

Headsoccer

Generated by Doxygen 1.9.4



<b>1 Head_soccer_game</b>	<b>1</b>
<b>2 Hierarchical Index</b>	<b>3</b>
2.1 Class Hierarchy	3
<b>3 Class Index</b>	<b>5</b>
3.1 Class List	5
<b>4 File Index</b>	<b>7</b>
4.1 File List	7
<b>5 Class Documentation</b>	<b>9</b>
5.1 BallObject Class Reference	9
5.1.1 Constructor & Destructor Documentation	9
5.1.1.1 BallObject() [1/2]	9
5.1.1.2 BallObject() [2/2]	10
5.1.2 Member Function Documentation	10
5.1.2.1 Move()	10
5.1.2.2 Reset()	10
5.1.3 Member Data Documentation	10
5.1.3.1 Radius	10
5.1.3.2 Stuck	10
5.2 Game Class Reference	11
5.2.1 Constructor & Destructor Documentation	11
5.2.1.1 Game()	11
5.2.1.2 ~Game()	11
5.2.2 Member Function Documentation	11
5.2.2.1 DoCollisions()	11
5.2.2.2 Init()	12
5.2.2.3 ProcessInput()	12
5.2.2.4 Render()	12
5.2.2.5 ResetPlayers()	12
5.2.2.6 Update()	12
5.2.3 Member Data Documentation	12
5.2.3.1 Height	12
5.2.3.2 Keys	12
5.2.3.3 State	13
5.2.3.4 Width	13
5.3 GameObject Class Reference	13
5.3.1 Constructor & Destructor Documentation	13
5.3.1.1 GameObject() [1/2]	14
5.3.1.2 GameObject() [2/2]	14
5.3.2 Member Function Documentation	14
5.3.2.1 Draw()	14

5.3.3 Member Data Documentation	14
5.3.3.1 Color	14
5.3.3.2 Destroyed	14
5.3.3.3 IsSolid	14
5.3.3.4 Position	15
5.3.3.5 Rotation	15
5.3.3.6 Size	15
5.3.3.7 Sprite	15
5.3.3.8 Velocity	15
5.4 ResourceManager Class Reference	15
5.4.1 Member Function Documentation	16
5.4.1.1 Clear()	16
5.4.1.2 GetShader()	16
5.4.1.3 GetTexture()	16
5.4.1.4 LoadShader()	16
5.4.1.5 LoadTexture()	16
5.4.2 Member Data Documentation	17
5.4.2.1 Shaders	17
5.4.2.2 Textures	17
5.5 Shader Class Reference	17
5.5.1 Constructor & Destructor Documentation	18
5.5.1.1 Shader() [1/2]	18
5.5.1.2 Shader() [2/2]	18
5.5.2 Member Function Documentation	18
5.5.2.1 Compile()	18
5.5.2.2 setBool()	18
5.5.2.3 SetFloat()	18
5.5.2.4 setFloat()	19
5.5.2.5 setInt()	19
5.5.2.6 SetInteger()	19
5.5.2.7 SetMatrix4()	19
5.5.2.8 SetVector2f() [1/2]	19
5.5.2.9 SetVector2f() [2/2]	19
5.5.2.10 SetVector3f() [1/2]	20
5.5.2.11 SetVector3f() [2/2]	20
5.5.2.12 SetVector4f() [1/2]	20
5.5.2.13 SetVector4f() [2/2]	20
5.5.2.14 Use()	20
5.5.2.15 use()	20
5.5.3 Member Data Documentation	21
5.5.3.1 ID	21
5.6 SpriteRenderer Class Reference	21

