WindowFlags [screenType](#) = SDL_WINDOW_SHOWN
- const SDL_Color [backgroundColor](#) { 0x3F, 0x3F, 0x3F, 0xFF }

5.2.1 Variable Documentation

5.2.1.1 backgroundColor

```
const SDL_Color Config::backgroundColor { 0x3F, 0x3F, 0x3F, 0xFF }
```

5.2.1.2 framerate

```
const int Config::framerate = 60
```

5.2.1.3 gameTitle

```
const std::string Config::gameTitle = "Lab Raid"
```

5.2.1.4 holdTimeThreshold

```
const float Config::holdTimeThreshold = 100
```

5.2.1.5 screenHeight

```
const int Config::screenHeight = 768
```

5.2.1.6 screenType

```
const SDL_WindowFlags Config::screenType = SDL_WINDOW_SHOWN
```

5.2.1.7 screenWidth

```
const int Config::screenWidth = 1280
```

5.2.1.8 volume

```
const int Config::volume = 50
```

5.3 Functions Namespace Reference

5.4 Global Namespace Reference

Functions

- void [init](#) ()

Variables

- `std::unique_ptr< FPSmanager > fpsManager`
- `std::unique_ptr< Views::Camera > playerCamera`
- `std::unique_ptr< Views::HUD > hudView`
- `std::unique_ptr< Views::HUD > menuView`
- `std::shared_ptr< Objects::Object > playerObject`
- `std::shared_ptr< Objects::Object > arrowObject1`
- `std::shared_ptr< Objects::Object > arrowObject2`
- `std::shared_ptr< Shapes::Circle > yellowCircle`
- `std::shared_ptr< Shapes::Circle > greenCircle`
- `std::shared_ptr< Shapes::Circle > blueCircle`
- `std::shared_ptr< Shapes::Circle > redCircle`
- `std::shared_ptr< Shapes::Circle > purpleCircle`
- `std::shared_ptr< Shapes::HollowCircle > hollowCircle1`
- `std::shared_ptr< Shapes::Line > line1`
- `std::shared_ptr< Shapes::Line > line2`
- `std::shared_ptr< Shapes::Line > line3`
- `std::shared_ptr< Shapes::Line > line4`
- `std::shared_ptr< Shapes::Line > crosshairLine1`
- `std::shared_ptr< Shapes::Line > crosshairLine2`
- `std::shared_ptr< Shapes::HollowCircle > crosshairCircle1`

5.4.1 Function Documentation

5.4.1.1 `init()`

```
void Global::init ( )
```

5.4.2 Variable Documentation

5.4.2.1 `arrowObject1`

```
std::shared_ptr<Objects::Object> Global::arrowObject1
```

5.4.2.2 `arrowObject2`

```
std::shared_ptr<Objects::Object> Global::arrowObject2 [extern]
```

5.4.2.3 `blueCircle`

```
std::shared_ptr<Shapes::Circle> Global::blueCircle [extern]
```

5.4.2.4 `crosshairCircle1`

```
std::shared_ptr<Shapes::HollowCircle> Global::crosshairCircle1 [extern]
```

5.4.2.5 crosshairLine1

`std::shared_ptr<Shapes::Line> Global::crosshairLine1 [extern]`

5.4.2.6 crosshairLine2

`std::shared_ptr<Shapes::Line> Global::crosshairLine2 [extern]`

5.4.2.7 fpsManager

`std::unique_ptr<FPSmanager> Global::fpsManager [extern]`

5.4.2.8 greenCircle

`std::shared_ptr<Shapes::Circle> Global::greenCircle [extern]`

5.4.2.9 hollowCircle1

`std::shared_ptr<Shapes::HollowCircle> Global::hollowCircle1 [extern]`

5.4.2.10 hudView

`std::unique_ptr<Views::HUD> Global::hudView [extern]`

5.4.2.11 line1

`std::shared_ptr<Shapes::Line> Global::line1 [extern]`

5.4.2.12 line2

`std::shared_ptr<Shapes::Line> Global::line2 [extern]`

5.4.2.13 line3

`std::shared_ptr<Shapes::Line> Global::line3 [extern]`

5.4.2.14 line4

`std::shared_ptr<Shapes::Line> Global::line4 [extern]`

5.4.2.15 menuView

```
std::unique_ptr<Views::HUD> Global::menuView [extern]
```

5.4.2.16 playerCamera

```
std::unique_ptr<Views::Camera> Global::playerCamera [extern]
```

5.4.2.17 playerObject

```
std::shared_ptr<Objects::Object> Global::playerObject [extern]
```

5.4.2.18 purpleCircle

```
std::shared_ptr<Shapes::Circle> Global::purpleCircle [extern]
```

5.4.2.19 redCircle

```
std::shared_ptr<Shapes::Circle> Global::redCircle [extern]
```

5.4.2.20 yellowCircle

```
std::shared_ptr<Shapes::Circle> Global::yellowCircle [extern]
```

5.5 Items Namespace Reference

Classes

- class [Item](#)

5.6 Objects Namespace Reference

Classes

- class [Button](#)
- class [Object](#)
 - [Object](#) type for all renderable objects in the world note: the texture won't be created until loaded into the renderer.*
- class [Player](#)

5.7 Shapes Namespace Reference

Classes

- class [Circle](#)
- class [HollowCircle](#)
- class [Line](#)
- class [Rect](#)
- class [Shape](#)

5.8 Views Namespace Reference

Classes

- class [Camera](#)
[Camera](#) for following object or stationary view.
- class [HUD](#)
- class [View](#)
[View](#): defines a view area, translates the objects' virtual rects to real rendering rects.

Variables

- const int [INIT_VIEW_WIDTH](#) = 1600
- const int [INIT_VIEW_HEIGHT](#) = 900

5.8.1 Variable Documentation

5.8.1.1 INIT_VIEW_HEIGHT

```
const int Views::INIT_VIEW_HEIGHT = 900
```

5.8.1.2 INIT_VIEW_WIDTH

```
const int Views::INIT_VIEW_WIDTH = 1600
```

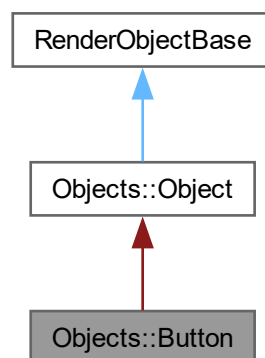
Chapter 6

Class Documentation

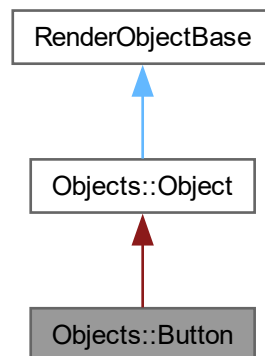
6.1 Objects::Button Class Reference

```
#include <button.h>
```

Inheritance diagram for Objects::Button:



Collaboration diagram for Objects::Button:



Public Member Functions

- `Button` (const `Views::View` *view, const `Vector2D` &position, const `Vector2D` &dimension, const `SDL_Color` &color, const `std::string` &text, `std::function`< void(void)> action)
- void `setHovered` (void) noexcept
- void `onClick` (void) noexcept
- void `update` (void) noexcept

6.1.1 Constructor & Destructor Documentation

6.1.1.1 Button()

```
Objects::Button::Button (
    const Views::View * view,
    const Vector2D & position,
    const Vector2D & dimension,
    const SDL_Color & color,
    const std::string & text,
    std::function< void(void)> action )
```

6.1.2 Member Function Documentation

6.1.2.1 onClick()

```
void Objects::Button::onClick (
    void ) [noexcept]
```

6.1.2.2 setHovered()

```
void Objects::Button::setHovered (
    void ) [noexcept]
```

6.1.2.3 update()

```
void Objects::Button::update (
    void ) [noexcept]
```

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

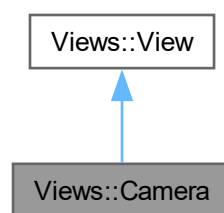
- [include/object/button.h](#)

6.2 Views::Camera Class Reference

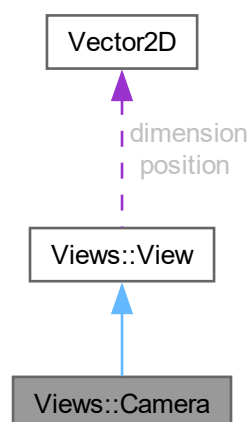
[Camera](#) for following object or stationary view.

```
#include <camera.h>
```

Inheritance diagram for Views::Camera:



Collaboration diagram for Views::Camera:



Public Member Functions

- [Camera](#) ()
- void [setPivotObject](#) (std::shared_ptr< [Objects::Object](#) > pivotObject) noexcept
Sets the pivot object of the camera.
- void [setPosition](#) (const [Vector2D](#) &newPosition) noexcept
Sets the position of the camera.
- void [setDimension](#) (const [Vector2D](#) &newDimension)
Sets the dimensions of the camera. The new dimension vector should be positive in both components. Throws std::invalid_argument if the new dimension vector is invalid.
- void [setZoom](#) (float zoom)
Sets the zoom level of the camera.
- float [getZoom](#) (void) const noexcept override
Gets the zoom level of the view.
- void [setAngle](#) (float angle) noexcept
Sets the rotation angle of the camera.
- void [rotate](#) (float diffAngle) noexcept
Rotates the view by @diffAngle.
- float [getAngle](#) (void) const noexcept override
Gets the rotation angle of the camera.
- SDL_FRect [getRect](#) (const [Objects::Object](#) &object) const noexcept override
Gets the render rect for.
- [Vector2D](#) [transform](#) (const [Vector2D](#) &position) const noexcept override
Gets the transformed render position of.
- [Vector2D](#) [transformFromRender](#) (const [Vector2D](#) &renderPosition) const noexcept override
Gets the virtual position of.

Public Member Functions inherited from [Views::View](#)

- virtual [~View](#) ()
- virtual [Vector2D](#) [getDimension](#) (void) const noexcept
Gets the virtual dimension of the view.

Additional Inherited Members

Protected Member Functions inherited from [Views::View](#)

- [View](#) (const [Vector2D](#) &_position, const [Vector2D](#) &_dimension)

Protected Attributes inherited from [Views::View](#)

- [Vector2D](#) position
- [Vector2D](#) dimension

6.2.1 Detailed Description

[Camera](#) for following object or stationary view.

6.2.2 Constructor & Destructor Documentation

6.2.2.1 Camera()

```
Views::Camera::Camera ( )
```

6.2.3 Member Function Documentation

6.2.3.1 getAngle()

```
float Views::Camera::getAngle (
    void ) const [override], [virtual], [noexcept]
```

Gets the rotation angle of the camera.

Returns

The rotation angle of the camera.

Reimplemented from [Views::View](#).

6.2.3.2 getRect()

```
SDL_FRect Views::Camera::getRect (
    const Objects::Object & object ) const [override], [virtual], [noexcept]
```

Gets the render rect for.

Parameters

<i>object.</i>	
<i>object</i>	The object to be rendered.

Returns

The render rect of
object.

Implements [Views::View](#).

6.2.3.3 getZoom()

```
float Views::Camera::getZoom (
    void ) const [override], [virtual], [noexcept]
```

Gets the zoom level of the view.

Returns

The zoom level of the view.

Reimplemented from [Views::View](#).

6.2.3.4 rotate()

```
void Views::Camera::rotate (
    float diffAngle ) [noexcept]
```

Rotates the view by @diffAngle.

Parameters

<i>diffAngle</i>	The angle to rotate by.
------------------	-------------------------

6.2.3.5 setAngle()

```
void Views::Camera::setAngle (
    float angle ) [noexcept]
```

Sets the rotation angle of the camera.

Parameters

<i>angle</i>	The rotation angle to be set.
--------------	-------------------------------

6.2.3.6 setDimension()

```
void Views::Camera::setDimension (
    const Vector2D & newDimension )
```

Sets the dimensions of the camera. The new dimension vector should be positive in both components. Throws `std::invalid_argument` if the new dimension vector is invalid.

Parameters

<i>newDimension</i>	The new dimensions of the camera.
---------------------	-----------------------------------

6.2.3.7 setPivotObject()

```
void Views::Camera::setPivotObject (
    std::shared_ptr< Objects::Object > pivotObject ) [noexcept]
```

Sets the pivot object of the camera.

Parameters

<i>pivotObject</i>	The object to pivot on.
--------------------	-------------------------

6.2.3.8 setPosition()

```
void Views::Camera::setPosition (
    const Vector2D & newPosition ) [noexcept]
```

Sets the position of the camera.

Parameters

<i>newPosition</i>	The new positions of the camera.
--------------------	----------------------------------

6.2.3.9 setZoom()

```
void Views::Camera::setZoom (
    float zoom )
```

Sets the zoom level of the camera.

Parameters

<i>zoom</i>	should be positive. Throws std::invalid_argument if
<i>zoom</i>	is invalid.
<i>zoom</i>	The zoom level to be set.

6.2.3.10 transform()

```
Vector2D Views::Camera::transform (
    const Vector2D & position ) const [override], [virtual], [noexcept]
```

Gets the transformed render position of.

Parameters

<i>position.</i>	
<i>position</i>	The virtual position to be transformed.

Returns

The render position after transformation.

Implements [Views::View](#).

6.2.3.11 transformFromRender()

```
Vector2D Views::Camera::transformFromRender (
    const Vector2D & renderPosition ) const [override], [virtual], [noexcept]
```

Gets the virtual position of.

Parameters

<i>renderPosition.</i>	
<i>renderPosition</i>	The render position to be transformed

Returns

The virtual position after transformation.

Implements [Views::View](#).

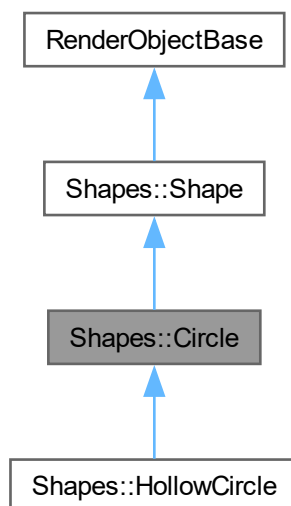
The documentation for this class was generated from the following file:

- [include/view/camera.h](#)

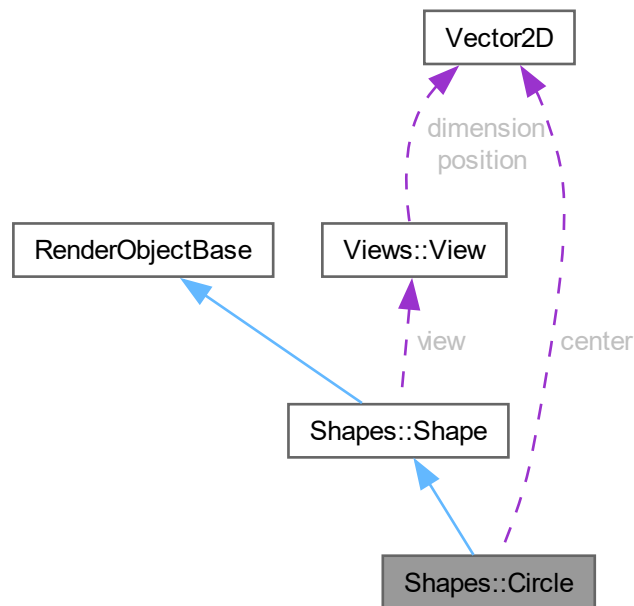
6.3 Shapes::Circle Class Reference

```
#include <circle.h>
```

Inheritance diagram for Shapes::Circle:



Collaboration diagram for Shapes::Circle:



Public Member Functions

- [Circle](#) ([Views::View](#) *[view](#), const [Vector2D](#) &[center](#), float [radius](#), [SDL_Color](#) [color](#)={ 0, 0, 0, 255 }) noexcept
- void [setCenter](#) (const [Vector2D](#) &[newCenter](#)) noexcept
- void [setRadius](#) (float [newRadius](#)) noexcept
- void [draw](#) ([SDL_Renderer](#) *[renderer](#)) const noexcept override

Public Member Functions inherited from [Shapes::Shape](#)

- [Shape](#) ([Views::View](#) *[view](#), const [SDL_Color](#) &[color](#)={ 0, 0, 0, 255 })
- virtual [~Shape](#) ()
- void [setColor](#) (const [SDL_Color](#) &[newColor](#)) noexcept
- [SDL_Color](#) [getColor](#) (void) const noexcept

Public Member Functions inherited from [RenderObjectBase](#)

- virtual void [debug](#) (void) const noexcept

Protected Attributes

- [Vector2D](#) [center](#)
- float [radius](#)

Protected Attributes inherited from [Shapes::Shape](#)

- const [Views::View](#) * [view](#)
Calls draw function after transforming coordinates with view.
- [SDL_Color](#) [color](#)

6.3.1 Constructor & Destructor Documentation

6.3.1.1 Circle()

```
Shapes::Circle::Circle (
    Views::View * view,
    const Vector2D & center,
    float radius,
    SDL\_Color color = { 0, 0, 0, 255 } ) [noexcept]
```

6.3.2 Member Function Documentation

6.3.2.1 draw()

```
void Shapes::Circle::draw (
    SDL\_Renderer * renderer ) const [override], [virtual], [noexcept]
```

Reimplemented from [Shapes::Shape](#).

Reimplemented in [Shapes::HollowCircle](#).

6.3.2.2 setCenter()

```
void Shapes::Circle::setCenter (
    const Vector2D & newCenter ) [noexcept]
```

6.3.2.3 setRadius()

```
void Shapes::Circle::setRadius (
    float newRadius ) [noexcept]
```

6.3.3 Member Data Documentation

6.3.3.1 center

```
Vector2D Shapes::Circle::center [protected]
```

6.3.3.2 radius

```
float Shapes::Circle::radius [protected]
```

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

- include/shape/[circle.h](#)

6.4 Commands::Command Class Reference

[Commands](#) base abstract class.

```
#include <command.h>
```

Classes

- class [ExecuteKey](#)

Public Member Functions

- virtual [~Command](#) ()
- virtual void [execute](#) ([ExecuteKey](#))

6.4.1 Detailed Description

[Commands](#) base abstract class.

6.4.2 Constructor & Destructor Documentation

6.4.2.1 ~Command()

```
virtual Commands::Command::~~Command ( ) [inline], [virtual]
```

6.4.3 Member Function Documentation

6.4.3.1 execute()

```
virtual void Commands::Command::execute (
    ExecuteKey ) [inline], [virtual]
```

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

- include/command/[command.h](#)

6.5 CommandManager Class Reference

Manages a map from key bindings to various functions. e.g. `player.move()`, `currentScene.set(mainMenu)`, or `renderer.drawCone()`.

```
#include <command_manager.h>
```

Public Member Functions

- bool [registerCommand](#) ([KeyBind](#) keyBind, std::shared_ptr< [Commands::Command](#) > command)
Registers a command for the specified key bind.
- void [update](#) () noexcept
Executes corresponding command if a key bind was matched. Note: beware of thread safety.

6.5.1 Detailed Description

Manages a map from key bindings to various functions. e.g. `player.move()`, `currentScene.set(mainMenu)`, or `renderer.drawCone()`.

6.5.2 Member Function Documentation

6.5.2.1 registerCommand()

```
bool CommandManager::registerCommand (
    KeyBind keyBind,
    std::shared_ptr< Commands::Command > command )
```

Registers a command for the specified key bind.

Parameters

<i>keyBind</i>	The key bind of this command.
<i>command</i>	The command to execute if the key bind is pressed.

Returns

Whether the command was successfully registered, fails if `keyBind` is already registered.

6.5.2.2 update()

```
void CommandManager::update ( ) [noexcept]
```

Executes corresponding command if a key bind was matched. Note: beware of thread safety.

