

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::Bullet Class Reference	15
6.1.1 Constructor & Destructor Documentation	17
6.1.1.1 Bullet()	17
6.1.2 Member Function Documentation	17
6.1.2.1 getAliveTime()	17
6.1.2.2 update()	18
6.2 Objects::Button Class Reference	18
6.2.1 Constructor & Destructor Documentation	19
6.2.1.1 Button()	19
6.2.2 Member Function Documentation	19
6.2.2.1 onClick()	19
6.2.2.2 setHovered()	19
6.2.2.3 update()	19
6.3 Views::Camera Class Reference	20
6.3.1 Detailed Description	21
6.3.2 Constructor & Destructor Documentation	21
6.3.2.1 Camera()	21
6.3.3 Member Function Documentation	22
6.3.3.1 getAngle()	22
6.3.3.2 getRect()	22
6.3.3.3 getZoom()	22
6.3.3.4 rotate()	22
6.3.3.5 setAngle()	23
6.3.3.6 setDimension()	23
6.3.3.7 setPivotObject()	23
6.3.3.8 setPosition()	23
6.3.3.9 setZoom()	24
6.3.3.10 transform()	24

6.3.3.11 transformFromRender()	24
6.4 Shapes::Circle Class Reference	25
6.4.1 Constructor & Destructor Documentation	27
6.4.1.1 Circle()	27
6.4.2 Member Function Documentation	27
6.4.2.1 draw()	27
6.4.2.2 setCenter()	27
6.4.2.3 setRadius()	27
6.4.3 Member Data Documentation	27
6.4.3.1 center	27
6.4.3.2 radius	28
6.5 Commands::Command Class Reference	28
6.5.1 Detailed Description	28
6.5.2 Constructor & Destructor Documentation	28
6.5.2.1 ~Command()	28
6.5.3 Member Function Documentation	28
6.5.3.1 execute()	28
6.6 CommandManager Class Reference	29
6.6.1 Detailed Description	29
6.6.2 Member Function Documentation	29
6.6.2.1 registerCommand()	29
6.6.2.2 update()	29
6.7 Commands::Command::ExecuteKey Class Reference	30
6.7.1 Friends And Related Symbol Documentation	30
6.7.1.1 CommandManager	30
6.8 GameManager Class Reference	30
6.9 Shapes::HollowCircle Class Reference	30
6.9.1 Constructor & Destructor Documentation	32
6.9.1.1 HollowCircle()	32
6.9.2 Member Function Documentation	32
6.9.2.1 draw()	32
6.9.2.2 setThickness()	32
6.9.3 Member Data Documentation	33
6.9.3.1 thickness	33
6.10 Views::HUD Class Reference	33
6.10.1 Constructor & Destructor Documentation	34
6.10.1.1 HUD()	34
6.10.2 Member Function Documentation	34
6.10.2.1 getRect()	34
6.10.2.2 transform()	35
6.10.2.3 transformFromRender()	35
6.11 InputHandler Class Reference	36

6.11.1 Detailed Description	36
6.11.2 Constructor & Destructor Documentation	36
6.11.2.1 InputHandler()	36
6.11.3 Member Function Documentation	37
6.11.3.1 getInstance()	37
6.11.3.2 getMousePosition()	37
6.11.3.3 holdTime() [1/2]	37
6.11.3.4 holdTime() [2/2]	37
6.11.3.5 isButtonDown()	37
6.11.3.6 isButtonUp()	37
6.11.3.7 isKeyDown()	37
6.11.3.8 isKeyUp()	38
6.11.3.9 operator=()	38
6.11.3.10 pollButtonPress()	38
6.11.3.11 pollButtonRelease()	38
6.11.3.12 pollKeyPress()	38
6.11.3.13 pollKeyRelease()	39
6.11.3.14 pollMouseScroll()	39
6.11.3.15 receiveEvent() [1/3]	39
6.11.3.16 receiveEvent() [2/3]	39
6.11.3.17 receiveEvent() [3/3]	39
6.12 Items::Item Class Reference	40
6.12.1 Constructor & Destructor Documentation	40
6.12.1.1 Item()	40
6.13 KeyBind Struct Reference	40
6.13.1 Detailed Description	41
6.13.2 Member Enumeration Documentation	41
6.13.2.1 Trigger	41
6.13.3 Constructor & Destructor Documentation	41
6.13.3.1 KeyBind()	41
6.13.4 Friends And Related Symbol Documentation	41
6.13.4.1 operator<	41
6.13.5 Member Data Documentation	41
6.13.5.1 buttons	41
6.13.5.2 ID	42
6.13.5.3 KeyBindCount	42
6.13.5.4 keys	42
6.14 Shapes::Line Class Reference	42
6.14.1 Constructor & Destructor Documentation	44
6.14.1.1 Line()	44
6.14.2 Member Function Documentation	44
6.14.2.1 draw()	44

6.14.2.2 setBeginPoint()	44
6.14.2.3 setEndPoint()	44
6.14.2.4 setThickness()	44
6.14.3 Member Data Documentation	44
6.14.3.1 beginPoint	44
6.14.3.2 endPoint	45
6.14.3.3 thickness	45
6.15 Objects::Object Class Reference	45
6.15.1 Detailed Description	47
6.15.2 Constructor & Destructor Documentation	47
6.15.2.1 Object()	47
6.15.2.2 ~Object()	47
6.15.3 Member Function Documentation	47
6.15.3.1 debug()	47
6.15.3.2 flipHorizontal()	47
6.15.3.3 flipVertical()	48
6.15.3.4 getAngle()	48
6.15.3.5 getDimension()	48
6.15.3.6 getFlipFlag()	48
6.15.3.7 getPosition()	48
6.15.3.8 getRenderAngle()	49
6.15.3.9 getRenderRect()	49
6.15.3.10 getTexture()	49
6.15.3.11 getTextureCount()	49
6.15.3.12 getVisibility()	49
6.15.3.13 lookAt()	49
6.15.3.14 move()	50
6.15.3.15 nextTexture()	50
6.15.3.16 previousTexture()	50
6.15.3.17 rotate()	50
6.15.3.18 setAngle()	50
6.15.3.19 setTexture()	51
6.15.3.20 setVisibility()	51
6.15.3.21 stretch()	51
6.15.3.22 stretchX()	52
6.15.3.23 stretchY()	52
6.15.3.24 update()	52
6.15.4 Friends And Related Symbol Documentation	52
6.15.4.1 TextureHandler	52
6.16 Objects::Player Class Reference	53
6.17 Shapes::Rect Class Reference	55
6.17.1 Member Data Documentation	56

6.17.1.1 dimension	56
6.17.1.2 position	56
6.18 Renderer Class Reference	56
6.18.1 Detailed Description	57
6.18.2 Constructor & Destructor Documentation	57
6.18.2.1 Renderer()	57
6.18.3 Member Function Documentation	58
6.18.3.1 clear()	58
6.18.3.2 createTexture()	58
6.18.3.3 debug()	58
6.18.3.4 getInstance()	58
6.18.3.5 moveLayerBottom()	58
6.18.3.6 moveLayerDown()	59
6.18.3.7 moveLayerTop()	59
6.18.3.8 moveLayerUp()	59
6.18.3.9 operator=()	59
6.18.3.10 registerObject()	59
6.18.3.11 removeObject()	60
6.18.3.12 render()	60
6.19 Renderer::RenderKey Class Reference	60
6.19.1 Constructor & Destructor Documentation	61
6.19.1.1 RenderKey() [1/2]	61
6.19.1.2 RenderKey() [2/2]	61
6.20 RenderObjectBase Class Reference	61
6.20.1 Detailed Description	61
6.20.2 Member Function Documentation	62
6.20.2.1 debug()	62
6.21 sdl_deleter Struct Reference	62
6.21.1 Detailed Description	62
6.21.2 Member Function Documentation	62
6.21.2.1 operator>() [1/12]	62
6.21.2.2 operator>() [2/12]	63
6.21.2.3 operator>() [3/12]	63
6.21.2.4 operator>() [4/12]	63
6.21.2.5 operator>() [5/12]	63
6.21.2.6 operator>() [6/12]	63
6.21.2.7 operator>() [7/12]	63
6.21.2.8 operator>() [8/12]	63
6.21.2.9 operator>() [9/12]	63
6.21.2.10 operator>() [10/12]	63
6.21.2.11 operator>() [11/12]	64
6.21.2.12 operator>() [12/12]	64

6.22 SelectionManager< T > Class Template Reference	64
6.22.1 Constructor & Destructor Documentation	65
6.22.1.1 SelectionManager() [1/2]	65
6.22.1.2 SelectionManager() [2/2]	65
6.22.2 Member Function Documentation	65
6.22.2.1 add()	65
6.22.2.2 get()	65
6.22.2.3 getSelectionId()	65
6.22.2.4 next()	66
6.22.2.5 prev()	66
6.22.2.6 remove()	66
6.22.2.7 set()	66
6.22.2.8 size()	66
6.22.3 Member Data Documentation	67
6.22.3.1 SELECTION_NOT_SET	67
6.23 Shapes::Shape Class Reference	67
6.23.1 Constructor & Destructor Documentation	68
6.23.1.1 Shape()	68
6.23.1.2 ~Shape()	69
6.23.2 Member Function Documentation	69
6.23.2.1 draw()	69
6.23.2.2 getColor()	69
6.23.2.3 setColor()	69
6.23.3 Member Data Documentation	69
6.23.3.1 color	69
6.23.3.2 view	69
6.24 TextureHandler Class Reference	69
6.24.1 Detailed Description	70
6.24.2 Constructor & Destructor Documentation	70
6.24.2.1 TextureHandler()	70
6.24.3 Member Function Documentation	70
6.24.3.1 getInstance()	70
6.24.3.2 getTexture()	70
6.24.3.3 operator=()	71
6.25 Vector2D Class Reference	71
6.25.1 Constructor & Destructor Documentation	72
6.25.1.1 Vector2D() [1/2]	72
6.25.1.2 Vector2D() [2/2]	72
6.25.2 Member Function Documentation	72
6.25.2.1 cross()	72
6.25.2.2 dot()	72
6.25.2.3 getX()	72

6.25.2.4 getY()	72
6.25.2.5 len()	72
6.25.2.6 len2()	72
6.25.2.7 norm()	73
6.25.2.8 rotate() [1/2]	73
6.25.2.9 rotate() [2/2]	73
6.25.2.10 zero()	73
6.25.3 Friends And Related Symbol Documentation	73
6.25.3.1 operator*	73
6.25.3.2 operator*= 6.25.3.3 operator+ 6.25.3.4 operator+= 6.25.3.5 operator- [1/2] 6.25.3.6 operator- [2/2] 6.25.3.7 operator-= 6.25.3.8 operator/ 6.25.3.9 operator/=	73 73 73 73 74 74 74 74 74
6.26 Views::View Class Reference	74
6.26.1 Detailed Description	76
6.26.2 Constructor & Destructor Documentation	76
6.26.2.1 View()	76
6.26.2.2 ~View()	76
6.26.3 Member Function Documentation	76
6.26.3.1 getAngle()	76
6.26.3.2 getDimension()	76
6.26.3.3 getPosition()	77
6.26.3.4 getRect()	77
6.26.3.5 getZoom()	77
6.26.3.6 transform()	77
6.26.3.7 transformFromRender()	78
6.26.4 Member Data Documentation	78
6.26.4.1 dimension	78
6.26.4.2 position	78
7 File Documentation	79
7.1 include/command/command.h File Reference	79
7.2 command.h	80
7.3 include/command_manager.h File Reference	81
7.4 command_manager.h	82
7.5 include/config.h File Reference	82
7.6 config.h	83
7.7 include/game_manager.h File Reference	84

7.8 game_manager.h	84
7.9 include/init.h File Reference	84
7.10 init.h	86
7.11 include/input_handler.h File Reference	86
7.11.1 Enumeration Type Documentation	87
7.11.1.1 MouseButton	87
7.12 input_handler.h	88
7.13 include/object/bullet.h File Reference	89
7.14 bullet.h	90
7.15 include/object/button.h File Reference	90
7.16 button.h	91
7.17 include/object/item/item.h File Reference	91
7.18 item.h	92
7.19 include/object/object.h File Reference	92
7.20 object.h	93
7.21 include/object/player.h File Reference	95
7.22 player.h	95
7.23 include/render_object_base.h File Reference	95
7.24 render_object_base.h	96
7.25 include/renderer.h File Reference	96
7.26 renderer.h	97
7.27 include/shape/circle.h File Reference	98
7.28 circle.h	99
7.29 include/shape/line.h File Reference	100
7.30 line.h	101
7.31 include/shape/rect.h File Reference	101
7.32 rect.h	102
7.33 include/shape/shape.h File Reference	102
7.34 shape.h	103
7.35 include/shape/shapes.h File Reference	104
7.36 shapes.h	105
7.37 include/texture/texture_handler.h File Reference	105
7.38 texture_handler.h	106
7.39 include/utility/functions.h File Reference	106
7.40 functions.h	106
7.41 include/utility/pointer_wrappers.h File Reference	107
7.41.1 Typedef Documentation	108
7.41.1.1 sdl_unique_ptr	108
7.41.2 Function Documentation	108
7.41.2.1 sdl_make_shared()	108
7.42 pointer_wrappers.h	108
7.43 include/utility/selection_manager.h File Reference	108

7.44 selection_manager.h	109
7.45 include/utility/utility.h File Reference	110
7.45.1 Macro Definition Documentation	111
7.45.1.1 _USE_MATH_DEFINES	111
7.45.2 Function Documentation	111
7.45.2.1 normalizeAngle()	111
7.45.2.2 polarToCartesian()	112
7.46 utility.h	112
7.47 include/utility/vector2d.h File Reference	112
7.48 vector2d.h	113
7.49 include/view/camera.h File Reference	114
7.50 camera.h	115
7.51 include/view/hud.h File Reference	115
7.52 hud.h	116
7.53 include/view/view.h File Reference	116
7.54 view.h	118
7.55 include/view/views.h File Reference	118
7.56 views.h	119
Index	121

