Lab Raid

Generated by Doxygen 1.10.0

Chapter 1

Namespace Index

1.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

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

Chapter 3

Class Index

3.1 Class List

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

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

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 = 1920∗7/10
- const int screenHeight = 1080∗7/10
- 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 = 1080*7/10
```

5.2.1.6 screenType

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

5.2.1.7 screenWidth

```
const int Config::screenWidth = 1920*7/10
```

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< Objects::Object > hudArrow
- std::shared_ptr< Shapes::Circle > hudCircle
- std::shared_ptr< Shapes::Line > crosshairLine1
- std::shared_ptr< Shapes::Line > crosshairLine2
- std::shared_ptr< Shapes::HollowCircle > crosshairCircle1
- std::shared_ptr< Objects::Object > cameraObject

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  [extern]
```

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 cameraObject

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

5.4.2.5 crosshairCircle1

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

5.4.2.6 crosshairLine1

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

5.4.2.7 crosshairLine2

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

5.4.2.8 fpsManager

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

5.4.2.9 greenCircle

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

5.4.2.10 hollowCircle1

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

5.4.2.11 hudArrow

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

5.4.2.12 hudCircle

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

5.4.2.13 hudView

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

5.4.2.14 line1

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

5.4.2.15 line2

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

5.4.2.16 line3

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

5.4.2.17 line4

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

5.4.2.18 menuView

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

5.4.2.19 playerCamera

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

5.4.2.20 playerObject

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

5.4.2.21 purpleCircle

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

5.4.2.22 redCircle

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

5.4.2.23 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
```

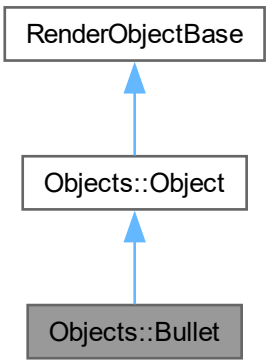
Chapter 6

Class Documentation

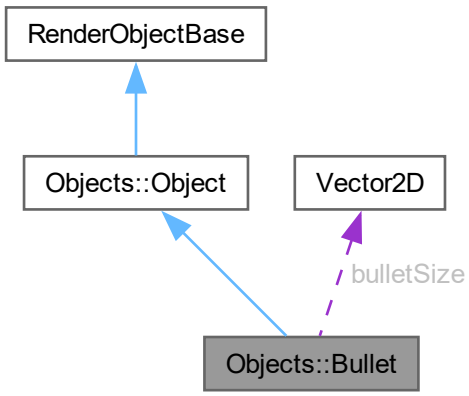
6.1 Objects::Bullet Class Reference

`#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=20.0f)
- 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, 2pi), 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.
- void setPosition (const Vector2D &newPosition) noexcept

  Sets the position of the object.
- Vector2D getDimension (void) const noexcept

Gets the dimension of the object.

- void setDimension (const Vector2D &newDimension)

  Sets the dimension of the object. Throws 'std::invalid_argument' if one of the dimensions is negative.
- 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.
- bool collideWith (const Object &other) const noexcept

  Check if this object collides with 'other' object.
- 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

Static Public Attributes

- static const Vector2D bulletSize

6.1.1 Constructor & Destructor Documentation

6.1.1.1 Bullet()

```
Objects::Bullet::Bullet (
            const Views::View * view,
            Vector2D position,
            float angle,
            float speed = 20.0f )  [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.

6.1.3 Member Data Documentation

6.1.3.1 bulletSize

```
const Vector2D Objects::Bullet::bulletSize  [static]
```

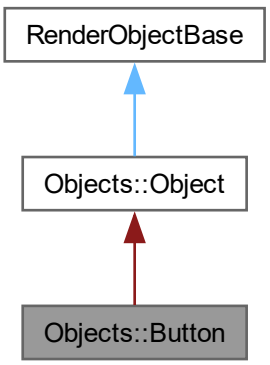
The documentation for this class was generated from the following file:

- include/object/bullet.h

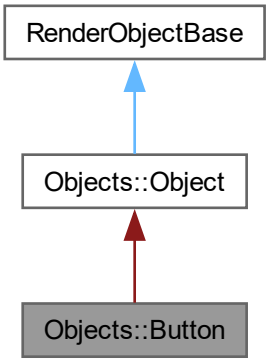
6.2 Objects::Button Class Reference

```
#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:

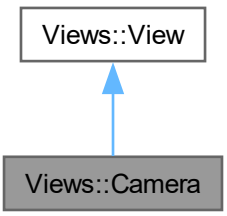
- include/object/button.h

6.3 Views::Camera Class Reference

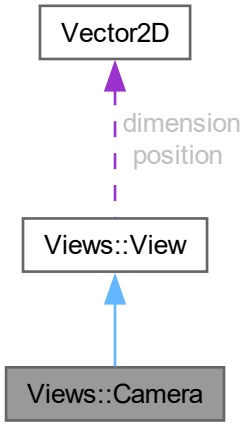
Camera for following object or stationary view.

```
#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

object.	
object	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

diffAngle	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

angle	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

newDimension	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

pivotObject	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

newPosition	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

zoom	should be positive. Throws std::invalid_argument if
zoom	is invalid.
zoom	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

position.	
position	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

renderPosition.	
renderPosition	The render position to be transformed

Returns

The virtual position after transformation.

Implements Views::View.

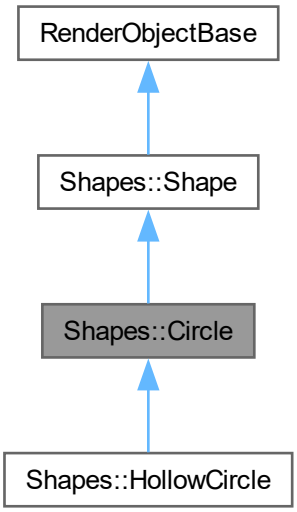
The documentation for this class was generated from the following file:

- include/view/camera.h

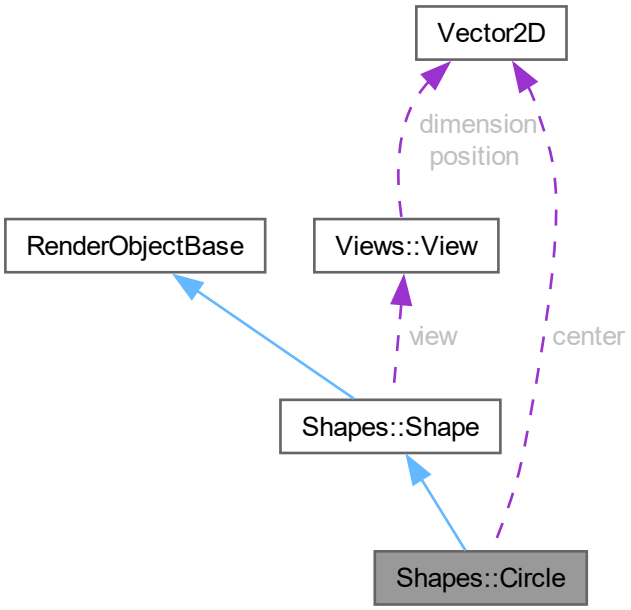
6.4 Shapes::Circle Class Reference

```
#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

keyBind	The key bind of this command.
command	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>
```

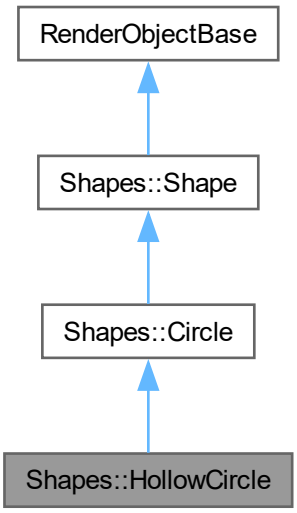
The documentation for this class was generated from the following file:

- include/game_manager.h

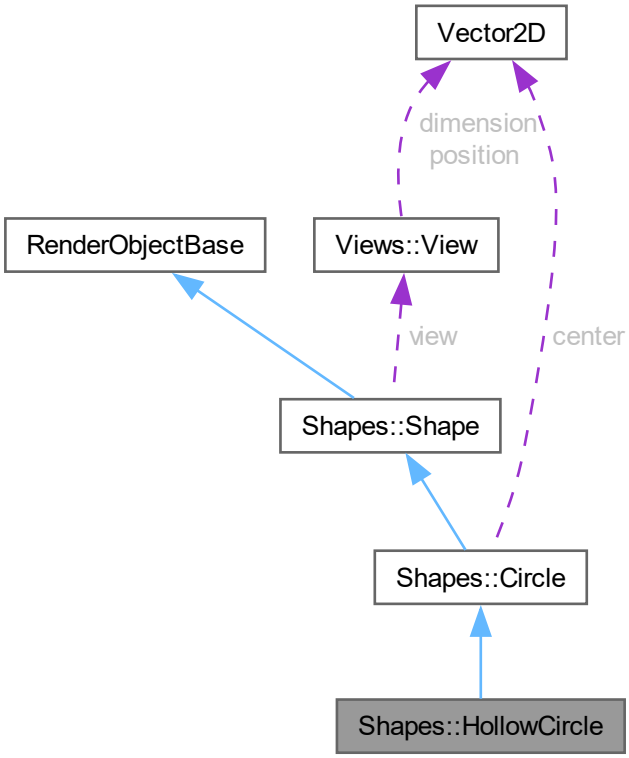
6.9 Shapes::HollowCircle Class Reference

`#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]
```

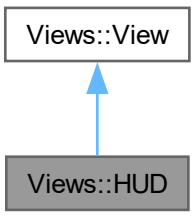
The documentation for this class was generated from the following file:

- include/shape/circle.h

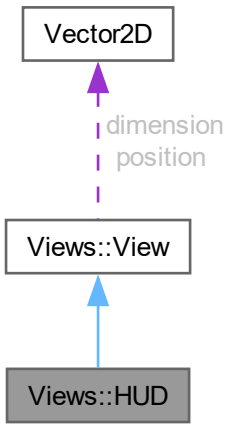
6.10 Views::HUD Class Reference