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

- include/[command_manager.h](#)

6.6 Commands::Command::ExecuteKey Class Reference

```
#include <command.h>
```

Friends

- class [CommandManager](#)

6.6.1 Friends And Related Symbol Documentation

6.6.1.1 CommandManager

```
friend class CommandManager [friend]
```

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

- include/command/[command.h](#)

6.7 GameManager Class Reference

```
#include <game_manager.h>
```

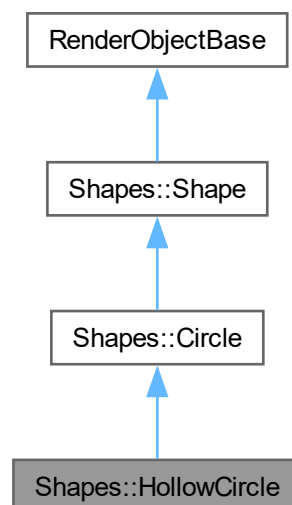
The documentation for this class was generated from the following file:

- include/[game_manager.h](#)

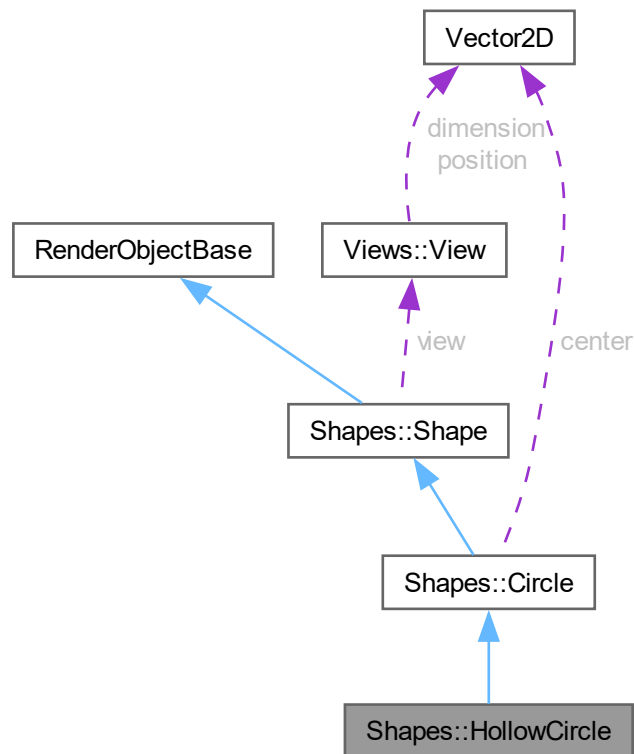
6.8 Shapes::HollowCircle Class Reference

```
#include <circle.h>
```

Inheritance diagram for Shapes::HollowCircle:



Collaboration diagram for Shapes::HollowCircle:



Public Member Functions

- [HollowCircle](#) ([Views::View](#) *[view](#), const [Vector2D](#) &[center](#), float [radius](#), uint8_t [thickness](#), [SDL_Color](#) [color](#)={ 0, 0, 0, 255 }) noexcept
- void [setThickness](#) (uint8_t newThickness) noexcept
- void [draw](#) ([SDL_Renderer](#) *[renderer](#)) const noexcept override

Public Member Functions inherited from [Shapes::Circle](#)

- [Circle](#) ([Views::View](#) *[view](#), const [Vector2D](#) &[center](#), float [radius](#), [SDL_Color](#) [color](#)={ 0, 0, 0, 255 }) noexcept
- void [setCenter](#) (const [Vector2D](#) &newCenter) noexcept
- void [setRadius](#) (float newRadius) noexcept

Public Member Functions inherited from [Shapes::Shape](#)

- [Shape](#) ([Views::View](#) *[view](#), const [SDL_Color](#) &[color](#)={ 0, 0, 0, 255 })
- virtual [~Shape](#) ()
- void [setColor](#) (const [SDL_Color](#) &newColor) noexcept
- [SDL_Color](#) [getColor](#) (void) const noexcept

Public Member Functions inherited from [RenderObjectBase](#)

- virtual void [debug](#) (void) const noexcept

Protected Attributes

- uint8_t [thickness](#)

Protected Attributes inherited from [Shapes::Circle](#)

- [Vector2D](#) [center](#)
- float [radius](#)

Protected Attributes inherited from [Shapes::Shape](#)

- const [Views::View](#) * [view](#)
Calls draw function after transforming coordinates with view.
- [SDL_Color](#) [color](#)

6.8.1 Constructor & Destructor Documentation

6.8.1.1 [HollowCircle\(\)](#)

```
Shapes::HollowCircle::HollowCircle (  
    Views::View * view,  
    const Vector2D & center,  
    float radius,  
    uint8_t thickness,  
    SDL\_Color color = { 0, 0, 0, 255 } ) [noexcept]
```

6.8.2 Member Function Documentation

6.8.2.1 [draw\(\)](#)

```
void Shapes::HollowCircle::draw (  
    SDL\_Renderer * renderer ) const [override], [virtual], [noexcept]
```

Reimplemented from [Shapes::Circle](#).

6.8.2.2 [setThickness\(\)](#)

```
void Shapes::HollowCircle::setThickness (  
    uint8_t newThickness ) [noexcept]
```

6.8.3 Member Data Documentation

6.8.3.1 thickness

```
uint8_t Shapes::HollowCircle::thickness [protected]
```

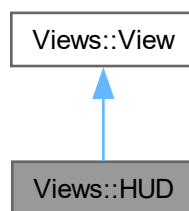
The documentation for this class was generated from the following file:

- [include/shape/circle.h](#)

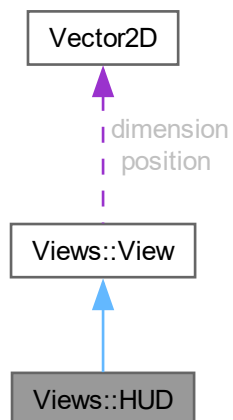
6.9 Views::HUD Class Reference

```
#include <hud.h>
```

Inheritance diagram for Views::HUD:



Collaboration diagram for Views::HUD:



Public Member Functions

- [HUD](#) ()
- `SDL_FRect` [getRect](#) (const [Objects::Object](#) &) const noexcept override
Gets the render rect for.
- [Vector2D](#) [transform](#) (const [Vector2D](#) &position) const noexcept override
Gets the transformed render position of.
- [Vector2D](#) [transformFromRender](#) (const [Vector2D](#) &renderPosition) const noexcept override
Gets the virtual position of.

Public Member Functions inherited from [Views::View](#)

- virtual [~View](#) ()
- virtual [Vector2D](#) [getPosition](#) (void) const noexcept
Gets the virtual position of the view.
- virtual [Vector2D](#) [getDimension](#) (void) const noexcept
Gets the virtual dimension of the view.
- virtual float [getAngle](#) (void) const noexcept
Gets the rotation angle of the view.
- virtual float [getZoom](#) (void) const noexcept
Gets the zoom level of the view.

Additional Inherited Members

Protected Member Functions inherited from [Views::View](#)

- [View](#) (const [Vector2D](#) &_position, const [Vector2D](#) &_dimension)

Protected Attributes inherited from [Views::View](#)

- [Vector2D](#) position
- [Vector2D](#) dimension

6.9.1 Constructor & Destructor Documentation

6.9.1.1 HUD()

```
Views::HUD::HUD ( )
```

6.9.2 Member Function Documentation

6.9.2.1 getRect()

```
SDL_FRect Views::HUD::getRect (
    const Objects::Object & object ) const [override], [virtual], [noexcept]
```

Gets the render rect for.

Parameters

<i>object.</i>	
<i>object</i>	The object to be rendered.

Returns

The render rect of
object.

Implements [Views::View](#).

6.9.2.2 transform()

```
Vector2D Views::HUD::transform (  
    const Vector2D & position ) const [override], [virtual], [noexcept]
```

Gets the transformed render position of.

Parameters

<i>position.</i>	
<i>position</i>	The virtual position to be transformed.

Returns

The render position after transformation.

Implements [Views::View](#).

6.9.2.3 transformFromRender()

```
Vector2D Views::HUD::transformFromRender (  
    const Vector2D & renderPosition ) const [override], [virtual], [noexcept]
```

Gets the virtual position of.

Parameters

<i>renderPosition.</i>	
<i>renderPosition</i>	The render position to be transformed

Returns

The virtual position after transformation.

Implements [Views::View](#).

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

- [include/view/hud.h](#)

6.10 InputHandler Class Reference

This is a global singleton class of handling user inputs. Wrapper class of `SDL_PollEvent` and events handling.

```
#include <input_handler.h>
```

Public Member Functions

- [InputHandler](#) (const [InputHandler](#) &)=delete
- void [operator=](#) (const [InputHandler](#) &)=delete
- bool [pollKeyPress](#) (SDL_Keycode key) noexcept
Checks if a key is pressed. (SDL_KeyDown) Is only true when the key was not held down.
- bool [pollKeyRelease](#) (SDL_Keycode key) noexcept
Checks if a key is released. (SDL_KeyUp)
- bool [isKeyDown](#) (SDL_Keycode key) const noexcept
Checks if a key is held down. (SDL_KeyDown)
- bool [isKeyUp](#) (SDL_Keycode key) const noexcept
Checks if a key is not held down.
- uint32_t [holdTime](#) (SDL_Keycode key) const noexcept
Gets the time a key was held down in SDL_Ticks.
- bool [pollButtonPress](#) ([MouseButton](#) button) noexcept
- bool [pollButtonRelease](#) ([MouseButton](#) button) noexcept
- bool [isButtonDown](#) ([MouseButton](#) button) const noexcept
- bool [isButtonUp](#) ([MouseButton](#) button) const noexcept
- uint32_t [holdTime](#) ([MouseButton](#) button) const noexcept
- [Vector2D](#) [getMousePosition](#) (void) const noexcept
- [Vector2D](#) [pollMouseScroll](#) (void) noexcept
- void [receiveEvent](#) (SDL_KeyboardEvent keyboardEvent) noexcept
- void [receiveEvent](#) (SDL_MouseButtonEvent mouseButtonEvent) noexcept
- void [receiveEvent](#) (SDL_MouseWheelEvent mouseWheelEvent) noexcept

Static Public Member Functions

- static [InputHandler](#) & [getInstance](#) (void) noexcept

6.10.1 Detailed Description

This is a global singleton class of handling user inputs. Wrapper class of `SDL_PollEvent` and events handling.

6.10.2 Constructor & Destructor Documentation

6.10.2.1 InputHandler()

```
InputHandler::InputHandler (
    const InputHandler & ) [delete]
```

6.10.3 Member Function Documentation

6.10.3.1 getInstance()

```
static InputHandler & InputHandler::getInstance (
    void ) [static], [noexcept]
```

6.10.3.2 getMousePosition()

```
Vector2D InputHandler::getMousePosition (
    void ) const [noexcept]
```

6.10.3.3 holdTime() [1/2]

```
uint32_t InputHandler::holdTime (
    MouseButton button ) const [noexcept]
```

6.10.3.4 holdTime() [2/2]

```
uint32_t InputHandler::holdTime (
    SDL_Keycode key ) const [noexcept]
```

Gets the time a key was held down in SDL_Ticks.

Returns

How long the key was held down.

6.10.3.5 isButtonDown()

```
bool InputHandler::isButtonDown (
    MouseButton button ) const [noexcept]
```

6.10.3.6 isButtonUp()

```
bool InputHandler::isButtonUp (
    MouseButton button ) const [noexcept]
```

6.10.3.7 isKeyDown()

```
bool InputHandler::isKeyDown (
    SDL_Keycode key ) const [noexcept]
```

Checks if a key is held down. (SDL_KeyDown)

Parameters

<i>key</i>	SDL_Keycode key value.
------------	------------------------

Returns

Whether the key was held down.

6.10.3.8 isKeyUp()

```
bool InputHandler::isKeyUp (
    SDL_Keycode key ) const [noexcept]
```

Checks if a key is not held down.

Parameters

<i>key</i>	SDL_Keycode key value.
------------	------------------------

Returns

Whether the key was not held down.

6.10.3.9 operator=()

```
void InputHandler::operator= (
    const InputHandler & ) [delete]
```

6.10.3.10 pollButtonPress()

```
bool InputHandler::pollButtonPress (
    MouseButton button ) [noexcept]
```

6.10.3.11 pollButtonRelease()

```
bool InputHandler::pollButtonRelease (
    MouseButton button ) [noexcept]
```

6.10.3.12 pollKeyPress()

```
bool InputHandler::pollKeyPress (
    SDL_Keycode key ) [noexcept]
```

Checks if a key is pressed. (SDL_KeyDown) Is only true when the key was not held down.

Parameters

<i>key</i>	SDL_Keycode key value.
------------	------------------------

Returns

Whether the key was pressed.

6.10.3.13 pollKeyRelease()

```
bool InputHandler::pollKeyRelease (
    SDL_Keycode key ) [noexcept]
```

Checks if a key is released. (SDL_KeyUp)

Parameters

<i>key</i>	SDL_Keycode key value.
------------	------------------------

Returns

Whether the key was released.

6.10.3.14 pollMouseScroll()

```
Vector2D InputHandler::pollMouseScroll (
    void ) [noexcept]
```

6.10.3.15 receiveEvent() [1/3]

```
void InputHandler::receiveEvent (
    SDL_KeyboardEvent keyboardEvent ) [noexcept]
```

6.10.3.16 receiveEvent() [2/3]

```
void InputHandler::receiveEvent (
    SDL_MouseButtonEvent mouseButtonEvent ) [noexcept]
```

6.10.3.17 receiveEvent() [3/3]

```
void InputHandler::receiveEvent (
    SDL_MouseWheelEvent mouseWheelEvent ) [noexcept]
```

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

- [include/input_handler.h](#)

6.11 Items::Item Class Reference

```
#include <item.h>
```

Public Member Functions

- [Item](#) (const std::vector< std::string > &instanceTextureNames, const std::vector< std::string > &inventoryObject, const std::string &itemName, uint8_t cap, uint8_t count)

6.11.1 Constructor & Destructor Documentation

6.11.1.1 Item()

```
Items::Item::Item (
    const std::vector< std::string > & instanceTextureNames,
    const std::vector< std::string > & inventoryObject,
    const std::string & itemName,
    uint8_t cap,
    uint8_t count )
```

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

- include/object/item/[item.h](#)

6.12 KeyBind Struct Reference

[KeyBind](#) structure for key bindings.

```
#include <command_manager.h>
```

Public Types

- enum class [Trigger](#) { [TAP](#) , [HOLD](#) , [RELEASE](#) }

Public Member Functions

- [KeyBind](#) (const std::map< SDL_Keycode, [Trigger](#) > &keys, const std::map< [MouseButton](#), [Trigger](#) > buttons)

Public Attributes

- int [ID](#)
- std::map< SDL_Keycode, [Trigger](#) > [keys](#)
- std::map< [MouseButton](#), [Trigger](#) > [buttons](#)

Static Public Attributes

- static unsigned int [KeyBindCount](#)

Friends

- bool [operator<](#) (const [KeyBind](#) &a, const [KeyBind](#) &b)

6.12.1 Detailed Description

[KeyBind](#) structure for key bindings.

6.12.2 Member Enumeration Documentation

6.12.2.1 Trigger

```
enum class KeyBind::Trigger [strong]
```

Enumerator

TAP	
HOLD	
RELEASE	

6.12.3 Constructor & Destructor Documentation

6.12.3.1 KeyBind()

```
KeyBind::KeyBind (
    const std::map< SDL_Keycode, Trigger > & keys,
    const std::map< MouseButton, Trigger > buttons ) [inline]
```

6.12.4 Friends And Related Symbol Documentation

6.12.4.1 operator<

```
bool operator< (
    const KeyBind & a,
    const KeyBind & b ) [friend]
```

6.12.5 Member Data Documentation

6.12.5.1 buttons

```
std::map<MouseButton, Trigger> KeyBind::buttons
```

6.12.5.2 ID

```
int KeyBind::ID
```

6.12.5.3 KeyBindCount

```
unsigned int KeyBind::KeyBindCount [static]
```

6.12.5.4 keys

```
std::map<SDL_Keycode, Trigger> KeyBind::keys
```

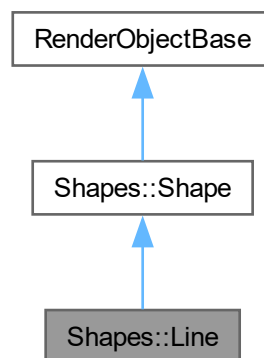
The documentation for this struct was generated from the following file:

- include/[command_manager.h](#)

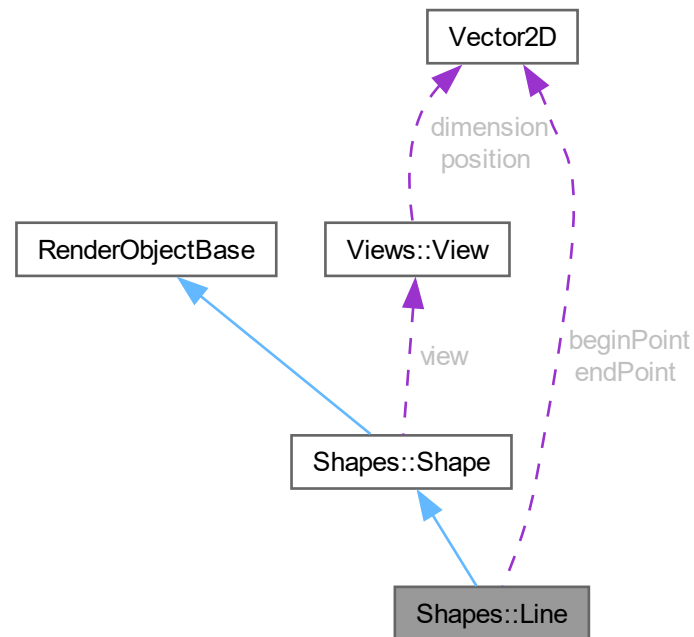
6.13 Shapes::Line Class Reference

```
#include <line.h>
```

Inheritance diagram for Shapes::Line:



Collaboration diagram for Shapes::Line:



Public Member Functions

- [Line](#) ([Views::View](#) *view, [Vector2D](#) _beginPoint, [Vector2D](#) _endPoint, uint8_t _thickness, [SDL_Color](#) color={0, 0, 0, 255}) noexcept
- void [setBeginPoint](#) ([Vector2D](#) newBeginPoint) noexcept
- void [setEndPoint](#) ([Vector2D](#) newEndPoint) noexcept
- void [setThickness](#) (uint8_t newThickness) noexcept
- void [draw](#) ([SDL_Renderer](#) *renderer) const noexcept override

Public Member Functions inherited from [Shapes::Shape](#)

- [Shape](#) ([Views::View](#) *view, const [SDL_Color](#) &color={ 0, 0, 0, 255 })
- virtual [~Shape](#) ()
- void [setColor](#) (const [SDL_Color](#) &newColor) noexcept
- [SDL_Color](#) [getColor](#) (void) const noexcept

Public Member Functions inherited from [RenderObjectBase](#)

- virtual void [debug](#) (void) const noexcept

Protected Attributes

- [Vector2D](#) [beginPoint](#)
- [Vector2D](#) [endPoint](#)
- uint8_t [thickness](#)

Protected Attributes inherited from [Shapes::Shape](#)

- const [Views::View](#) * [view](#)
Calls draw function after transforming coordinates with view.
- [SDL_Color](#) [color](#)

6.13.1 Constructor & Destructor Documentation

6.13.1.1 Line()

```
Shapes::Line::Line (
    Views::View * view,
    Vector2D _beginPoint,
    Vector2D _endPoint,
    uint8_t _thickness,
    SDL\_Color color = {0, 0, 0, 255} ) [noexcept]
```

6.13.2 Member Function Documentation

6.13.2.1 draw()

```
void Shapes::Line::draw (
    SDL\_Renderer * renderer ) const [override], [virtual], [noexcept]
```

Reimplemented from [Shapes::Shape](#).