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	28
CommandManager	29
Commands::Command::ExecuteKey	30
GameManager	30
InputHandler	36
Items::Item	40
KeyBind	40
Renderer	56
Renderer::RenderKey	60
RenderObjectBase	61
Objects::Object	45
Objects::Bullet	15
Objects::Button	18
Objects::Player	53
Shapes::Shape	67
Shapes::Circle	25
Shapes::HollowCircle	30
Shapes::Line	42
Shapes::Rect	55
sdl_deleter	62
SelectionManager< T >	64
SelectionManager< SDL_Texture * >	64
TextureHandler	69
Vector2D	71
Views::View	74
Views::Camera	20
Views::HUD	33

Chapter 3

Class Index

3.1 Class List

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

Objects::Bullet	15
Objects::Button	18
Views::Camera	
Camera for following object or stationary view	20
Shapes::Circle	25
Commands::Command	
Commands base abstract class	28
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>	29
Commands::Command::ExecuteKey	30
GameManager	30
Shapes::HollowCircle	30
Views::HUD	33
InputHandler	
This is a global singleton class of handling user inputs. Wrapper class of <code>SDL_PollEvent</code> and events handling	36
Items::Item	40
KeyBind	
KeyBind structure for key bindings	40
Shapes::Line	42
Objects::Object	
Object type for all renderable objects in the world note: the texture won't be created until loaded into the renderer	45
Objects::Player	53
Shapes::Rect	55
Renderer	
Required key to call render() in	56
Renderer::RenderKey	60
RenderObjectBase	
Empty render object base class category	61
sdl_deleter	
Generic deleter functor for SDL resources. For use with std smart pointers	62
SelectionManager< T >	64
Shapes::Shape	67

TextureHandler	
This is a global singleton class for texture handling	69
Vector2D	71
Views::View	
View : defines a view area, translates the objects' virtual rects to real rendering rects	74

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

include/command_manager.h	81
include/config.h	82
include/game_manager.h	84
include/init.h	84
include/input_handler.h	86
include/render_object_base.h	95
include/renderer.h	96
include/command/command.h	79
include/object/bullet.h	89
include/object/button.h	90
include/object/object.h	92
include/object/player.h	95
include/object/item/item.h	91
include/shape/circle.h	98
include/shape/line.h	100
include/shape/rect.h	101
include/shape/shape.h	102
include/shape/shapes.h	104
include/texture/texture_handler.h	105
include/utility/functions.h	106
include/utility/pointer_wrappers.h	107
include/utility/selection_manager.h	108
include/utility/utility.h	110
include/utility/vector2d.h	112
include/view/camera.h	114
include/view/hud.h	115
include/view/view.h	116
include/view/views.h	118

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 [Bullet](#)
- 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
```

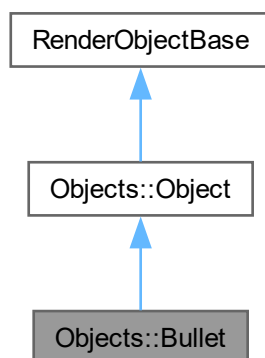
Chapter 6

Class Documentation

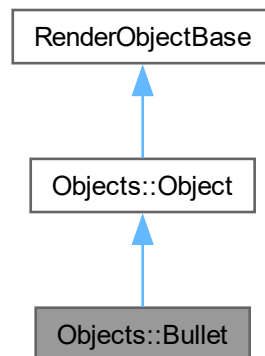
6.1 Objects::Bullet Class Reference

```
#include <bullet.h>
```

Inheritance diagram for Objects::Bullet:



Collaboration diagram for Objects::Bullet:



Public Member Functions

- **Bullet** (const [Views::View](#) *view, [Vector2D](#) position, float angle, float speed=0.5f)
- [Uint32](#) **getAliveTime** (void) const noexcept
Gets the alive time of this bullet.
- void **update** (void) noexcept override
Updates the object state.

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.
- float **getRenderAngle** (void) const noexcept
Gets the render angle of the object.
- 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 `debug` (void) const noexcept override

6.1.1 Constructor & Destructor Documentation

6.1.1.1 Bullet()

```
Objects::Bullet::Bullet (
    const Views::View * view,
    Vector2D position,
    float angle,
    float speed = 0.5f ) [inline]
```

6.1.2 Member Function Documentation

6.1.2.1 getAliveTime()

```
Uint32 Objects::Bullet::getAliveTime (
    void ) const [noexcept]
```

Gets the alive time of this bullet.

Returns

The alive time of this bullet.

6.1.2.2 update()

```
void Objects::Bullet::update (  
    void ) [override], [virtual], [noexcept]
```

Updates the object state.

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

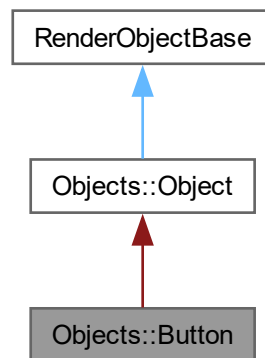
The documentation for this class was generated from the following file:

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

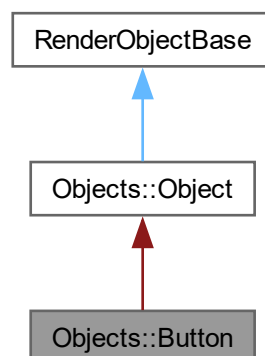
6.2 Objects::Button Class Reference

```
#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

Updates the object state.

6.2.1 Constructor & Destructor Documentation

6.2.1.1 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 )
```

6.2.2 Member Function Documentation

6.2.2.1 onClick()

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

6.2.2.2 setHovered()

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

6.2.2.3 update()

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

Updates the object state.

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

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

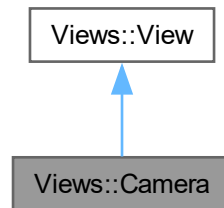
- include/object/[button.h](#)

6.3 Views::Camera Class Reference

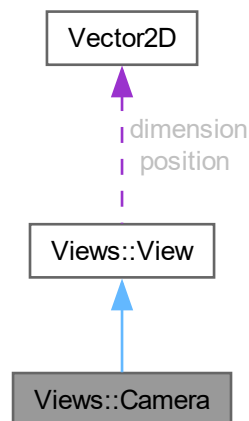
[Camera](#) for following object or stationary view.

```
#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

6.3.1 Detailed Description

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

6.3.2 Constructor & Destructor Documentation

6.3.2.1 Camera()

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

6.3.3 Member Function Documentation

6.3.3.1 `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](#).

6.3.3.2 `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](#).

6.3.3.3 `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](#).

6.3.3.4 `rotate()`

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

Rotates the view by @diffAngle.

Parameters

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

6.3.3.5 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.
--------------	-------------------------------

6.3.3.6 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.
---------------------	-----------------------------------

6.3.3.7 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.
--------------------	-------------------------

6.3.3.8 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.
--------------------	----------------------------------

6.3.3.9 setZoom()

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

Sets the zoom level of the camera.

Parameters

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

6.3.3.10 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](#).

6.3.3.11 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](#).

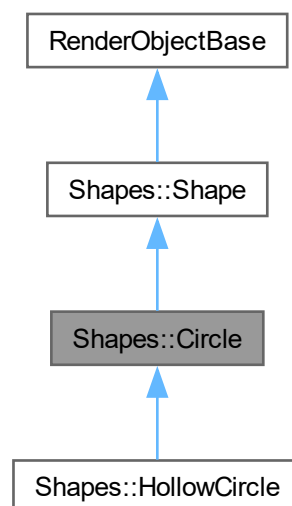
The documentation for this class was generated from the following file:

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

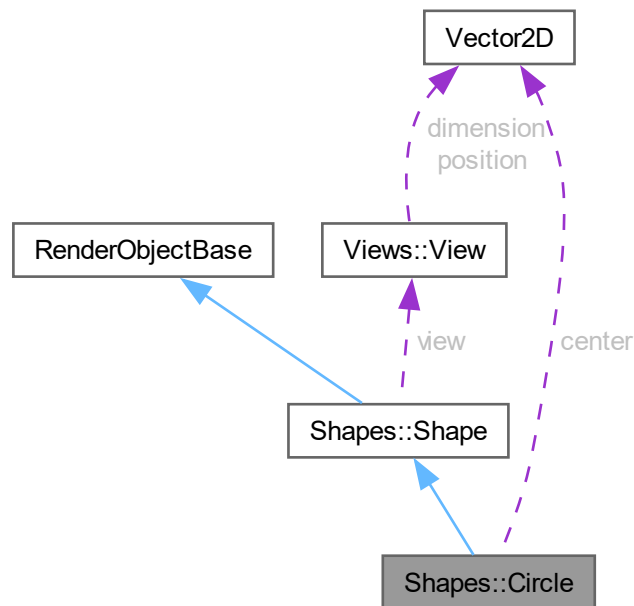
6.4 Shapes::Circle Class Reference

```
#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](#) ()=default
- 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*
- [SDL_Color](#) *color*

6.4.1 Constructor & Destructor Documentation

6.4.1.1 Circle()

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

6.4.2 Member Function Documentation

6.4.2.1 draw()

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

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

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

6.4.2.2 setCenter()

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

6.4.2.3 setRadius()

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

6.4.3 Member Data Documentation

6.4.3.1 center

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

6.4.3.2 radius

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

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

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

6.5 Commands::Command Class Reference

[Commands](#) base abstract class.

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

Classes

- class [ExecuteKey](#)

Public Member Functions

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

6.5.1 Detailed Description

[Commands](#) base abstract class.

6.5.2 Constructor & Destructor Documentation

6.5.2.1 ~Command()

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

6.5.3 Member Function Documentation

6.5.3.1 execute()

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

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

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

6.6 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.

6.6.1 Detailed Description

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

6.6.2 Member Function Documentation

6.6.2.1 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.

6.6.2.2 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](#)

6.7 Commands::Command::ExecuteKey Class Reference

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

Friends

- class [CommandManager](#)

6.7.1 Friends And Related Symbol Documentation

6.7.1.1 CommandManager

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

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

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

6.8 GameManager Class Reference

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

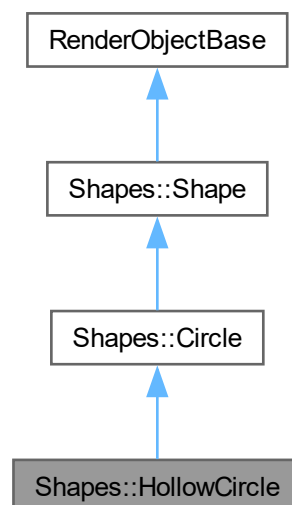
The documentation for this class was generated from the following file:

- include/[game_manager.h](#)

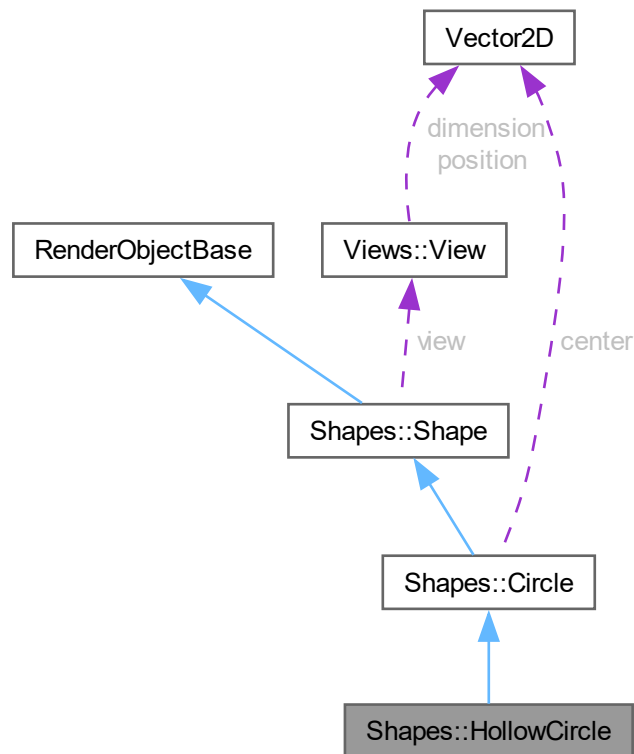
6.9 Shapes::HollowCircle Class Reference

```
#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](#) ()=default
- 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](#)
- [SDL_Color](#) [color](#)

6.9.1 Constructor & Destructor Documentation

6.9.1.1 [HollowCircle\(\)](#)

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

6.9.2 Member Function Documentation

6.9.2.1 [draw\(\)](#)

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

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

6.9.2.2 [setThickness\(\)](#)

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

6.9.3 Member Data Documentation

6.9.3.1 thickness

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

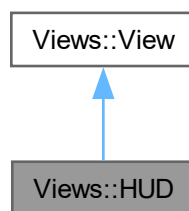
The documentation for this class was generated from the following file:

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

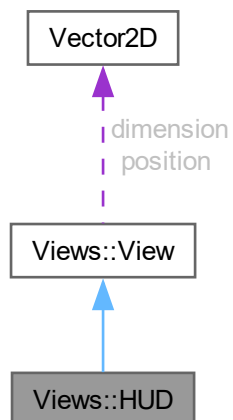
6.10 Views::HUD Class Reference

```
#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](#)

6.10.1 Constructor & Destructor Documentation

6.10.1.1 HUD()

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

6.10.2 Member Function Documentation

6.10.2.1 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](#).

6.10.2.2 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](#).

6.10.2.3 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](#)

6.11 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
Polls if a key is pressed. (SDL_KeyDown) Is only true when the key was not held down in the previous tick.
- bool [pollKeyRelease](#) (SDL_Keycode key) noexcept
Checks if a key is released. (SDL_KeyUp) Is only true when the key was held down in the last tick.
- 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

6.11.1 Detailed Description

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

6.11.2 Constructor & Destructor Documentation

6.11.2.1 InputHandler()

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


6.11.3 Member Function Documentation

6.11.3.1 getInstance()

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

6.11.3.2 getMousePosition()

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

6.11.3.3 holdTime() [1/2]

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

6.11.3.4 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.

6.11.3.5 isButtonDown()

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

6.11.3.6 isButtonUp()

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

6.11.3.7 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.

6.11.3.8 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.

6.11.3.9 operator=()

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

6.11.3.10 pollButtonPress()

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

6.11.3.11 pollButtonRelease()

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

6.11.3.12 pollKeyPress()

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

Polls if a key is pressed. (SDL_KeyDown) Is only true when the key was not held down in the previous tick.

Parameters

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

Returns

Whether the key was pressed.

6.11.3.13 pollKeyRelease()

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

Checks if a key is released. (SDL_KeyUp) Is only true when the key was held down in the last tick.

Parameters

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

Returns

Whether the key was released.

6.11.3.14 pollMouseScroll()

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

6.11.3.15 receiveEvent() [1/3]

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

6.11.3.16 receiveEvent() [2/3]

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

6.11.3.17 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](#)

6.12 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)

6.12.1 Constructor & Destructor Documentation

6.12.1.1 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](#)

6.13 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)

6.13.1 Detailed Description

[KeyBind](#) structure for key bindings.

6.13.2 Member Enumeration Documentation

6.13.2.1 Trigger

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

Enumerator

TAP	
HOLD	
RELEASE	

6.13.3 Constructor & Destructor Documentation

6.13.3.1 KeyBind()

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

6.13.4 Friends And Related Symbol Documentation

6.13.4.1 operator<

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

6.13.5 Member Data Documentation

6.13.5.1 buttons

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

6.13.5.2 ID

```
int KeyBind::ID
```

6.13.5.3 KeyBindCount

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

6.13.5.4 keys

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

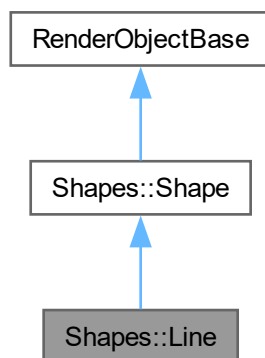
The documentation for this struct was generated from the following file:

- include/[command_manager.h](#)

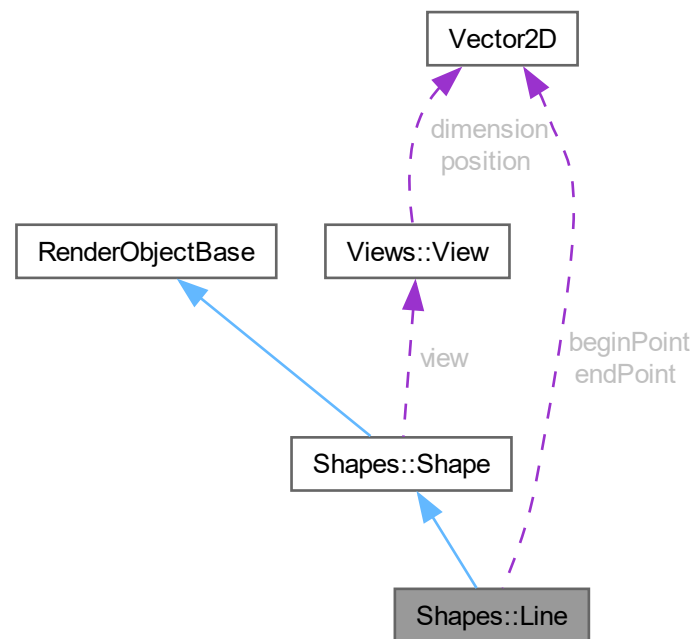
6.14 Shapes::Line Class Reference

```
#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](#) ()=default
- 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](#)
- [SDL_Color](#) [color](#)

6.14.1 Constructor & Destructor Documentation

6.14.1.1 Line()

```
Shapes::Line::Line (
    Views::View * view,
    Vector2D \_beginPoint,
    Vector2D \_endPoint,
    uint8\_t \_thickness,
    SDL\_Color color = {0, 0, 0, 255} ) [noexcept]
```

6.14.2 Member Function Documentation

6.14.2.1 draw()

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

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

6.14.2.2 setBeginPoint()

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

6.14.2.3 setEndPoint()

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

6.14.2.4 setThickness()

```
void Shapes::Line::setThickness (
    uint8\_t newThickness ) [noexcept]
```

6.14.3 Member Data Documentation

6.14.3.1 beginPoint

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


6.14.3.2 endPoint

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

6.14.3.3 thickness

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

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

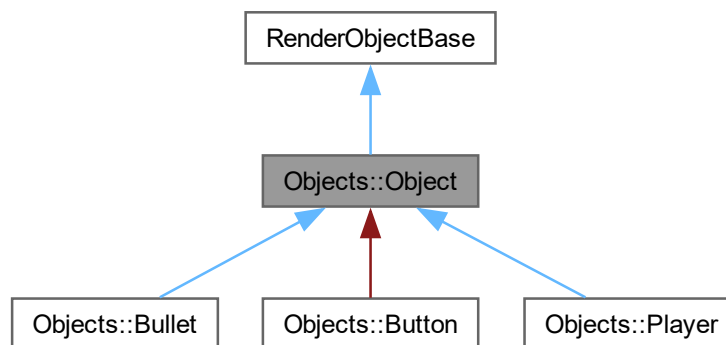
- [include/shape/line.h](#)

6.15 Objects::Object Class Reference

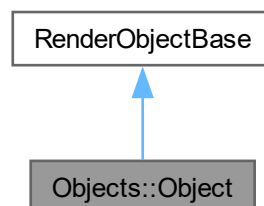
[Object](#) type for all renderable objects in the world note: the texture won't be created until loaded into the renderer.

```
#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 $[0, 2\pi)$, with 0 set at positive x direction, and going counter-clockwise.
- float [getRenderAngle](#) (void) const noexcept
Gets the render angle of the object.
- 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.
- virtual void [update](#) (void) noexcept
Updates the object state.
- void [debug](#) (void) const noexcept override

Friends

- class [TextureHandler](#)

6.15.1 Detailed Description

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

6.15.2 Constructor & Destructor Documentation

6.15.2.1 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)

6.15.2.2 ~Object()

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

6.15.3 Member Function Documentation

6.15.3.1 debug()

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

Reimplemented from [RenderObjectBase](#).

6.15.3.2 flipHorizontal()

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

Flips the object horizontally.

6.15.3.3 flipVertical()

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

Flips the object vertically.

6.15.3.4 getAngle()

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

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

Returns

The angle which the object is facing.

6.15.3.5 getDimension()

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

Gets the dimension of the object.

Returns

The object's dimension.

6.15.3.6 getFlipFlag()

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

Returns the flip flag used by SDL.

Returns

A SDL_RendererFlip flag.

6.15.3.7 getPosition()

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

Gets the position of the object.

Returns

The object's location.

6.15.3.8 getRenderAngle()

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

Gets the render angle of the object.

Returns

The render angle of the object

6.15.3.9 getRenderRect()

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

Gets render rectangle for rendering.

Returns

The SDL_FRect for rendering.

6.15.3.10 getTexture()

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

Gets current texture.

Returns

The current texture the object is using.

6.15.3.11 getTextureCount()

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

Gets the number of textures this object has.

Returns

Number of textures.

6.15.3.12 getVisibility()

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

Gets the object's visibility.

Returns

The object's visibility.

6.15.3.13 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.

6.15.3.14 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.
------------------	-------------------------------

6.15.3.15 nextTexture()

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

Set to next texture, texture ID wraps around.

6.15.3.16 previousTexture()

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

Set to previous texture, texture ID wraps around.

6.15.3.17 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.

6.15.3.18 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)

6.15.3.19 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).

6.15.3.20 setVisibility()

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

Sets the object's visibility.

Parameters

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

6.15.3.21 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.

6.15.3.22 stretchX()

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

Stretches the object's width by.

Parameters

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

6.15.3.23 stretchY()

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

Stretches the object's height by.

Parameters

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

6.15.3.24 update()

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

Updates the object state.

Reimplemented in [Objects::Button](#), and [Objects::Bullet](#).

6.15.4 Friends And Related Symbol Documentation

6.15.4.1 TextureHandler

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

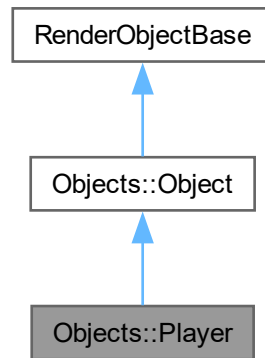
The documentation for this class was generated from the following file:

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

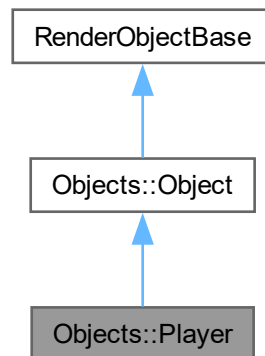
6.16 Objects::Player Class Reference

```
#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.
- float `getRenderAngle` (void) const noexcept
Gets the render angle of the object.
- 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.
- virtual void `update` (void) noexcept
Updates the object state.
- void `debug` (void) const noexcept override

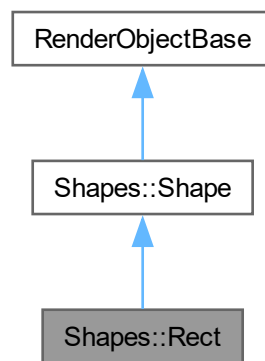
The documentation for this class was generated from the following file:

- include/object/player.h

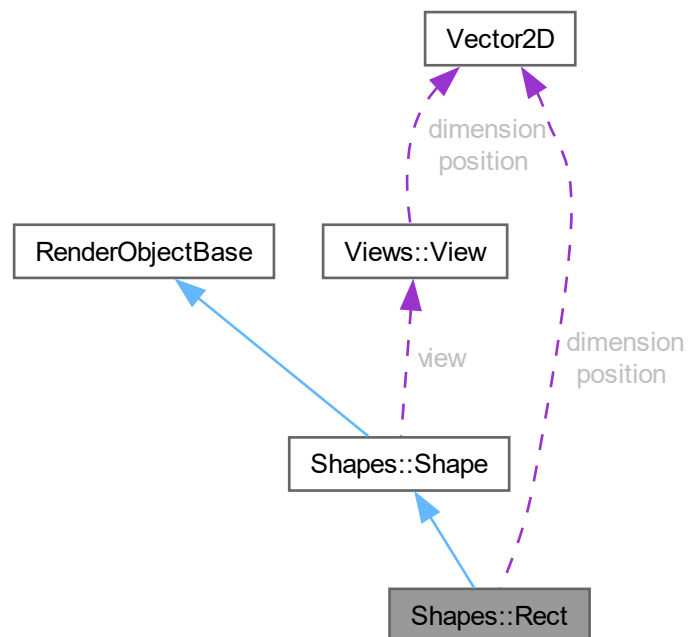
6.17 Shapes::Rect Class Reference

```
#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
- [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](#) ()=default
- 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

6.17.1 Member Data Documentation

6.17.1.1 dimension

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

6.17.1.2 position

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

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

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

6.18 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::weak_ptr< [RenderObjectBase](#) > objectPtr) noexcept
Unregisters the object for rendering.
- void [render](#) (const [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 if @objectPtr is not registered.
- void [moveLayerDown](#) (std::shared_ptr< [RenderObjectBase](#) > objectPtr)
Moves the object down one layer. Throws std::invalid_argument if @objectPtr is not registered.
- void [moveLayerTop](#) (std::shared_ptr< [RenderObjectBase](#) > objectPtr)
Moves the object to the top layer. Throws std::invalid_argument if @objectPtr is not registered.
- void [moveLayerBottom](#) (std::shared_ptr< [RenderObjectBase](#) > objectPtr)
Moves the object to the bottom layer. Throws std::invalid_argument if @objectPtr is not registered.
- 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

6.18.1 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.

6.18.2 Constructor & Destructor Documentation

6.18.2.1 [Renderer\(\)](#)

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

6.18.3 Member Function Documentation

6.18.3.1 clear()

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

Clears object set and unloads all textures.

6.18.3.2 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.

6.18.3.3 debug()

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

Prints renderer debug info.

6.18.3.4 getInstance()

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

6.18.3.5 moveLayerBottom()

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

Moves the object to the bottom layer. Throws std::invalid_argument if @objectPtr is not registered.

Parameters

<i>objectPtr</i>	The object to be moved.
------------------	-------------------------

6.18.3.6 moveLayerDown()

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

Moves the object down one layer. Throws `std::invalid_argument` if `@objectPtr` is not registered.

Parameters

<i>objectPtr</i>	The object to be moved.
------------------	-------------------------

6.18.3.7 moveLayerTop()

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

Moves the object to the top layer. Throws `std::invalid_argument` if `@objectPtr` is not registered.

Parameters

<i>objectPtr</i>	The object to be moved.
------------------	-------------------------

6.18.3.8 moveLayerUp()

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

Moves the object up one layer. Throws `std::invalid_argument` if `@objectPtr` is not registered.

Parameters

<i>objectPtr</i>	The object to be moved.
------------------	-------------------------

6.18.3.9 operator=()

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

6.18.3.10 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>	std::shared_ptr of the object
------------------	-------------------------------

Returns

Whether the object was successfully registered

6.18.3.11 removeObject()

```
bool Renderer::removeObject (
    std::weak_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.