5.6.1 Constructor & Destructor Documentation	21
5.6.1.1 SpriteRenderer()	21
5.6.1.2 ~SpriteRenderer()	21
5.6.2 Member Function Documentation	21
5.6.2.1 DrawSprite()	22
5.7 Texture2D Class Reference	22
5.7.1 Constructor & Destructor Documentation	22
5.7.1.1 Texture2D()	22
5.7.2 Member Function Documentation	23
5.7.2.1 Bind()	23
5.7.2.2 Generate()	23
5.7.3 Member Data Documentation	23
5.7.3.1 Filter_Max	23
5.7.3.2 Filter_Min	23
5.7.3.3 Height	23
5.7.3.4 ID	23
5.7.3.5 Image_Format	24
5.7.3.6 Internal_Format	24
5.7.3.7 Width	24
5.7.3.8 Wrap_S	24
5.7.3.9 Wrap_T	24
<b>6 File Documentation</b>	<b>25</b>
6.1 include/game.h File Reference	25
6.1.1 Enumeration Type Documentation	25
6.1.1.1 GameState	25
6.2 game.h	26
6.3 include/game_object.h File Reference	26
6.4 game_object.h	26
6.5 include/resource_manager.h File Reference	27
6.6 resource_manager.h	27
6.7 include/shader.h File Reference	28
6.8 shader.h	28
6.9 include/shader_s.h File Reference	29
6.10 shader_s.h	29
6.11 include/sprite_renderer.h File Reference	31
6.12 sprite_renderer.h	31
6.13 include/texture.h File Reference	31
6.14 texture.h	32
6.15 README.md File Reference	32
6.16 src/game.cpp File Reference	32
6.16.1 Typedef Documentation	33

6.16.1.1 Collision	33
6.16.2 Enumeration Type Documentation	33
6.16.2.1 Direction	33
6.16.3 Function Documentation	33
6.16.3.1 CheckCollision() [1/2]	34
6.16.3.2 CheckCollision() [2/2]	34
6.16.3.3 INITIAL_BALL_VELOCITY()	34
6.16.3.4 PLAYER_SIZE()	34
6.16.3.5 PLAYER_VELOCITY()	34
6.16.3.6 VectorDirection()	34
6.16.4 Variable Documentation	34
6.16.4.1 Ball	35
6.16.4.2 BALL_RADIUS	35
6.16.4.3 Player1	35
6.16.4.4 Player2	35
6.16.4.5 Renderer	35
6.17 src/game_object.cpp File Reference	35
6.18 src/main.cpp File Reference	35
6.18.1 Function Documentation	36
6.18.1.1 framebuffer_size_callback()	36
6.18.1.2 key_callback()	36
6.18.1.3 main()	36
6.18.2 Variable Documentation	36
6.18.2.1 Headsoccer	37
6.18.2.2 SCREEN_HEIGHT	37
6.18.2.3 SCREEN_WIDTH	37
6.19 src/resource_manager.cpp File Reference	37
6.20 src/shader.cpp File Reference	37
6.21 src/sprite_renderer.cpp File Reference	37
6.22 src/texture.cpp File Reference	37

<b>Index</b>	<b>39</b>
--------------	-----------

## Chapter 1

# Head\_soccer\_game

**Game** of skill that uses OpenGL API (GLFW library)

Useful links: <https://learnopengl.com>





## Chapter 2

# Hierarchical Index

### 2.1 Class Hierarchy

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

Game . . . . .	11
GameObject . . . . .	13
BallObject . . . . .	9
ResourceManager . . . . .	15
Shader . . . . .	17
SpriteRenderer . . . . .	21
Texture2D . . . . .	22



## Chapter 3

# Class Index

### 3.1 Class List

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

<a href="#">BallObject</a>	9
<a href="#">Game</a>	11
<a href="#">GameObject</a>	13
<a href="#">ResourceManager</a>	15
<a href="#">Shader</a>	17
<a href="#">SpriteRenderer</a>	21
<a href="#">Texture2D</a>	22



## Chapter 4

# File Index

### 4.1 File List

Here is a list of all files with brief descriptions:

<a href="#">include/game.h</a>	25
<a href="#">include/game_object.h</a>	26
<a href="#">include/resource_manager.h</a>	27
<a href="#">include/shader.h</a>	28
<a href="#">include/shader_s.h</a>	29
<a href="#">include/sprite_renderer.h</a>	31
<a href="#">include/texture.h</a>	31
<a href="#">src/game.cpp</a>	32
<a href="#">src/game_object.cpp</a>	35
<a href="#">src/main.cpp</a>	35
<a href="#">src/resource_manager.cpp</a>	37
<a href="#">src/shader.cpp</a>	37
<a href="#">src/sprite_renderer.cpp</a>	37
<a href="#">src/texture.cpp</a>	37



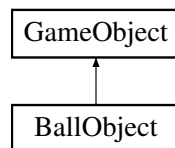
## Chapter 5

# Class Documentation

### 5.1 BallObject Class Reference

```
#include <game_object.h>
```

Inheritance diagram for BallObject:



#### Public Member Functions

- [BallObject](#) ()
- [BallObject](#) (glm::vec2 pos, float radius, glm::vec2 velocity, [Texture2D](#) sprite)
- glm::vec2 [Move](#) (float dt, unsigned int window\_width, unsigned int window\_height)
- void [Reset](#) (glm::vec2 position, glm::vec2 velocity)

#### Public Attributes

- float [Radius](#)
- bool [Stuck](#)

#### 5.1.1 Constructor & Destructor Documentation

##### 5.1.1.1 BallObject() [1/2]

```
BallObject::BallObject ( )
```

#### 5.1.1.2 BallObject() [2/2]

```
BallObject::BallObject (
    glm::vec2 pos,
    float radius,
    glm::vec2 velocity,
    Texture2D sprite )
```

### 5.1.2 Member Function Documentation

#### 5.1.2.1 Move()

```
glm::vec2 BallObject::Move (
    float dt,
    unsigned int window_width,
    unsigned int window_height )
```

#### 5.1.2.2 Reset()

```
void BallObject::Reset (
    glm::vec2 position,
    glm::vec2 velocity )
```

### 5.1.3 Member Data Documentation

#### 5.1.3.1 Radius

```
float BallObject::Radius
```

#### 5.1.3.2 Stuck

```
bool BallObject::Stuck
```

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

- [include/game\\_object.h](#)
- [src/game\\_object.cpp](#)



## 5.2 Game Class Reference

```
#include <game.h>
```

### Public Member Functions

- [Game](#) (unsigned int width, unsigned int height)
- [~Game](#) ()
- void [Init](#) ()
- void [ProcessInput](#) (float dt)
- void [Update](#) (float dt)
- void [Render](#) ()
- void [DoCollisions](#) ()
- void [ResetPlayers](#) ()

### Public Attributes

- [GameState](#) State
- bool [Keys](#) [1024]
- unsigned int [Width](#)
- unsigned int [Height](#)

### 5.2.1 Constructor & Destructor Documentation

#### 5.2.1.1 Game()

```
Game::Game (
    unsigned int width,
    unsigned int height )
```

#### 5.2.1.2 ~Game()

```
Game::~~Game ( )
```

### 5.2.2 Member Function Documentation

#### 5.2.2.1 DoCollisions()

```
void Game::DoCollisions ( )
```

#### 5.2.2.2 Init()

```
void Game::Init ( )
```

#### 5.2.2.3 ProcessInput()

```
void Game::ProcessInput (
    float dt )
```

#### 5.2.2.4 Render()

```
void Game::Render ( )
```

#### 5.2.2.5 ResetPlayers()

```
void Game::ResetPlayers ( )
```

#### 5.2.2.6 Update()

```
void Game::Update (
    float dt )
```

### 5.2.3 Member Data Documentation

#### 5.2.3.1 Height

```
unsigned int Game::Height
```

#### 5.2.3.2 Keys

```
bool Game::Keys[1024]
```

### 5.2.3.3 State

`GameState` `Game::State`

### 5.2.3.4 Width

`unsigned int` `Game::Width`

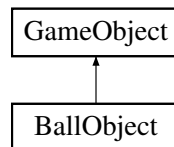
The documentation for this class was generated from the following files:

- [include/game.h](#)
- [src/game.cpp](#)

## 5.3 GameObject Class Reference

```
#include <game_object.h>
```

Inheritance diagram for `GameObject`:



### Public Member Functions

- [GameObject](#) ()
- [GameObject](#) (`glm::vec2` pos, `glm::vec2` size, [Texture2D](#) sprite, `glm::vec3` color=`glm::vec3`(1.0f), `glm::vec2` velocity=`glm::vec2`(0.0f, 0.0f))
- virtual void [Draw](#) ([SpriteRenderer](#) &renderer)

### Public Attributes

- `glm::vec2` [Position](#)
- `glm::vec2` [Size](#)
- `glm::vec2` [Velocity](#)
- `glm::vec3` [Color](#)
- float [Rotation](#)
- bool [IsSolid](#)
- bool [Destroyed](#)
- [Texture2D](#) [Sprite](#)