6.13.2.2 setBeginPoint()

```
void Shapes::Line::setBeginPoint (
    Vector2D newBeginPoint ) [noexcept]
```

6.13.2.3 setEndPoint()

```
void Shapes::Line::setEndPoint (
    Vector2D newEndPoint ) [noexcept]
```

6.13.2.4 setThickness()

```
void Shapes::Line::setThickness (
    uint8_t newThickness ) [noexcept]
```

6.13.3 Member Data Documentation

6.13.3.1 beginPoint

```
Vector2D Shapes::Line::beginPoint [protected]
```

6.13.3.2 endPoint

```
Vector2D Shapes::Line::endPoint [protected]
```

6.13.3.3 thickness

```
uint8_t Shapes::Line::thickness [protected]
```

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

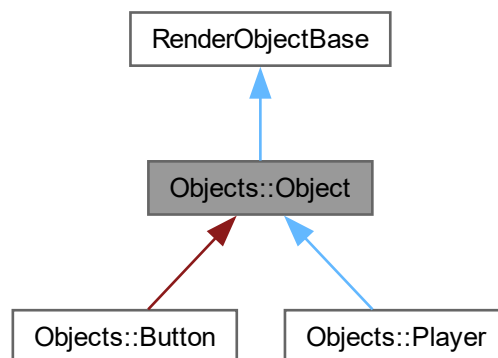
- [include/shape/line.h](#)

6.14 Objects::Object Class Reference

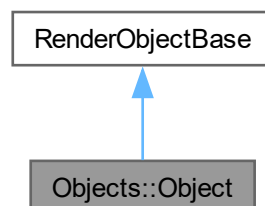
[Object](#) type for all renderable objects in the world note: the texture won't be created until loaded into the renderer.

```
#include <object.h>
```

Inheritance diagram for Objects::Object:



Collaboration diagram for Objects::Object:



Public Member Functions

- **Object** (const std::vector< std::string > &textureNames, const **Views::View** *_view, const **Vector2D** &_↵ position, const **Vector2D** &_dimension)
Constructs a new object.
- virtual **~Object** ()=default
- float **getAngle** (void) const noexcept
Returns the angle of the object in radians. The returned angle will be in [0, 2pi), with 0 set at positive x direction, and going counter-clockwise.
- void **setAngle** (float newAngle) noexcept
Sets rotation angle to.
- void **rotate** (float diffAngle) noexcept
Rotates the object by.
- **SDL_RendererFlip** **getFlipFlag** (void) const noexcept
Returns the flip flag used by SDL.
- **Vector2D** **getPosition** (void) const noexcept
Gets the position of the object.
- **Vector2D** **getDimension** (void) const noexcept
Gets the dimension of the object.
- void **move** (const **Vector2D** &translate) noexcept
Moves the object by the translate vector.
- void **stretchX** (float ratio) noexcept
Stretches the object's width by.
- void **stretchY** (float ratio) noexcept
Stretches the object's height by.
- void **stretch** (float ratio) noexcept
Stretches both the object's width and height by.
- void **flipHorizontal** (void) noexcept
Flips the object horizontally.
- void **flipVertical** (void) noexcept
Flips the object vertically.
- void **setVisibility** (bool visibility) noexcept
Sets the object's visibility.
- bool **getVisibility** (void) const noexcept
Gets the object's visibility.
- void **nextTexture** (void) noexcept
Set to next texture, texture ID wraps around.
- void **previousTexture** (void) noexcept
Set to previous texture, texture ID wraps around.
- void **setTexture** (int textureId) noexcept
Sets texture to.
- size_t **getTextureCount** (void) const noexcept
Gets the number of textures this object has.
- **SDL_Texture *** **getTexture** (void) const noexcept
Gets current texture.
- virtual void **lookAt** (const **Vector2D** &position) noexcept
Make the object face.
- **SDL_FRect** **getRenderRect** (void) const noexcept
Gets render rectangle for rendering.
- void **update** (void) noexcept
Updates the object state.
- void **debug** (void) const noexcept

Friends

- class [TextureHandler](#)

6.14.1 Detailed Description

[Object](#) type for all renderable objects in the world note: the texture won't be created until loaded into the renderer.

6.14.2 Constructor & Destructor Documentation

6.14.2.1 Object()

```
Objects::Object::Object (
    const std::vector< std::string > & textureNames,
    const Views::View * _view,
    const Vector2D & _position,
    const Vector2D & _dimension )
```

Constructs a new object.

Parameters

<i>textureNames</i>	The list of texture names.
<i>_view</i>	The viewport of the object.
<i>_position</i>	Initial position. (x, y)
<i>_dimension</i>	Initial Dimension. (width, height)

6.14.2.2 ~Object()

```
virtual Objects::Object::~~Object ( ) [virtual], [default]
```

6.14.3 Member Function Documentation

6.14.3.1 debug()

```
void Objects::Object::debug (
    void ) const [virtual], [noexcept]
```

Reimplemented from [RenderObjectBase](#).

6.14.3.2 flipHorizontal()

```
void Objects::Object::flipHorizontal (
    void ) [noexcept]
```

Flips the object horizontally.

6.14.3.3 flipVertical()

```
void Objects::Object::flipVertical (
    void ) [noexcept]
```

Flips the object vertically.

6.14.3.4 getAngle()

```
float Objects::Object::getAngle (
    void ) const [noexcept]
```

Returns the angle of the object in radians. The returned angle will be in $[0, 2\pi]$, with 0 set at positive x direction, and going counter-clockwise.

Returns

The angle which the object is facing.

6.14.3.5 getDimension()

```
Vector2D Objects::Object::getDimension (
    void ) const [noexcept]
```

Gets the dimension of the object.

Returns

The object's dimension.

6.14.3.6 getFlipFlag()

```
SDL_RendererFlip Objects::Object::getFlipFlag (
    void ) const [noexcept]
```

Returns the flip flag used by SDL.

Returns

A `SDL_RendererFlip` flag.

6.14.3.7 getPosition()

```
Vector2D Objects::Object::getPosition (
    void ) const [noexcept]
```

Gets the position of the object.

Returns

The object's location.

6.14.3.8 getRenderRect()

```
SDL_FRect Objects::Object::getRenderRect (
    void ) const [noexcept]
```

Gets render rectangle for rendering.

Returns

The SDL_FRect for rendering.

6.14.3.9 getTexture()

```
SDL_Texture * Objects::Object::getTexture (
    void ) const [noexcept]
```

Gets current texture.

Returns

The current texture the object is using.

6.14.3.10 getTextureCount()

```
size_t Objects::Object::getTextureCount (
    void ) const [noexcept]
```

Gets the number of textures this object has.

Returns

Number of textures.

6.14.3.11 getVisibility()

```
bool Objects::Object::getVisibility (
    void ) const [noexcept]
```

Gets the object's visibility.

Returns

The object's visibility.

6.14.3.12 lookAt()

```
virtual void Objects::Object::lookAt (
    const Vector2D & position ) [virtual], [noexcept]
```

Make the object face.

Parameters

<i>position</i>	coordinates.
<i>position</i>	The coordinate of where the object should look at.

6.14.3.13 move()

```
void Objects::Object::move (
    const Vector2D & translate ) [noexcept]
```

Moves the object by the translate vector.

Parameters

<i>translate</i>	The offset vector to move by.
------------------	-------------------------------

6.14.3.14 nextTexture()

```
void Objects::Object::nextTexture (
    void ) [noexcept]
```

Set to next texture, texture ID wraps around.

6.14.3.15 previousTexture()

```
void Objects::Object::previousTexture (
    void ) [noexcept]
```

Set to previous texture, texture ID wraps around.

6.14.3.16 rotate()

```
void Objects::Object::rotate (
    float diffAngle ) [noexcept]
```

Rotates the object by.

Parameters

<i>diffAngle</i>	radians in the counter-clockwise direction.
<i>diffAngle</i>	Rotation angle.

6.14.3.17 setAngle()

```
void Objects::Object::setAngle (
```

```
float newAngle ) [noexcept]
```

Sets rotation angle to.

Parameters

<i>newAngle</i>	radians.
<i>newAngle</i>	The new angle to set to. (in radians)

6.14.3.18 `setTexture()`

```
void Objects::Object::setTexture (
    int textureId ) [noexcept]
```

Sets texture to.

Parameters

<i>textureId.</i>	
<i>textureId</i>	The ID of the texture to be set. Should be in [0, textureCount).

6.14.3.19 `setVisibility()`

```
void Objects::Object::setVisibility (
    bool visibility ) [noexcept]
```

Sets the object's visibility.

Parameters

<i>visibility</i>	The object's visibility.
-------------------	--------------------------

6.14.3.20 `stretch()`

```
void Objects::Object::stretch (
    float ratio ) [noexcept]
```

Stretches both the object's width and height by.

Parameters

<i>ratio.</i>	
<i>ratio</i>	Stretch ratio.

6.14.3.21 stretchX()

```
void Objects::Object::stretchX (
    float ratio ) [noexcept]
```

Stretches the object's width by.

Parameters

<i>ratio</i> .	
<i>ratio</i>	Stretch ratio.

6.14.3.22 stretchY()

```
void Objects::Object::stretchY (
    float ratio ) [noexcept]
```

Stretches the object's height by.

Parameters

<i>ratio</i> .	
<i>ratio</i>	Stretch ratio.

6.14.3.23 update()

```
void Objects::Object::update (
    void ) [noexcept]
```

Updates the object state.

6.14.4 Friends And Related Symbol Documentation

6.14.4.1 TextureHandler

```
friend class TextureHandler [friend]
```

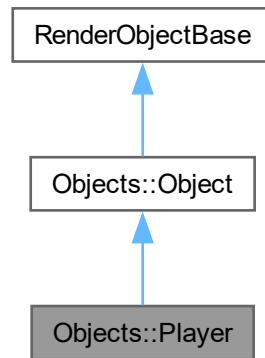
The documentation for this class was generated from the following file:

- include/object/[object.h](#)

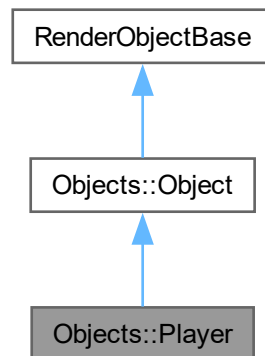
6.15 Objects::Player Class Reference

```
#include <player.h>
```

Inheritance diagram for Objects::Player:



Collaboration diagram for Objects::Player:



Additional Inherited Members

Public Member Functions inherited from [Objects::Object](#)

- [Object](#) (const std::vector< std::string > &textureNames, const [Views::View](#) *_view, const [Vector2D](#) &_↵ position, const [Vector2D](#) &_dimension)
Constructs a new object.

- virtual `~Object()` = default
- float `getAngle` (void) const noexcept
Returns the angle of the object in radians. The returned angle will be in $[0, 2\pi)$, with 0 set at positive x direction, and going counter-clockwise.
- void `setAngle` (float newAngle) noexcept
Sets rotation angle to.
- void `rotate` (float diffAngle) noexcept
Rotates the object by.
- SDL_RendererFlip `getFlipFlag` (void) const noexcept
Returns the flip flag used by SDL.
- `Vector2D` `getPosition` (void) const noexcept
Gets the position of the object.
- `Vector2D` `getDimension` (void) const noexcept
Gets the dimension of the object.
- void `move` (const `Vector2D` &translate) noexcept
Moves the object by the translate vector.
- void `stretchX` (float ratio) noexcept
Stretches the object's width by.
- void `stretchY` (float ratio) noexcept
Stretches the object's height by.
- void `stretch` (float ratio) noexcept
Stretches both the object's width and height by.
- void `flipHorizontal` (void) noexcept
Flips the object horizontally.
- void `flipVertical` (void) noexcept
Flips the object vertically.
- void `setVisibility` (bool visibility) noexcept
Sets the object's visibility.
- bool `getVisibility` (void) const noexcept
Gets the object's visibility.
- void `nextTexture` (void) noexcept
Set to next texture, texture ID wraps around.
- void `previousTexture` (void) noexcept
Set to previous texture, texture ID wraps around.
- void `setTexture` (int textureId) noexcept
Sets texture to.
- size_t `getTextureCount` (void) const noexcept
Gets the number of textures this object has.
- `SDL_Texture` * `getTexture` (void) const noexcept
Gets current texture.
- virtual void `lookAt` (const `Vector2D` &position) noexcept
Make the object face.
- `SDL_FRect` `getRenderRect` (void) const noexcept
Gets render rectangle for rendering.
- void `update` (void) noexcept
Updates the object state.
- void `debug` (void) const noexcept

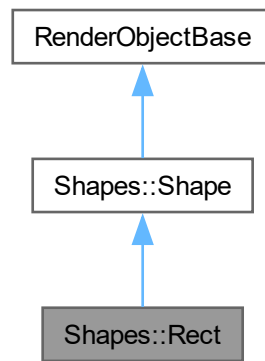
The documentation for this class was generated from the following file:

- include/object/player.h

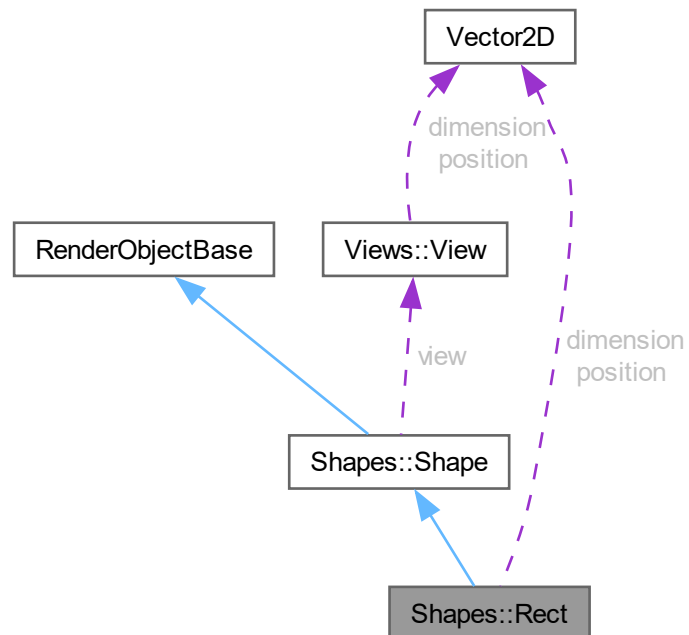
6.16 Shapes::Rect Class Reference

```
#include <rect.h>
```

Inheritance diagram for Shapes::Rect:



Collaboration diagram for Shapes::Rect:



Protected Attributes

- [Vector2D](#) position
- [Vector2D](#) dimension

Protected Attributes inherited from [Shapes::Shape](#)

- const [Views::View](#) * [view](#)
Calls draw function after transforming coordinates with view.
- [SDL_Color](#) [color](#)

Additional Inherited Members**Public Member Functions inherited from [Shapes::Shape](#)**

- virtual void [draw](#) ([SDL_Renderer](#) *renderer) const noexcept
- [Shape](#) ([Views::View](#) *view, const [SDL_Color](#) &color={ 0, 0, 0, 255 })
- virtual [~Shape](#) ()
- void [setColor](#) (const [SDL_Color](#) &newColor) noexcept
- [SDL_Color](#) [getColor](#) (void) const noexcept

Public Member Functions inherited from [RenderObjectBase](#)

- virtual void [debug](#) (void) const noexcept

6.16.1 Member Data Documentation**6.16.1.1 dimension**

[Vector2D](#) [Shapes::Rect::dimension](#) [protected]

6.16.1.2 position

[Vector2D](#) [Shapes::Rect::position](#) [protected]

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

- include/shape/[rect.h](#)

6.17 Renderer Class Reference

Required key to call [render\(\)](#) in.

```
#include <renderer.h>
```

Classes

- class [RenderKey](#)

Public Member Functions

- [Renderer](#) (const [Renderer](#) &)=delete
- void [operator=](#) (const [Renderer](#) &)=delete
- [SDL_Texture](#) * [createTexture](#) (CreateTextureKey key, [SDL_Surface](#) *surface) const
Creates a texture from a SDL_Surface.
- bool [registerObject](#) (std::shared_ptr< [RenderObjectBase](#) > objectPtr) noexcept
Get underlying SDL_Renderer renderer.
- bool [removeObject](#) (std::shared_ptr< [RenderObjectBase](#) > objectPtr) noexcept
Unregisters the object for rendering.
- void [render](#) ([RenderKey](#) key)
Renders every registered object. Note: SDL has built-in out of boundaries check.
- void [moveLayerUp](#) (std::shared_ptr< [RenderObjectBase](#) > objectPtr)
Moves the object up one layer. Throws std::invalid_argument.
- void [moveLayerDown](#) (std::shared_ptr< [RenderObjectBase](#) > objectPtr)
- void [moveLayerTop](#) (std::shared_ptr< [RenderObjectBase](#) > objectPtr)
- void [moveLayerBottom](#) (std::shared_ptr< [RenderObjectBase](#) > objectPtr)
- void [clear](#) () noexcept
Clears object set and unloads all textures.
- void [debug](#) (void) const noexcept
Prints renderer debug info.

Static Public Member Functions

- static [Renderer](#) & [getInstance](#) (void) noexcept

6.17.1 Detailed Description

Required key to call [render\(\)](#) in.

This is a global singleton class for rendering. Keeps track of current objects, shapes and renders everything onto a set window.

6.17.2 Constructor & Destructor Documentation

6.17.2.1 [Renderer\(\)](#)

```
Renderer::Renderer (
    const Renderer & ) [delete]
```

6.17.3 Member Function Documentation

6.17.3.1 clear()

```
void Renderer::clear ( ) [noexcept]
```

Clears object set and unloads all textures.

6.17.3.2 createTexture()

```
SDL_Texture * Renderer::createTexture (
    CreateTextureKey key,
    SDL_Surface * surface ) const
```

Creates a texture from a `SDL_Surface`.

Parameters

<i>key</i>	Required key to use this function.
<i>surface</i>	The source surface.

Returns

A pointer to the allocated `SDL_Texture` object.

6.17.3.3 debug()

```
void Renderer::debug (
    void ) const [noexcept]
```

Prints renderer debug info.

6.17.3.4 getInstance()

```
static Renderer & Renderer::getInstance (
    void ) [static], [noexcept]
```

6.17.3.5 moveLayerBottom()

```
void Renderer::moveLayerBottom (
    std::shared_ptr< RenderObjectBase > objectPtr )
```

6.17.3.6 moveLayerDown()

```
void Renderer::moveLayerDown (
    std::shared_ptr< RenderObjectBase > objectPtr )
```

6.17.3.7 moveLayerTop()

```
void Renderer::moveLayerTop (
    std::shared_ptr< RenderObjectBase > objectPtr )
```

6.17.3.8 moveLayerUp()

```
void Renderer::moveLayerUp (
    std::shared_ptr< RenderObjectBase > objectPtr )
```

Moves the object up one layer. Throws `std::invalid_argument`.

Parameters

<i>objectPtr</i>	
------------------	--

6.17.3.9 operator=()

```
void Renderer::operator= (
    const Renderer & ) [delete]
```

6.17.3.10 registerObject()

```
bool Renderer::registerObject (
    std::shared_ptr< RenderObjectBase > objectPtr ) [noexcept]
```

Get underlying `SDL_Renderer` renderer.

Returns

The underlying renderer.

Registers the object for rendering.

Parameters

<i>objectPtr</i>	<code>std::shared_ptr</code> of the object
------------------	--

Returns

Whether the object was successfully registered

6.17.3.11 removeObject()

```
bool Renderer::removeObject (
    std::shared_ptr< RenderObjectBase > objectPtr ) [noexcept]
```

Unregisters the object for rendering.

Parameters

<i>objectPtr</i>	std::shared_ptr of the object
------------------	-------------------------------

Returns

Whether the object was successfully unregistered.

6.17.3.12 render()

```
void Renderer::render (
    RenderKey key )
```

Renders every registered object. Note: SDL has built-in out of boundaries check.

Parameters

<i>key</i>	Access Control Key
------------	--------------------

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

- include/[renderer.h](#)

6.18 Renderer::RenderKey Class Reference

```
#include <renderer.h>
```

Public Member Functions

- [RenderKey](#) ()=default
- [RenderKey](#) (const [RenderKey](#) &)=default

6.18.1 Constructor & Destructor Documentation**6.18.1.1 RenderKey() [1/2]**

```
Renderer::RenderKey::RenderKey ( ) [default]
```

6.18.1.2 RenderKey() [2/2]

```
Renderer::RenderKey::RenderKey (
    const RenderKey & ) [default]
```

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

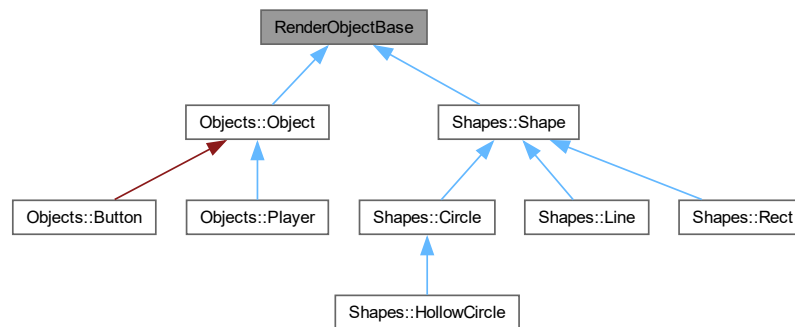
- include/[renderer.h](#)

6.19 RenderObjectBase Class Reference

Empty render object base class category.

```
#include <render_object_base.h>
```

Inheritance diagram for RenderObjectBase:



Public Member Functions

- virtual void [debug](#) (void) const noexcept

6.19.1 Detailed Description

Empty render object base class category.