6.18.3.12 render()

```
void Renderer::render (
    const 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](#)

6.19 Renderer::RenderKey Class Reference

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

Public Member Functions

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

6.19.1 Constructor & Destructor Documentation

6.19.1.1 RenderKey() [1/2]

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

6.19.1.2 RenderKey() [2/2]

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

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

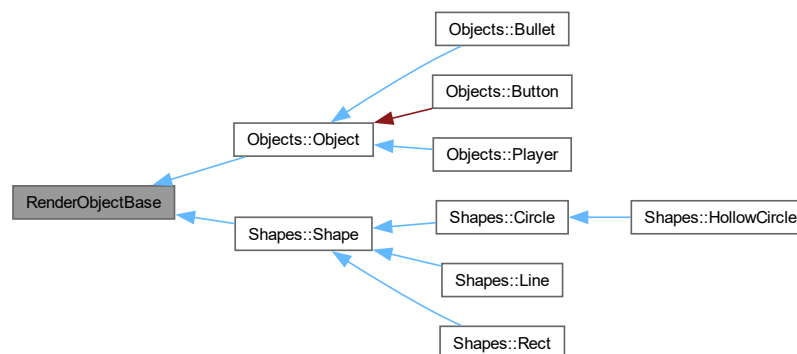
- include/[renderObjectBase.h](#)

6.20 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

6.20.1 Detailed Description

Empty render object base class category.

6.20.2 Member Function Documentation

6.20.2.1 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](#)

6.21 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

6.21.1 Detailed Description

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

6.21.2 Member Function Documentation

6.21.2.1 operator>() [1/12]

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

6.21.2.2 operator>() [2/12]

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

6.21.2.3 operator>() [3/12]

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

6.21.2.4 operator>() [4/12]

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

6.21.2.5 operator>() [5/12]

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

6.21.2.6 operator>() [6/12]

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

6.21.2.7 operator>() [7/12]

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

6.21.2.8 operator>() [8/12]

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

6.21.2.9 operator>() [9/12]

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

6.21.2.10 operator>() [10/12]

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

6.21.2.11 operator>() [11/12]

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

6.21.2.12 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](#)

6.22 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

6.22.1 Constructor & Destructor Documentation

6.22.1.1 SelectionManager() [1/2]

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

6.22.1.2 SelectionManager() [2/2]

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

6.22.2 Member Function Documentation

6.22.2.1 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.

6.22.2.2 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.

6.22.2.3 getSelectionId()

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

Gets the current selection ID.

Returns

The current selection ID.

6.22.2.4 next()

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

Set to next selection.

6.22.2.5 prev()

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

Set to previous selection.

6.22.2.6 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.

6.22.2.7 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.

6.22.2.8 size()

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

Gets the count of available selections.

Returns

The count of available selections.

6.22.3 Member Data Documentation

6.22.3.1 SELECTION_NOT_SET

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

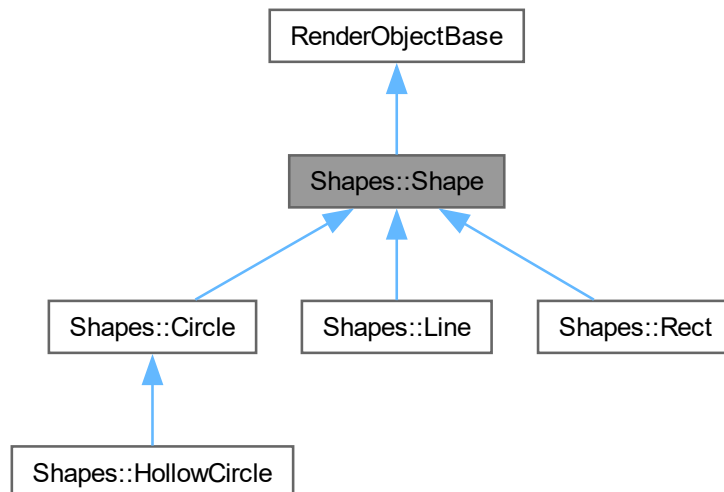
The documentation for this class was generated from the following file:

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

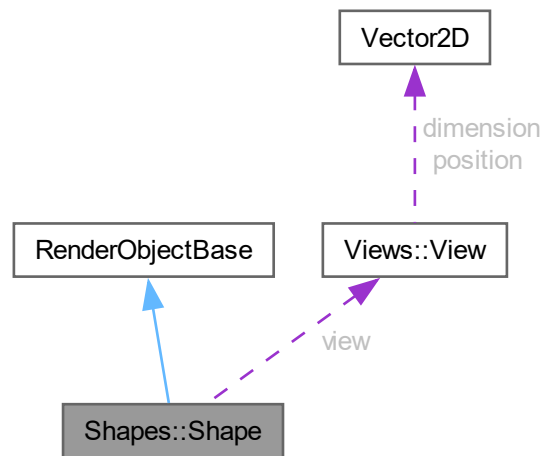
6.23 Shapes::Shape Class Reference

```
#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` ()=default
- 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`
- SDL_Color `color`

6.23.1 Constructor & Destructor Documentation

6.23.1.1 Shape()

```

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

```


6.23.1.2 ~Shape()

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

6.23.2 Member Function Documentation

6.23.2.1 draw()

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

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

6.23.2.2 getColor()

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

6.23.2.3 setColor()

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

6.23.3 Member Data Documentation

6.23.3.1 color

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

6.23.3.2 view

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

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

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

6.24 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)

6.24.1 Detailed Description

This is a global singleton class for texture handling.

Required key to request texture from.

6.24.2 Constructor & Destructor Documentation

6.24.2.1 TextureHandler()

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

6.24.3 Member Function Documentation

6.24.3.1 getInstance()

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

6.24.3.2 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.

6.24.3.3 operator=()

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

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

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

6.25 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

6.25.1 Constructor & Destructor Documentation

6.25.1.1 Vector2D() [1/2]

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

6.25.1.2 Vector2D() [2/2]

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

6.25.2 Member Function Documentation

6.25.2.1 cross()

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

6.25.2.2 dot()

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

6.25.2.3 getX()

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

6.25.2.4 getY()

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

6.25.2.5 len()

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

6.25.2.6 len2()

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

6.25.2.7 norm()

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

6.25.2.8 rotate() [1/2]

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

6.25.2.9 rotate() [2/2]

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

6.25.2.10 zero()

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

6.25.3 Friends And Related Symbol Documentation

6.25.3.1 operator*

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

6.25.3.2 operator*==

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

6.25.3.3 operator+

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

6.25.3.4 operator+=

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

6.25.3.5 operator- [1/2]

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

6.25.3.6 operator- [2/2]

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

6.25.3.7 operator-=

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

6.25.3.8 operator/

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

6.25.3.9 operator/=

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

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

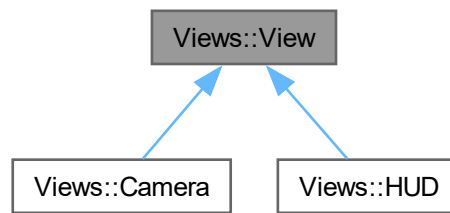
- include/utility/[vector2d.h](#)

6.26 Views::View Class Reference

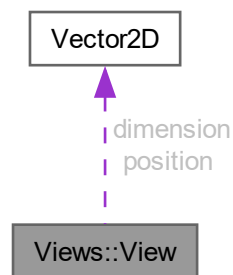
[View](#): defines a view area, translates the objects' virtual rects to real rendering rects.

```
#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

6.26.1 Detailed Description

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

6.26.2 Constructor & Destructor Documentation

6.26.2.1 View()

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

6.26.2.2 ~View()

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

6.26.3 Member Function Documentation

6.26.3.1 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](#).

6.26.3.2 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.

6.26.3.3 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.

6.26.3.4 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](#).

6.26.3.5 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](#).

6.26.3.6 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](#).

6.26.3.7 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](#).

6.26.4 Member Data Documentation**6.26.4.1 dimension**

```
Vector2D Views::View::dimension [protected]
```

6.26.4.2 position

```
Vector2D Views::View::position [protected]
```

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

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

Chapter 7

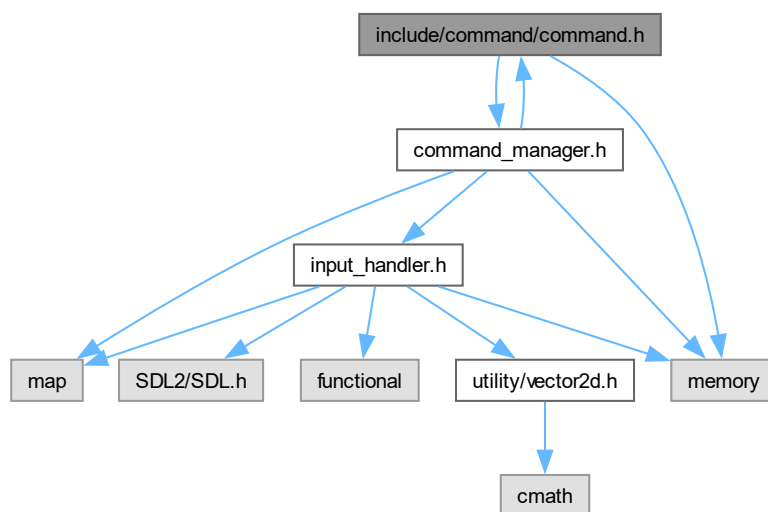
File Documentation

7.1 include/command/command.h File Reference

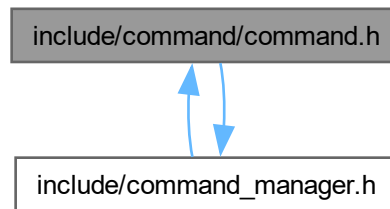
```
#include <command_manager.h>
```

```
#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.](#)

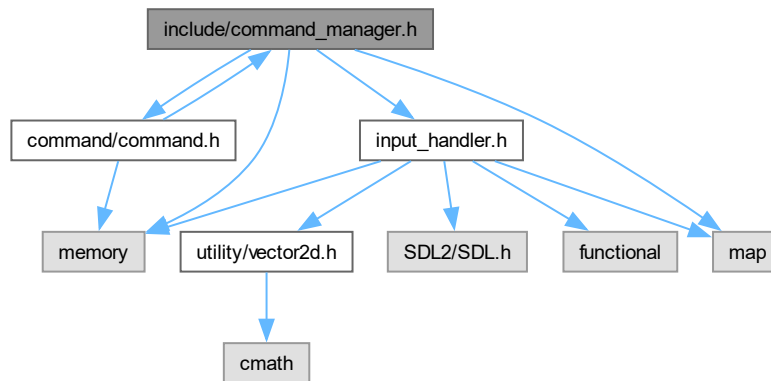
```

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(const ExecuteKey&) {};
00024     };
00025 }
  
```

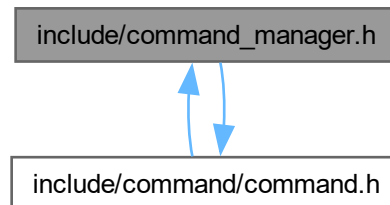
7.3 include/command_manager.h File Reference

```
#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 };

```

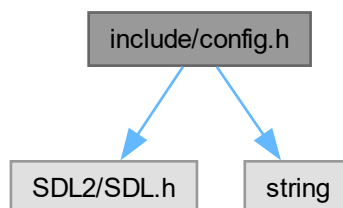
7.5 include/config.h File Reference

```

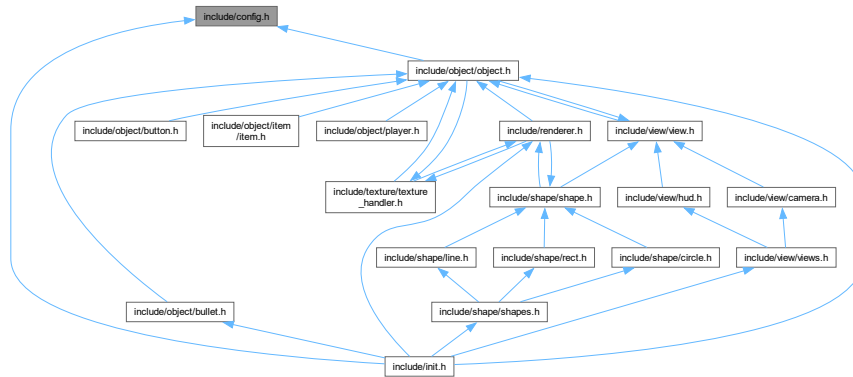
#include <SDL2/SDL.h>
#include <string>

```

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

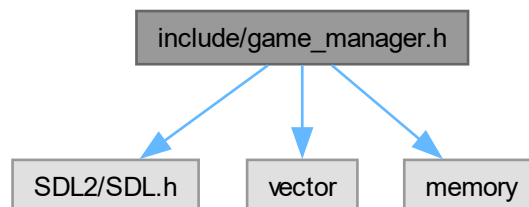
[Go to the documentation of this file.](#)

```
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 };
```