```
#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

object.	
object	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

position.	
position	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

renderPosition.	
renderPosition	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

key	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

key	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

key	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

key	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 > &inventory↩ Object, 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
```

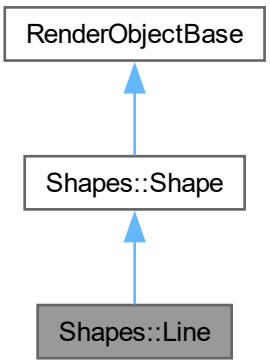
The documentation for this struct was generated from the following file:

- include/command_manager.h

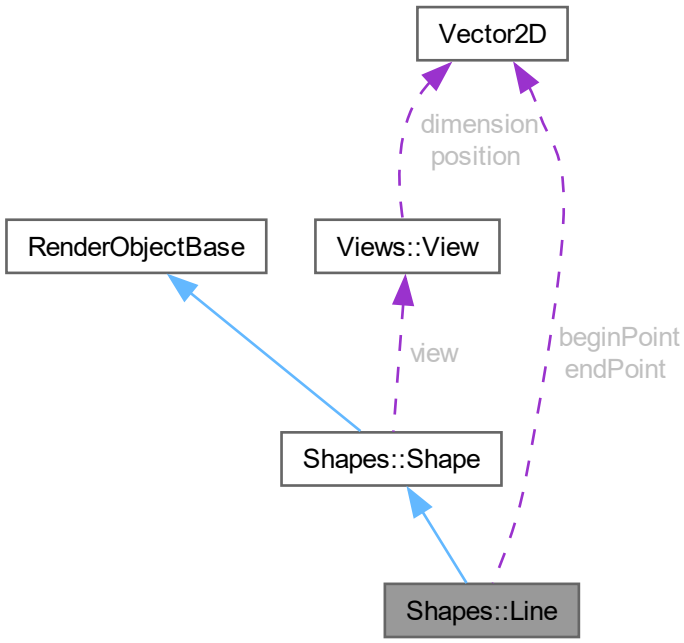
6.14 Shapes::Line Class Reference

```
#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:

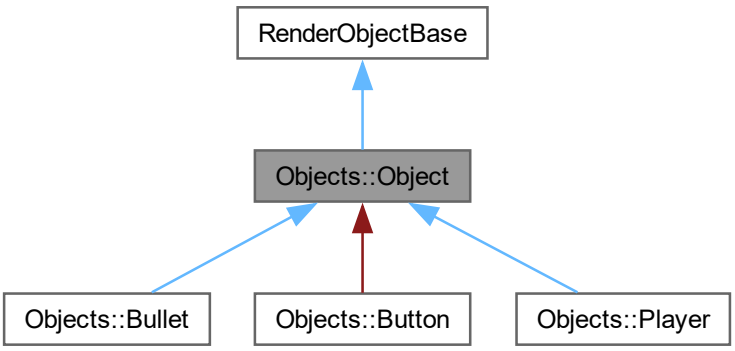
- include/shape/line.h

6.15 Objects::Object Class Reference

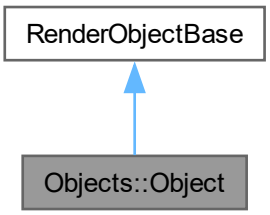
Object type for all renderable objects in the world note: the texture won't be created until loaded into the renderer.

```
#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, 2pi), 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.
- void setPosition (const Vector2D &newPosition) noexcept

  Sets the position of the object.
- Vector2D getDimension (void) const noexcept

  Gets the dimension of the object.
- void setDimension (const Vector2D &newDimension)

  Sets the dimension of the object. Throws 'std::invalid_argument' if one of the dimensions is negative.
- 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.
- bool collideWith (const Object &other) const noexcept

  Check if this object collides with 'other' object.
- 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

textureNames	The list of texture names.
_view	The viewport of the object.
_position	Initial position. (x, y)
_dimension	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 collideWith()

```
bool Objects::Object::collideWith (
            const Object & other ) const  [noexcept]
```

Check if this object collides with 'other' object.

Parameters

other	The other object.

Returns

If collided.

6.15.3.2 debug()

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

Reimplemented from RenderObjectBase.

6.15.3.3 flipHorizontal()

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

Flips the object horizontally.

6.15.3.4 flipVertical()

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

Flips the object vertically.

6.15.3.5 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.

6.15.3.6 getDimension()

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

Gets the dimension of the object.

Returns

The object's dimension.

6.15.3.7 getFlipFlag()

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

Returns the flip flag used by SDL.

Returns

A SDL_RendererFlip flag.

6.15.3.8 getPosition()

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

Gets the position of the object.

Returns

The object's location.

6.15.3.9 getRenderAngle()

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

Gets the render angle of the object.

Returns

The render angle of the object

6.15.3.10 getRenderRect()

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

Gets render rectangle for rendering.

Returns

The SDL_FRect for rendering.

6.15.3.11 getTexture()

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

Gets current texture.

Returns

The current texture the object is using.

6.15.3.12 getTextureCount()

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

Gets the number of textures this object has.

Returns

Numbeer of textures.

6.15.3.13 getVisibility()

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

Gets the object's visibility.

Returns

The object's visibility.

6.15.3.14 lookAt()

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

Make the object face.

Parameters

position	coordinates.
position	The coordinate of where the object should look at.

6.15.3.15 move()

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

Moves the object by the translate vector.

Parameters

translate	The offset vector to move by.

6.15.3.16 nextTexture()

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

Set to next texture, texture ID wraps around.

6.15.3.17 previousTexture()

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

Set to previous texture, texture ID wraps around.

6.15.3.18 rotate()

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

Rotates the object by.

Parameters

diffAngle	radians in the counter-clockwise direction.
diffAngle	Rotation angle.

6.15.3.19 setAngle()

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

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

Sets rotation angle to.

Parameters

newAngle	radians.
newAngle	The new angle to set to. (in radians)

6.15.3.20 setDimension()

```
void Objects::Object::setDimension (
            const Vector2D & newDimension )
```

Sets the dimension of the object. Throws 'std::invalid_argument' if one of the dimensions is negative.

6.15.3.21 setPosition()

```
void Objects::Object::setPosition (
            const Vector2D & newPosition )  [noexcept]
```

Sets the position of the object.

6.15.3.22 setTexture()

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

Sets texture to.

Parameters

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

6.15.3.23 setVisibility()

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

Sets the object's visibility.

Parameters

visibility	The object's visibility.

6.15.3.24 stretch()

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

Stretches both the object's width and height by.

Parameters

ratio.	
ratio	Stretch ratio.

6.15.3.25 stretchX()

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

Stretches the object's width by.

Parameters

ratio.	
ratio	Stretch ratio.

6.15.3.26 stretchY()

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

Stretches the object's height by.

Parameters

ratio.	
ratio	Stretch ratio.

6.15.3.27 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]
```

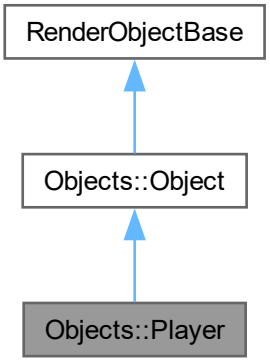
The documentation for this class was generated from the following file:

- include/object/object.h

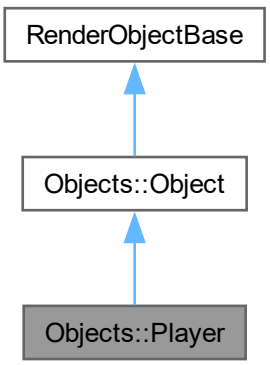
6.16 Objects::Player Class Reference

```
#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, 2pi), 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.
- void setPosition (const Vector2D &newPosition) noexcept

  Sets the position of the object.
- Vector2D getDimension (void) const noexcept

  Gets the dimension of the object.
- void setDimension (const Vector2D &newDimension)

  Sets the dimension of the object. Throws 'std::invalid_argument' if one of the dimensions is negative.
- 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.
- bool collideWith (const Object &other) const noexcept

  Check if this object collides with 'other' object.
- 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

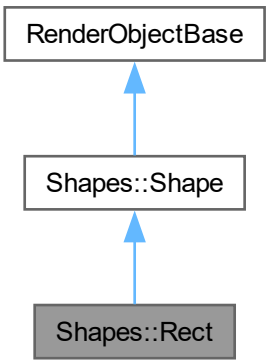
The documentation for this class was generated from the following file:

- include/object/player.h

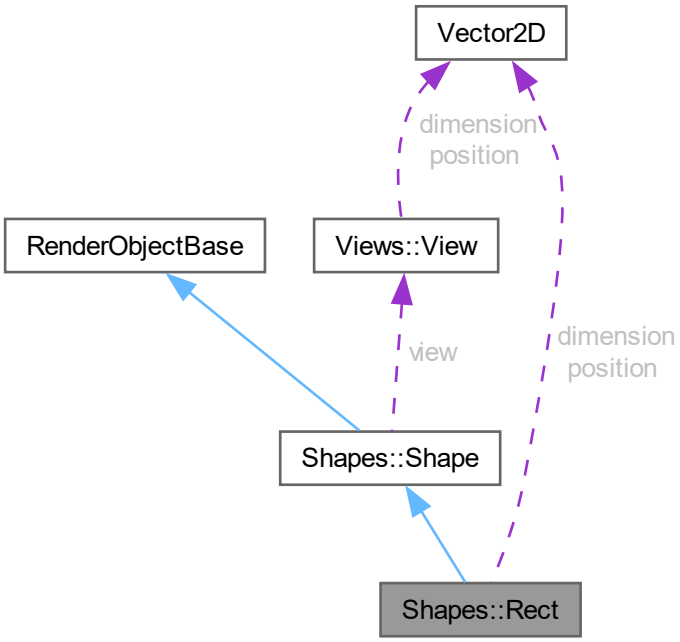
6.17 Shapes::Rect Class Reference

`#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_Window * getWindow (void) noexcept

  Gets game window.
- SDL_Texture * createTexture (CreateTextureKey key, SDL_Surface *surface) const

  Creates a texture from a SDL_Surface.
- Vector2D getWindowSize (void) const noexcept

  Get underlying SDL_Renderer renderer.
- bool registerObject (std::shared_ptr< RenderObjectBase > objectPtr) noexcept

  Registers the object for rendering.
- 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

key	Required key to use this function.
surface	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 getWindow()

```
SDL_Window * Renderer::getWindow (
            void  )  [noexcept]
```

Gets game window.

Returns

The game window.

6.18.3.6 getWindowSize()