6.19.2 Member Function Documentation

6.19.2.1 debug()

```
virtual void RenderObjectBase::debug (
    void ) const [virtual], [noexcept]
```

Reimplemented in [Objects::Object](#).

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

- include/[render_object_base.h](#)

6.20 sdl_deleter Struct Reference

Generic deleter functor for SDL resources. For use with std smart pointers.

```
#include <pointer_wrappers.h>
```

Public Member Functions

- void [operator\(\)](#) (SDL_RWops *thing) const noexcept
- void [operator\(\)](#) (SDL_cond *thing) const noexcept
- void [operator\(\)](#) (SDL_Cursor *thing) const noexcept
- void [operator\(\)](#) (SDL_PixelFormat *thing) const noexcept
- void [operator\(\)](#) (SDL_mutex *thing) const noexcept
- void [operator\(\)](#) (SDL_Palette *thing) const noexcept
- void [operator\(\)](#) (SDL_Renderer *thing) const noexcept
- void [operator\(\)](#) (SDL_sem *thing) const noexcept
- void [operator\(\)](#) (SDL_Surface *thing) const noexcept
- void [operator\(\)](#) (SDL_Texture *thing) const noexcept
- void [operator\(\)](#) (Uint8 *thing) const noexcept
- void [operator\(\)](#) (SDL_Window *thing) const noexcept

6.20.1 Detailed Description

Generic deleter functor for SDL resources. For use with std smart pointers.

6.20.2 Member Function Documentation

6.20.2.1 [operator\(\)](#) [1/12]

```
void sdl_deleter::operator() (
    SDL_cond * thing ) const    [inline], [noexcept]
```

6.20.2.2 [operator\(\)](#) [2/12]

```
void sdl_deleter::operator() (
    SDL_Cursor * thing ) const    [inline], [noexcept]
```

6.20.2.3 [operator\(\)](#) [3/12]

```
void sdl_deleter::operator() (
    SDL_mutex * thing ) const    [inline], [noexcept]
```

6.20.2.4 [operator\(\)](#) [4/12]

```
void sdl_deleter::operator() (
    SDL_Palette * thing ) const    [inline], [noexcept]
```

6.20.2.5 [operator\(\)](#) [5/12]

```
void sdl_deleter::operator() (
    SDL_PixelFormat * thing ) const    [inline], [noexcept]
```

6.20.2.6 operator>() [6/12]

```
void sdl_deleter::operator() (
    SDL_Renderer * thing ) const [inline], [noexcept]
```

6.20.2.7 operator>() [7/12]

```
void sdl_deleter::operator() (
    SDL_RWops * thing ) const [inline], [noexcept]
```

6.20.2.8 operator>() [8/12]

```
void sdl_deleter::operator() (
    SDL_sem * thing ) const [inline], [noexcept]
```

6.20.2.9 operator>() [9/12]

```
void sdl_deleter::operator() (
    SDL_Surface * thing ) const [inline], [noexcept]
```

6.20.2.10 operator>() [10/12]

```
void sdl_deleter::operator() (
    SDL_Texture * thing ) const [inline], [noexcept]
```

6.20.2.11 operator>() [11/12]

```
void sdl_deleter::operator() (
    SDL_Window * thing ) const [inline], [noexcept]
```

6.20.2.12 operator>() [12/12]

```
void sdl_deleter::operator() (
    Uint8 * thing ) const [inline], [noexcept]
```

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

- [include/utility/pointer_wrappers.h](#)

6.21 SelectionManager< T > Class Template Reference

```
#include <selection_manager.h>
```


Public Member Functions

- [SelectionManager](#) ()
- [SelectionManager](#) (const std::vector< T > &selections)
- void [next](#) (void) const noexcept
Set to next selection.
- void [prev](#) (void) const noexcept
Set to previous selection.
- void [set](#) (int newSelection) const
Set current selection ID to.
- size_t [size](#) (void) const noexcept
Gets the count of available selections.
- void [add](#) (T newSelection) noexcept
Adds.
- void [remove](#) (size_t selectionId)
Removes the selection at.
- T [get](#) (void) const
Gets the current selection. Throws std::logic_error is current selection is SELECTION_NOT_SET.
- int [getSelectionId](#) (void) const noexcept
Gets the current selection ID.

Static Public Attributes

- static const int [SELECTION_NOT_SET](#) = -1

6.21.1 Constructor & Destructor Documentation

6.21.1.1 SelectionManager() [1/2]

```
template<class T >
SelectionManager< T >::SelectionManager ( )
```

6.21.1.2 SelectionManager() [2/2]

```
template<class T >
SelectionManager< T >::SelectionManager (
    const std::vector< T > & selections )
```

6.21.2 Member Function Documentation

6.21.2.1 add()

```
template<class T >
void SelectionManager< T >::add (
    T newSelection ) [noexcept]
```

Adds.

Parameters

<i>newSelection</i>	to the manager.
<i>newSelection</i>	The new selection.

6.21.2.2 get()

```
template<class T >
T SelectionManager< T >::get (
    void ) const
```

Gets the current selection. Throws `std::logic_error` is current selection is `SELECTION_NOT_SET`.

Returns

The current selection.

6.21.2.3 getSelectionId()

```
template<class T >
int SelectionManager< T >::getSelectionId (
    void ) const [noexcept]
```

Gets the current selection ID.

Returns

The current selection ID.

6.21.2.4 next()

```
template<class T >
void SelectionManager< T >::next (
    void ) const [noexcept]
```

Set to next selection.

6.21.2.5 prev()

```
template<class T >
void SelectionManager< T >::prev (
    void ) const [noexcept]
```

Set to previous selection.

6.21.2.6 remove()

```
template<class T >
void SelectionManager< T >::remove (
    size_t selectionId )
```

Removes the selection at.

Parameters

<i>selectionId.</i>	Throws std::out_of_range if selectionId is invalid.
<i>selectionId</i>	The position of where the selection is at.

6.21.2.7 set()

```
template<class T >
void SelectionManager< T >::set (
    int newSelection ) const
```

Set current selection ID to.

Parameters

<i>newSelection.</i>	Throws std::out_of_range if ID is not in range of [0, size) or SELECTION_NOT_SET.
<i>newSelection</i>	The new selection ID.

6.21.2.8 size()

```
template<class T >
size_t SelectionManager< T >::size (
    void ) const [noexcept]
```

Gets the count of available selections.

Returns

The count of available selections.

6.21.3 Member Data Documentation**6.21.3.1 SELECTION_NOT_SET**

```
template<class T >
const int SelectionManager< T >::SELECTION_NOT_SET = -1 [static]
```

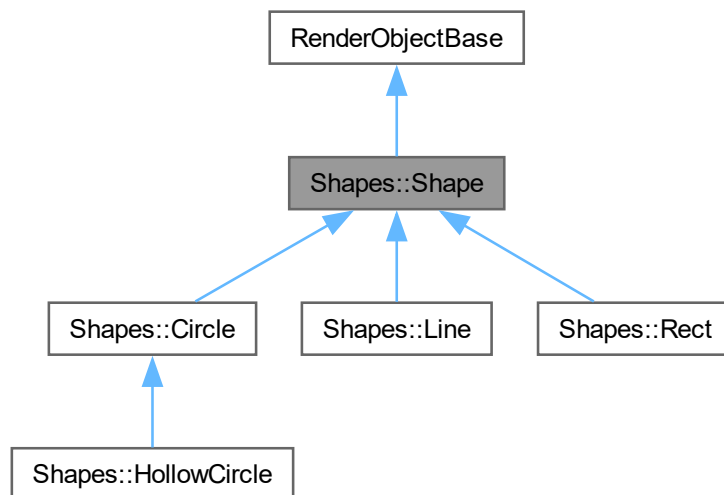
The documentation for this class was generated from the following file:

- include/utility/[selection_manager.h](#)

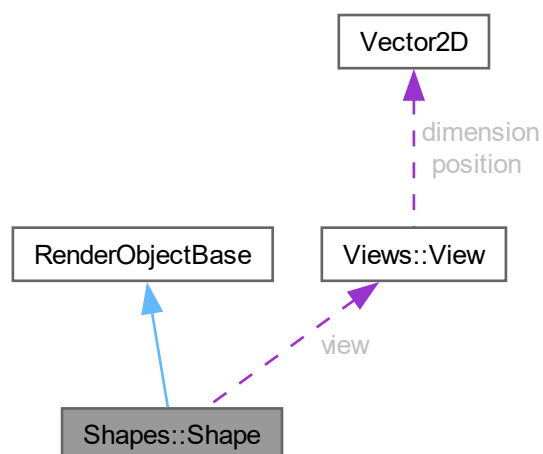
6.22 Shapes::Shape Class Reference

```
#include <shape.h>
```

Inheritance diagram for Shapes::Shape:



Collaboration diagram for Shapes::Shape:



Public Member Functions

- virtual void [draw](#) (SDL_Renderer *renderer) const noexcept
- [Shape](#) ([Views::View](#) *view, const SDL_Color &color={ 0, 0, 0, 255 })
- virtual [~Shape](#) ()
- void [setColor](#) (const SDL_Color &newColor) noexcept
- SDL_Color [getColor](#) (void) const noexcept

Public Member Functions inherited from [RenderObjectBase](#)

- virtual void [debug](#) (void) const noexcept

Protected Attributes

- const [Views::View](#) * [view](#)
Calls draw function after transforming coordinates with view.
- SDL_Color [color](#)

6.22.1 Constructor & Destructor Documentation

6.22.1.1 Shape()

```
Shapes::Shape::Shape (
    Views::View * view,
    const SDL_Color & color = { 0, 0, 0, 255 } )
```

6.22.1.2 ~Shape()

```
virtual Shapes::Shape::~Shape ( ) [inline], [virtual]
```

6.22.2 Member Function Documentation

6.22.2.1 draw()

```
virtual void Shapes::Shape::draw (
    SDL_Renderer * renderer ) const [inline], [virtual], [noexcept]
```

Reimplemented in [Shapes::Circle](#), [Shapes::HollowCircle](#), and [Shapes::Line](#).

6.22.2.2 getColor()

```
SDL_Color Shapes::Shape::getColor (
    void ) const [noexcept]
```

6.22.2.3 setColor()

```
void Shapes::Shape::setColor (
    const SDL_Color & newColor ) [noexcept]
```

6.22.3 Member Data Documentation

6.22.3.1 color

```
SDL_Color Shapes::Shape::color [protected]
```

6.22.3.2 view

```
const Views::View* Shapes::Shape::view [protected]
```

Calls draw function after transforming coordinates with view.

Parameters

<i>view</i>	The target view port.
-------------	-----------------------

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

- include/shape/[shape.h](#)

6.23 TextureHandler Class Reference

This is a global singleton class for texture handling.

```
#include <texture_handler.h>
```

Public Member Functions

- `SDL_Texture * getTexture (TextureRequestKey key, const std::string &textureName)`
Gets a weak pointer pointing to the requested texture.
- `TextureHandler (const TextureHandler &)=delete`
- `void operator= (const TextureHandler &)=delete`

Static Public Member Functions

- `static TextureHandler & getInstance (void)`

6.23.1 Detailed Description

This is a global singleton class for texture handling.

Required key to request texture from.

6.23.2 Constructor & Destructor Documentation

6.23.2.1 TextureHandler()

```
TextureHandler::TextureHandler (
    const TextureHandler & ) [delete]
```

6.23.3 Member Function Documentation

6.23.3.1 getInstance()

```
static TextureHandler & TextureHandler::getInstance (
    void ) [static]
```

6.23.3.2 getTexture()

```
SDL_Texture * TextureHandler::getTexture (
    TextureRequestKey key,
    const std::string & textureName )
```

Gets a weak pointer pointing to the requested texture.

Parameters

<i>key</i>	Access Control Key
<i>textureName</i>	The name of the texture.

Returns

The raw pointer of the requested texture.

6.23.3.3 operator=()

```
void TextureHandler::operator= (
    const TextureHandler & ) [delete]
```

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

- include/texture/[texture_handler.h](#)

6.24 Vector2D Class Reference

```
#include <vector2d.h>
```

Public Member Functions

- [Vector2D](#) (void) noexcept
- [Vector2D](#) (float _x, float _y) noexcept
- float [getX](#) (void) const noexcept
- float [getY](#) (void) const noexcept
- [Vector2D norm](#) (void) const noexcept
- float [len](#) (void) const noexcept
- float [len2](#) (void) const noexcept
- [Vector2D rotate](#) (float theta) const noexcept

Static Public Member Functions

- static [Vector2D zero](#) (void) noexcept
- static float [dot](#) (const [Vector2D](#) &, const [Vector2D](#) &) noexcept
- static float [cross](#) (const [Vector2D](#) &, const [Vector2D](#) &) noexcept
- static [Vector2D rotate](#) ([Vector2D](#), float) noexcept

Friends

- [Vector2D operator+](#) (const [Vector2D](#) &, const [Vector2D](#) &) noexcept
- [Vector2D operator-](#) (const [Vector2D](#) &) noexcept
- [Vector2D operator-](#) (const [Vector2D](#) &, const [Vector2D](#) &) noexcept
- [Vector2D operator*](#) (const [Vector2D](#) &, float) noexcept
- [Vector2D operator/](#) (const [Vector2D](#) &, float) noexcept
- [Vector2D & operator+=](#) ([Vector2D](#) &, const [Vector2D](#) &) noexcept
- [Vector2D & operator-=](#) ([Vector2D](#) &, const [Vector2D](#) &) noexcept
- [Vector2D & operator*=](#) ([Vector2D](#) &, float) noexcept
- [Vector2D & operator/=](#) ([Vector2D](#) &, float) noexcept

6.24.1 Constructor & Destructor Documentation

6.24.1.1 [Vector2D\(\)](#) [1/2]

```
Vector2D::Vector2D (  
    void ) [noexcept]
```

6.24.1.2 [Vector2D\(\)](#) [2/2]

```
Vector2D::Vector2D (  
    float _x,  
    float _y ) [noexcept]
```


6.24.2 Member Function Documentation

6.24.2.1 cross()

```
static float Vector2D::cross (
    const Vector2D & ,
    const Vector2D & ) [static], [noexcept]
```

6.24.2.2 dot()

```
static float Vector2D::dot (
    const Vector2D & ,
    const Vector2D & ) [static], [noexcept]
```

6.24.2.3 getX()

```
float Vector2D::getX (
    void ) const [noexcept]
```

6.24.2.4 getY()

```
float Vector2D::getY (
    void ) const [noexcept]
```

6.24.2.5 len()

```
float Vector2D::len (
    void ) const [noexcept]
```

6.24.2.6 len2()

```
float Vector2D::len2 (
    void ) const [noexcept]
```

6.24.2.7 norm()

```
Vector2D Vector2D::norm (
    void ) const [noexcept]
```

6.24.2.8 rotate() [1/2]

```
Vector2D Vector2D::rotate (
    float theta ) const [noexcept]
```

6.24.2.9 rotate() [2/2]

```
static Vector2D Vector2D::rotate (
    Vector2D ,
    float ) [static], [noexcept]
```

6.24.2.10 zero()

```
static Vector2D Vector2D::zero (
    void ) [static], [noexcept]
```

6.24.3 Friends And Related Symbol Documentation

6.24.3.1 operator*

```
Vector2D operator* (
    const Vector2D & ,
    float ) [friend]
```

6.24.3.2 operator*==

```
Vector2D & operator*== (
    Vector2D & ,
    float ) [friend]
```

6.24.3.3 operator+

```
Vector2D operator+ (
    const Vector2D & ,
    const Vector2D & ) [friend]
```

6.24.3.4 operator+=

```
Vector2D & operator+= (
    Vector2D & ,
    const Vector2D & ) [friend]
```

6.24.3.5 operator- [1/2]

```
Vector2D operator- (
    const Vector2D & ) [friend]
```

6.24.3.6 operator- [2/2]

```
Vector2D operator- (
    const Vector2D & ,
    const Vector2D & ) [friend]
```

6.24.3.7 operator-=

```
Vector2D & operator-= (
    Vector2D & ,
    const Vector2D & ) [friend]
```

6.24.3.8 operator/

```
Vector2D operator/ (
    const Vector2D & ,
    float ) [friend]
```

6.24.3.9 operator/=

```
Vector2D & operator/= (
    Vector2D & ,
    float ) [friend]
```

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

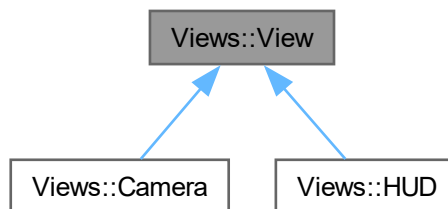
- include/utility/[vector2d.h](#)

6.25 Views::View Class Reference

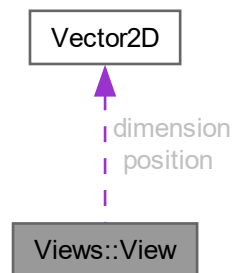
[View](#): defines a view area, translates the objects' virtual rects to real rendering rects.

```
#include <view.h>
```

Inheritance diagram for Views::View:



Collaboration diagram for Views::View:



Public Member Functions

- virtual [~View](#) ()
- virtual `SDL_FRect` [getRect](#) (const [Objects::Object](#) &object) const noexcept=0
Gets the render rect for.
- virtual [Vector2D](#) [transform](#) (const [Vector2D](#) &position) const noexcept=0
Gets the transformed render position of.
- virtual [Vector2D](#) [transformFromRender](#) (const [Vector2D](#) &renderPosition) const noexcept=0
Gets the virtual position of.
- virtual [Vector2D](#) [getPosition](#) (void) const noexcept
Gets the virtual position of the view.
- virtual [Vector2D](#) [getDimension](#) (void) const noexcept
Gets the virtual dimension of the view.
- virtual float [getAngle](#) (void) const noexcept
Gets the rotation angle of the view.
- virtual float [getZoom](#) (void) const noexcept
Gets the zoom level of the view.

Protected Member Functions

- [View](#) (const [Vector2D](#) &_position, const [Vector2D](#) &_dimension)

Protected Attributes

- [Vector2D](#) position
- [Vector2D](#) dimension

6.25.1 Detailed Description

[View](#): defines a view area, translates the objects' virtual rects to real rendering rects.

6.25.2 Constructor & Destructor Documentation

6.25.2.1 View()

```
Views::View::View (
    const Vector2D & _position,
    const Vector2D & _dimension ) [inline], [protected]
```

6.25.2.2 ~View()

```
virtual Views::View::~~View ( ) [inline], [virtual]
```

6.25.3 Member Function Documentation

6.25.3.1 getAngle()

```
virtual float Views::View::getAngle (
    void ) const [inline], [virtual], [noexcept]
```

Gets the rotation angle of the view.

Returns

The virtual angle of the view.

Reimplemented in [Views::Camera](#).

6.25.3.2 getDimension()

```
virtual Vector2D Views::View::getDimension (
    void ) const [inline], [virtual], [noexcept]
```

Gets the virtual dimension of the view.

Returns

The virtual dimension of the view.

6.25.3.3 getPosition()

```
virtual Vector2D Views::View::getPosition (
    void ) const [inline], [virtual], [noexcept]
```

Gets the virtual position of the view.

Returns

The virtual position of the view.

6.25.3.4 getRect()

```
virtual SDL\_FRect Views::View::getRect (
    const Objects::Object & object ) const [pure virtual], [noexcept]
```

Gets the render rect for.

Parameters

<i>object.</i>	
<i>object</i>	The object to be rendered.

Returns

The render rect of
object.

Implemented in [Views::HUD](#), and [Views::Camera](#).

6.25.3.5 getZoom()

```
virtual float Views::View::getZoom (
    void ) const [inline], [virtual], [noexcept]
```

Gets the zoom level of the view.

Returns

The zoom level of the view.

Reimplemented in [Views::Camera](#).