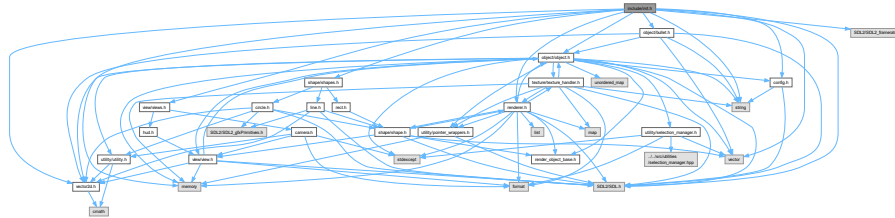
7.9 include/init.h File Reference

```
#include <object/object.h>
#include <object/bullet.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 <object/bullet.h>
00005 #include <view/views.h>
00006 #include <renderer.h>
00007 #include <config.h>
00008 #include <utility/vector2d.h>
00009 #include <shape/shapes.h>
00010 #include <SDL2/SDL.h>
00011 #include <SDL2/SDL2_framerate.h>
00012 #include <memory>
00013 #include <string>
00014 #include <vector>
00015
00016 namespace Global {
00017     extern std::unique_ptr<FPSmanager> fpsManager;
00018     extern std::unique_ptr<Views::Camera> playerCamera;
00019     extern std::unique_ptr<Views::HUD> hudView;
00020     extern std::unique_ptr<Views::HUD> menuView;
00021
00022     extern std::shared_ptr<Objects::Object> playerObject, arrowObject1;
00023     extern std::shared_ptr<Objects::Object> arrowObject2;
00024     extern std::shared_ptr<Shapes::Circle> yellowCircle;
00025     extern std::shared_ptr<Shapes::Circle> greenCircle;
00026     extern std::shared_ptr<Shapes::Circle> blueCircle;
00027     extern std::shared_ptr<Shapes::Circle> redCircle;
00028     extern std::shared_ptr<Shapes::Circle> purpleCircle;
00029
00030     extern std::shared_ptr<Shapes::HollowCircle> hollowCircle1;
00031     extern std::shared_ptr<Shapes::Line> line1;
00032     extern std::shared_ptr<Shapes::Line> line2;
00033     extern std::shared_ptr<Shapes::Line> line3;
00034     extern std::shared_ptr<Shapes::Line> line4;
00035
00036     extern std::shared_ptr<Shapes::Line> crosshairLine1;
00037     extern std::shared_ptr<Shapes::Line> crosshairLine2;
00038     extern std::shared_ptr<Shapes::HollowCircle> crosshairCircle1;
00039
00040     void init();
00041 }

```

7.11 include/input_handler.h File Reference

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

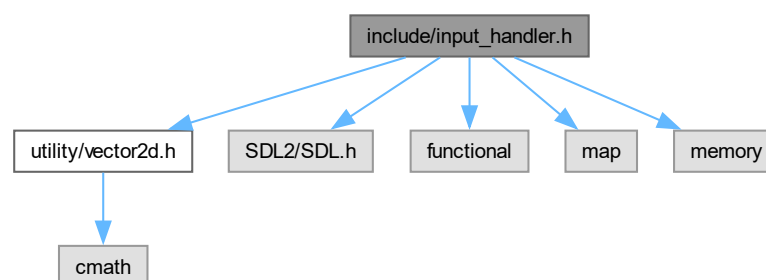
```
#include <SDL2/SDL.h>
```

```
#include <functional>
```

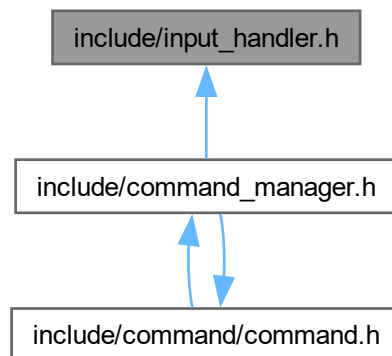
```
#include <map>
```

```
#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     // Keyboard functions
00062
00063     bool pollKeyPress(SDL_Keycode key) noexcept;
00064
00065     bool pollKeyRelease(SDL_Keycode key) noexcept;
00066
00067     bool isKeyDown(SDL_Keycode key) const noexcept;
00068
00069     bool isKeyUp(SDL_Keycode key) const noexcept;
00070
00071     uint32_t holdTime(SDL_Keycode key) const noexcept;
00072
00073     // Mouse functions
00074
00075     bool pollButtonPress(MouseButton button) noexcept;
00076     bool pollButtonRelease(MouseButton button) noexcept;
00077     bool isButtonDown(MouseButton button) const noexcept;
00078     bool isButtonUp(MouseButton button) const noexcept;
00079     uint32_t holdTime(MouseButton button) const noexcept;
00080
00081     Vector2D getMousePosition(void) const noexcept;
00082
00083     Vector2D pollMouseScroll(void) noexcept;
00084
00085
00086
00087
00088
00089
00090
00091
00092
00093
00094
00095
00096
00097
00098
00099
00100
00101
00102
00103
00104
00105
00106
00107
00108
00109
00110
00111
00112

```

```

00113     // Event Receivers
00114
00115     void receiveEvent(SDL_KeyboardEvent keyboardEvent) noexcept;
00116     void receiveEvent(SDL_MouseButtonEvent mouseButtonEvent) noexcept;
00117     void receiveEvent(SDL_MouseWheelEvent mouseWheelEvent) noexcept;
00118     //void receiveEvent(SDL_MouseMotionEvent mouseMotionEvent) noexcept;
00119 };

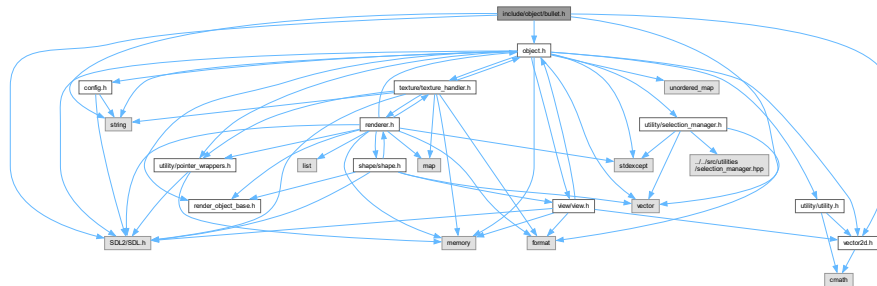
```

7.13 include/object/bullet.h File Reference

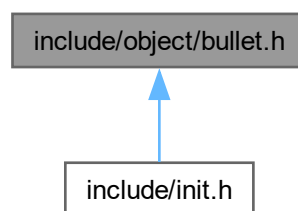
```

#include "object.h"
#include <utility/vector2d.h>
#include <SDL2/SDL.h>
#include <vector>
#include <string>
Include dependency graph for bullet.h:

```



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



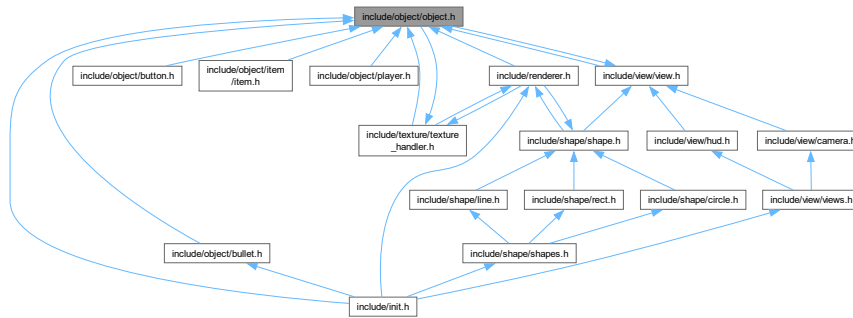
Classes

- class [Objects::Bullet](#)

Namespaces

- namespace [Objects](#)

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



Classes

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

Namespaces

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

7.20 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/textur_handler.h>
00009 #include <view/view.h>
00010 #include <config.h>
00011 #include <SDL2/SDL.h>
00012 #include <memory>
00013 #include <string>
00014 #include <vector>
00015 #include <unordered_map>
00016 #include <stdexcept>
00017
00018 namespace Views { class View; }
00019 class TextureHandler;
00020
00021 namespace Objects {
00022
00023     // TODO: add 'shapes' field to `Objects::Object`
00024
00029     class Object : public RenderObjectBase {
00030     friend class TextureHandler;
00031     private:
00032         SelectionManager<SDL_Texture*> textures;
00033         bool visible;
00034
00035         float angle; // stored as radians
00036         SDL_RendererFlip flipFlag;
00037         // SDL_Color colorMask; // color mod mask
00038         Vector2D position; // actual position in the world
00039         Vector2D dimension; // height and width
00040         const Views::View* view;

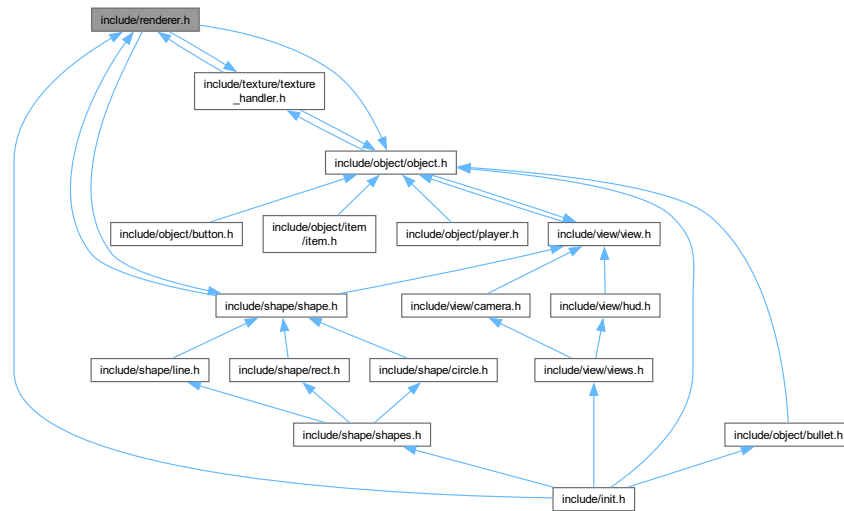
```

```

00041     public:
00042
00050         Object(
00051             const std::vector<std::string>& textureNames,
00052             const Views::View* _view,
00053             const Vector2D& _position,
00054             const Vector2D& _dimension
00055         );
00056
00057         virtual ~Object() = default;
00058
00065         float getAngle(void) const noexcept;
00066
00071         float getRenderAngle(void) const noexcept;
00072
00077         void setAngle(float newAngle) noexcept;
00078
00084         void rotate(float diffAngle) noexcept;
00085
00090         SDL_RendererFlip getFlipFlag(void) const noexcept;
00091
00096         Vector2D getPosition(void) const noexcept;
00097
00102         Vector2D getDimension(void) const noexcept;
00103
00108         void move(const Vector2D& translate) noexcept;
00109
00114         void stretchX(float ratio) noexcept;
00115
00120         void stretchY(float ratio) noexcept;
00121
00126         void stretch(float ratio) noexcept;
00127
00131         void flipHorizontal(void) noexcept;
00132
00136         void flipVertical(void) noexcept;
00137
00142         void setVisibility(bool visibility) noexcept;
00143
00148         bool getVisibility(void) const noexcept;
00149
00150
00151         /* TEXTURES */
00152
00156         void nextTexture(void) noexcept;
00157
00161         void previousTexture(void) noexcept;
00162
00167         void setTexture(int textureId) noexcept;
00168
00173         size_t getTextureCount(void) const noexcept;
00174
00179         SDL_Texture* getTexture(void) const noexcept;
00180
00181         /* TEXTURES */
00182
00183
00188         virtual void lookAt(const Vector2D& position) noexcept;
00189
00194         SDL_FRect getRenderRect(void) const noexcept;
00195         //Vector2D getRenderRelativePosition(Vector2D renderPosition) const noexcept;
00196
00200         virtual void update(void) noexcept;
00201
00202         // debug
00203         void debug(void) const noexcept override;
00204     };
00205 }

```


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](#)

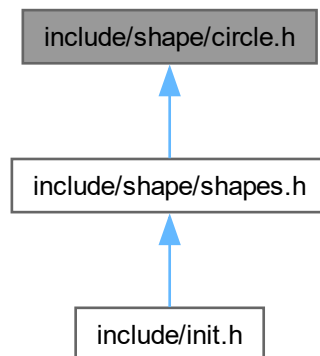
7.26 renderer.h

[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 class Renderer {
00023     class CreateTextureKey {
00024         friend class TextureHandler;
00025     private:
00026         CreateTextureKey() = default;
00027     };
00028 };
  
```


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



Classes

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

Namespaces

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

7.28 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
00009 namespace Views {
00010     class View;
00011 };
00012
00013 namespace Shapes {
00014     class Circle : public Shape {
00015     protected:
00016         Vector2D center;
00017         float radius;
00018     public:
00019         Circle(
00020             Views::View* view,
00021             const Vector2D& center,
00022             float radius,
00023             SDL_Color color = { 0, 0, 0, 255 }
00024         ) noexcept;
00025         void setCenter(const Vector2D& newCenter) noexcept;
00026         void setRadius(float newRadius) noexcept;
00027         void draw(SDL_Renderer* renderer) const noexcept override;
00028     };
  
```


Classes

- class Shapes::Line

Namespaces

- namespace **Shapes**

7.30 line.h

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