### 5.3.1 Constructor & Destructor Documentation

#### 5.3.1.1 `GameObject()` [1/2]

```
GameObject::GameObject ( )
```

#### 5.3.1.2 `GameObject()` [2/2]

```
GameObject::GameObject (
    glm::vec2 pos,
    glm::vec2 size,
    Texture2D sprite,
    glm::vec3 color = glm::vec3(1.0f),
    glm::vec2 velocity = glm::vec2(0.0f, 0.0f) )
```

### 5.3.2 Member Function Documentation

#### 5.3.2.1 `Draw()`

```
void GameObject::Draw (
    SpriteRenderer & renderer ) [virtual]
```

### 5.3.3 Member Data Documentation

#### 5.3.3.1 `Color`

```
glm::vec3 GameObject::Color
```

#### 5.3.3.2 `Destroyed`

```
bool GameObject::Destroyed
```

#### 5.3.3.3 `IsSolid`

```
bool GameObject::IsSolid
```

#### 5.3.3.4 Position

```
glm::vec2 GameObject::Position
```

#### 5.3.3.5 Rotation

```
float GameObject::Rotation
```

#### 5.3.3.6 Size

```
glm::vec2 GameObject::Size
```

#### 5.3.3.7 Sprite

```
Texture2D GameObject::Sprite
```

#### 5.3.3.8 Velocity

```
glm::vec2 GameObject::Velocity
```

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

- [include/game\\_object.h](#)
- [src/game\\_object.cpp](#)

## 5.4 ResourceManager Class Reference

```
#include <resource_manager.h>
```

### Static Public Member Functions

- static [Shader LoadShader](#) (const char \*vShaderFile, const char \*fShaderFile, const char \*gShaderFile, std::string name)
- static [Shader GetShader](#) (std::string name)
- static [Texture2D LoadTexture](#) (const char \*file, bool alpha, std::string name)
- static [Texture2D GetTexture](#) (std::string name)
- static void [Clear](#) ()

## Static Public Attributes

- static std::map< std::string, [Shader](#) > [Shaders](#)
- static std::map< std::string, [Texture2D](#) > [Textures](#)

## 5.4.1 Member Function Documentation

### 5.4.1.1 Clear()

```
void ResourceManager::Clear ( ) [static]
```

### 5.4.1.2 GetShader()

```
Shader ResourceManager::GetShader (
    std::string name ) [static]
```

### 5.4.1.3 GetTexture()

```
Texture2D ResourceManager::GetTexture (
    std::string name ) [static]
```

### 5.4.1.4 LoadShader()

```
Shader ResourceManager::LoadShader (
    const char * vShaderFile,
    const char * fShaderFile,
    const char * gShaderFile,
    std::string name ) [static]
```

### 5.4.1.5 LoadTexture()

```
Texture2D ResourceManager::LoadTexture (
    const char * file,
    bool alpha,
    std::string name ) [static]
```

## 5.4.2 Member Data Documentation

### 5.4.2.1 Shaders

```
std::map< std::string, Shader > ResourceManager::Shaders [static]
```

### 5.4.2.2 Textures

```
std::map< std::string, Texture2D > ResourceManager::Textures [static]
```

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

- include/resource\_manager.h
- src/resource\_manager.cpp

## 5.5 Shader Class Reference

```
#include <shader.h>
```

### Public Member Functions

- [Shader](#) ()
- [Shader & Use](#) ()
- void [Compile](#) (const char \*vertexSource, const char \*fragmentSource, const char \*geometrySource=nullptr)
- void [SetFloat](#) (const char \*name, float value, bool useShader=false)
- void [SetInteger](#) (const char \*name, int value, bool useShader=false)
- void [SetVector2f](#) (const char \*name, float x, float y, bool useShader=false)
- void [SetVector2f](#) (const char \*name, const glm::vec2 &value, bool useShader=false)
- void [SetVector3f](#) (const char \*name, float x, float y, float z, bool useShader=false)
- void [SetVector3f](#) (const char \*name, const glm::vec3 &value, bool useShader=false)
- void [SetVector4f](#) (const char \*name, float x, float y, float z, float w, bool useShader=false)
- void [SetVector4f](#) (const char \*name, const glm::vec4 &value, bool useShader=false)
- void [SetMatrix4](#) (const char \*name, const glm::mat4 &matrix, bool useShader=false)
- [Shader](#) (const char \*vertexPath, const char \*fragmentPath)
- void [use](#) ()
- void [setBool](#) (const std::string &name, bool value) const
- void [setInt](#) (const std::string &name, int value) const
- void [setFloat](#) (const std::string &name, float value) const