```
Vector2D Renderer::getWindowSize (
            void  ) const  [noexcept]
```

Get underlying SDL_Renderer renderer.

Returns

The underlying renderer.

Gets current window size.

Returns

Current window size.

6.18.3.7 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

objectPtr	The object to be moved.

6.18.3.8 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

objectPtr	The object to be moved.

6.18.3.9 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

objectPtr	The object to be moved.

6.18.3.10 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

objectPtr	The object to be moved.

6.18.3.11 operator=()

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

6.18.3.12 registerObject()

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

Registers the object for rendering.

Parameters

objectPtr	std::shared_ptr of the object

Returns

Whether the object was successfully registered

6.18.3.13 removeObject()

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

Unregisters the object for rendering.

Parameters

objectPtr	std::shared_ptr of the object

Returns

Whether the object was successfully unregistered.

6.18.3.14 render()

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

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

Parameters

key	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]

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

6.19.1.2 RenderKey() [2/2]

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

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

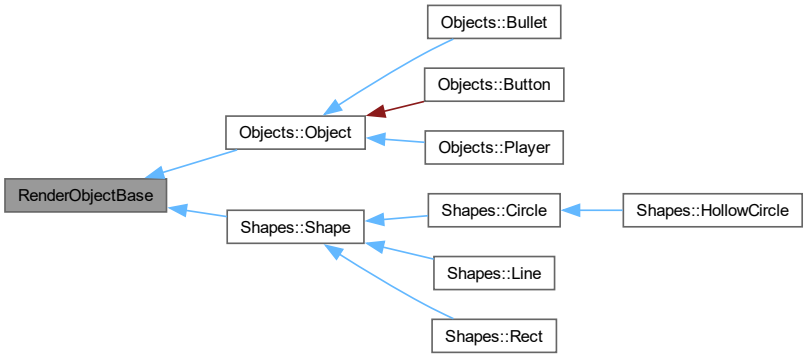
- include/renderer.h

6.20 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

newSelection	to the manager.
newSelection	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

selectionId.	Throws std::out_of_range if selectionId is inavlid.
selectionId	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

newSelection.	Throws std::out_of_range if ID is not in range of [0, size) or SELECTION_NOT_SET.
newSelection	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]
```

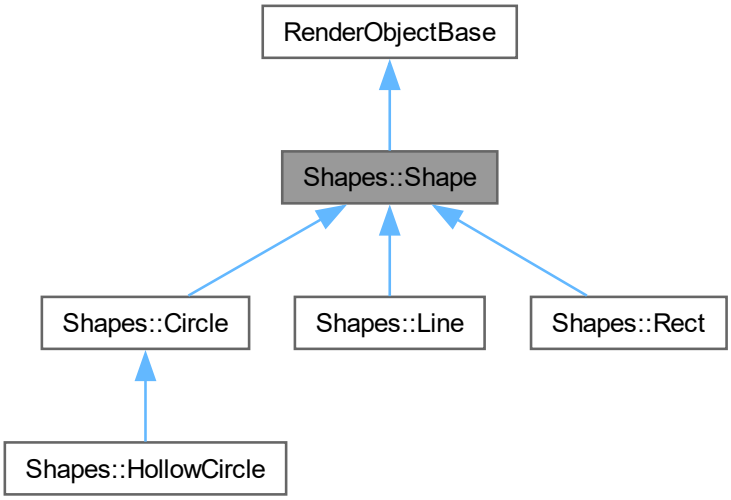
The documentation for this class was generated from the following file:

- include/utility/selection_manager.h

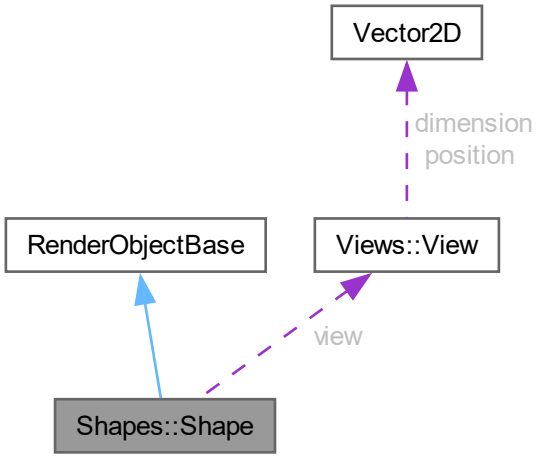
6.23 Shapes::Shape Class Reference

```
#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

key	Access Control Key
textureName	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∗ (float, const Vector2D &) 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* [1/2]

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

6.25.3.2 operator* [2/2]

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

6.25.3.3 operator*=

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

6.25.3.4 operator+

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

6.25.3.5 operator+=

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

6.25.3.6 operator- [1/2]

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

6.25.3.7 operator- [2/2]

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

6.25.3.8 operator-=

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

6.25.3.9 operator/

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

6.25.3.10 operator/=

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

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

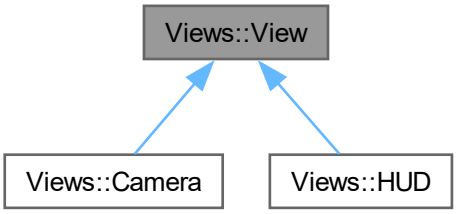
- include/utility/vector2d.h

6.26 Views::View Class Reference

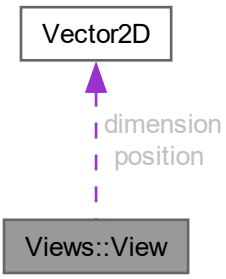
View: defines a view area, translates the objects' virtual rects to real rendering rects.

`#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

object.	
object	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

position.	
position	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

renderPosition.	
renderPosition	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

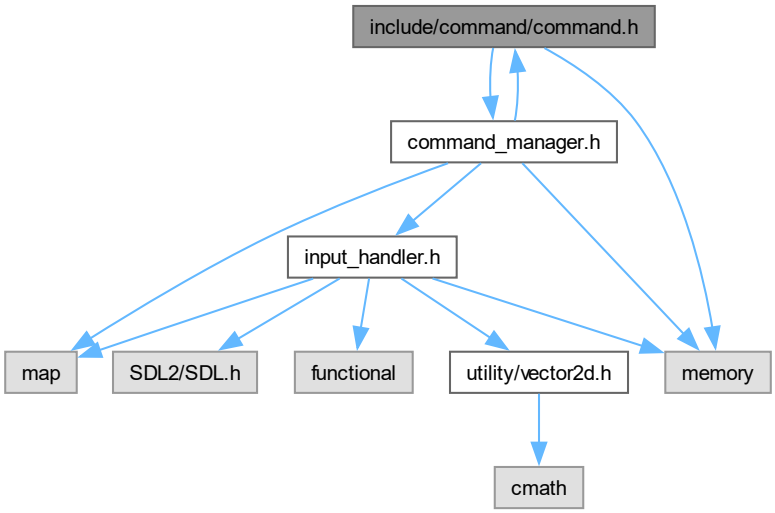
Chapter 7

File Documentation

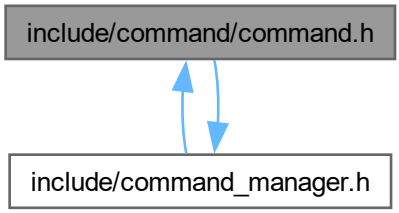
7.1 include/command/command.h File Reference

```
#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

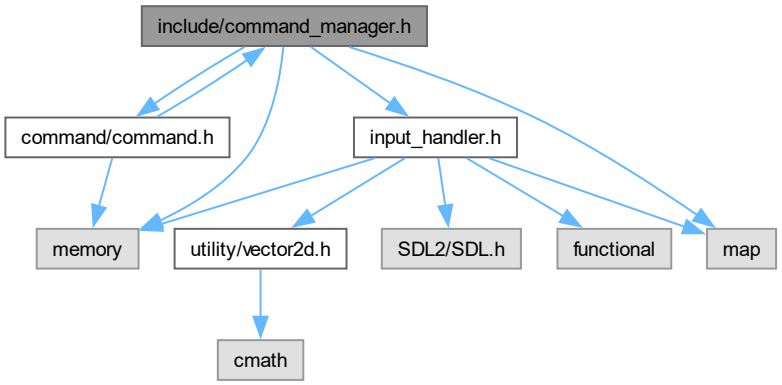
Go to the documentation of this file.

```
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 }
```

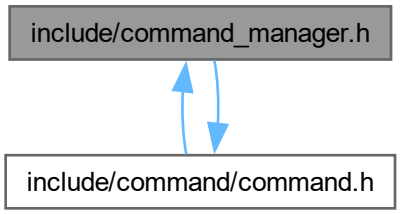
7.3 include/command_manager.h File Reference

```
#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

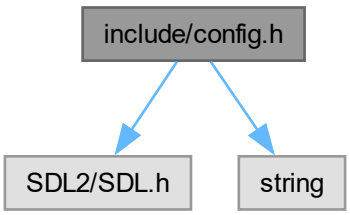
Go to the documentation of this file.

```
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 };
```

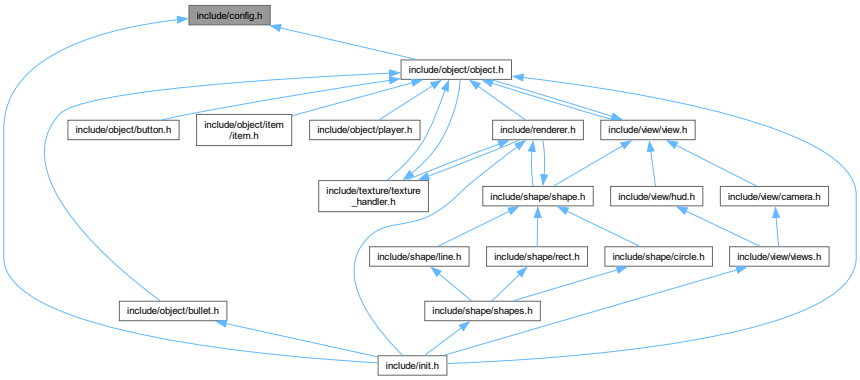
7.5 include/config.h File Reference

```
#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 = 1920∗7/10
- const int Config::screenHeight = 1080∗7/10
- 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