6.25.3.6 transform()

```
virtual Vector2D Views::View::transform (
    const Vector2D & position ) const [pure virtual], [noexcept]
```

Gets the transformed render position of.

Parameters

<i>position.</i>	
<i>position</i>	The virtual position to be transformed.

Returns

The render position after transformation.

Implemented in [Views::Camera](#), and [Views::HUD](#).

6.25.3.7 transformFromRender()

```
virtual Vector2D Views::View::transformFromRender (
    const Vector2D & renderPosition ) const [pure virtual], [noexcept]
```

Gets the virtual position of.

Parameters

<i>renderPosition.</i>	
<i>renderPosition</i>	The render position to be transformed

Returns

The virtual position after transformation.

Implemented in [Views::Camera](#), and [Views::HUD](#).

6.25.4 Member Data Documentation

6.25.4.1 dimension

[Vector2D](#) Views::View::dimension [protected]

6.25.4.2 position

[Vector2D](#) Views::View::position [protected]

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

- include/view/[view.h](#)

Chapter 7

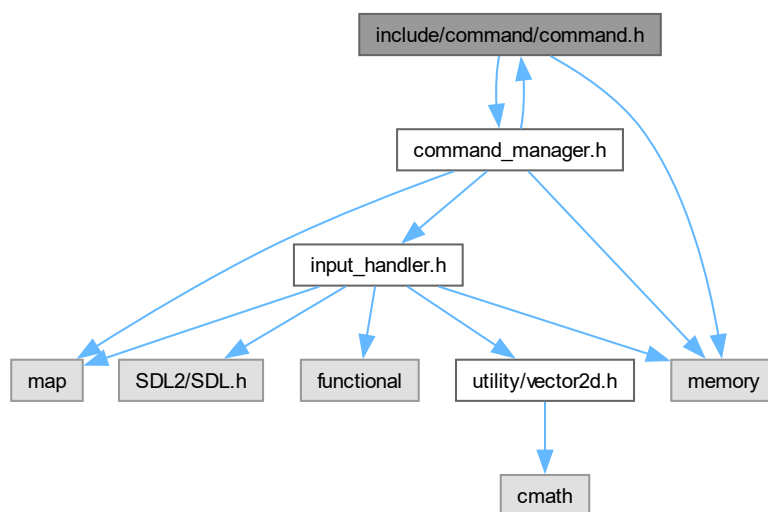
File Documentation

7.1 include/command/command.h File Reference

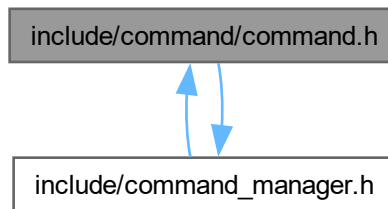
```
#include <command_manager.h>
```

```
#include <memory>
```

Include dependency graph for command.h:



This graph shows which files directly or indirectly include this file:



Classes

- class [Commands::Command](#)
Commands base abstract class.
- class [Commands::Command::ExecuteKey](#)

Namespaces

- namespace [Commands](#)

7.2 command.h

[Go to the documentation of this file.](#)

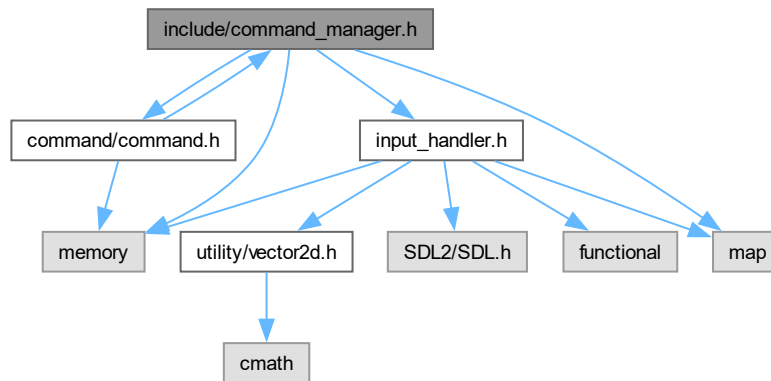
```

00001 #pragma once
00002
00003 #include <command_manager.h>
00004 #include <memory>
00005
00006 class CommandManager;
00007
00008 namespace Commands {
00009
00013     class Command {
00014     protected:
00015         class ExecuteKey {
00016             friend class CommandManager;
00017         private:
00018             ExecuteKey() = default;
00019             ExecuteKey(const ExecuteKey&) = default;
00020         };
00021     public:
00022         virtual ~Command() {};
00023         virtual void execute(ExecuteKey) {};
00024     };
00025 }
  
```

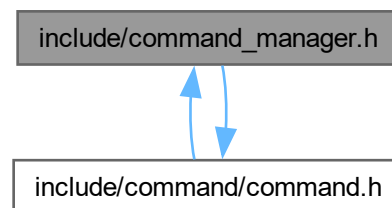
7.3 include/command_manager.h File Reference

```
#include <command/command.h>
#include <input_handler.h>
#include <map>
#include <memory>
```

Include dependency graph for command_manager.h:



This graph shows which files directly or indirectly include this file:



Classes

- struct [KeyBind](#)
KeyBind structure for key bindings.
- class [CommandManager](#)
Manages a map from key bindings to various functions. e.g. `player.move()`, `currentScene.set(mainMenu)`, or `renderer.drawCone()`.

Namespaces

- namespace [Commands](#)

7.4 command_manager.h

[Go to the documentation of this file.](#)

```

00001 #pragma once
00002
00003 #include <command/command.h>
00004 #include <input_handler.h>
00005 #include <map>
00006 #include <memory>
00007
00008 namespace Commands { class Command; }
00009
00010 enum class MouseButton : uint8_t;
00014 struct KeyBind {
00015     static unsigned int KeyBindCount;
00016     int ID; // only used for sorting
00017     enum class Trigger { TAP, HOLD, RELEASE };
00018     std::map<SDL_Keycode, Trigger> keys;
00019     std::map<MouseButton, Trigger> buttons;
00020     KeyBind(const std::map<SDL_Keycode, Trigger>& keys, const std::map<MouseButton, Trigger> buttons):
00021         keys(keys), buttons(buttons) {
00022         ID = KeyBind::KeyBindCount++;
00023     }
00024     friend bool operator < (const KeyBind& a, const KeyBind& b) {
00025         return a.ID < b.ID;
00026     }
00027 };
00028
00033 class CommandManager {
00034 private:
00035     std::map<KeyBind, std::shared_ptr<Commands::Command> commandDB;
00036 public:
00037
00044     bool registerCommand(KeyBind keyBind, std::shared_ptr<Commands::Command> command);
00045
00050     void update() noexcept;
00051 };

```

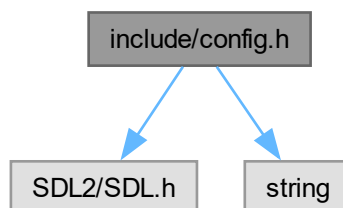
7.5 include/config.h File Reference

```

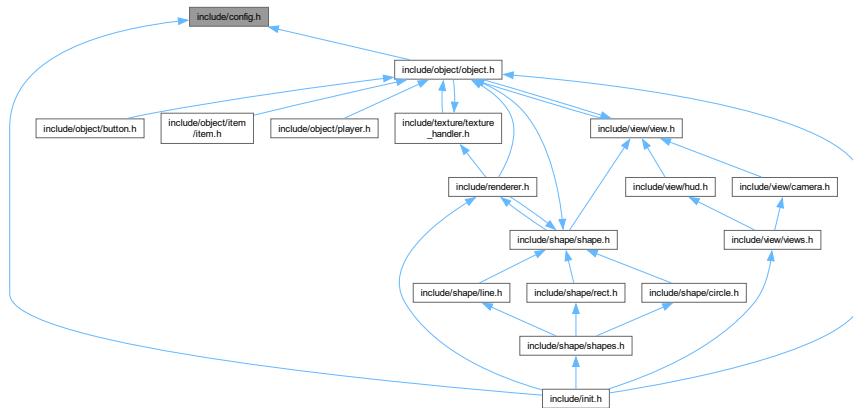
#include <SDL2/SDL.h>
#include <string>

```

Include dependency graph for config.h:



This graph shows which files directly or indirectly include this file:



Namespaces

- namespace [Config](#)

Variables

- const std::string [Config::gameTitle](#) = "Lab Raid"
- const int [Config::screenWidth](#) = 1280
- const int [Config::screenHeight](#) = 768
- const int [Config::volume](#) = 50
- const int [Config::framerate](#) = 60
- const float [Config::holdTimeThreshold](#) = 100
- const SDL_WindowFlags [Config::screenType](#) = SDL_WINDOW_SHOWN
- const SDL_Color [Config::backgroundColor](#) { 0x3F, 0x3F, 0x3F, 0xFF }

7.6 config.h

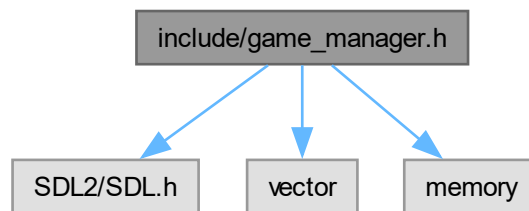
[Go to the documentation of this file.](#)

```
00001 #pragma once
00002
00003 #include <SDL2/SDL.h>
00004 #include <string>
00005
00006 namespace Config {
00007     const std::string gameTitle = "Lab Raid";
00008     const int screenWidth = 1280;
00009     const int screenHeight = 768;
00010     const int volume = 50;
00011     const int framerate = 60;
00012     const float holdTimeThreshold = 100;
00013     const SDL_WindowFlags screenType = SDL_WINDOW_SHOWN;
00014     //const SDL_Color backgroundColor{ 0x1F, 0x1E, 0x33, 0x7F };
00015     const SDL_Color backgroundColor{ 0x3F, 0x3F, 0x3F, 0xFF };
00016 }
```

7.7 include/game_manager.h File Reference

```
#include <SDL2/SDL.h>
#include <vector>
#include <memory>
```

Include dependency graph for game_manager.h:



Classes

- class [GameManager](#)

7.8 game_manager.h

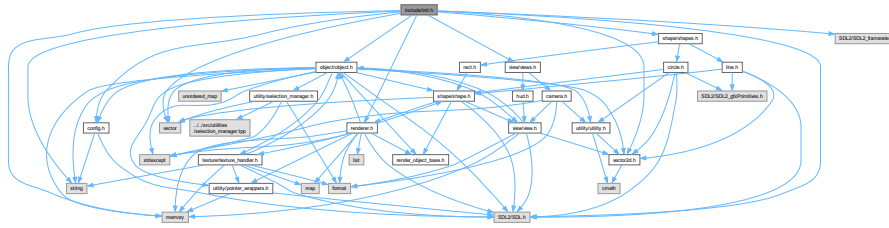
[Go to the documentation of this file.](#)

```
00001 #pragma once
00002
00003 #include <SDL2/SDL.h>
00004 #include <vector>
00005 #include <memory>
00006
00007 class GameManager {
00008 private:
00009     bool paused;
00010     enum {
00011         GAME_TITLE = 1,
00012         GAME_LEVEL = 2,
00013         GAME_END = 3
00014     } state;
00015
00016 };
```

7.9 include/init.h File Reference

```
#include <object/object.h>
#include <view/views.h>
#include <renderer.h>
#include <config.h>
#include <utility/vector2d.h>
#include <shape/shapes.h>
#include <SDL2/SDL.h>
#include <SDL2/SDL2_framerate.h>
```

```
#include <memory>
#include <string>
#include <vector>
Include dependency graph for init.h:
```



Namespaces

- namespace **Global**

Functions

- void Global::init ()

Variables

- std::unique_ptr< FPSManager > Global::fpsManager
- std::unique_ptr< Views::Camera > Global::playerCamera
- std::unique_ptr< Views::HUD > Global::hudView
- std::unique_ptr< Views::HUD > Global::menuView
- std::shared_ptr< Objects::Object > Global::playerObject
- std::shared_ptr< Objects::Object > Global::arrowObject1
- std::shared_ptr< Objects::Object > Global::arrowObject2
- std::shared_ptr< Shapes::Circle > Global::yellowCircle
- std::shared_ptr< Shapes::Circle > Global::greenCircle
- std::shared_ptr< Shapes::Circle > Global::blueCircle
- std::shared_ptr< Shapes::Circle > Global::redCircle
- std::shared_ptr< Shapes::Circle > Global::purpleCircle
- std::shared_ptr< Shapes::HollowCircle > Global::hollowCircle1
- std::shared_ptr< Shapes::Line > Global::line1
- std::shared_ptr< Shapes::Line > Global::line2
- std::shared_ptr< Shapes::Line > Global::line3
- std::shared_ptr< Shapes::Line > Global::line4
- std::shared_ptr< Shapes::Line > Global::crosshairLine1
- std::shared_ptr< Shapes::Line > Global::crosshairLine2
- std::shared_ptr< Shapes::HollowCircle > Global::crosshairCircle1

7.10 init.h

[Go to the documentation of this file.](#)

```

00001 #pragma once
00002
00003 #include <object/object.h>
00004 #include <view/views.h>
00005 #include <renderer.h>
00006 #include <config.h>
00007 #include <utility/vector2d.h>
00008 #include <shape/shapes.h>
00009 #include <SDL2/SDL.h>
00010 #include <SDL2/SDL2_framerate.h>
00011 #include <memory>
00012 #include <string>
00013 #include <vector>
00014
00015 namespace Global {
00016     extern std::unique_ptr<FPSmanager> fpsManager;
00017     extern std::unique_ptr<Views::Camera> playerCamera;
00018     extern std::unique_ptr<Views::HUD> hudView;
00019     extern std::unique_ptr<Views::HUD> menuView;
00020
00021     extern std::shared_ptr<Objects::Object> playerObject, arrowObject1;
00022     extern std::shared_ptr<Objects::Object> arrowObject2;
00023     extern std::shared_ptr<Shapes::Circle> yellowCircle;
00024     extern std::shared_ptr<Shapes::Circle> greenCircle;
00025     extern std::shared_ptr<Shapes::Circle> blueCircle;
00026     extern std::shared_ptr<Shapes::Circle> redCircle;
00027     extern std::shared_ptr<Shapes::Circle> purpleCircle;
00028
00029     extern std::shared_ptr<Shapes::HollowCircle> hollowCircle1;
00030     extern std::shared_ptr<Shapes::Line> line1;
00031     extern std::shared_ptr<Shapes::Line> line2;
00032     extern std::shared_ptr<Shapes::Line> line3;
00033     extern std::shared_ptr<Shapes::Line> line4;
00034
00035     extern std::shared_ptr<Shapes::Line> crosshairLine1;
00036     extern std::shared_ptr<Shapes::Line> crosshairLine2;
00037     extern std::shared_ptr<Shapes::HollowCircle> crosshairCircle1;
00038
00039     void init();
00040 }

```

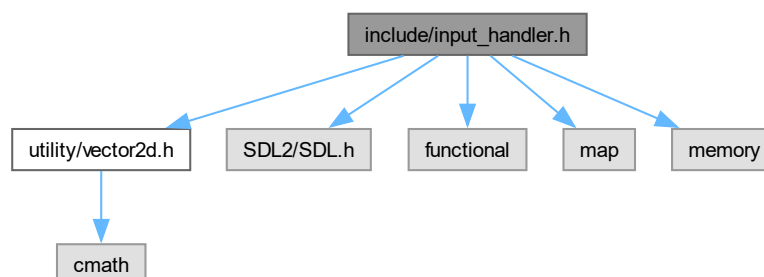
7.11 include/input_handler.h File Reference

```

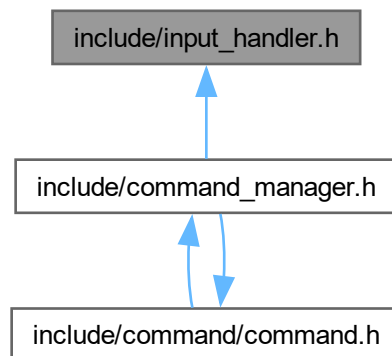
#include <utility/vector2d.h>
#include <SDL2/SDL.h>
#include <functional>
#include <map>
#include <memory>

```

Include dependency graph for input_handler.h:



This graph shows which files directly or indirectly include this file:



Classes

- class [InputHandler](#)

This is a global singleton class of handling user inputs. Wrapper class of `SDL_PollEvent` and events handling.

Enumerations

- enum class [MouseButton](#) : `uint8_t` {
`LEFT` = `SDL_BUTTON_LEFT` , `MIDDLE` = `SDL_BUTTON_MIDDLE` , `RIGHT` = `SDL_BUTTON_RIGHT` , `X1` = `SDL_BUTTON_X1` ,
`X2` = `SDL_BUTTON_X2` }

7.11.1 Enumeration Type Documentation

7.11.1.1 MouseButton

```
enum class MouseButton : uint8_t [strong]
```

Enumerator

LEFT	
MIDDLE	
RIGHT	
X1	
X2	

7.12 input_handler.h

[Go to the documentation of this file.](#)

```

00001 #pragma once
00002
00003 #include <utility/vector2d.h>
00004 #include <SDL2/SDL.h>
00005 #include <functional>
00006 #include <map>
00007 #include <memory>
00008
00009 enum class MouseButton : uint8_t {
00010     LEFT    = SDL_BUTTON_LEFT,
00011     MIDDLE  = SDL_BUTTON_MIDDLE,
00012     RIGHT   = SDL_BUTTON_RIGHT,
00013     X1      = SDL_BUTTON_X1,
00014     X2      = SDL_BUTTON_X2
00015 };
00016
00021 class InputHandler {
00022 private:
00023     struct KeyState {
00024         enum { PRESSED, RELEASED, NONE } toggle;
00025         enum { UP, DOWN } hold;
00026         uint32_t holdStart; // The tick this key was first held down.
00027         KeyState() :
00028             toggle(NONE),
00029             hold(UP),
00030             holdStart(0) {}
00031         void toggleDown(void) noexcept {
00032             if (hold == UP) {
00033                 toggle = PRESSED;
00034                 holdStart = SDL_GetTicks();
00035             }
00036             hold = DOWN;
00037         }
00038         void toggleUp(void) noexcept {
00039             if (hold == DOWN) {
00040                 toggle = RELEASED;
00041             }
00042             hold = UP;
00043         }
00044         uint32_t getHoldTime(void) const noexcept {
00045             if (hold == DOWN)
00046                 return SDL_GetTicks() - holdStart;
00047             return 0;
00048         }
00049     };
00050     std::map<SDL_Keycode, KeyState> keyStateDB;
00051     std::map<MouseButton, KeyState> mouseButtonStateDB;
00052     Vector2D mouseScroll;
00053
00054     InputHandler();
00055 public:
00056     InputHandler(const InputHandler&) = delete;
00057     void operator = (const InputHandler&) = delete;
00058
00059     static InputHandler& getInstance(void) noexcept;
00060
00061
00062     // Keyboard functions
00063
00070     bool pollKeyPress(SDL_Keycode key) noexcept;
00071
00077     bool pollKeyRelease(SDL_Keycode key) noexcept;
00078
00084     bool isKeyDown(SDL_Keycode key) const noexcept;
00085
00091     bool isKeyUp(SDL_Keycode key) const noexcept;
00092
00097     uint32_t holdTime(SDL_Keycode key) const noexcept;
00098
00099
00100     // Mouse functions
00101
00102     bool pollButtonPress(MouseButton button) noexcept;
00103     bool pollButtonRelease(MouseButton button) noexcept;
00104     bool isButtonDown(MouseButton button) const noexcept;
00105     bool isButtonUp(MouseButton button) const noexcept;
00106     uint32_t holdTime(MouseButton button) const noexcept;
00107
00108     Vector2D getMousePosition(void) const noexcept;
00109
00110     Vector2D pollMouseScroll(void) noexcept;
00111