```

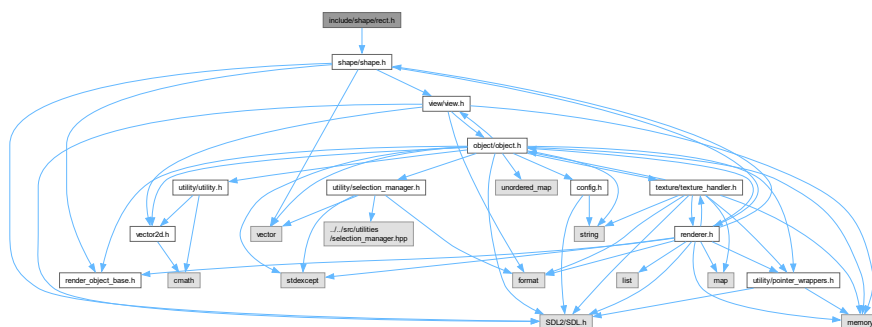
00001 #pragma once
00002
00003 #include <shape/shape.h>
00004 #include <utility/vector2d.h>
00005 #include <SDL2/SDL.h>
00006 #include <SDL2/SDL2_gfxPrimitives.h>
00007
00008 namespace Shapes {
00009     class Line : public Shape {
00010     protected:
00011         Vector2D beginPoint;
00012         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 }

```

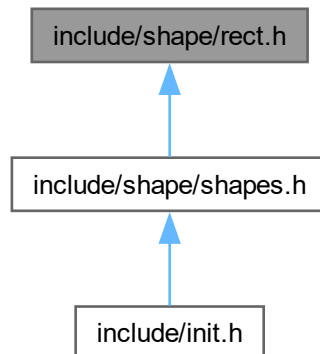
7.31 include/shape/rect.h File Reference

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

Include dependency graph for rect.h:



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



Classes

- class [Shapes::Rect](#)

Namespaces

- namespace [Shapes](#)

7.32 rect.h

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

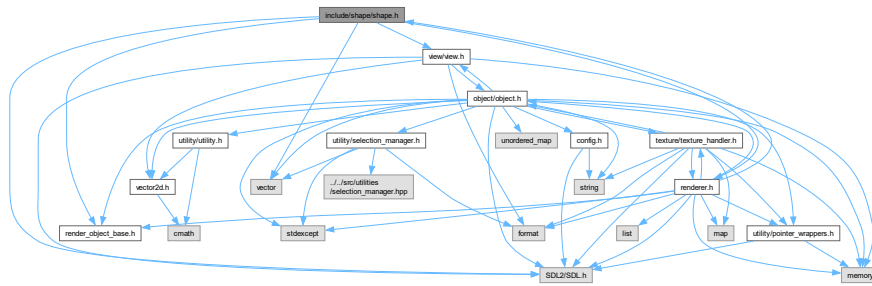
```
00001 #pragma once
00002
00003 #include <shape/shape.h>
00004
00005 namespace Shapes {
00006     class Rect : public Shape {
00007     private:
00008         //void draw()
00009     protected:
00010         Vector2D position;
00011         Vector2D dimension;
00012     };
00013 }
```

7.33 include/shape/shape.h File Reference

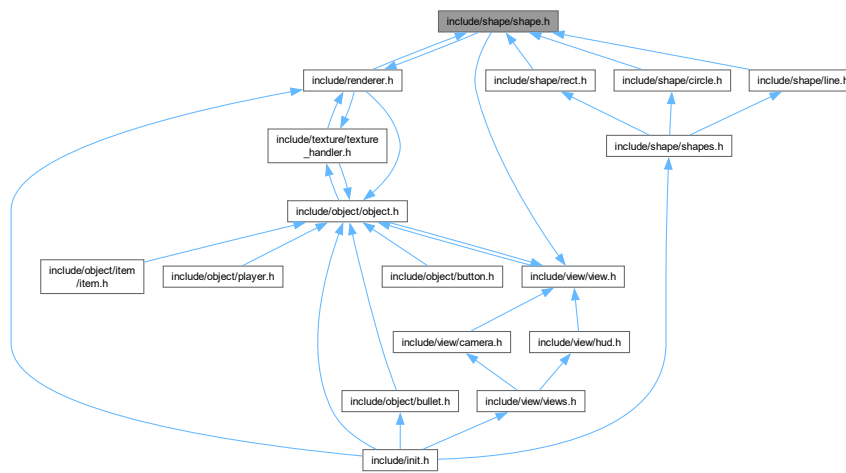
```
#include <render_object_base.h>
#include <view/view.h>
#include <renderer.h>
#include <SDL2/SDL.h>
```

```
#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 shape.h

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

```
00001 #pragma once
00002
00003 #include <render_object_base.h>
00004 #include <view/view.h>
00005 #include <renderer.h>
00006 #include <SDL2/SDL.h>
```


7.36 shapes.h

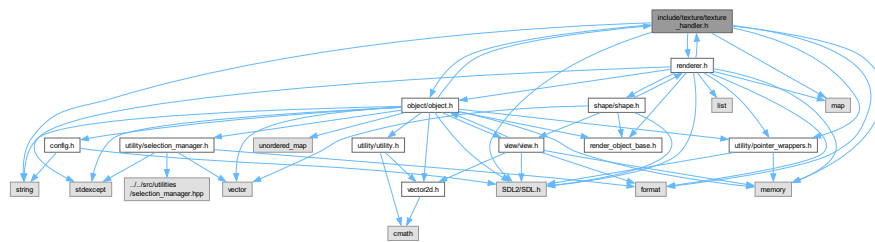
[Go to the documentation of this file.](#)

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

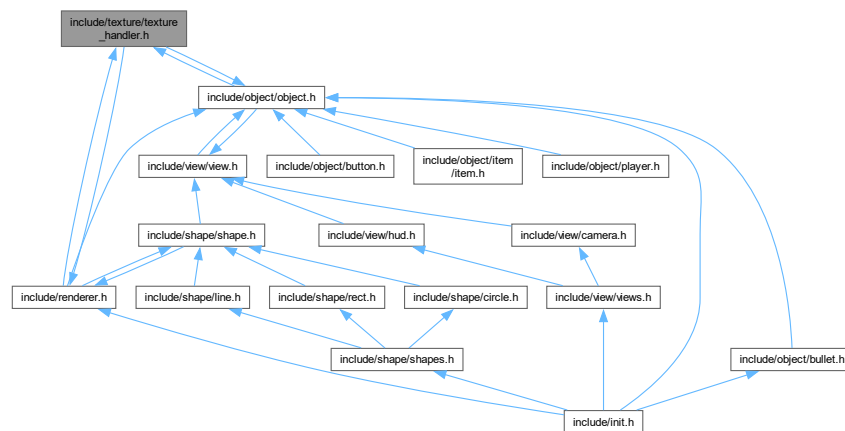
7.37 include/texture/texture_handler.h File Reference

```
#include <renderer.h>
#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](#)

7.38 texture_handler.h

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

```
00001 #pragma once
00002
00003 #include <renderer.h>
00004 #include <utility/pointer_wrappers.h>
00005 #include <object/object.h>
00006 #include <SDL2/SDL.h>
00007 #include <string>
00008 #include <map>
00009 #include <memory>
00010 #include <format>
00011
00012 namespace Objects {
00013     class Object;
00014 }
00015
00016 // TODO: Add support for text textures.
00017
00021 class TextureHandler {
00025     class TextureRequestKey {
00026         friend class Objects::Object;
00027     private:
00028         TextureRequestKey() = default;
00029         TextureRequestKey(const TextureRequestKey&) = default;
00030     };
00031
00032 private:
00033     static const std::string errorTextureName;
00034     std::map<std::string, sdl_unique_ptr<SDL_Texture> textureDB;
00035
00039     TextureHandler();
00040
00041     void loadTexture(const std::string& textureName);
00042
00043 public:
00050     SDL_Texture* getTexture(TextureRequestKey key, const std::string& textureName);
00051
00052 public:
00053     TextureHandler(const TextureHandler&) = delete;
00054     void operator = (const TextureHandler&) = delete;
00055     static TextureHandler& getInstance(void);
00056 };
```

7.39 include/utility/functions.h File Reference

Namespaces

- namespace [Functions](#)

7.40 functions.h

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

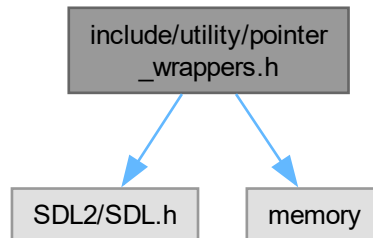
```
00001 #pragma once
00002
00003 namespace Functions {
00004
00005 }
```

7.41 include/utility/pointer_wrappers.h File Reference

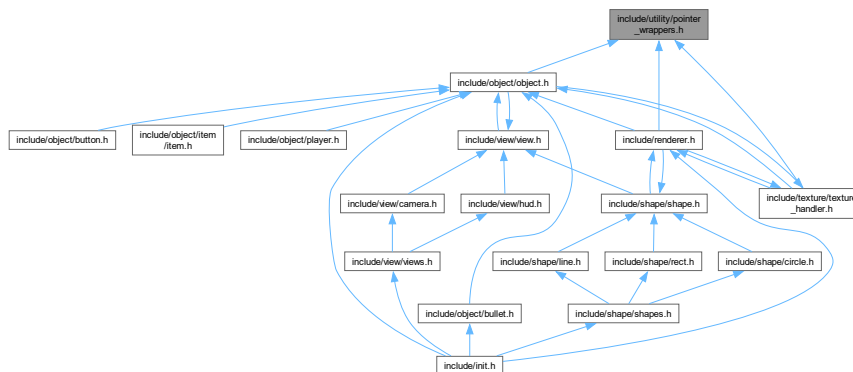
```
#include <SDL2/SDL.h>
```

```
#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)

7.41.1 Typedef Documentation

7.41.1.1 sdl_unique_ptr

```
template<typename Resource >
using sdl_unique_ptr = std::unique_ptr<Resource, sdl_deleter>
```

7.41.2 Function Documentation

7.41.2.1 sdl_make_shared()

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

7.42 pointer_wrappers.h

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

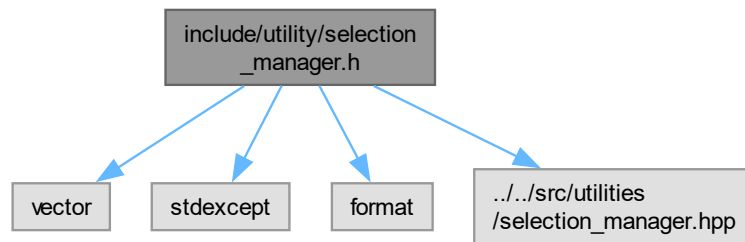
```
00001 #pragma once
00002
00003 #include <SDL2/SDL.h>
00004 #include <memory>
00005
00009 struct sdl_deleter {
00010     inline void operator () (SDL_RWops* thing) const noexcept { if (thing) SDL_FreeRW(thing); }
00011     inline void operator () (SDL_cond* thing) const noexcept { if (thing)
    SDL_DestroyCond(thing); }
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 }
```

7.43 include/utility/selection_manager.h File Reference

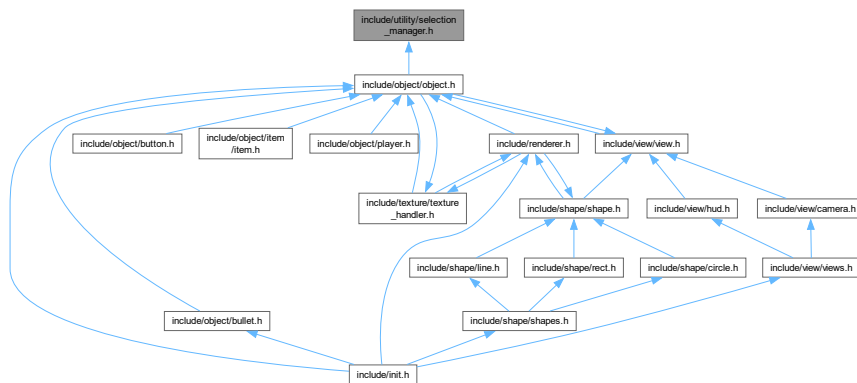
```
#include <vector>
#include <stdexcept>
#include <format>
```



```
#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 >`

7.44 selection_manager.h

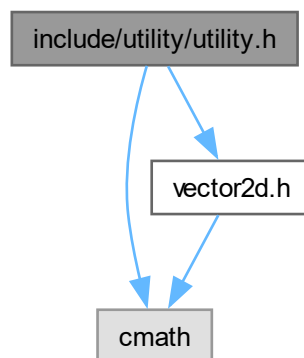
[Go to the documentation of this file.](#)

```
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;
```

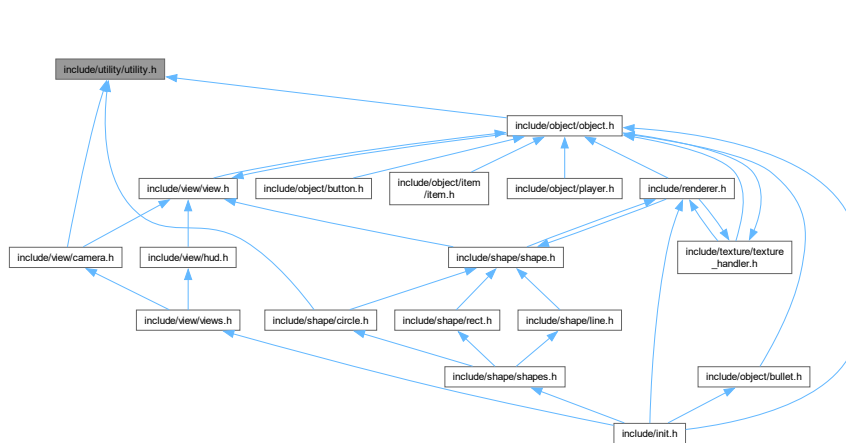
```
00016
00017     SelectionManager();
00018     SelectionManager(const std::vector<T>& selections);
00019
00023     void next(void) const noexcept;
00024
00028     void prev(void) const noexcept;
00029
00035     void set(int newSelection) const;
00036
00041     size_t size(void) const noexcept;
00042
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"
```

7.45 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 $[0, 2\pi)$
- [Vector2D polarToCartesian](#) (float radius, float theta)
Helper function to transform polar coordinates to cartesian coordinates.

7.45.1 Macro Definition Documentation

7.45.1.1 `_USE_MATH_DEFINES`

```
#define __USE_MATH_DEFINES
```

7.45.2 Function Documentation

7.45.2.1 normalizeAngle()

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

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

Parameters

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

Returns

normalized angle

7.45.2.2 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

7.46 utility.h

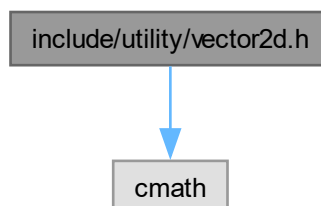
[Go to the documentation of this file.](#)