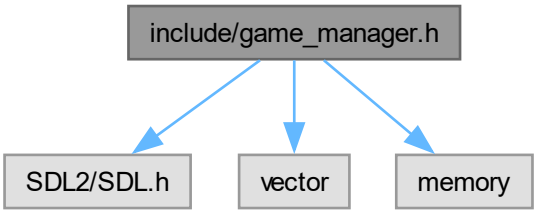
Go to the documentation of this file.

```
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 = 1920*7/10;
00009     const int screenHeight = 1080*7/10;
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.

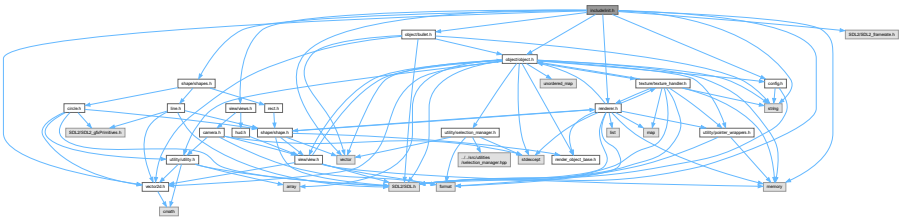
```
00001 #pragma once
00002
00003 #include <SDL2/SDL.h>
00004 #include <vector>
00005 #include <memory>
00006
00007 class GameManager {
00008 private:
00009     bool paused;
00010     enum {
00011         GAME_TITLE = 1,
00012         GAME_LEVEL = 2,
00013         GAME_END = 3
00014     } state;
00015
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< Objects::Object > Global::hudArrow
- std::shared_ptr< Shapes::Circle > Global::hudCircle
- std::shared_ptr< Shapes::Line > Global::crosshairLine1
- std::shared_ptr< Shapes::Line > Global::crosshairLine2
- std::shared_ptr< Shapes::HollowCircle > Global::crosshairCircle1
- std::shared_ptr< Objects::Object > Global::cameraObject

7.10 init.h

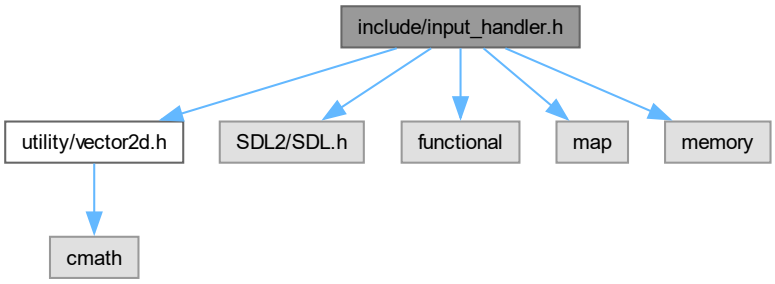
Go to the documentation of this file.

```
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;
00023     extern std::shared_ptr<Objects::Object> arrowObject1;
00024     extern std::shared_ptr<Objects::Object> arrowObject2;
00025     extern std::shared_ptr<Shapes::Circle> yellowCircle;
00026     extern std::shared_ptr<Shapes::Circle> greenCircle;
00027     extern std::shared_ptr<Shapes::Circle> blueCircle;
00028     extern std::shared_ptr<Shapes::Circle> redCircle;
00029     extern std::shared_ptr<Shapes::Circle> purpleCircle;
00030
00031     extern std::shared_ptr<Shapes::HollowCircle> hollowCircle1;
00032     extern std::shared_ptr<Shapes::Line> line1;
00033     extern std::shared_ptr<Shapes::Line> line2;
00034     extern std::shared_ptr<Shapes::Line> line3;
00035     extern std::shared_ptr<Shapes::Line> line4;
00036
00037     extern std::shared_ptr<Objects::Object> hudArrow;
00038     extern std::shared_ptr<Shapes::Circle> hudCircle;
00039
00040     extern std::shared_ptr<Shapes::Line> crosshairLine1;
00041     extern std::shared_ptr<Shapes::Line> crosshairLine2;
00042     extern std::shared_ptr<Shapes::HollowCircle> crosshairCircle1;
00043
00044     extern std::shared_ptr<Objects::Object> cameraObject;
00045
00046     void init();
00047 }
```

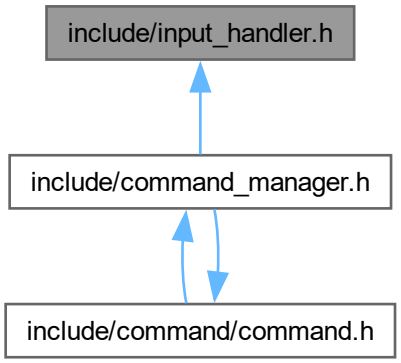
7.11 include/input_handler.h File Reference

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

Include dependency graph for input_handler.h:



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



Classes

- class InputHandler

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

Enumerations

- enum class MouseButton : uint8_t {
  LEFT = SDL_BUTTON_LEFT , MIDDLE = SDL_BUTTON_MIDDLE , RIGHT = SDL_BUTTON_RIGHT , X1 = SDL_BUTTON_X1 ,
  X2 = SDL_BUTTON_X2 }

7.11.1 Enumeration Type Documentation

7.11.1.1 MouseButton

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

Enumerator

LEFT	
MIDDLE	
RIGHT	
X1	
X2	

7.12 input_handler.h

Go to the documentation of this file.

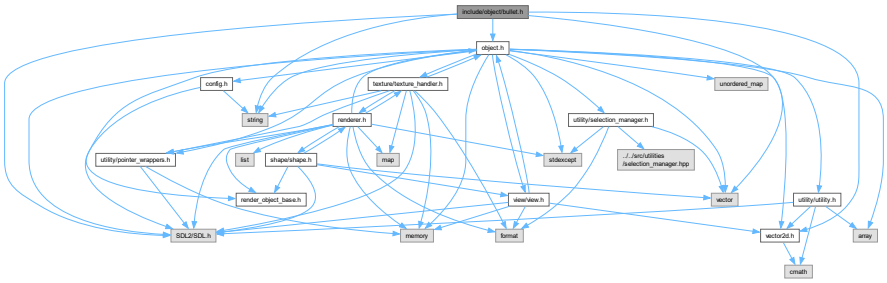
```
00001 #pragma once
00002
00003 #include <utility/vector2d.h>
00004 #include <SDL2/SDL.h>
00005 #include <functional>
00006 #include <map>
00007 #include <memory>
00008
00009 enum class MouseButton : uint8_t {
00010     LEFT    = SDL_BUTTON_LEFT,
00011     MIDDLE    = SDL_BUTTON_MIDDLE,
00012     RIGHT  = SDL_BUTTON_RIGHT,
00013     X1        = SDL_BUTTON_X1,
00014     X2        = SDL_BUTTON_X2
00015 };
00016
00021 class InputHandler {
00022 private:
00023     struct KeyState {
00024         enum { PRESSED, RELEASED, NONE } toggle;
00025         enum { UP, DOWN } hold;
00026         uint32_t holdStart; // The tick this key was first held down.
00027         KeyState() :
00028             toggle(NONE),
00029             hold(UP),
00030             holdStart(0) {}
00031         void toggleDown(void) noexcept {
00032             if (hold == UP) {
00033                 toggle = PRESSED;
00034                 holdStart = SDL_GetTicks();
00035             }
00036             hold = DOWN;
00037         }
00038         void toggleUp(void) noexcept {
00039             if (hold == DOWN) {
00040                 toggle = RELEASED;
00041             }
00042             hold = UP;
00043         }
00044         uint32_t getHoldTime(void) const noexcept {
00045             if (hold == DOWN)
00046                 return SDL_GetTicks() - holdStart;
00047             return 0;
00048         }
00049     };
00050     std::map<SDL_Keycode, KeyState> keyStateDB;
00051     std::map<MouseButton, KeyState> mouseButtonStateDB;
00052     Vector2D mouseScroll;
00053
00054     InputHandler();
00055 public:
00056     InputHandler(const InputHandler&) = delete;
00057     void operator = (const InputHandler&) = delete;
00058
00059     static InputHandler& getInstance(void) noexcept;
00060
00061
00062     // Keyboard functions
00063
00070     bool pollKeyPress(SDL_Keycode key) noexcept;
00071
00078     bool pollKeyRelease(SDL_Keycode key) noexcept;
00079
00085     bool isKeyDown(SDL_Keycode key) const noexcept;
00086
```

```
00092     bool isKeyUp(SDL_Keycode key) const noexcept;
00093
00098     uint32_t holdTime(SDL_Keycode key) const noexcept;
00099
00100
00101     // Mouse functions
00102
00103     bool pollButtonPress(MouseButton button) noexcept;
00104     bool pollButtonRelease(MouseButton button) noexcept;
00105     bool isButtonDown(MouseButton button) const noexcept;
00106     bool isButtonUp(MouseButton button) const noexcept;
00107     uint32_t holdTime(MouseButton button) const noexcept;
00108
00109     Vector2D getMousePosition(void) const noexcept;
00110
00111     Vector2D pollMouseScroll(void) noexcept;
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 };
```

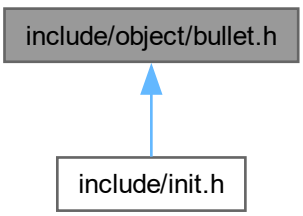
7.13 include/object/bullet.h File Reference

```
#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

7.14 bullet.h

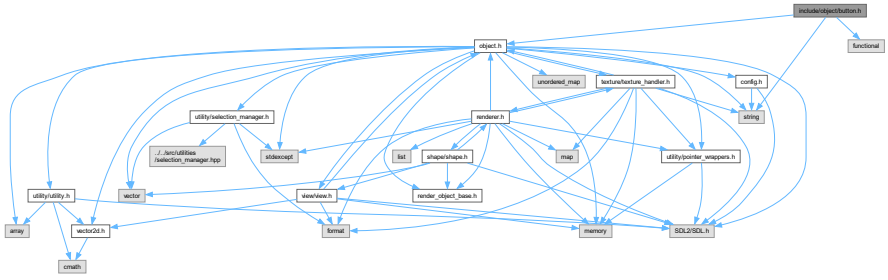
Go to the documentation of this file.

```
00001 #pragma once
00002
00003 #include "object.h"
00004 #include <utility/vector2d.h>
00005 #include <SDL2/SDL.h>
00006 #include <vector>
00007 #include <string>
00008
00009 namespace Objects {
00010     class Bullet : public Object {
00011     private:
00012         Uint32 createdTick;
00013         float speed;
00014     public:
00015         const static Vector2D bulletSize;
00016         Bullet(
00017             const Views::View* view,
00018             Vector2D position,
00019             float angle,
00020             float speed = 20.0f
00021         ) : Object(
00022                 std::vector<std::string> {"bullet"},
00023                 view,
00024                 position,
00025                 bulletSize
00026             ),
00027             createdTick(SDL_GetTicks()),
00028             speed(speed) {
00029             this->setAngle(angle);
00030         }
00031
00036         Uint32 getAliveTime(void) const noexcept;
00037
00038         void update(void) noexcept override;
00039     };
00040 }
```

7.15 include/object/button.h File Reference

```
#include "object.h"
#include <string>
#include <functional>
```

Include dependency graph for button.h:



Classes

- class Objects::Button

Namespaces

- namespace Objects

7.16 button.h

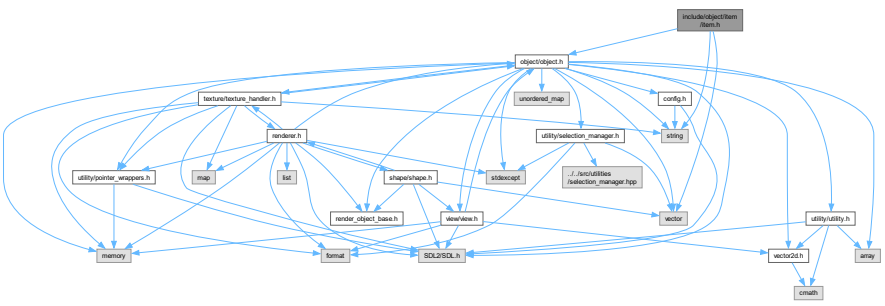
Go to the documentation of this file.