```

```

00112     // Event Receivers
00113
00114     void receiveEvent(SDL_KeyboardEvent keyboardEvent) noexcept;
00115     void receiveEvent(SDL_MouseButtonEvent mouseButtonEvent) noexcept;
00116     void receiveEvent(SDL_MouseWheelEvent mouseWheelEvent) noexcept;
00117     //void receiveEvent(SDL_MouseMotionEvent mouseMotionEvent) noexcept;
00118 };

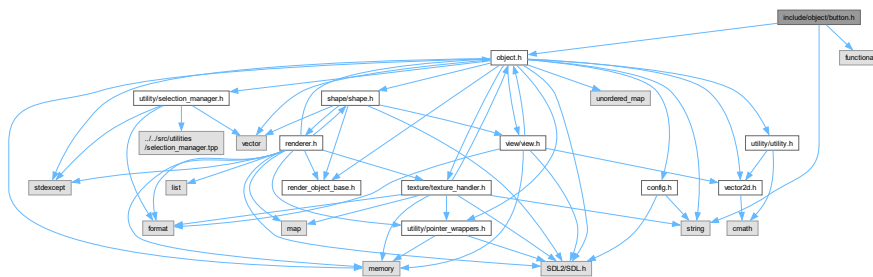
```

7.13 include/object/button.h File Reference

```

#include "object.h"
#include <string>
#include <functional>
Include dependency graph for button.h:

```



Classes

- class [Objects::Button](#)

Namespaces

- namespace [Objects](#)

7.14 button.h

[Go to the documentation of this file.](#)

```

00001 #pragma once
00002
00003 #include "object.h"
00004 #include <string>
00005 #include <functional>
00006
00007 namespace Objects {
00008     class Button : private Object {
00009     private:
00010         std::string text;
00011         bool hover;
00012         std::function<void(void)> actionFunc;
00013
00014         bool pollHover(void) noexcept;
00015
00016     public:
00017         Button(
00018             const Views::View* view,
00019             const Vector2D& position,
00020             const Vector2D& dimension,
00021             const SDL_Color& color,
00022             const std::string& text,

```

```

00023         std::function<void(void)> action
00024     );
00025
00026     void setHovered(void) noexcept;
00027
00028     void onClick(void) noexcept;
00029
00030     void update(void) noexcept;
00031 };
00032 }

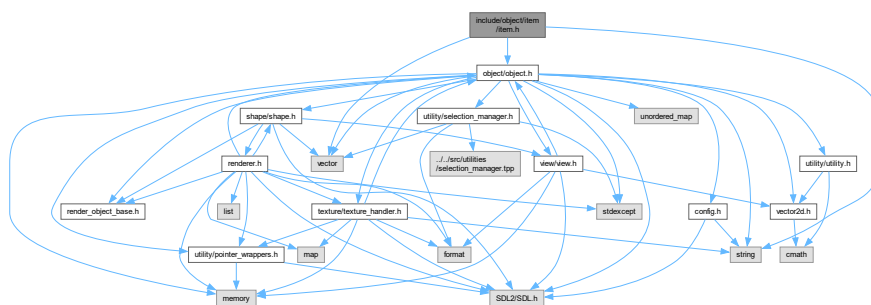
```

7.15 include/object/item/item.h File Reference

```

#include <object/object.h>
#include <vector>
#include <string>
Include dependency graph for item.h:

```



Classes

- class [Items::Item](#)

Namespaces

- namespace [Items](#)

7.16 item.h

[Go to the documentation of this file.](#)

```

00001 #pragma once
00002
00003 #include <object/object.h>
00004 #include <vector>
00005 #include <string>
00006
00007 namespace Items {
00008     class Item {
00009     private:
00010         std::string itemName;
00011         const uint8_t cap;
00012         uint8_t count;
00013         std::unique_ptr<Objects::Object> instanceObject;
00014         std::unique_ptr<Objects::Object> inventoryObject;
00015     public:
00016         Item(
00017             const std::vector<std::string>& instanceTextureNames,
00018             const std::vector<std::string>& inventoryObject,
00019             const std::string& itemName,
00020             uint8_t cap,
00021             uint8_t count
00022         );
00023     };
00024 }

```


7.18 object.h

[Go to the documentation of this file.](#)

```

00001 #pragma once
00002
00003 #include <render_object_base.h>
00004 #include <utility/utility.h>
00005 #include <utility/pointer_wrappers.h>
00006 #include <utility/vector2d.h>
00007 #include <utility/selection_manager.h>
00008 #include <texture/texture_handler.h>
00009 #include <view/view.h>
00010 #include <config.h>
00011 #include <shape/shape.h>
00012 #include <SDL2/SDL.h>
00013 #include <memory>
00014 #include <string>
00015 #include <vector>
00016 #include <unordered_map>
00017 #include <stdexcept>
00018
00019 namespace Views { class View; }
00020 class TextureHandler;
00021 namespace Shapes { class Shape; }
00022
00023 namespace Objects {
00024
00025     // TODO: add 'shapes' field to `Objects::Object`
00026
00031     class Object : public RenderObjectBase {
00032     friend class TextureHandler;
00033     private:
00034         SelectionManager<SDL_Texture*> textures;
00035         bool visible;
00036
00037         float angle; // stored as radians
00038         SDL_RendererFlip flipFlag;
00039         // SDL_Color colorMask; // color mod mask
00040         Vector2D position; // actual position in the world
00041         Vector2D dimension; // height and width
00042         const Views::View* view;
00043     public:
00044
00052         Object(
00053             const std::vector<std::string>& textureNames,
00054             const Views::View* _view,
00055             const Vector2D& _position,
00056             const Vector2D& _dimension
00057         );
00058
00059         virtual ~Object() = default;
00060
00067         float getAngle(void) const noexcept;
00068
00073         void setAngle(float newAngle) noexcept;
00074
00080         void rotate(float diffAngle) noexcept;
00081
00086         SDL_RendererFlip getFlipFlag(void) const noexcept;
00087
00092         Vector2D getPosition(void) const noexcept;
00093
00098         Vector2D getDimension(void) const noexcept;
00099
00104         void move(const Vector2D& translate) noexcept;
00105
00110         void stretchX(float ratio) noexcept;
00111
00116         void stretchY(float ratio) noexcept;
00117
00122         void stretch(float ratio) noexcept;
00123
00127         void flipHorizontal(void) noexcept;
00128
00132         void flipVertical(void) noexcept;
00133
00138         void setVisibility(bool visibility) noexcept;
00139
00144         bool getVisibility(void) const noexcept;
00145
00146
00147         /* TEXTURES */
00148
00152         void nextTexture(void) noexcept;
00153

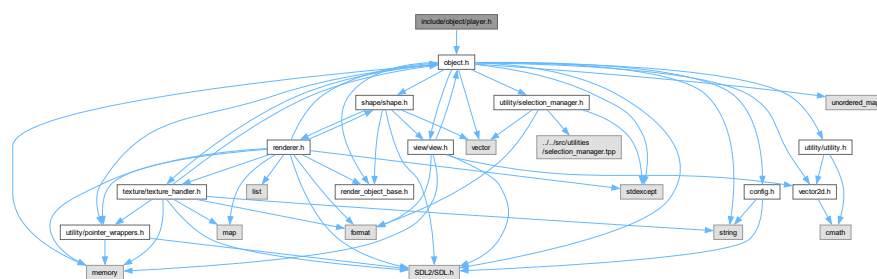
```

```
00157 void previousTexture(void) noexcept;
00158
00163 void setTexture(int textureId) noexcept;
00164
00169 size_t getTextureCount(void) const noexcept;
00170
00175 SDL_Texture* getTexture(void) const noexcept;
00176
00177 /* TEXTURES */
00178
00179
00184 virtual void lookAt(const Vector2D& position) noexcept;
00185
00190 SDL_FRect getRenderRect(void) const noexcept;
00191
00192 //Vector2D getRenderRelativePosition(Vector2D renderPosition) const noexcept;
00193
00197 void update(void) noexcept;
00198
00199 // debug
00200 void debug(void) const noexcept;
00201 };
00202 }
```

7.19 include/object/player.h File Reference

```
#include "object.h"
```

Include dependency graph for player.h:



Classes

- class **Objects::Player**

Namespaces

- namespace **Objects**

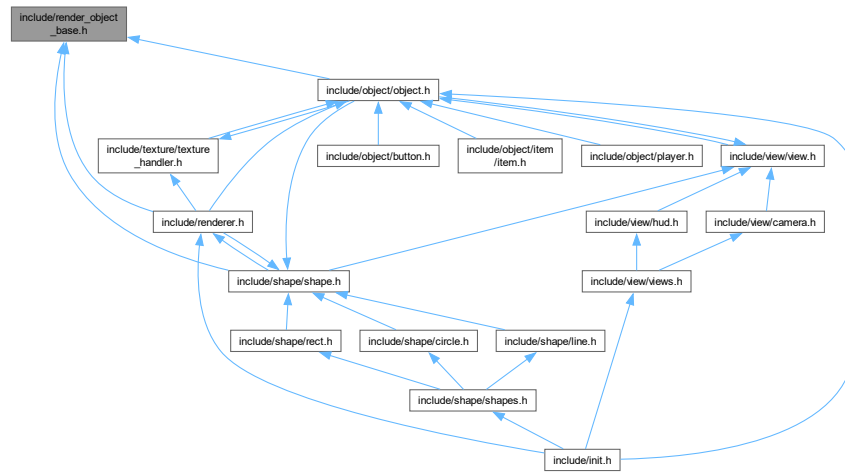
7.20 player.h

[Go to the documentation of this file.](#)

```
00001 #pragma once
00002
00003 #include "object.h"
00004
00005 namespace Objects {
00006     class Player : public Object {
00007
00008     };
00009 }
```

7.21 include/render_object_base.h File Reference

This graph shows which files directly or indirectly include this file:



Classes

- class [RenderObjectBase](#)
Empty render object base class category.

7.22 render_object_base.h

[Go to the documentation of this file.](#)

```

00001 #pragma once
00002
00006 class RenderObjectBase {
00007 public:
00008     virtual void debug(void) const noexcept;
00009 };

```

7.23 include/renderer.h File Reference

```

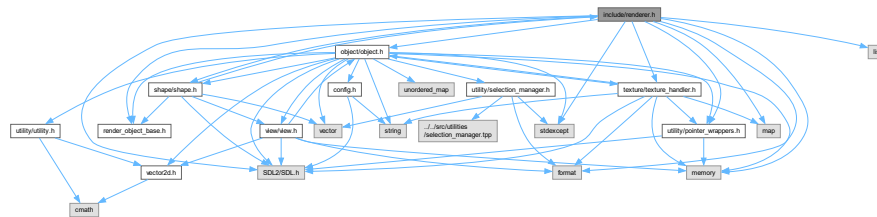
#include <render_object_base.h>
#include <object/object.h>
#include <utility/pointer_wrappers.h>
#include <texture/texture_handler.h>
#include <shape/shape.h>
#include <SDL2/SDL.h>
#include <memory>
#include <list>
#include <map>
#include <stdexcept>

```

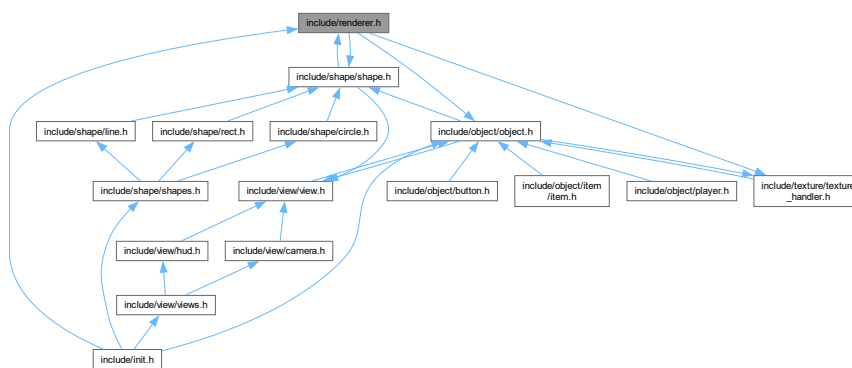


```
#include <format>
```

Include dependency graph for renderer.h:



This graph shows which files directly or indirectly include this file:



Classes

- class `Renderer`
Required key to call `render()` in.
- class `Renderer::RenderKey`

Namespaces

- namespace **Objects**

7.24 `renderer.h`

[Go to the documentation of this file.](#)

```
00001 #pragma once
00002
00003 #include <render_object_base.h>
00004 #include <object/object.h>
00005 #include <utility/pointer_wrappers.h>
00006 #include <texture/texture_handler.h>
00007 #include <shape/shape.h>
00008 #include <SDL2/SDL.h>
00009 #include <memory>
00010 #include <list>
00011 #include <map>
00012 #include <stdexcept>
```

```

00013 #include <format>
00014
00015 namespace Objects {
00016     class Object;
00017 }
00018
00019 // TODO: Consider wrapping object layer management into a LayerManager class.
00020
00021 // Singleton is needed as the renderer can only be initialized at runtime.
00022
00023 class Renderer {
00024     class CreateTextureKey {
00025         friend class TextureHandler;
00026     private:
00027         CreateTextureKey() = default;
00028         CreateTextureKey(const CreateTextureKey&) = default;
00029     };
00030
00031 public: // TODO: change this to private, this is for testing purposes.
00032     class RenderKey {
00033     public: // TODO: change this to private, this is for testing purposes.
00034         RenderKey() = default;
00035         RenderKey(const RenderKey&) = default;
00036     };
00037
00038 private:
00039     using ObjectWeakPtr = std::weak_ptr<RenderObjectBase>;
00040     using ObjectList = std::list<ObjectWeakPtr>;
00041
00042 private:
00043     sdl_unique_ptr<SDL_Window> window;
00044     sdl_unique_ptr<SDL_Renderer> renderer;
00045     std::map<ObjectWeakPtr, ObjectList::iterator, std::owner_less<ObjectWeakPtr> > objectListMap;
00046     ObjectList objectList;
00047
00048     Renderer();
00049 public:
00050     /* SINGLETON PATTERN */
00051     Renderer(const Renderer&) = delete;
00052     void operator = (const Renderer&) = delete;
00053     static Renderer& getInstance(void) noexcept;
00054     /* SINGLETON PATTERN */
00055
00056     SDL_Texture* createTexture(CreateTextureKey key, SDL_Surface* surface) const;
00057
00058     //SDL_Renderer* getRenderer(void) noexcept;
00059
00060     bool registerObject(std::shared_ptr<RenderObjectBase> objectPtr) noexcept;
00061
00062     bool removeObject(std::shared_ptr<RenderObjectBase> objectPtr) noexcept;
00063
00064     void render(RenderKey key);
00065
00066     void moveLayerUp(std::shared_ptr<RenderObjectBase> objectPtr);
00067     void moveLayerDown(std::shared_ptr<RenderObjectBase> objectPtr);
00068     void moveLayerTop(std::shared_ptr<RenderObjectBase> objectPtr);
00069     void moveLayerBottom(std::shared_ptr<RenderObjectBase> objectPtr);
00070
00071     void clear() noexcept;
00072
00073     void debug(void) const noexcept;
00074 };

```

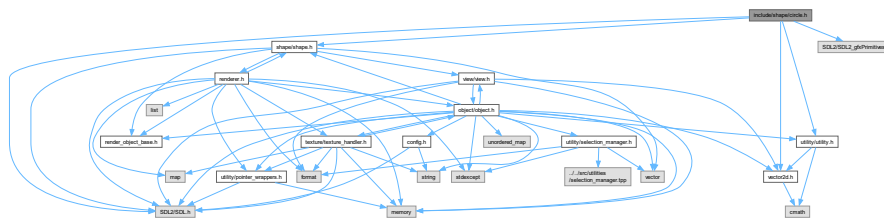
7.25 include/shape/circle.h File Reference

```

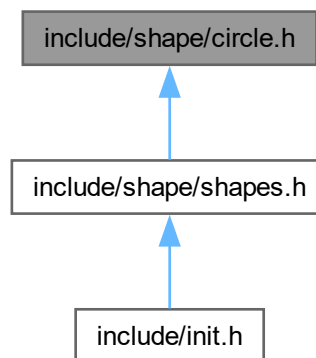
#include <shape/shape.h>
#include <utility/vector2d.h>
#include <utility/utility.h>
#include <SDL2/SDL.h>
#include <SDL2/SDL2_gfxPrimitives.h>

```

Include dependency graph for circle.h:



This graph shows which files directly or indirectly include this file:



Classes

- class [Shapes::Circle](#)
- class [Shapes::HollowCircle](#)

Namespaces

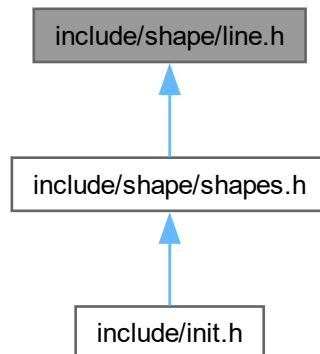
- namespace [Views](#)
- namespace [Shapes](#)

7.26 circle.h

[Go to the documentation of this file.](#)

```
00001 #pragma once
00002
00003 #include <shape/shape.h>
00004 #include <utility/vector2d.h>
00005 #include <utility/utility.h>
00006 #include <SDL2/SDL.h>
00007 #include <SDL2/SDL2_gfxPrimitives.h>
00008
```


This graph shows which files directly or indirectly include this file:



Classes

- class [Shapes::Line](#)

Namespaces

- namespace [Shapes](#)

7.28 line.h

[Go to the documentation of this file.](#)

```

00001 #pragma once
00002
00003 #include <shape/shape.h>
00004 #include <utility/vector2d.h>
00005 #include <SDL2/SDL.h>
00006 #include <SDL2/SDL2_gfxPrimitives.h>
00007
00008 namespace Shapes {
00009     class Line : public Shape {
00010     protected:
00011         Vector2D beginPoint;
00012         Vector2D endPoint;
00013         uint8_t thickness;
00014     public:
00015         Line(
00016             Views::View* view,
00017             Vector2D _beginPoint,
00018             Vector2D _endPoint,
00019             uint8_t _thickness,
00020             SDL_Color color = {0, 0, 0, 255}
00021         ) noexcept;
00022         void setBeginPoint(Vector2D newBeginPoint) noexcept;
00023         void setEndPoint(Vector2D newEndPoint) noexcept;
00024         void setThickness(uint8_t newThickness) noexcept;
00025         void draw(SDL_Renderer* renderer) const noexcept override;
00026     };
00027 }

```



```

00005 namespace Shapes {
00006     class Rect : public Shape {
00007     private:
00008         //void draw()
00009     protected:
00010         Vector2D position;
00011         Vector2D dimension;
00012     };
00013 }

```

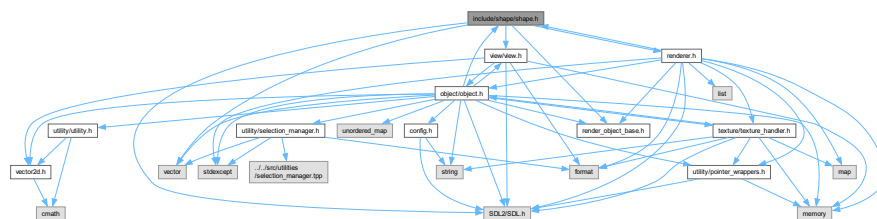
7.31 include/shape/shape.h File Reference

```

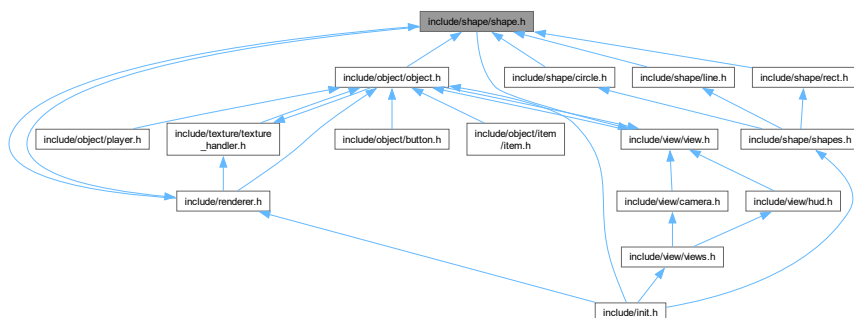
#include <render_object_base.h>
#include <view/view.h>
#include <renderer.h>
#include <SDL2/SDL.h>
#include <vector>

```

Include dependency graph for shape.h:



This graph shows which files directly or indirectly include this file:



Classes

- class [Shapes::Shape](#)

Namespaces

- namespace [Views](#)
- namespace [Shapes](#)

7.34 shapes.h

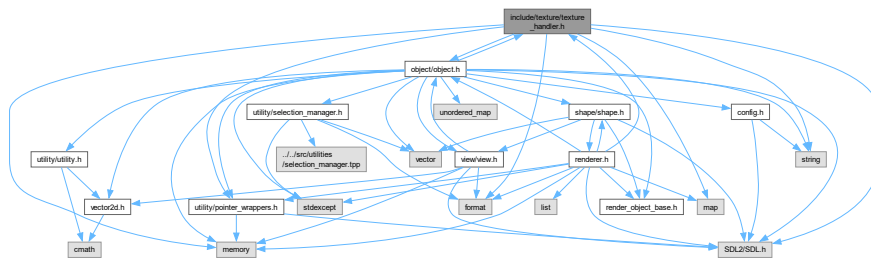
[Go to the documentation of this file.](#)

```
00001 #pragma once
00002
00003 #include "line.h"
00004 #include "circle.h"
00005 #include "rect.h"
00006
00007 // TODO: add more shapes: pie, triangle
```

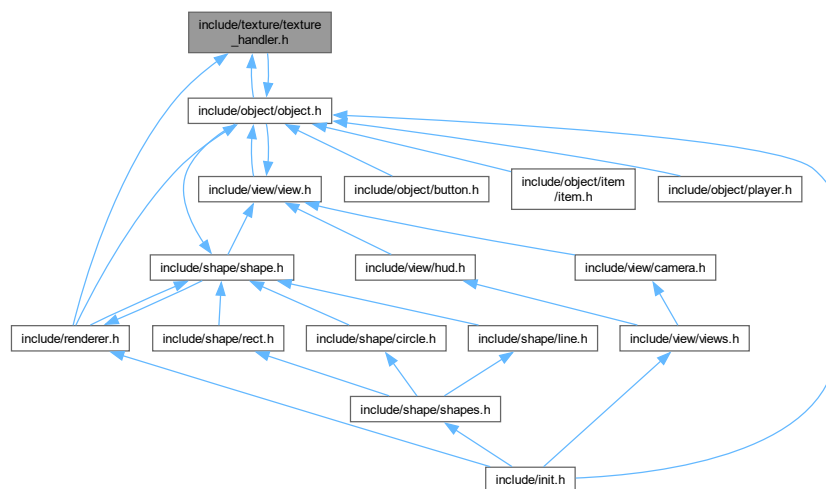
7.35 include/texture/texture_handler.h File Reference

```
#include <utility/pointer_wrappers.h>
#include <object/object.h>
#include <SDL2/SDL.h>
#include <string>
#include <map>
#include <memory>
#include <format>
```

Include dependency graph for texture_handler.h:



This graph shows which files directly or indirectly include this file:



Classes

- class [TextureHandler](#)

This is a global singleton class for texture handling.