```
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);
```

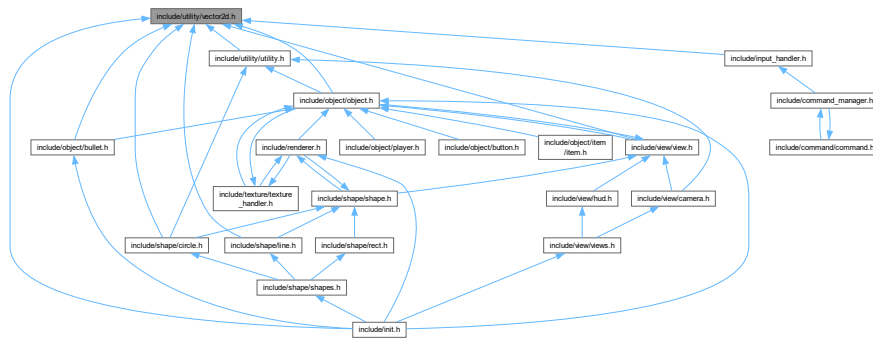
7.47 include/utility/vector2d.h File Reference

```
#include <cmath>
```

Include dependency graph for vector2d.h:



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



Classes

- class [Vector2D](#)

7.48 vector2d.h

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

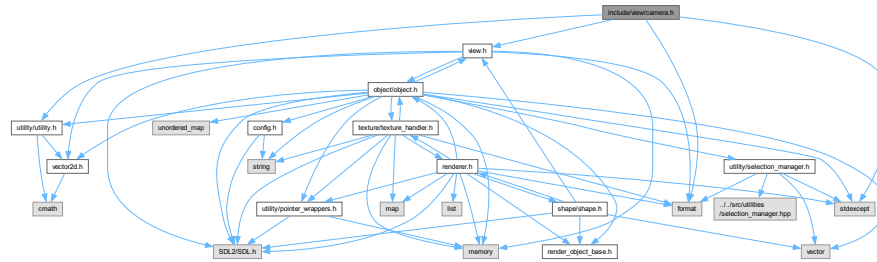
```

00001 #pragma once
00002
00003 #include <cmath>
00004
00005 class Vector2D {
00006 private:
00007     float x;
00008     float y;
00009 public:
00010     // Constructors
00011     Vector2D(void) noexcept;
00012     Vector2D(float _x, float _y) noexcept;
00013
00014     // Member Functions
00015     float getX(void) const noexcept; // x factor
00016     float getY(void) const noexcept; // y factor
00017     Vector2D norm(void) const noexcept; // normalized vector
00018     float len(void) const noexcept; // length of vector
00019     float len2(void) const noexcept; // squared length of vector
00020
00021     Vector2D rotate(float theta) const noexcept; // rotates the vector by @param theta radians
00022
00023     // Static functions
00024     static Vector2D zero(void) noexcept; // returns a zero-vector
00025
00026     // Operators
00027     friend Vector2D operator + (const Vector2D&, const Vector2D&) noexcept;
00028     friend Vector2D operator - (const Vector2D&) noexcept;
00029     friend Vector2D operator - (const Vector2D&, const Vector2D&) noexcept;
00030     friend Vector2D operator * (const Vector2D&, float) noexcept;
00031     friend Vector2D operator / (const Vector2D&, float) noexcept;
00032     friend Vector2D& operator += (Vector2D&, const Vector2D&) noexcept;
00033     friend Vector2D& operator -= (Vector2D&, const Vector2D&) noexcept;
00034     friend Vector2D& operator *= (Vector2D&, float) noexcept;
00035     friend Vector2D& operator /= (Vector2D&, float) noexcept;
00036     static float dot(const Vector2D&, const Vector2D&) noexcept;
00037     static float cross(const Vector2D&, const Vector2D&) noexcept;
00038     static Vector2D rotate(Vector2D, float) noexcept;
00039 };

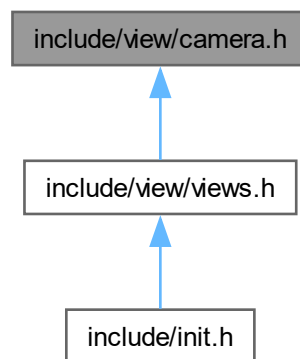
```

7.49 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](#)

7.50 camera.h

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

```

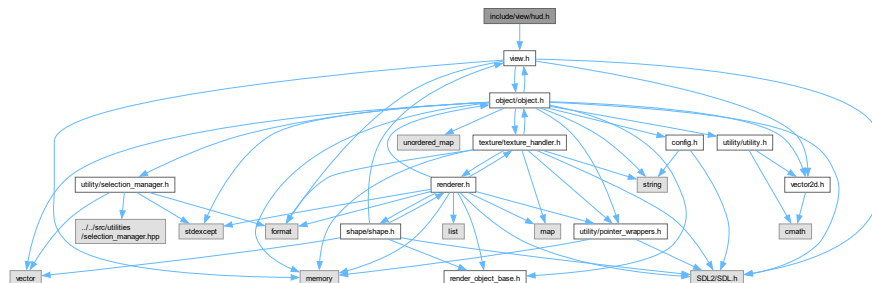
00001 #pragma once
00002
00003 #include <utility/utility.h>
00004 #include "view.h"
00005 #include <stdexcept>
00006 #include <format>
00007
00008 namespace Views {
00009
00013     class Camera : public View {
00014     private:
00015         std::weak_ptr<Objects::Object> pivotObject;
00016
00017         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 }

```

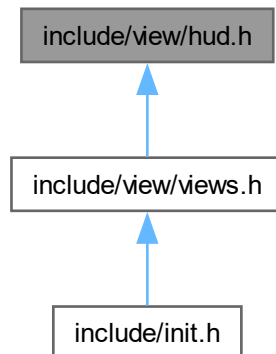
7.51 include/view/hud.h File Reference

```
#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](#)

7.52 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 }
```

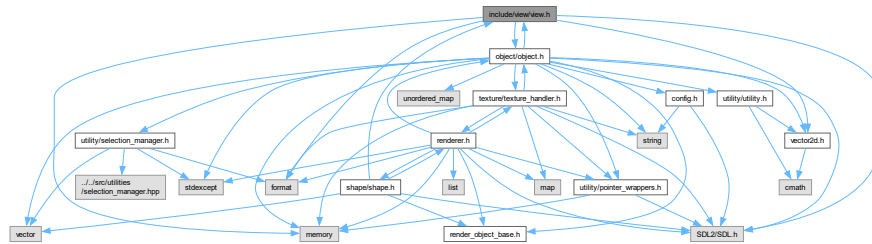
7.53 include/view/view.h File Reference

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

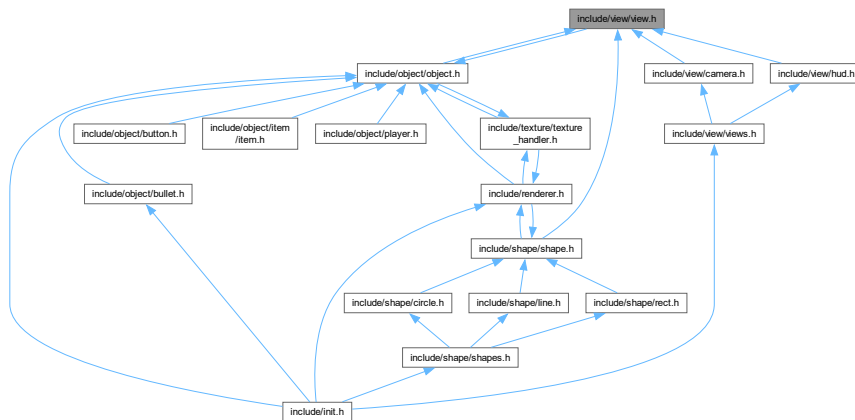


```
#include <format>
```

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

7.54 view.h

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

```

00001 #pragma once
00002
00003 #include <object/object.h>
00004 #include <utility/vector2d.h>
00005 #include <SDL2/SDL.h>
00006 #include <memory>
00007 #include <format>
00008
00009 namespace Objects {
00010     class Object;
00011 }
00012 namespace Views {
00013
00014     class View {
00015     protected:
00016         Vector2D position;
00017         Vector2D dimension;
00018
00019     public:
00020         View(const Vector2D& _position, const Vector2D& _dimension) :
00021             position(_position), dimension(_dimension) {}
00022
00023         virtual ~View() {};
00024
00025         virtual SDL_FRect getRect(const Objects::Object& object) const noexcept = 0;
00026
00027         virtual Vector2D transform(const Vector2D& position) const noexcept = 0;
00028
00029         virtual Vector2D transformFromRender(const Vector2D& renderPosition) const noexcept = 0;
00030
00031         virtual Vector2D getPosition(void) const noexcept { return position; }
00032
00033         virtual Vector2D getDimension(void) const noexcept { return dimension; }
00034
00035         virtual float getAngle(void) const noexcept { return 0.0f; }
00036
00037         virtual float getZoom(void) const noexcept { return 1.0f; }
00038     };
00039
00040     const int INIT_VIEW_WIDTH = 1600;
00041     const int INIT_VIEW_HEIGHT = 900;
00042 }

```

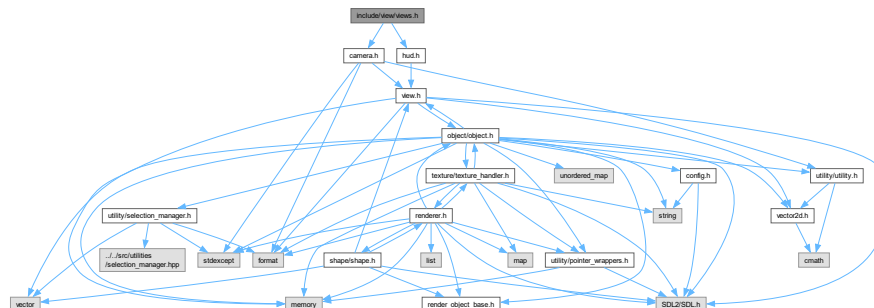
7.55 include/view/views.h File Reference