```
00001 #pragma once
00002
00003 #include "object.h"
00004 #include <string>
00005 #include <functional>
00006
00007 namespace Objects {
00008     class Button : private Object {
00009     private:
00010         std::string text;
00011         bool hover;
00012         std::function<void(void)> actionFunc;
00013
00014         bool pollHover(void) noexcept;
00015
00016     public:
00017         Button(
00018             const Views::View* view,
00019             const Vector2D& position,
00020             const Vector2D& dimension,
00021             const SDL_Color& color,
00022             const std::string& text,
00023             std::function<void(void)> action
00024         );
00025
00026         void setHovered(void) noexcept;
00027
00028         void onClick(void) noexcept;
00029
00030         void update(void) noexcept;
00031     };
00032 }
```

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

```
#include <object/object.h>
#include <vector>
#include <string>
```

Include dependency graph for item.h:



Classes

- class Items::Item

Namespaces

- namespace Items

7.18 item.h

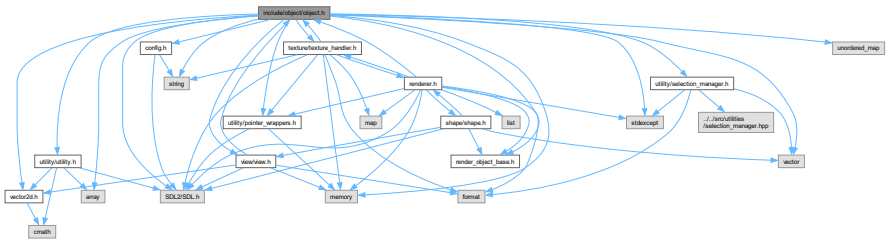
Go to the documentation of this file.

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

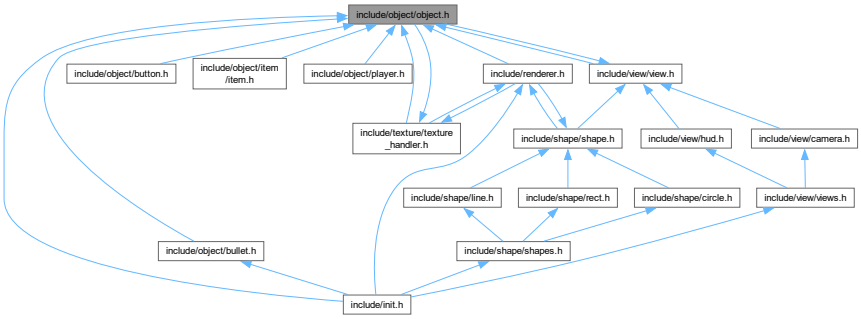
7.19 include/object/object.h File Reference

```
#include <render_object_base.h>
#include <utility/utility.h>
#include <utility/pointer_wrappers.h>
#include <utility/vector2d.h>
#include <utility/selection_manager.h>
#include <texture/texture_handler.h>
#include <view/view.h>
#include <config.h>
#include <SDL2/SDL.h>
#include <memory>
#include <string>
#include <vector>
#include <array>
#include <unordered_map>
```

`#include <stdexcept>`
Include dependency graph for object.h:



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.

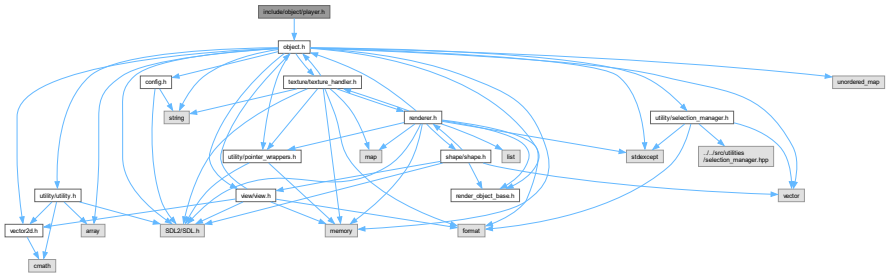
```
00001 #pragma once
00002
00003 #include <render_object_base.h>
00004 #include <utility/utility.h>
00005 #include <utility/pointer_wrappers.h>
00006 #include <utility/vector2d.h>
00007 #include <utility/selection_manager.h>
00008 #include <texture/texture_handler.h>
00009 #include <view/view.h>
00010 #include <config.h>
00011 #include <SDL2/SDL.h>
00012 #include <memory>
00013 #include <string>
00014 #include <vector>
```

```
00015 #include <array>
00016 #include <unordered_map>
00017 #include <stdexcept>
00018
00019 namespace Views { class View; }
00020 class TextureHandler;
00021
00022 namespace Objects {
00023
00024     // TODO: add 'shapes' field to `Objects::Object`
00025
00030     class Object : public RenderObjectBase {
00031         friend class TextureHandler;
00032     private:
00033         SelectionManager<SDL_Texture*> textures;
00034         bool visible;
00035
00036         float angle; // stored as radians
00037         SDL_RendererFlip flipFlag;
00038 //      SDL_Color colorMask; // color mod mask
00039         Vector2D position; // actual position in the world
00040         Vector2D dimension; // height and width
00041         const Views::View* view;
00042     public:
00043
00051         Object(
00052             const std::vector<std::string>& textureNames,
00053             const Views::View* _view,
00054             const Vector2D& _position,
00055             const Vector2D& _dimension
00056         );
00057
00058         virtual ~Object() = default;
00059
00066         float getAngle(void) const noexcept;
00067
00072         float getRenderAngle(void) const noexcept;
00073
00078         void setAngle(float newAngle) noexcept;
00079
00085         void rotate(float diffAngle) noexcept;
00086
00091         SDL_RendererFlip getFlipFlag(void) const noexcept;
00092
00097         Vector2D getPosition(void) const noexcept;
00098
00102         void setPosition(const Vector2D& newPosition) noexcept;
00103
00108         Vector2D getDimension(void) const noexcept;
00109
00113         void setDimension(const Vector2D& newDimension);
00114
00119         void move(const Vector2D& translate) noexcept;
00120
00125         void stretchX(float ratio) noexcept;
00126
00131         void stretchY(float ratio) noexcept;
00132
00137         void stretch(float ratio) noexcept;
00138
00142         void flipHorizontal(void) noexcept;
00143
00147         void flipVertical(void) noexcept;
00148
00153         void setVisibility(bool visibility) noexcept;
00154
00159         bool getVisibility(void) const noexcept;
00160
00164         bool collideWith(const Object& other) const noexcept;
00165
00166         /* TEXTURES */
00167
00171         void nextTexture(void) noexcept;
00172
00176         void previousTexture(void) noexcept;
00177
00182         void setTexture(int textureId) noexcept;
00183
00188         size_t getTextureCount(void) const noexcept;
00189
00194         SDL_Texture* getTexture(void) const noexcept;
00195
00196         /* TEXTURES */
00197
00198
00203         virtual void lookAt(const Vector2D& position) noexcept;
00204
```

```
00209         SDL_FRect getRenderRect(void) const noexcept;
00210         //Vector2D getRenderRelativePosition(Vector2D renderPosition) const noexcept;
00211
00215         virtual void update(void) noexcept;
00216
00217         // debug
00218         void debug(void) const noexcept override;
00219     };
00220 }
```

7.21 include/object/player.h File Reference

`#include "object.h"`
Include dependency graph for player.h:



Classes

- class Objects::Player

Namespaces

- namespace Objects

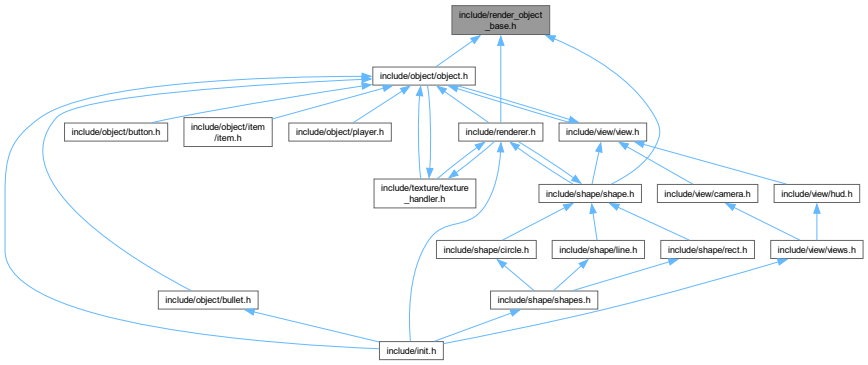
7.22 player.h

Go to the documentation of this file.

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

7.23 include/render_object_base.h File Reference

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



Classes

- class RenderObjectBase

  Empty render object base class category.

7.24 render_object_base.h

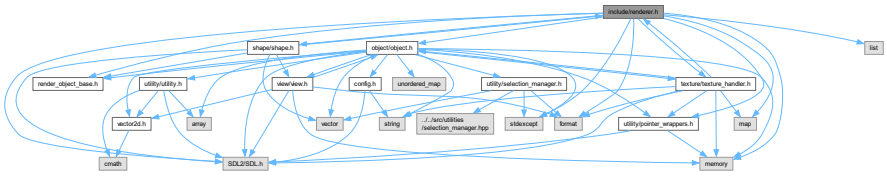
Go to the documentation of this file.

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

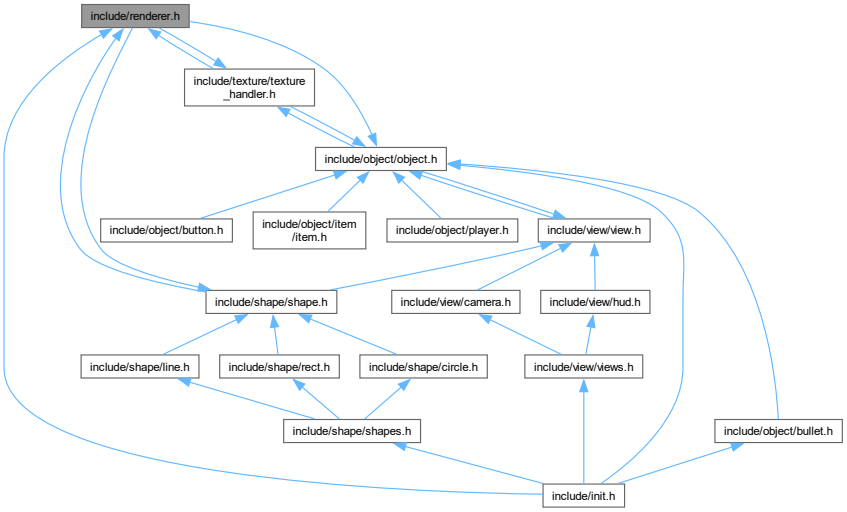
7.25 include/renderer.h File Reference

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

`#include <format>`
Include dependency graph for renderer.h:



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



Classes

- class Renderer

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

Namespaces

- namespace Objects

7.26 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>
```