Namespaces

- namespace [Objects](#)

7.36 texture_handler.h

[Go to the documentation of this file.](#)

```
00001 #pragma once
00002
00003 #include <utility/pointer_wrappers.h>
00004 #include <object/object.h>
00005 #include <SDL2/SDL.h>
00006 #include <string>
00007 #include <map>
00008 #include <memory>
00009 #include <format>
00010
00011 namespace Objects {
00012     class Object;
00013 }
00014
00015 // TODO: Add support for text textures.
00016
00020 class TextureHandler {
00024     class TextureRequestKey {
00025         friend class Objects::Object;
00026     private:
00027         TextureRequestKey() = default;
00028         TextureRequestKey(const TextureRequestKey&) = default;
00029     };
00030
00031 private:
00032     static const std::string errorTextureName;
00033     std::map<std::string, sdl_unique_ptr<SDL_Texture> textureDB;
00034
00038     TextureHandler();
00039
00040     void loadTexture(const std::string& textureName);
00041
00042 public:
00049     SDL_Texture* getTexture(TextureRequestKey key, const std::string& textureName);
00050
00051 public:
00052     TextureHandler(const TextureHandler&) = delete;
00053     void operator = (const TextureHandler&) = delete;
00054     static TextureHandler& getInstance(void);
00055 };
```

7.37 include/utility/functions.h File Reference

Namespaces

- namespace [Functions](#)

7.38 functions.h

[Go to the documentation of this file.](#)

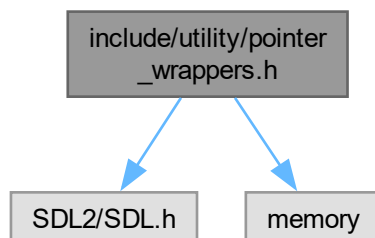
```
00001 #pragma once
00002
00003 namespace Functions {
00004
00005 }
```

7.39 include/utility/pointer_wrappers.h File Reference

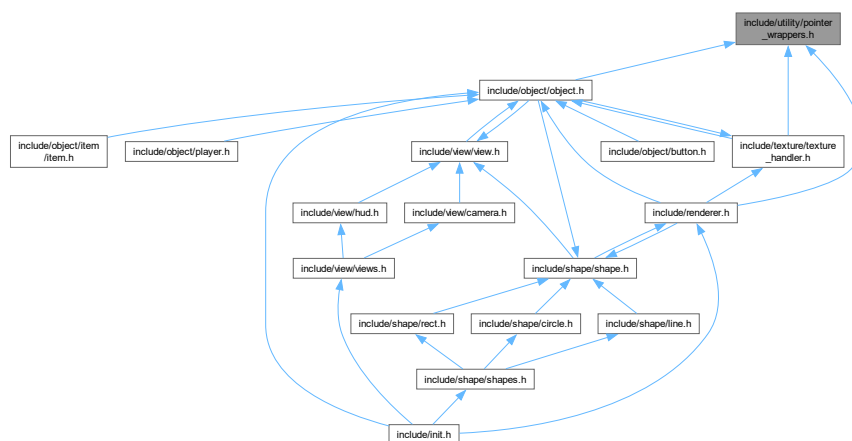
```
#include <SDL2/SDL.h>
```

```
#include <memory>
```

Include dependency graph for pointer_wrappers.h:



This graph shows which files directly or indirectly include this file:



Classes

- struct [sdl_deleter](#)

Generic deleter functor for SDL resources. For use with std smart pointers.

Typedefs

- template<typename Resource >
using [sdl_unique_ptr](#) = std::unique_ptr<Resource, [sdl_deleter](#)>

Functions

- `template<typename Resource >`
`std::shared_ptr< Resource > sdl_make_shared (Resource *resource)`

7.39.1 Typedef Documentation

7.39.1.1 `sdl_unique_ptr`

```
template<typename Resource >
using sdl\_unique\_ptr = std::unique_ptr<Resource, sdl\_deleter>
```

7.39.2 Function Documentation

7.39.2.1 `sdl_make_shared()`

```
template<typename Resource >
std::shared_ptr< Resource > sdl\_make\_shared (
    Resource * resource )
```

7.40 `pointer_wrappers.h`

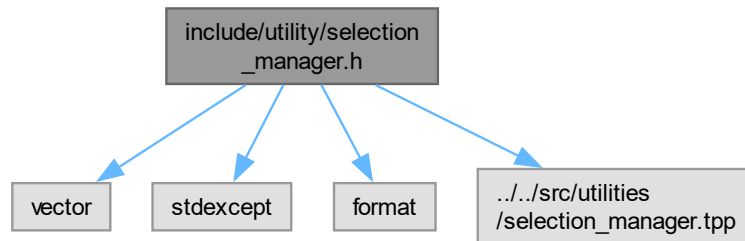
[Go to the documentation of this file.](#)

```
00001 #pragma once
00002
00003 #include <SDL2/SDL.h>
00004 #include <memory>
00005
00009 struct sdl\_deleter {
00010     inline void operator () (SDL_RWops* thing) const noexcept { if (thing) SDL\_FreeRW(thing); }
00011     inline void operator () (SDL_cond* thing) const noexcept { if (thing)
    SDL\_DestroyCond(thing); }
00012     inline void operator () (SDL_Cursor* thing) const noexcept { if (thing)
    SDL\_FreeCursor(thing); }
00013     inline void operator () (SDL_PixelFormat* thing) const noexcept { if (thing)
    SDL\_FreeFormat(thing); }
00014     inline void operator () (SDL_mutex* thing) const noexcept { if (thing)
    SDL\_DestroyMutex(thing); }
00015     inline void operator () (SDL_Palette* thing) const noexcept { if (thing)
    SDL\_FreePalette(thing); }
00016     inline void operator () (SDL_Renderer* thing) const noexcept { if (thing)
    SDL\_DestroyRenderer(thing); }
00017     inline void operator () (SDL_sem* thing) const noexcept { if (thing)
    SDL\_DestroySemaphore(thing); }
00018     inline void operator () (SDL_Surface* thing) const noexcept { if (thing)
    SDL\_FreeSurface(thing); }
00019     inline void operator () (SDL_Texture* thing) const noexcept { if (thing)
    SDL\_DestroyTexture(thing); }
00020     inline void operator () (Uint8* thing) const noexcept { if (thing) SDL\_FreeWAV(thing); }
00021     inline void operator () (SDL_Window* thing) const noexcept { if (thing)
    SDL\_DestroyWindow(thing); }
00022 };
00023
00024 template <typename Resource>
00025 using sdl\_unique\_ptr = std::unique_ptr<Resource, sdl\_deleter>;
00026
00027 template <typename Resource>
00028 std::shared_ptr<Resource> sdl\_make\_shared(Resource* resource) {
00029     return std::shared_ptr<Resource>(resource, sdl\_deleter());
00030 }
```

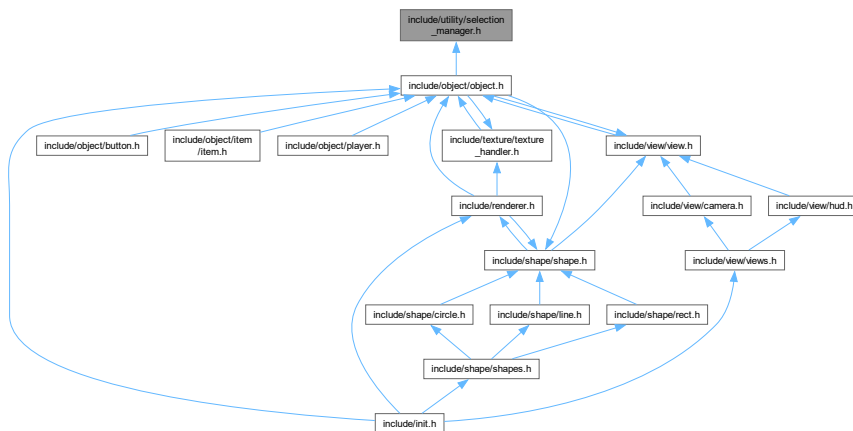
7.41 include/utility/selection_manager.h File Reference

```
#include <vector>
#include <stdexcept>
#include <format>
#include "../src/utilities/selection_manager.hpp"
```

Include dependency graph for selection_manager.h:



This graph shows which files directly or indirectly include this file:



Classes

- class [SelectionManager< T >](#)

7.42 selection_manager.h

[Go to the documentation of this file.](#)

```
00001 #pragma once
00002
00003 #include <vector>
00004 #include <stdexcept>
```

```

00005 #include <format>
00006
00007 // TODO: Complete SelectionManager.
00008
00009 template<class T>
00010 class SelectionManager {
00011 private:
00012     std::vector<T> selections;
00013     mutable int currentSelection; // mutable: this field should ALWAYS be modifiable.
00014 public:
00015     static const int SELECTION_NOT_SET = -1;
00016
00017     SelectionManager();
00018     SelectionManager(const std::vector<T>& selections);
00019
00023     void next(void) const noexcept;
00024
00028     void prev(void) const noexcept;
00029
00035     void set(int newSelection) const;
00036
00041     size_t size(void) const noexcept;
00042
00047     void add(T newSelection) noexcept;
00048
00054     void remove(size_t selectionId);
00055
00061     T get(void) const;
00062
00067     int getSelectionId(void) const noexcept;
00068 };
00069
00070 #include "../src/utilities/selection_manager.hpp"

```

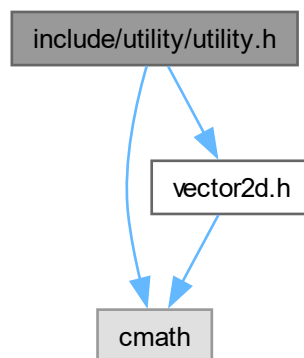
7.43 include/utility/utility.h File Reference

```

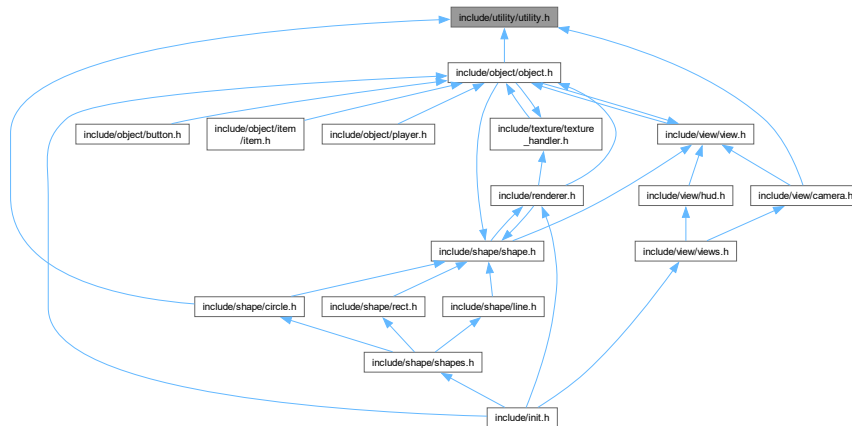
#include <cmath>
#include "vector2d.h"

```

Include dependency graph for utility.h:



This graph shows which files directly or indirectly include this file:



Macros

- `#define` [_USE_MATH_DEFINES](#)

Functions

- float [normalizeAngle](#) (float angle) noexcept
Helper function to normalize angle to [0, 2pi)
- [Vector2D polarToCartesian](#) (float radius, float theta)
Helper function to transform polar coordinates to cartesian coordinates.

7.43.1 Macro Definition Documentation

7.43.1.1 _USE_MATH_DEFINES

```
#define _USE_MATH_DEFINES
```

7.43.2 Function Documentation

7.43.2.1 normalizeAngle()

```
float normalizeAngle (
    float angle ) [noexcept]
```

Helper function to normalize angle to [0, 2pi)

Parameters

<i>angle</i>	input angle
--------------	-------------

Returns

normalized angle

7.43.2.2 polarToCartesian()

```
Vector2D polarToCartesian (
    float radius,
    float theta )
```

Helper function to to transform polar coordinates to cartesian coordinates.

Parameters

<i>radius</i>	input radius
<i>theta</i>	input anegele (radians)

Returns

the transformed cartesian coordinates

7.44 utility.h

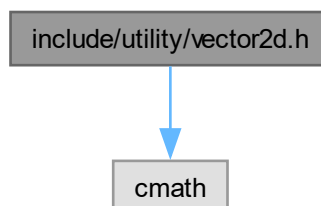
[Go to the documentation of this file.](#)

```
00001 #pragma once
00002
00003 #define _USE_MATH_DEFINES
00004 #include <cmath>
00005 #include "vector2d.h"
00006
00012 float normalizeAngle(float angle) noexcept;
00013
00020 Vector2D polarToCartesian(float radius, float theta);
```

7.45 include/utility/vector2d.h File Reference

```
#include <cmath>
```

Include dependency graph for vector2d.h:



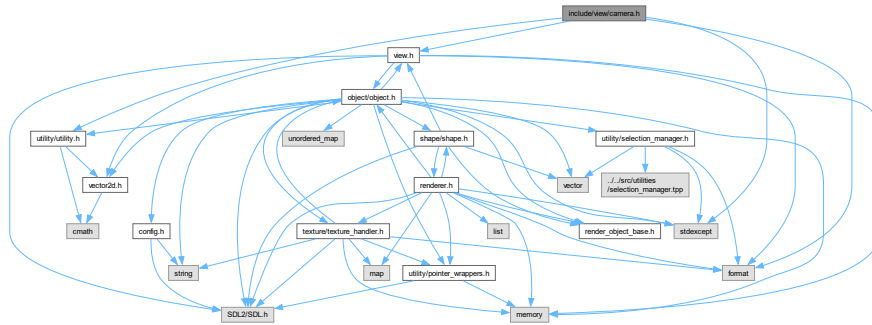
- class **Vector2D**

[Go to the documentation of this file.](#)

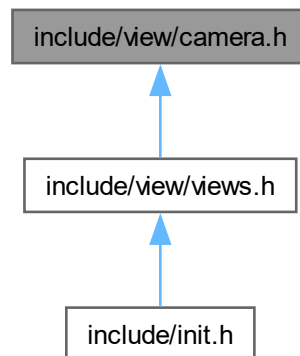
Generated by Doxygen

7.47 include/view/camera.h File Reference

```
#include <utility/utility.h>
#include "view.h"
#include <stdexcept>
#include <format>
Include dependency graph for camera.h:
```



This graph shows which files directly or indirectly include this file:



Classes

- class [Views::Camera](#)
Camera for following object or stationary view.

Namespaces

- namespace [Views](#)

7.48 camera.h

[Go to the documentation of this file.](#)

```

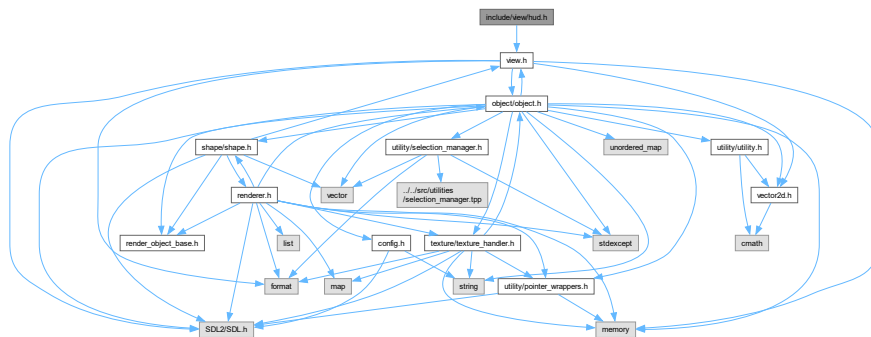
00001 #pragma once
00002
00003 #include <utility/utility.h>
00004 #include "view.h"
00005 #include <stdexcept>
00006 #include <format>
00007
00008 namespace Views {
00009
00013     class Camera : public View {
00014     private:
00015         std::weak_ptr<Objects::Object> pivotObject;
00016
00017         float zoom;
00018         float angle;
00019
00020         Vector2D getPosition(void) const noexcept;
00021     public:
00022         Camera();
00023
00028         void setPivotObject(std::shared_ptr<Objects::Object> pivotObject) noexcept;
00029         // const std::weak_ptr<Objects::Object> getPivotObject(void) const noexcept;
00030
00035         void setPosition(const Vector2D& newPosition) noexcept;
00036
00042         void setDimension(const Vector2D& newDimension);
00043
00049         void setZoom(float zoom);
00050
00051         float getZoom(void) const noexcept override;
00052
00057         void setAngle(float angle) noexcept;
00058
00063         void rotate(float diffAngle) noexcept;
00064
00069         float getAngle(void) const noexcept override;
00070
00071         SDL_FRect getRect(const Objects::Object& object) const noexcept override;
00072         Vector2D transform(const Vector2D& position) const noexcept override;
00073         Vector2D transformFromRender(const Vector2D& renderPosition) const noexcept override;
00074     };
00075 }

```

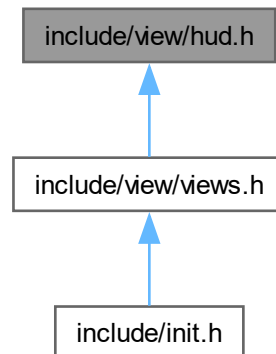
7.49 include/view/hud.h File Reference

```
#include "view.h"
```

Include dependency graph for hud.h:



This graph shows which files directly or indirectly include this file:



Classes

- class [Views::HUD](#)

Namespaces

- namespace [Views](#)

7.50 hud.h

[Go to the documentation of this file.](#)

```

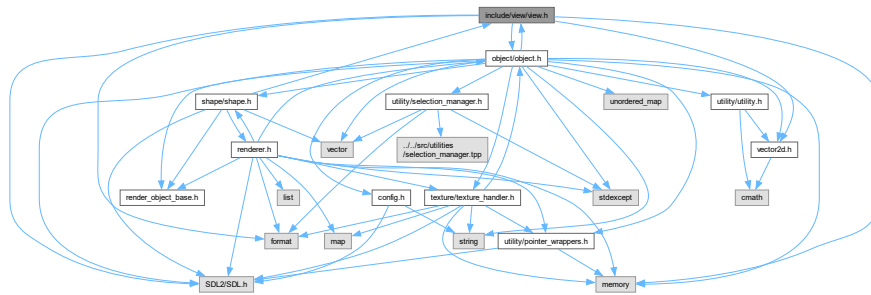
00001 #pragma once
00002
00003 #include "view.h"
00004
00005 namespace Views {
00006     class HUD : public View {
00007     public:
00008         HUD();
00009         SDL_FRect getRect(const Objects::Object&) const noexcept override;
00010         Vector2D transform(const Vector2D& position) const noexcept override;
00011         Vector2D transformFromRender(const Vector2D& renderPosition) const noexcept override;
00012     };
00013 }
  
```

7.51 include/view/view.h File Reference

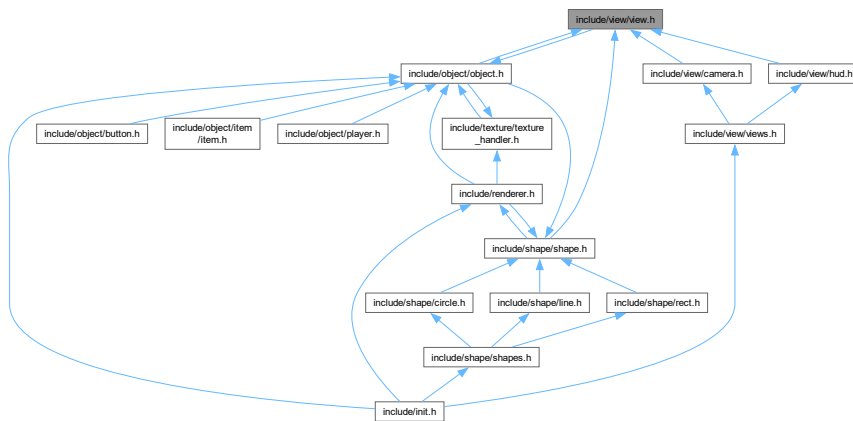
```

#include <object/object.h>
#include <utility/vector2d.h>
#include <SDL2/SDL.h>
#include <memory>
  
```

Include dependency graph for view.h:



This graph shows which files directly or indirectly include this file:



Classes

- class `Views::View`
 - View*: defines a view area, translates the objects' virtual rects to real rendering rects.