### Public Attributes

- unsigned int [ID](#)

## 5.5.1 Constructor & Destructor Documentation

### 5.5.1.1 Shader() [1/2]

```
Shader::Shader ( ) [inline]
```

### 5.5.1.2 Shader() [2/2]

```
Shader::Shader (
    const char * vertexPath,
    const char * fragmentPath ) [inline]
```

## 5.5.2 Member Function Documentation

### 5.5.2.1 Compile()

```
void Shader::Compile (
    const char * vertexSource,
    const char * fragmentSource,
    const char * geometrySource = nullptr )
```

### 5.5.2.2 setBool()

```
void Shader::setBool (
    const std::string & name,
    bool value ) const [inline]
```

### 5.5.2.3 SetFloat()

```
void Shader::SetFloat (
    const char * name,
    float value,
    bool useShader = false )
```



#### 5.5.2.4 setFloat()

```
void Shader::setFloat (
    const std::string & name,
    float value ) const [inline]
```

#### 5.5.2.5 setInt()

```
void Shader::setInt (
    const std::string & name,
    int value ) const [inline]
```

#### 5.5.2.6 SetInteger()

```
void Shader::SetInteger (
    const char * name,
    int value,
    bool useShader = false )
```

#### 5.5.2.7 SetMatrix4()

```
void Shader::SetMatrix4 (
    const char * name,
    const glm::mat4 & matrix,
    bool useShader = false )
```

#### 5.5.2.8 SetVector2f() [1/2]

```
void Shader::SetVector2f (
    const char * name,
    const glm::vec2 & value,
    bool useShader = false )
```

#### 5.5.2.9 SetVector2f() [2/2]

```
void Shader::SetVector2f (
    const char * name,
    float x,
    float y,
    bool useShader = false )
```

#### 5.5.2.10 SetVector3f() [1/2]

```
void Shader::SetVector3f (
    const char * name,
    const glm::vec3 & value,
    bool useShader = false )
```

#### 5.5.2.11 SetVector3f() [2/2]

```
void Shader::SetVector3f (
    const char * name,
    float x,
    float y,
    float z,
    bool useShader = false )
```

#### 5.5.2.12 SetVector4f() [1/2]

```
void Shader::SetVector4f (
    const char * name,
    const glm::vec4 & value,
    bool useShader = false )
```

#### 5.5.2.13 SetVector4f() [2/2]

```
void Shader::SetVector4f (
    const char * name,
    float x,
    float y,
    float z,
    float w,
    bool useShader = false )
```

#### 5.5.2.14 Use()

```
Shader & Shader::Use ( )
```

#### 5.5.2.15 use()

```
void Shader::use ( ) [inline]
```

### 5.5.3 Member Data Documentation

#### 5.5.3.1 ID

```
unsigned int Shader::ID
```

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

- include/[shader.h](#)
- include/[shader\\_s.h](#)
- src/[shader.cpp](#)

## 5.6 SpriteRenderer Class Reference

```
#include <sprite_renderer.h>
```

### Public Member Functions

- [SpriteRenderer](#) (const [Shader](#) &shader)
- [~SpriteRenderer](#) ()
- void [DrawSprite](#) (const [Texture2D](#) &texture, glm::vec2 position, glm::vec2 size=glm::vec2(10.0f, 10.0f), float rotate=0.0f, glm::vec3 color=glm::vec3(1.0f))

### 5.6.1 Constructor & Destructor Documentation

#### 5.6.1.1 SpriteRenderer()

```
SpriteRenderer::SpriteRenderer (  
    const Shader & shader )
```

#### 5.6.1.2 ~SpriteRenderer()

```
SpriteRenderer::~~SpriteRenderer ( )
```

### 5.6.2 Member Function Documentation

### 5.6.2.1 DrawSprite()

```
void SpriteRenderer::DrawSprite (
    const Texture2D & texture,
    glm::vec2 position,
    glm::vec2 size = glm::vec2(10.0f, 10.0f),
    float rotate = 0.0f,
    glm::vec3 color = glm::vec3(1.0f) )
```