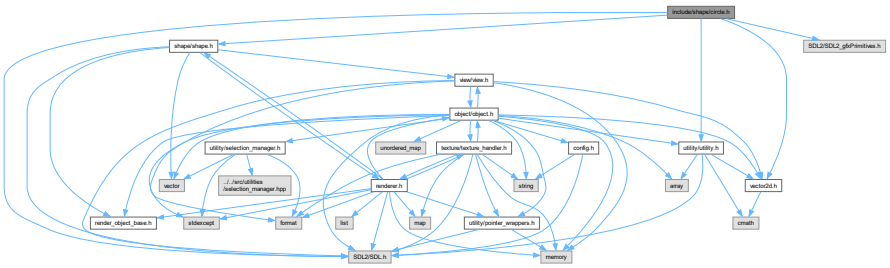
```
00008 #include <SDL2/SDL.h>
00009 #include <memory>
00010 #include <list>
00011 #include <map>
00012 #include <stdexcept>
00013 #include <format>
00014
00015 namespace Objects {
00016     class Object;
00017 }
00018
00019 // TODO: Consider wrapping object layer management into a LayerManager class.
00020
00021 // Singleton is needed as the renderer can only be initialized at runtime.
00026 class Renderer {
00030     class CreateTextureKey {
00031         friend class TextureHandler;
00032     private:
00033         CreateTextureKey() = default;
00034         CreateTextureKey(const CreateTextureKey&) = default;
00035     };
00036
00037 public: // TODO: change this to private, this is for testing purposes.
00041     class RenderKey {
00042     public: // TODO: change this to private, this is for testing purposes.
00043         RenderKey() = default;
00044         RenderKey(const RenderKey&) = default;
00045     };
00046
00047 private:
00048     using ObjectWeakPtr = std::weak_ptr<RenderObjectBase>;
00049     using ObjectList = std::list<ObjectWeakPtr>;
00050
00051 private:
00052     sdl_unique_ptr<SDL_Window> window;
00053     sdl_unique_ptr<SDL_Renderer> renderer;
00054     std::map<ObjectWeakPtr, ObjectList::iterator, std::owner_less<ObjectWeakPtr>> objectListMap;
00055     ObjectList objectList;
00056
00061     Renderer();
00062 public:
00063     /* SINGLETON PATTERN */
00064     Renderer(const Renderer&) = delete;
00065     void operator = (const Renderer&) = delete;
00066     static Renderer& getInstance(void) noexcept;
00067     /* SINGLETON PATTERN */
00068
00071     SDL_Window* getWindow(void) noexcept;
00072
00079     SDL_Texture* createTexture(CreateTextureKey key, SDL_Surface* surface) const;
00080
00085     //SDL_Renderer* getRenderer(void) noexcept;
00086
00089     Vector2D getWindowSize(void) const noexcept;
00090
00096     bool registerObject(std::shared_ptr<RenderObjectBase> objectPtr) noexcept;
00097
00103     bool removeObject(std::weak_ptr<RenderObjectBase> objectPtr) noexcept;
00104
00110     void render(const RenderKey& key);
00111
00117     void moveLayerUp(std::shared_ptr<RenderObjectBase> objectPtr);
00118
00124     void moveLayerDown(std::shared_ptr<RenderObjectBase> objectPtr);
00125
00131     void moveLayerTop(std::shared_ptr<RenderObjectBase> objectPtr);
00132
00138     void moveLayerBottom(std::shared_ptr<RenderObjectBase> objectPtr);
00139
00143     void clear() noexcept;
00144
00148     void debug(void) const noexcept;
00149 };
```

7.27 include/shape/circle.h File Reference

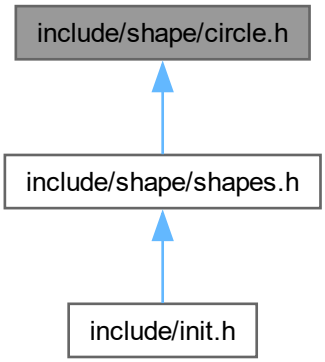
```
#include <shape/shape.h>
#include <utility/vector2d.h>
#include <utility/utility.h>
#include <SDL2/SDL.h>
```

```
#include <SDL2/SDL2_gfxPrimitives.h>
```

Include dependency graph for circle.h:



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



Classes

- class Shapes::Circle
- class Shapes::HollowCircle

Namespaces

- namespace Views
- namespace Shapes

7.28 circle.h

Go to the documentation of this file.

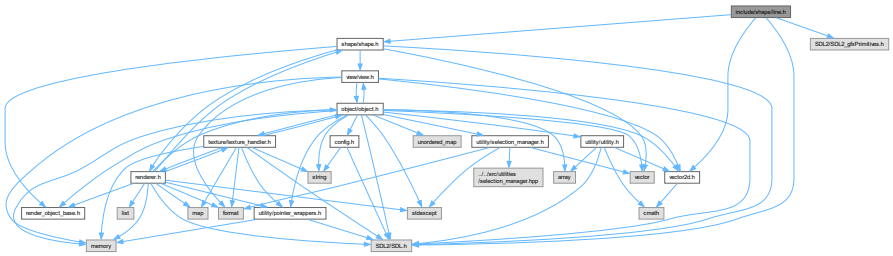
```
00001 #pragma once
00002
00003 #include <shape/shape.h>
00004 #include <utility/vector2d.h>
```

```
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     };
00029
00030     class HollowCircle : public Circle {
00031     private:
00032         static const int renderEdges = 36;
00033     protected:
00034         uint8_t thickness;
00035     public:
00036         HollowCircle(
00037             Views::View* view,
00038             const Vector2D& center,
00039             float radius,
00040             uint8_t thickness,
00041             SDL_Color color = { 0, 0, 0, 255 }
00042         ) noexcept;
00043         void setThickness(uint8_t newThickness) noexcept;
00044         void draw(SDL_Renderer* renderer) const noexcept override;
00045     };
00046 }
```

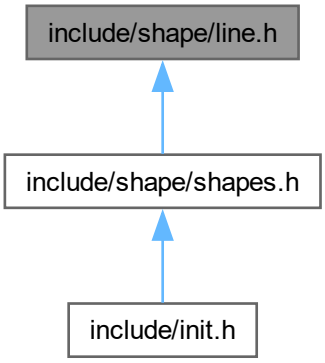
7.29 include/shape/line.h File Reference

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

Include dependency graph for line.h:



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



Classes

- class Shapes::Line

Namespaces

- namespace Shapes

7.30 line.h

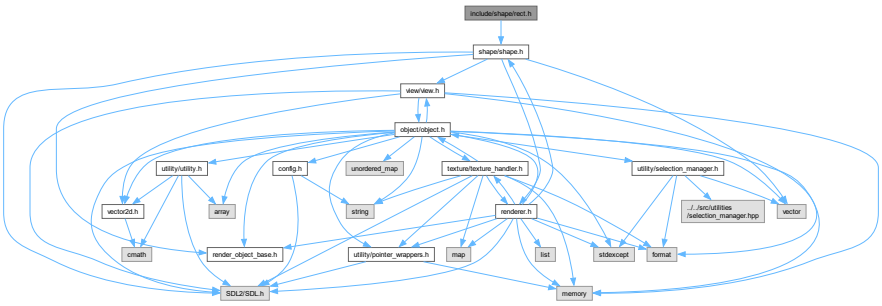
Go to the documentation of this file.

```
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 }
```

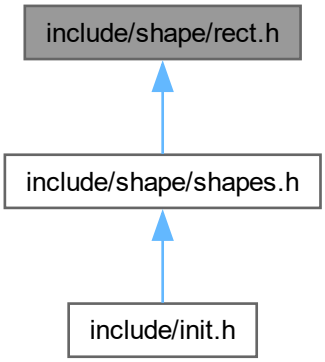
7.31 include/shape/rect.h File Reference

```
#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

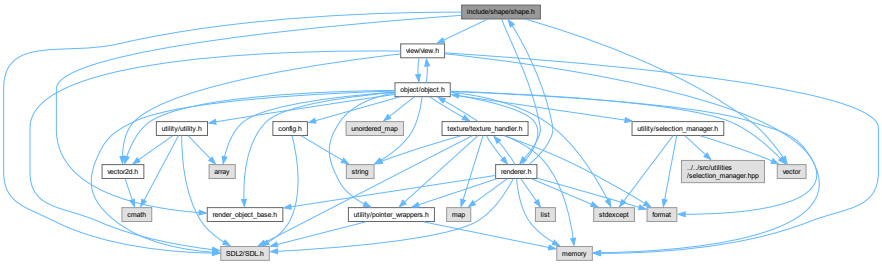
Go to the documentation of this file.