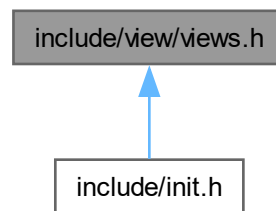
Namespaces

- namespace **Objects**
- namespace **Views**

Variables

- `const int Views::INIT_VIEW_WIDTH = 1600`
- `const int Views::INIT_VIEW_HEIGHT = 900`

This graph shows which files directly or indirectly include this file:



7.54 views.h

[Go to the documentation of this file.](#)

```
00001 #pragma once
00002
00003 #include "hud.h"
00004 #include "camera.h"
```


Index

- `_USE_MATH_DEFINES`
 - `utility.h`, [107](#)
 - `~Command`
 - `Commands::Command`, [25](#)
 - `~Object`
 - `Objects::Object`, [44](#)
 - `~Shape`
 - `Shapes::Shape`, [65](#)
 - `~View`
 - `Views::View`, [73](#)
- `add`
 - `SelectionManager< T >`, [61](#)
- `arrowObject1`
 - `Global`, [11](#)
- `arrowObject2`
 - `Global`, [11](#)
- `backgroundColor`
 - `Config`, [9](#)
- `beginPoint`
 - `Shapes::Line`, [41](#)
- `blueCircle`
 - `Global`, [11](#)
- `Button`
 - `Objects::Button`, [16](#)
- `buttons`
 - `KeyBind`, [38](#)
- `Camera`
 - `Views::Camera`, [19](#)
- `center`
 - `Shapes::Circle`, [24](#)
- `Circle`
 - `Shapes::Circle`, [24](#)
- `clear`
 - `Renderer`, [55](#)
- `color`
 - `Shapes::Shape`, [66](#)
- `CommandManager`, [26](#)
 - `Commands::Command::ExecuteKey`, [27](#)
 - `registerCommand`, [26](#)
 - `update`, [26](#)
- `Commands`, [9](#)
- `Commands::Command`, [25](#)
 - `~Command`, [25](#)
 - `execute`, [25](#)
- `Commands::Command::ExecuteKey`, [27](#)
 - `CommandManager`, [27](#)
- `Config`, [9](#)
- `backgroundColor`, [9](#)
- `framerate`, [9](#)
- `gameTitle`, [9](#)
- `holdTimeThreshold`, [10](#)
- `screenHeight`, [10](#)
- `screenType`, [10](#)
- `screenWidth`, [10](#)
- `volume`, [10](#)
- `createTexture`
 - `Renderer`, [55](#)
- `cross`
 - `Vector2D`, [69](#)
- `crosshairCircle1`
 - `Global`, [11](#)
- `crosshairLine1`
 - `Global`, [11](#)
- `crosshairLine2`
 - `Global`, [12](#)
- `debug`
 - `Objects::Object`, [44](#)
 - `Renderer`, [55](#)
 - `RenderObjectBase`, [58](#)
- `dimension`
 - `Shapes::Rect`, [53](#)
 - `Views::View`, [75](#)
- `dot`
 - `Vector2D`, [69](#)
- `draw`
 - `Shapes::Circle`, [24](#)
 - `Shapes::HollowCircle`, [29](#)
 - `Shapes::Line`, [41](#)
 - `Shapes::Shape`, [65](#)
- `endPoint`
 - `Shapes::Line`, [41](#)
- `execute`
 - `Commands::Command`, [25](#)
- `flipHorizontal`
 - `Objects::Object`, [44](#)
- `flipVertical`
 - `Objects::Object`, [44](#)
- `fpsManager`
 - `Global`, [12](#)
- `framerate`
 - `Config`, [9](#)
- `Functions`, [10](#)
- `GameManager`, [27](#)

gameTitle
 Config, 9
 get
 SelectionManager< T >, 62
 getAngle
 Objects::Object, 45
 Views::Camera, 19
 Views::View, 73
 getColor
 Shapes::Shape, 65
 getDimension
 Objects::Object, 45
 Views::View, 73
 getFlipFlag
 Objects::Object, 45
 getInstance
 InputHandler, 34
 Renderer, 55
 TextureHandler, 67
 getMousePosition
 InputHandler, 34
 getPosition
 Objects::Object, 45
 Views::View, 73
 getRect
 Views::Camera, 19
 Views::HUD, 31
 Views::View, 73
 getRenderRect
 Objects::Object, 45
 getSelectionId
 SelectionManager< T >, 62
 getTexture
 Objects::Object, 46
 TextureHandler, 67
 getTextureCount
 Objects::Object, 46
 getVisibility
 Objects::Object, 46
 getX
 Vector2D, 69
 getY
 Vector2D, 69
 getZoom
 Views::Camera, 19
 Views::View, 74
 Global, 10
 arrowObject1, 11
 arrowObject2, 11
 blueCircle, 11
 crosshairCircle1, 11
 crosshairLine1, 11
 crosshairLine2, 12
 fpsManager, 12
 greenCircle, 12
 hollowCircle1, 12
 hudView, 12
 init, 11
 line1, 12
 line2, 12
 line3, 12
 line4, 12
 menuView, 12
 playerCamera, 13
 playerObject, 13
 purpleCircle, 13
 redCircle, 13
 yellowCircle, 13
 greenCircle
 Global, 12
 HOLD
 KeyBind, 38
 holdTime
 InputHandler, 34
 holdTimeThreshold
 Config, 10
 HollowCircle
 Shapes::HollowCircle, 29
 hollowCircle1
 Global, 12
 HUD
 Views::HUD, 31
 hudView
 Global, 12
 ID
 KeyBind, 38
 include/command/command.h, 77, 78
 include/command_manager.h, 79, 80
 include/config.h, 80, 81
 include/game_manager.h, 82
 include/init.h, 82, 84
 include/input_handler.h, 84, 86
 include/object/button.h, 87
 include/object/item/item.h, 88
 include/object/object.h, 89, 90
 include/object/player.h, 91
 include/render_object_base.h, 92
 include/renderer.h, 92, 93
 include/shape/circle.h, 94, 95
 include/shape/line.h, 96, 97
 include/shape/rect.h, 98
 include/shape/shape.h, 99, 100
 include/shape/shapes.h, 100, 101
 include/texture/texture_handler.h, 101, 102
 include/utility/functions.h, 102
 include/utility/pointer_wrappers.h, 103, 104
 include/utility/selection_manager.h, 105
 include/utility/utility.h, 106, 108
 include/utility/vector2d.h, 108, 109
 include/view/camera.h, 110, 111
 include/view/hud.h, 111, 112
 include/view/view.h, 112, 114
 include/view/views.h, 114, 115
 init
 Global, 11

- INIT_VIEW_HEIGHT
 - Views, [14](#)
- INIT_VIEW_WIDTH
 - Views, [14](#)
- input_handler.h
 - LEFT, [85](#)
 - MIDDLE, [85](#)
 - MouseButton, [85](#)
 - RIGHT, [85](#)
 - X1, [85](#)
 - X2, [85](#)
- InputHandler, [33](#)
 - getInstance, [34](#)
 - getMousePosition, [34](#)
 - holdTime, [34](#)
 - InputHandler, [33](#)
 - isButtonDown, [34](#)
 - isButtonUp, [34](#)
 - isKeyDown, [34](#)
 - isKeyUp, [35](#)
 - operator=, [35](#)
 - pollButtonPress, [35](#)
 - pollButtonRelease, [35](#)
 - pollKeyPress, [35](#)
 - pollKeyRelease, [36](#)
 - pollMouseScroll, [36](#)
 - receiveEvent, [36](#)
- isButtonDown
 - InputHandler, [34](#)
- isButtonUp
 - InputHandler, [34](#)
- isKeyDown
 - InputHandler, [34](#)
- isKeyUp
 - InputHandler, [35](#)
- Item
 - Items::Item, [37](#)
- Items, [13](#)
- Items::Item, [37](#)
 - Item, [37](#)
- KeyBind, [37](#)
 - buttons, [38](#)
 - HOLD, [38](#)
 - ID, [38](#)
 - KeyBind, [38](#)
 - KeyBindCount, [39](#)
 - keys, [39](#)
 - operator<, [38](#)
 - RELEASE, [38](#)
 - TAP, [38](#)
 - Trigger, [38](#)
- KeyBindCount
 - KeyBind, [39](#)
- keys
 - KeyBind, [39](#)
- LEFT
 - input_handler.h, [85](#)
- len
 - Vector2D, [69](#)
- len2
 - Vector2D, [69](#)
- Line
 - Shapes::Line, [41](#)
- line1
 - Global, [12](#)
- line2
 - Global, [12](#)
- line3
 - Global, [12](#)
- line4
 - Global, [12](#)
- lookAt
 - Objects::Object, [46](#)
- menuView
 - Global, [12](#)
- MIDDLE
 - input_handler.h, [85](#)
- MouseButton
 - input_handler.h, [85](#)
- move
 - Objects::Object, [47](#)
- moveLayerBottom
 - Renderer, [55](#)
- moveLayerDown
 - Renderer, [55](#)
- moveLayerTop
 - Renderer, [55](#)
- moveLayerUp
 - Renderer, [56](#)
- next
 - SelectionManager< T >, [62](#)
- nextTexture
 - Objects::Object, [47](#)
- norm
 - Vector2D, [69](#)
- normalizeAngle
 - utility.h, [107](#)
- Object
 - Objects::Object, [44](#)
- Objects, [13](#)
- Objects::Button, [15](#)
 - Button, [16](#)
 - onClick, [16](#)
 - setHovered, [16](#)
 - update, [16](#)
- Objects::Object, [42](#)
 - ~Object, [44](#)
 - debug, [44](#)
 - flipHorizontal, [44](#)
 - flipVertical, [44](#)
 - getAngle, [45](#)
 - getDimension, [45](#)
 - getFlipFlag, [45](#)

- getPosition, [45](#)
- getRenderRect, [45](#)
- getTexture, [46](#)
- getTextureCount, [46](#)
- getVisibility, [46](#)
- lookAt, [46](#)
- move, [47](#)
- nextTexture, [47](#)
- Object, [44](#)
- previousTexture, [47](#)
- rotate, [47](#)
- setAngle, [47](#)
- setTexture, [48](#)
- setVisibility, [48](#)
- stretch, [48](#)
- stretchX, [48](#)
- stretchY, [49](#)
- TextureHandler, [49](#)
- update, [49](#)
- Objects::Player, [50](#)
- onClick
 - Objects::Button, [16](#)
- operator<
 - KeyBind, [38](#)
- operator()
 - sdl_deleter, [59](#), [60](#)
- operator+
 - Vector2D, [70](#)
- operator+=
 - Vector2D, [70](#)
- operator-
 - Vector2D, [70](#)
- operator-=
 - Vector2D, [70](#)
- operator/
 - Vector2D, [71](#)
- operator/=
 - Vector2D, [71](#)
- operator=
 - InputHandler, [35](#)
 - Renderer, [56](#)
 - TextureHandler, [67](#)
- operator*
 - Vector2D, [70](#)
- operator*=
 - Vector2D, [70](#)
- playerCamera
 - Global, [13](#)
- playerObject
 - Global, [13](#)
- pointer_wrappers.h
 - sdl_make_shared, [104](#)
 - sdl_unique_ptr, [104](#)
- polarToCartesian
 - utility.h, [108](#)
- pollButtonPress
 - InputHandler, [35](#)
- pollButtonRelease
 - InputHandler, [35](#)
- pollKeyPress
 - InputHandler, [35](#)
- pollKeyRelease
 - InputHandler, [36](#)
- pollMouseScroll
 - InputHandler, [36](#)
- position
 - Shapes::Rect, [53](#)
 - Views::View, [75](#)
- prev
 - SelectionManager< T >, [62](#)
- previousTexture
 - Objects::Object, [47](#)
- purpleCircle
 - Global, [13](#)
- radius
 - Shapes::Circle, [24](#)
- receiveEvent
 - InputHandler, [36](#)
- redCircle
 - Global, [13](#)
- registerCommand
 - CommandManager, [26](#)
- registerObject
 - Renderer, [56](#)
- RELEASE
 - KeyBind, [38](#)
- remove
 - SelectionManager< T >, [62](#)
- removeObject
 - Renderer, [56](#)
- render
 - Renderer, [57](#)
- Renderer, [53](#)
 - clear, [55](#)
 - createTexture, [55](#)
 - debug, [55](#)
 - getInstance, [55](#)
 - moveLayerBottom, [55](#)
 - moveLayerDown, [55](#)
 - moveLayerTop, [55](#)
 - moveLayerUp, [56](#)
 - operator=, [56](#)
 - registerObject, [56](#)
 - removeObject, [56](#)
 - render, [57](#)
 - Renderer, [54](#)
- Renderer::RenderKey, [57](#)
 - RenderKey, [57](#)
- RenderKey
 - Renderer::RenderKey, [57](#)
- RenderObjectBase, [58](#)
 - debug, [58](#)
- RIGHT
 - input_handler.h, [85](#)
- rotate
 - Objects::Object, [47](#)

- Vector2D, 69
- Views::Camera, 19
- screenHeight
 - Config, 10
- screenType
 - Config, 10
- screenWidth
 - Config, 10
- sdl_deleter, 58
 - operator(), 59, 60
- sdl_make_shared
 - pointer_wrappers.h, 104
- sdl_unique_ptr
 - pointer_wrappers.h, 104
- SELECTION_NOT_SET
 - SelectionManager< T >, 63
- SelectionManager
 - SelectionManager< T >, 61
- SelectionManager< T >, 60
 - add, 61
 - get, 62
 - getSelectionId, 62
 - next, 62
 - prev, 62
 - remove, 62
 - SELECTION_NOT_SET, 63
 - SelectionManager, 61
 - set, 63
 - size, 63
- set
 - SelectionManager< T >, 63
- setAngle
 - Objects::Object, 47
 - Views::Camera, 20
- setBeginPoint
 - Shapes::Line, 41
- setCenter
 - Shapes::Circle, 24
- setColor
 - Shapes::Shape, 65
- setDimension
 - Views::Camera, 20
- setEndPoint
 - Shapes::Line, 41
- setHovered
 - Objects::Button, 16
- setPivotObject
 - Views::Camera, 20
- setPosition
 - Views::Camera, 20
- setRadius
 - Shapes::Circle, 24
- setTexture
 - Objects::Object, 48
- setThickness
 - Shapes::HollowCircle, 29
 - Shapes::Line, 41
- setVisibility
 - Objects::Object, 48
- setZoom
 - Views::Camera, 21
- Shape
 - Shapes::Shape, 65
- Shapes, 14
- Shapes::Circle, 22
 - center, 24
 - Circle, 24
 - draw, 24
 - radius, 24
 - setCenter, 24
 - setRadius, 24
- Shapes::HollowCircle, 27
 - draw, 29
 - HollowCircle, 29
 - setThickness, 29
 - thickness, 30
- Shapes::Line, 39
 - beginPoint, 41
 - draw, 41
 - endPoint, 41
 - Line, 41
 - setBeginPoint, 41
 - setEndPoint, 41
 - setThickness, 41
 - thickness, 42
- Shapes::Rect, 52
 - dimension, 53
 - position, 53
- Shapes::Shape, 64
 - ~Shape, 65
 - color, 66
 - draw, 65
 - getColor, 65
 - setColor, 65
 - Shape, 65
 - view, 66
- size
 - SelectionManager< T >, 63
- stretch
 - Objects::Object, 48
- stretchX
 - Objects::Object, 48
- stretchY
 - Objects::Object, 49
- TAP
 - KeyBind, 38
- TextureHandler, 66
 - getInstance, 67
 - getTexture, 67
 - Objects::Object, 49
 - operator=, 67
 - TextureHandler, 67
- thickness
 - Shapes::HollowCircle, 30
 - Shapes::Line, 42
- transform

- Views::Camera, 21
- Views::HUD, 32
- Views::View, 74
- transformFromRender
 - Views::Camera, 21
 - Views::HUD, 32
 - Views::View, 74
- Trigger
 - KeyBind, 38
- update
 - CommandManager, 26
 - Objects::Button, 16
 - Objects::Object, 49
- utility.h
 - _USE_MATH_DEFINES, 107
 - normalizeAngle, 107
 - polarToCartesian, 108
- Vector2D, 68
 - cross, 69
 - dot, 69
 - getX, 69
 - getY, 69
 - len, 69
 - len2, 69
 - norm, 69
 - operator+, 70
 - operator+=, 70
 - operator-, 70
 - operator-=, 70
 - operator/, 71
 - operator/=: 71
 - operator*, 70
 - operator*=: 70
 - rotate, 69
 - Vector2D, 68
 - zero, 70
- View
 - Views::View, 73
- view
 - Shapes::Shape, 66
- Views, 14
 - INIT_VIEW_HEIGHT, 14
 - INIT_VIEW_WIDTH, 14
- Views::Camera, 17
 - Camera, 19
 - getAngle, 19
 - getRect, 19
 - getZoom, 19
 - rotate, 19
 - setAngle, 20
 - setDimension, 20
 - setPivotObject, 20
 - setPosition, 20
 - setZoom, 21
 - transform, 21
 - transformFromRender, 21
- Views::HUD, 30
 - getRect, 31
 - HUD, 31
 - transform, 32
 - transformFromRender, 32
- Views::View, 71
 - ~View, 73
 - dimension, 75
 - getAngle, 73
 - getDimension, 73
 - getPosition, 73
 - getRect, 73
 - getZoom, 74
 - position, 75
 - transform, 74
 - transformFromRender, 74
 - View, 73
- volume
 - Config, 10
- X1
 - input_handler.h, 85
- X2
 - input_handler.h, 85
- yellowCircle
 - Global, 13
- zero
 - Vector2D, 70