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

- [include/sprite\\_renderer.h](#)
- [src/sprite\\_renderer.cpp](#)

## 5.7 Texture2D Class Reference

```
#include <texture.h>
```

### Public Member Functions

- [Texture2D](#) ()
- void [Generate](#) (unsigned int width, unsigned int height, unsigned char \*data)
- void [Bind](#) () const

### Public Attributes

- unsigned int [ID](#)
- unsigned int [Width](#)
- unsigned int [Height](#)
- unsigned int [Internal\\_Format](#)
- unsigned int [Image\\_Format](#)
- unsigned int [Wrap\\_S](#)
- unsigned int [Wrap\\_T](#)
- unsigned int [Filter\\_Min](#)
- unsigned int [Filter\\_Max](#)

### 5.7.1 Constructor & Destructor Documentation

#### 5.7.1.1 Texture2D()

```
Texture2D::Texture2D ( )
```

## 5.7.2 Member Function Documentation

### 5.7.2.1 Bind()

```
void Texture2D::Bind ( ) const
```

### 5.7.2.2 Generate()

```
void Texture2D::Generate (
    unsigned int width,
    unsigned int height,
    unsigned char * data )
```

## 5.7.3 Member Data Documentation

### 5.7.3.1 Filter\_Max

```
unsigned int Texture2D::Filter_Max
```

### 5.7.3.2 Filter\_Min

```
unsigned int Texture2D::Filter_Min
```

### 5.7.3.3 Height

```
unsigned int Texture2D::Height
```

### 5.7.3.4 ID

```
unsigned int Texture2D::ID
```

#### 5.7.3.5 Image\_Format

```
unsigned int Texture2D::Image_Format
```

#### 5.7.3.6 Internal\_Format

```
unsigned int Texture2D::Internal_Format
```

#### 5.7.3.7 Width

```
unsigned int Texture2D::Width
```

#### 5.7.3.8 Wrap\_S

```
unsigned int Texture2D::Wrap_S
```

#### 5.7.3.9 Wrap\_T

```
unsigned int Texture2D::Wrap_T
```

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

- [include/texture.h](#)
- [src/texture.cpp](#)

## Chapter 6

# File Documentation

### 6.1 include/game.h File Reference

```
#include <glad/glad.h>
#include <GLFW/glfw3.h>
```

#### Classes

- class [Game](#)

#### Enumerations

- enum [GameState](#) { [GAME\\_ACTIVE](#) , [GAME\\_MENU](#) , [GAME\\_WIN](#) }

#### 6.1.1 Enumeration Type Documentation

##### 6.1.1.1 GameState

enum [GameState](#)

##### Enumerator

GAME_ACTIVE	
GAME_MENU	
GAME_WIN	

## 6.2 game.h

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

```
1 #ifndef GAME_H
2 #define GAME_H
3
4 #include <glad/glad.h>
5 #include <GLFW/glfw3.h>
6
7 // Represents the current state of the game
8 enum GameState {
9     GAME_ACTIVE,
10    GAME_MENU,
11    GAME_WIN
12 };
13
14 // Game holds all game-related state and functionality.
15 // Combines all game-related data into a single class for
16 // easy access to each of the components and manageability.
17 class Game
18 {
19 public:
20     // game state
21     GameState          State;
22     bool               Keys[1024];
23     unsigned int       Width, Height;
24     // constructor/destructor
25     Game(unsigned int width, unsigned int height);
26     ~Game();
27     // initialize game state (load all shaders/textures/levels)
28     void Init();
29     // game loop
30     void ProcessInput(float dt);
31     void Update(float dt);
32     void Render();
33
34     void DoCollisions();
35     // reset
36     void ResetPlayers();
37 };
38
39 #endif
```

## 6.3 include/game\_object.h File Reference

```
#include <glad/glad.h>
#include <glm/glm.hpp>
#include "texture.h"
#include "sprite_renderer.h"
```

### Classes

- class [GameObject](#)
- class [BallObject](#)