```
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 }
```

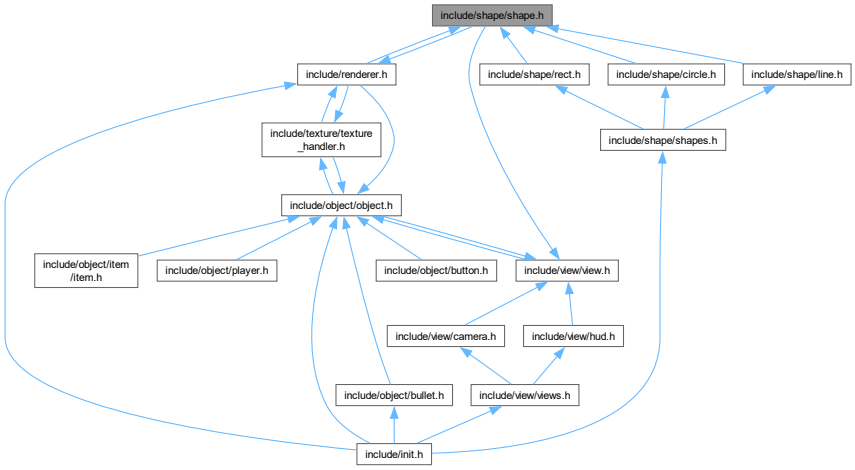
7.33 include/shape/shape.h File Reference

```
#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

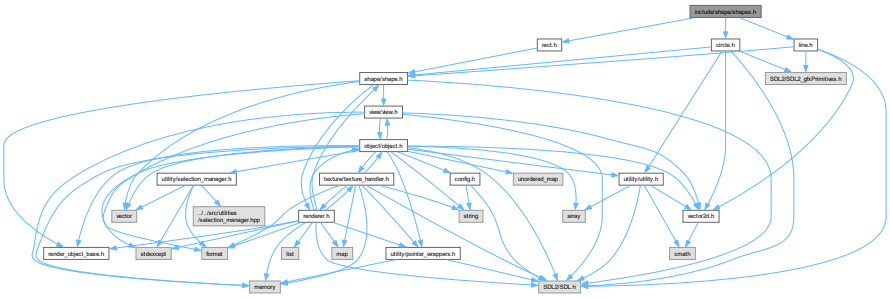
Go to the documentation of this file.

```
00001 #pragma once
00002
00003 #include <render_object_base.h>
00004 #include <view/view.h>
00005 #include <renderer.h>
00006 #include <SDL2/SDL.h>
00007 #include <vector>
00008
00009 namespace Views { class View; }
00010
00011 namespace Shapes {
00012     class Shape : public RenderObjectBase {
00013     private:
00014     protected:
00015         const Views::View* view;
00016         SDL_Color color;
00017     public:
00018         virtual void draw(SDL_Renderer* renderer) const noexcept {}
00019         Shape(Views::View* view, const SDL_Color& color = { 0, 0, 0, 255 });
00020         virtual ~Shape() = default;
00021
00022         void setColor(const SDL_Color& newColor) noexcept;
00023         SDL_Color getColor(void) const noexcept;
00024     };
00025 }
```

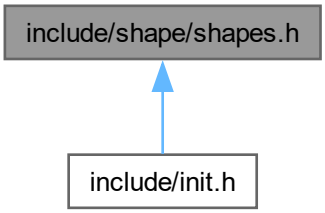
7.35 include/shape/shapes.h File Reference

```
#include "line.h"
#include "circle.h"
#include "rect.h"
```

Include dependency graph for shapes.h:



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



7.36 shapes.h

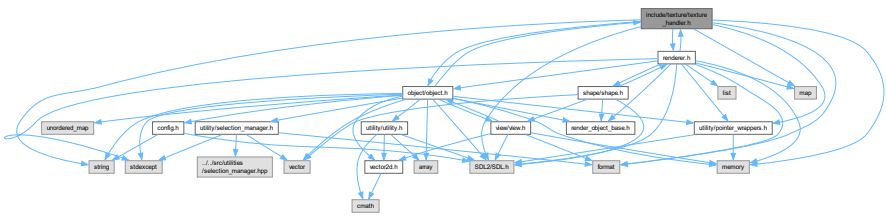
Go to the documentation of this file.

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

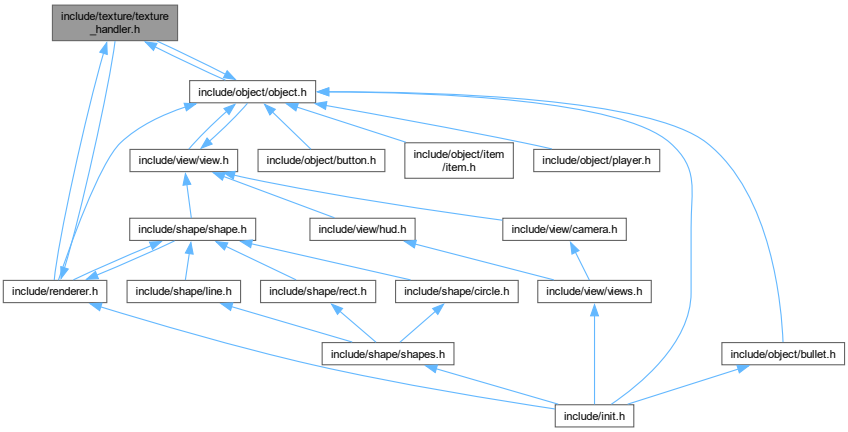
7.37 include/texture/texture_handler.h File Reference

```
#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

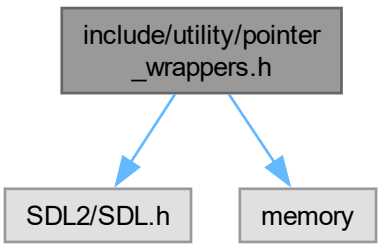
Go to the documentation of this file.

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

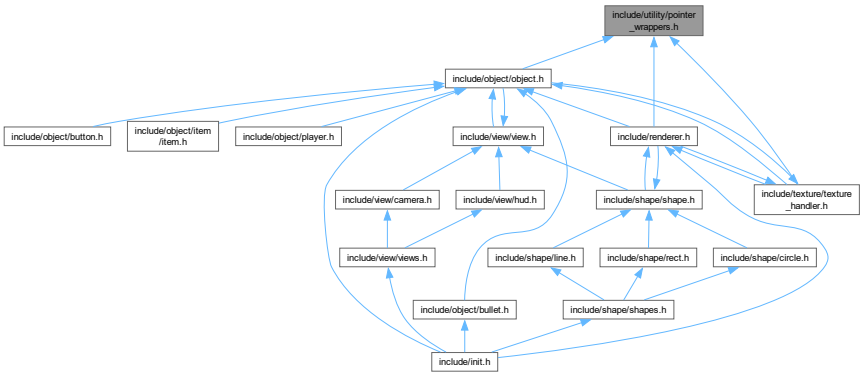
7.41 include/utility/pointer_wrappers.h File Reference

```
#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

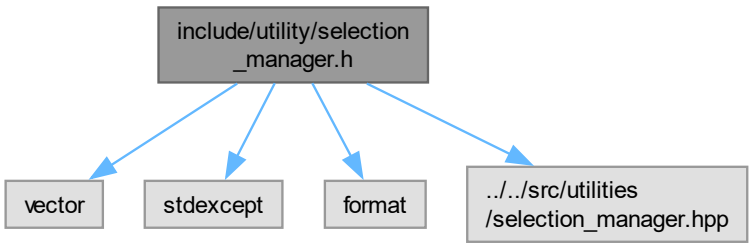
Go to the documentation of this file.

```
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 }
```

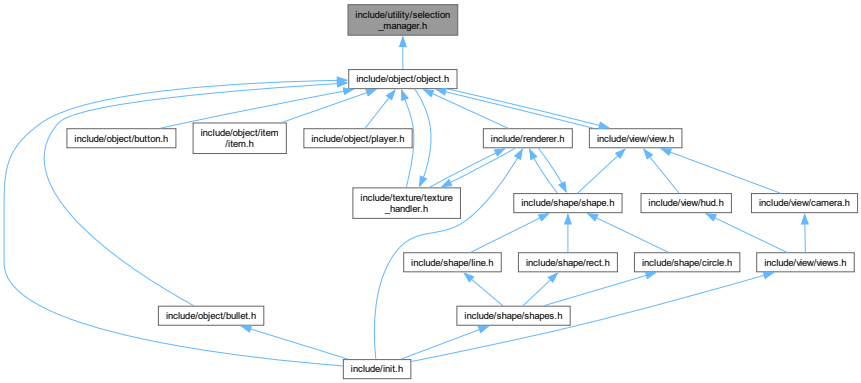
7.43 include/utility/selection_manager.h File Reference

```
#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

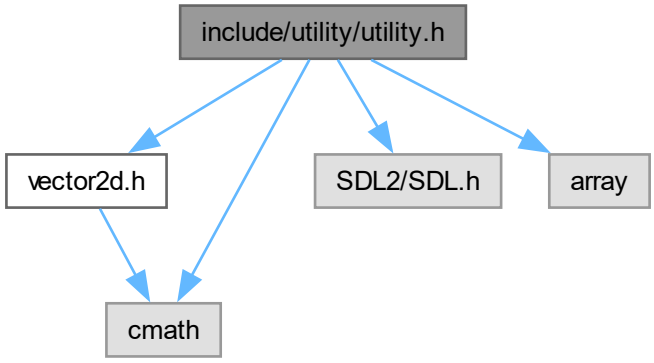
Go to the documentation of this file.

```
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"
```

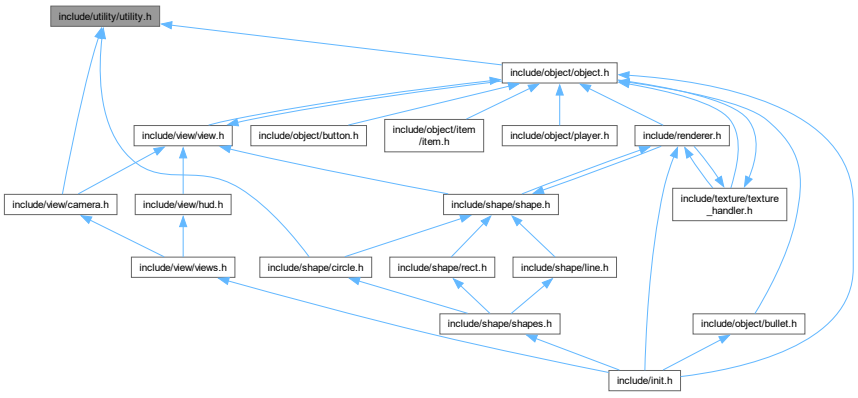
7.45 include/utility/utility.h File Reference

```
#include "vector2d.h"
#include <SDL2/SDL.h>
#include <cmath>
#include <array>
```

Include dependency graph for utility.h:



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



Macros

- #define _USE_MATH_DEFINES

Functions

- float normalizeAngle (float angle) noexcept

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

- Vector2D polarToCartesian (float radius, float theta)

  Helper function to transform polar coordinates to cartesian coordinates.
- bool rectCollide (const SDL_FRect &rect1, float angle1, const SDL_FRect &rect2, float angle2)

  Checks if two rectangles collides.

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, 2pi)

Parameters

angle	input angle

Returns

normalized angle

7.45.2.2 polarToCartesian()

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

Helper function to transform polar coordinates to cartesian coordinates.

Parameters

radius	input radius
theta	input angle (radians)

Returns

the transformed cartesian coordinates

7.45.2.3 rectCollide()

```
bool rectCollide (
            const SDL_FRect & rect1,
            float angle1,
            const SDL_FRect & rect2,
            float angle2 )
```

Checks if two rectangles collides.

Parameters

rect1	First rect.
angle1	The rotation of the first rect.
rect2	Second rect.
angle2	The rotation of the second rect.

Returns

If the rectangles collides.

7.46 utility.h

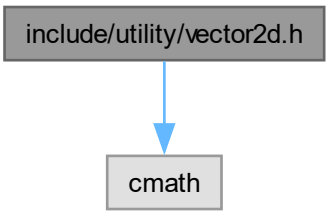
Go to the documentation of this file.