```

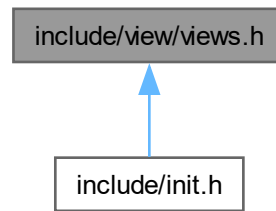
#include "hud.h"
#include "camera.h"

```

Include dependency graph for views.h:



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



7.56 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`, [111](#)
- `~Command`
 - `Commands::Command`, [28](#)
- `~Object`
 - `Objects::Object`, [47](#)
- `~Shape`
 - `Shapes::Shape`, [68](#)
- `~View`
 - `Views::View`, [76](#)
- `add`
 - `SelectionManager< T >`, [65](#)
- `arrowObject1`
 - Global, [11](#)
- `arrowObject2`
 - Global, [11](#)
- `backgroundColor`
 - `Config`, [9](#)
- `beginPoint`
 - `Shapes::Line`, [44](#)
- `blueCircle`
 - Global, [11](#)
- `Bullet`
 - `Objects::Bullet`, [17](#)
- `Button`
 - `Objects::Button`, [19](#)
- `buttons`
 - `KeyBind`, [41](#)
- `Camera`
 - `Views::Camera`, [21](#)
- `center`
 - `Shapes::Circle`, [27](#)
- `Circle`
 - `Shapes::Circle`, [27](#)
- `clear`
 - `Renderer`, [58](#)
- `color`
 - `Shapes::Shape`, [69](#)
- `CommandManager`, [29](#)
 - `Commands::Command::ExecuteKey`, [30](#)
 - `registerCommand`, [29](#)
 - `update`, [29](#)
- `Commands`, [9](#)
- `Commands::Command`, [28](#)
 - `~Command`, [28](#)
 - `execute`, [28](#)
- `Commands::Command::ExecuteKey`, [30](#)
- `CommandManager`, [30](#)
- `Config`, [9](#)
 - `backgroundColor`, [9](#)
 - `framerate`, [9](#)
 - `gameTitle`, [9](#)
 - `holdTimeThreshold`, [10](#)
 - `screenHeight`, [10](#)
 - `screenType`, [10](#)
 - `screenWidth`, [10](#)
 - `volume`, [10](#)
- `createTexture`
 - `Renderer`, [58](#)
- `cross`
 - `Vector2D`, [72](#)
- `crosshairCircle1`
 - Global, [11](#)
- `crosshairLine1`
 - Global, [11](#)
- `crosshairLine2`
 - Global, [12](#)
- `debug`
 - `Objects::Object`, [47](#)
 - `Renderer`, [58](#)
 - `RenderObjectBase`, [62](#)
- `dimension`
 - `Shapes::Rect`, [56](#)
 - `Views::View`, [78](#)
- `dot`
 - `Vector2D`, [72](#)
- `draw`
 - `Shapes::Circle`, [27](#)
 - `Shapes::HollowCircle`, [32](#)
 - `Shapes::Line`, [44](#)
 - `Shapes::Shape`, [69](#)
- `endPoint`
 - `Shapes::Line`, [44](#)
- `execute`
 - `Commands::Command`, [28](#)
- `flipHorizontal`
 - `Objects::Object`, [47](#)
- `flipVertical`
 - `Objects::Object`, [47](#)
- `fpsManager`
 - Global, [12](#)
- `framerate`
 - `Config`, [9](#)
- `Functions`, [10](#)

- GameManager, 30
- gameTitle
 - Config, 9
- get
 - SelectionManager< T >, 65
- getAliveTime
 - Objects::Bullet, 17
- getAngle
 - Objects::Object, 48
 - Views::Camera, 22
 - Views::View, 76
- getColor
 - Shapes::Shape, 69
- getDimension
 - Objects::Object, 48
 - Views::View, 76
- getFlipFlag
 - Objects::Object, 48
- getInstance
 - InputHandler, 37
 - Renderer, 58
 - TextureHandler, 70
- getMousePosition
 - InputHandler, 37
- getPosition
 - Objects::Object, 48
 - Views::View, 76
- getRect
 - Views::Camera, 22
 - Views::HUD, 34
 - Views::View, 77
- getRenderAngle
 - Objects::Object, 48
- getRenderRect
 - Objects::Object, 49
- getSelectionId
 - SelectionManager< T >, 65
- getTexture
 - Objects::Object, 49
 - TextureHandler, 70
- getTextureCount
 - Objects::Object, 49
- getVisibility
 - Objects::Object, 49
- getX
 - Vector2D, 72
- getY
 - Vector2D, 72
- getZoom
 - Views::Camera, 22
 - Views::View, 77
- 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, 41
- holdTime
 - InputHandler, 37
- holdTimeThreshold
 - Config, 10
- HollowCircle
 - Shapes::HollowCircle, 32
- hollowCircle1
 - Global, 12
- HUD
 - Views::HUD, 34
- hudView
 - Global, 12
- ID
 - KeyBind, 41
- include/command/command.h, 79, 80
- include/command_manager.h, 81, 82
- include/config.h, 82, 83
- include/game_manager.h, 84
- include/init.h, 84, 86
- include/input_handler.h, 86, 88
- include/object/bullet.h, 89, 90
- include/object/button.h, 90, 91
- include/object/item/item.h, 91, 92
- include/object/object.h, 92, 93
- include/object/player.h, 95
- include/render_object_base.h, 95, 96
- include/renderer.h, 96, 97
- include/shape/circle.h, 98, 99
- include/shape/line.h, 100, 101
- include/shape/rect.h, 101, 102
- include/shape/shape.h, 102, 103
- include/shape/shapes.h, 104, 105
- include/texture/texture_handler.h, 105, 106
- include/utility/functions.h, 106
- include/utility/pointer_wrappers.h, 107, 108
- include/utility/selection_manager.h, 108, 109
- include/utility/utility.h, 110, 112
- include/utility/vector2d.h, 112, 113

- include/view/camera.h, [114](#), [115](#)
- include/view/hud.h, [115](#), [116](#)
- include/view/view.h, [116](#), [118](#)
- include/view/views.h, [118](#), [119](#)
- init
 - Global, [11](#)
- INIT_VIEW_HEIGHT
 - Views, [14](#)
- INIT_VIEW_WIDTH
 - Views, [14](#)
- input_handler.h
 - LEFT, [87](#)
 - MIDDLE, [87](#)
 - MouseButton, [87](#)
 - RIGHT, [87](#)
 - X1, [87](#)
 - X2, [87](#)
- InputHandler, [36](#)
 - getInstance, [37](#)
 - getMousePosition, [37](#)
 - holdTime, [37](#)
 - InputHandler, [36](#)
 - isButtonDown, [37](#)
 - isButtonUp, [37](#)
 - isKeyDown, [37](#)
 - isKeyUp, [38](#)
 - operator=, [38](#)
 - pollButtonPress, [38](#)
 - pollButtonRelease, [38](#)
 - pollKeyPress, [38](#)
 - pollKeyRelease, [39](#)
 - pollMouseScroll, [39](#)
 - receiveEvent, [39](#)
- isButtonDown
 - InputHandler, [37](#)
- isButtonUp
 - InputHandler, [37](#)
- isKeyDown
 - InputHandler, [37](#)
- isKeyUp
 - InputHandler, [38](#)
- Item
 - Items::Item, [40](#)
- Items, [13](#)
- Items::Item, [40](#)
 - Item, [40](#)
- KeyBind, [40](#)
 - buttons, [41](#)
 - HOLD, [41](#)
 - ID, [41](#)
 - KeyBind, [41](#)
 - KeyBindCount, [42](#)
 - keys, [42](#)
 - operator<, [41](#)
 - RELEASE, [41](#)
 - TAP, [41](#)
 - Trigger, [41](#)
- KeyBindCount
 - KeyBind, [42](#)
- keys
 - KeyBind, [42](#)
- LEFT
 - input_handler.h, [87](#)
- len
 - Vector2D, [72](#)
- len2
 - Vector2D, [72](#)
- Line
 - Shapes::Line, [44](#)
- line1
 - Global, [12](#)
- line2
 - Global, [12](#)
- line3
 - Global, [12](#)
- line4
 - Global, [12](#)
- lookAt
 - Objects::Object, [49](#)
- menuView
 - Global, [12](#)
- MIDDLE
 - input_handler.h, [87](#)
- MouseButton
 - input_handler.h, [87](#)
- move
 - Objects::Object, [50](#)
- moveLayerBottom
 - Renderer, [58](#)
- moveLayerDown
 - Renderer, [59](#)
- moveLayerTop
 - Renderer, [59](#)
- moveLayerUp
 - Renderer, [59](#)
- next
 - SelectionManager< T >, [65](#)
- nextTexture
 - Objects::Object, [50](#)
- norm
 - Vector2D, [72](#)
- normalizeAngle
 - utility.h, [111](#)
- Object
 - Objects::Object, [47](#)
- Objects, [13](#)
- Objects::Bullet, [15](#)
 - Bullet, [17](#)
 - getAliveTime, [17](#)
 - update, [17](#)
- Objects::Button, [18](#)
 - Button, [19](#)
 - onClick, [19](#)

- setHovered, [19](#)
 - update, [19](#)
- Objects::Object, [45](#)
 - ~Object, [47](#)
 - debug, [47](#)
 - flipHorizontal, [47](#)
 - flipVertical, [47](#)
 - getAngle, [48](#)
 - getDimension, [48](#)
 - getFlipFlag, [48](#)
 - getPosition, [48](#)
 - getRenderAngle, [48](#)
 - getRenderRect, [49](#)
 - getTexture, [49](#)
 - getTextureCount, [49](#)
 - getVisibility, [49](#)
 - lookAt, [49](#)
 - move, [50](#)
 - nextTexture, [50](#)
 - Object, [47](#)
 - previousTexture, [50](#)
 - rotate, [50](#)
 - setAngle, [50](#)
 - setTexture, [51](#)
 - setVisibility, [51](#)
 - stretch, [51](#)
 - stretchX, [51](#)
 - stretchY, [52](#)
 - TextureHandler, [52](#)
 - update, [52](#)
- Objects::Player, [53](#)
- onClick
 - Objects::Button, [19](#)
- operator<
 - KeyBind, [41](#)
- operator()
 - sdl_deleter, [62–64](#)
- operator+
 - Vector2D, [73](#)
- operator+=
 - Vector2D, [73](#)
- operator-
 - Vector2D, [73, 74](#)
- operator-=
 - Vector2D, [74](#)
- operator/
 - Vector2D, [74](#)
- operator/=
 - Vector2D, [74](#)
- operator=
 - InputHandler, [38](#)
 - Renderer, [59](#)
 - TextureHandler, [71](#)
- operator*
 - Vector2D, [73](#)
- operator*=
 - Vector2D, [73](#)
- playerCamera
 - Global, [13](#)
- playerObject
 - Global, [13](#)
- pointer_wrappers.h
 - sdl_make_shared, [108](#)
 - sdl_unique_ptr, [108](#)
- polarToCartesian
 - utility.h, [112](#)
- pollButtonPress
 - InputHandler, [38](#)
- pollButtonRelease
 - InputHandler, [38](#)
- pollKeyPress
 - InputHandler, [38](#)
- pollKeyRelease
 - InputHandler, [39](#)
- pollMouseScroll
 - InputHandler, [39](#)
- position
 - Shapes::Rect, [56](#)
 - Views::View, [78](#)
- prev
 - SelectionManager< T >, [66](#)
- previousTexture
 - Objects::Object, [50](#)
- purpleCircle
 - Global, [13](#)
- radius
 - Shapes::Circle, [27](#)
- receiveEvent
 - InputHandler, [39](#)
- redCircle
 - Global, [13](#)
- registerCommand
 - CommandManager, [29](#)
- registerObject
 - Renderer, [59](#)
- RELEASE
 - KeyBind, [41](#)
- remove
 - SelectionManager< T >, [66](#)
- removeObject
 - Renderer, [60](#)
- render
 - Renderer, [60](#)
- Renderer, [56](#)
 - clear, [58](#)
 - createTexture, [58](#)
 - debug, [58](#)
 - getInstance, [58](#)
 - moveLayerBottom, [58](#)
 - moveLayerDown, [59](#)
 - moveLayerTop, [59](#)
 - moveLayerUp, [59](#)
 - operator=, [59](#)
 - registerObject, [59](#)
 - removeObject, [60](#)
 - render, [60](#)

- Renderer, [57](#)
- Renderer::RenderKey, [60](#)
 - RenderKey, [61](#)
- RenderKey
 - Renderer::RenderKey, [61](#)
- RenderObjectBase, [61](#)
 - debug, [62](#)
- RIGHT
 - input_handler.h, [87](#)
- rotate
 - Objects::Object, [50](#)
 - Vector2D, [73](#)
 - Views::Camera, [22](#)
- screenHeight
 - Config, [10](#)
- screenType
 - Config, [10](#)
- screenWidth
 - Config, [10](#)
- sdl_deleter, [62](#)
 - operator(), [62–64](#)
- sdl_make_shared
 - pointer_wrappers.h, [108](#)
- sdl_unique_ptr
 - pointer_wrappers.h, [108](#)
- SELECTION_NOT_SET
 - SelectionManager< T >, [67](#)
- SelectionManager
 - SelectionManager< T >, [65](#)
- SelectionManager< T >, [64](#)
 - add, [65](#)
 - get, [65](#)
 - getSelectionId, [65](#)
 - next, [65](#)
 - prev, [66](#)
 - remove, [66](#)
 - SELECTION_NOT_SET, [67](#)
 - SelectionManager, [65](#)
 - set, [66](#)
 - size, [66](#)
- set
 - SelectionManager< T >, [66](#)
- setAngle
 - Objects::Object, [50](#)
 - Views::Camera, [23](#)
- setBeginPoint
 - Shapes::Line, [44](#)
- setCenter
 - Shapes::Circle, [27](#)
- setColor
 - Shapes::Shape, [69](#)
- setDimension
 - Views::Camera, [23](#)
- setEndPoint
 - Shapes::Line, [44](#)
- setHovered
 - Objects::Button, [19](#)
- setPivotObject
 - Views::Camera, [23](#)
- setPosition
 - Views::Camera, [23](#)
- setRadius
 - Shapes::Circle, [27](#)
- setTexture
 - Objects::Object, [51](#)
- setThickness
 - Shapes::HollowCircle, [32](#)
 - Shapes::Line, [44](#)
- setVisibility
 - Objects::Object, [51](#)
- setZoom
 - Views::Camera, [24](#)
- Shape
 - Shapes::Shape, [68](#)
- Shapes, [14](#)
- Shapes::Circle, [25](#)
 - center, [27](#)
 - Circle, [27](#)
 - draw, [27](#)
 - radius, [27](#)
 - setCenter, [27](#)
 - setRadius, [27](#)
- Shapes::HollowCircle, [30](#)
 - draw, [32](#)
 - HollowCircle, [32](#)
 - setThickness, [32](#)
 - thickness, [33](#)
- Shapes::Line, [42](#)
 - beginPoint, [44](#)
 - draw, [44](#)
 - endPoint, [44](#)
 - Line, [44](#)
 - setBeginPoint, [44](#)
 - setEndPoint, [44](#)
 - setThickness, [44](#)
 - thickness, [45](#)
- Shapes::Rect, [55](#)
 - dimension, [56](#)
 - position, [56](#)
- Shapes::Shape, [67](#)
 - ~Shape, [68](#)
 - color, [69](#)
 - draw, [69](#)
 - getColor, [69](#)
 - setColor, [69](#)
 - Shape, [68](#)
 - view, [69](#)
- size
 - SelectionManager< T >, [66](#)
- stretch
 - Objects::Object, [51](#)
- stretchX
 - Objects::Object, [51](#)
- stretchY
 - Objects::Object, [52](#)
- TAP

- KeyBind, 41
- TextureHandler, 69
 - getInstance, 70
 - getTexture, 70
 - Objects::Object, 52
 - operator=, 71
 - TextureHandler, 70
- thickness
 - Shapes::HollowCircle, 33
 - Shapes::Line, 45
- transform
 - Views::Camera, 24
 - Views::HUD, 35
 - Views::View, 77
- transformFromRender
 - Views::Camera, 24
 - Views::HUD, 35
 - Views::View, 78
- Trigger
 - KeyBind, 41
- update
 - CommandManager, 29
 - Objects::Bullet, 17
 - Objects::Button, 19
 - Objects::Object, 52
- utility.h
 - _USE_MATH_DEFINES, 111
 - normalizeAngle, 111
 - polarToCartesian, 112
- Vector2D, 71
 - cross, 72
 - dot, 72
 - getX, 72
 - getY, 72
 - len, 72
 - len2, 72
 - norm, 72
 - operator+, 73
 - operator+=, 73
 - operator-, 73, 74
 - operator-=, 74
 - operator/, 74
 - operator/=: 74
 - operator*, 73
 - operator*=: 73
 - rotate, 73
 - Vector2D, 72
 - zero, 73
- View
 - Views::View, 76
- view
 - Shapes::Shape, 69
- Views, 14
 - INIT_VIEW_HEIGHT, 14
 - INIT_VIEW_WIDTH, 14
- Views::Camera, 20
 - Camera, 21
 - getAngle, 22
 - getRect, 22
 - getZoom, 22
 - rotate, 22
 - setAngle, 23
 - setDimension, 23
 - setPivotObject, 23
 - setPosition, 23
 - setZoom, 24
 - transform, 24
 - transformFromRender, 24
- Views::HUD, 33
 - getRect, 34
 - HUD, 34
 - transform, 35
 - transformFromRender, 35
- Views::View, 74
 - ~View, 76
 - dimension, 78
 - getAngle, 76
 - getDimension, 76
 - getPosition, 76
 - getRect, 77
 - getZoom, 77
 - position, 78
 - transform, 77
 - transformFromRender, 78
 - View, 76
- volume
 - Config, 10
- X1
 - input_handler.h, 87
- X2
 - input_handler.h, 87
- yellowCircle
 - Global, 13
- zero
 - Vector2D, 73