```
00001 #pragma once
00002
00003 #include "vector2d.h"
00004 #include <SDL2/SDL.h>
00005 #define _USE_MATH_DEFINES
00006 #include <cmath>
00007 #include <array>
00008
00014 float normalizeAngle(float angle) noexcept;
00015
00022 Vector2D polarToCartesian(float radius, float theta);
00023
00032 bool rectCollide(const SDL_FRect& rect1, float angle1, const SDL_FRect& rect2, float angle2);
```

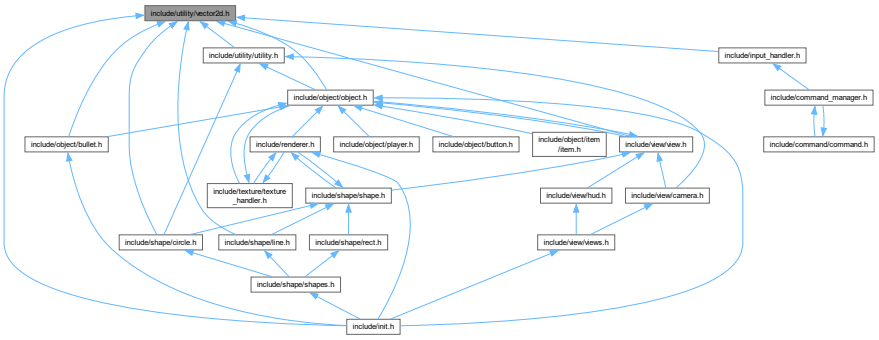
7.47 include/utility/vector2d.h File Reference

`#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

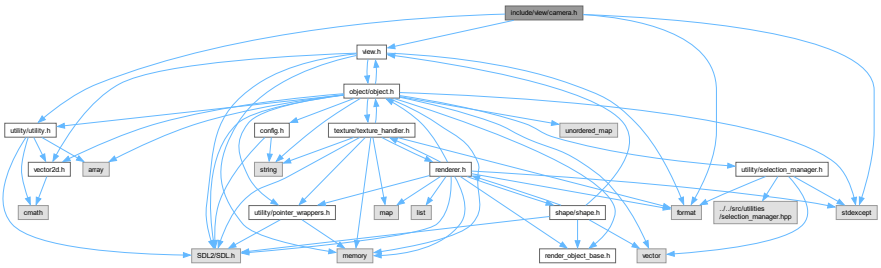
Go to the documentation of this file.

```
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 * (float, const Vector2D&) noexcept;
00032     friend Vector2D operator / (const Vector2D&, float) noexcept;
00033     friend Vector2D& operator += (Vector2D&, const Vector2D&) noexcept;
00034     friend Vector2D& operator -= (Vector2D&, const Vector2D&) noexcept;
00035     friend Vector2D& operator *= (Vector2D&, float) noexcept;
00036     friend Vector2D& operator /= (Vector2D&, float) noexcept;
00037     static float dot(const Vector2D&, const Vector2D&) noexcept;
00038     static float cross(const Vector2D&, const Vector2D&) noexcept;
00039     static Vector2D rotate(Vector2D, float) noexcept;
00040 };
```

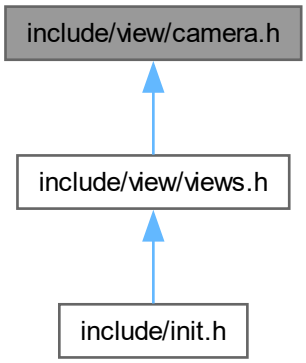
7.49 include/view/camera.h File Reference

```
#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

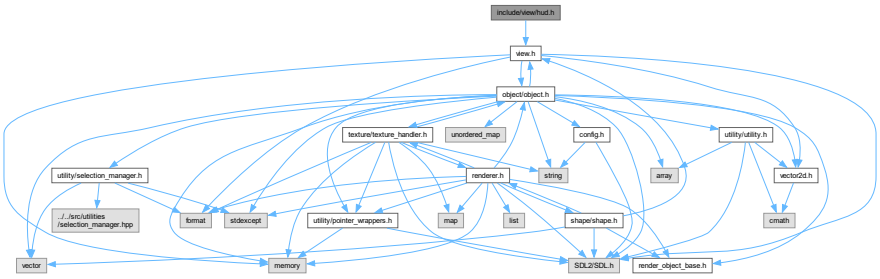
Go to the documentation of this file.

```
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 }
```

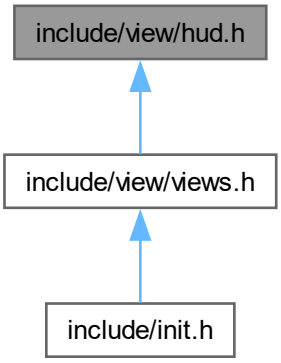
7.51 include/view/hud.h File Reference

`#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.

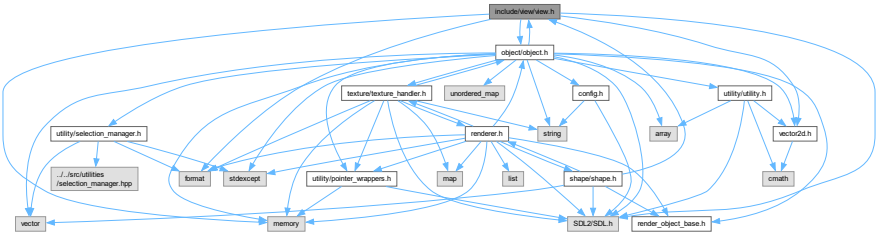
```
00001 #pragma once
00002
00003 #include "view.h"
00004
00005 namespace Views {
00006     class HUD : public View {
00007     public:
00008         HUD();
00009         SDL_FRect getRect(const Objects::Object&) const noexcept override;
00010         Vector2D transform(const Vector2D& position) const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

7.53 include/view/view.h File Reference

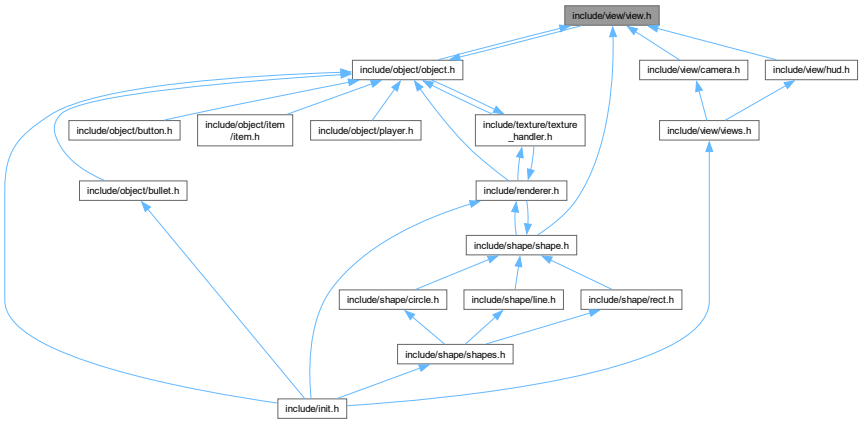
```
#include <object/object.h>
#include <utility/vector2d.h>
#include <SDL2/SDL.h>
#include <memory>
```

```
#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

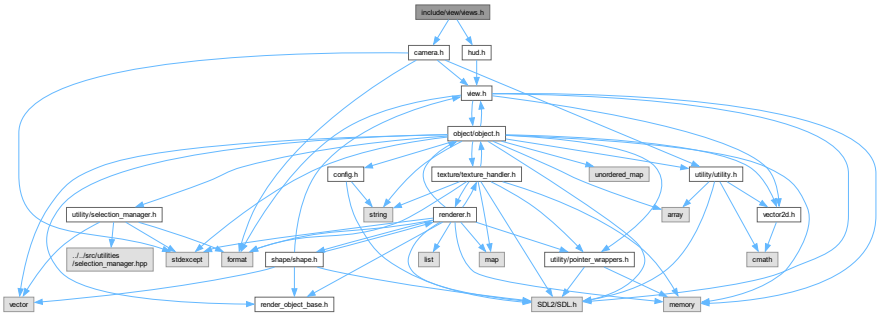
Go to the documentation of this file.

```
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
00018     class View {
00019     protected:
00020         Vector2D position;
00021         Vector2D dimension;
00022
00023         View(const Vector2D& _position, const Vector2D& _dimension) :
00024             position(_position), dimension(_dimension) {}
00025     public:
00026         virtual ~View() {};
00027
00033         virtual SDL_FRect getRect(const Objects::Object& object) const noexcept = 0;
00034
00040         virtual Vector2D transform(const Vector2D& position) const noexcept = 0;
00041
00047         virtual Vector2D transformFromRender(const Vector2D& renderPosition) const noexcept = 0;
00048
00053         virtual Vector2D getPosition(void) const noexcept { return position; }
00054
00059         virtual Vector2D getDimension(void) const noexcept { return dimension; }
00060
00065         virtual float getAngle(void) const noexcept { return 0.0f; }
00066
00071         virtual float getZoom(void) const noexcept { return 1.0f; }
00072     };
00073
00074     const int INIT_VIEW_WIDTH = 1600;
00075     const int INIT_VIEW_HEIGHT = 900;
00076 }
```

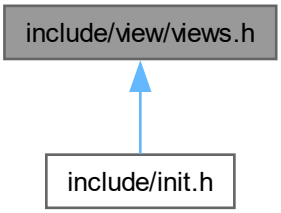
7.55 include/view/views.h File Reference

```
#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