## 6.4 game\_object.h

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

```
1 #ifndef GAMEOBJECT_H
2 #define GAMEOBJECT_H
3
4 #include <glad/glad.h>
5 #include <glm/glm.hpp>
6
7 #include "texture.h"
```



```

8 #include "sprite_renderer.h"
9
10
11 /* GAMEOBJECT CLASS*/
12 // Container object for holding all state relevant for a single
13 // game object entity. Each object in the game likely needs the
14 // minimal of state as described within GameObject.
15 class GameObject
16 {
17 public:
18     // object state
19     glm::vec2    Position, Size, Velocity;
20     glm::vec3    Color;
21     float        Rotation;
22     bool         IsSolid;
23     bool         Destroyed;
24     // render state
25     Texture2D    Sprite;
26     // constructor(s)
27     GameObject();
28     GameObject(glm::vec2 pos, glm::vec2 size, Texture2D sprite, glm::vec3 color = glm::vec3(1.0f),
29               glm::vec2 velocity = glm::vec2(0.0f, 0.0f));
30     // draw sprite
31     virtual void Draw(SpriteRenderer &renderer);
32 };
33
34 /* BALLOBJECT CLASS*/
35 class BallObject : public GameObject
36 {
37 public:
38     //ball state
39     float Radius;
40     bool Stuck;
41
42     BallObject();
43     BallObject(glm::vec2 pos, float radius, glm::vec2 velocity, Texture2D sprite);
44
45     glm::vec2 Move(float dt, unsigned int window_width, unsigned int window_height);
46     void      Reset(glm::vec2 position, glm::vec2 velocity);
47 };
48 #endif

```

## 6.5 include/resource\_manager.h File Reference

```

#include <map>
#include <string>
#include <glad/glad.h>
#include "texture.h"
#include "shader.h"

```

### Classes

- class [ResourceManager](#)

## 6.6 resource\_manager.h

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

```

1 #ifndef RESOURCE_MANAGER_H
2 #define RESOURCE_MANAGER_H
3
4 #include <map>
5 #include <string>
6
7 #include <glad/glad.h>
8
9 #include "texture.h"
10 #include "shader.h"

```

```

11
12
13 // A static singleton ResourceManager class that hosts several
14 // functions to load Textures and Shaders. Each loaded texture
15 // and/or shader is also stored for future reference by string
16 // handles. All functions and resources are static and no
17 // public constructor is defined.
18 class ResourceManager
19 {
20 public:
21     // resource storage
22     static std::map<std::string, Shader>    Shaders;
23     static std::map<std::string, Texture2D> Textures;
24     // loads (and generates) a shader program from file loading vertex, fragment (and geometry) shader's
25     // source code. If gShaderFile is not nullptr, it also loads a geometry shader
26     static Shader    LoadShader(const char *vShaderFile, const char *fShaderFile, const char
27     *gShaderFile, std::string name);
28     // retrieves a stored sader
29     static Shader    GetShader(std::string name);
30     // loads (and generates) a texture from file
31     static Texture2D LoadTexture(const char *file, bool alpha, std::string name);
32     // retrieves a stored texture
33     static Texture2D GetTexture(std::string name);
34     // properly de-allocates all loaded resources
35     static void      Clear();
36 private:
37     // private constructor, that is we do not want any actual resource manager objects. Its members and
38     // functions should be publicly available (static).
39     ResourceManager() { }
40     // loads and generates a shader from file
41     static Shader    loadShaderFromFile(const char *vShaderFile, const char *fShaderFile, const char
42     *gShaderFile = nullptr);
43     // loads a single texture from file
44     static Texture2D loadTextureFromFile(const char *file, bool alpha);
45 };
46 #endif

```

## 6.7 include/shader.h File Reference

```

#include <string>
#include <glad/glad.h>
#include <glm/glm.hpp>
#include <glm/gtc/type_ptr.hpp>

```

### Classes

- class [Shader](#)

## 6.8 shader.h

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

```

1 #ifndef SHADER_H
2 #define SHADER_H
3
4 #include <string>
5
6 #include <glad/glad.h>
7 #include <glm/glm.hpp>
8 #include <glm/gtc/type_ptr.hpp>
9
10
11 // General purpsoe shader object. Compiles from file, generates
12 // compile/link-time error messages and hosts several utility
13 // functions for easy management.
14 class Shader
15 {
16 public:

```

```

17     // state
18     unsigned int ID;
19     // constructor
20     Shader() { }
21     // sets the current shader as active
22     Shader &Use();
23     // compiles the shader from given source code
24     void Compile(const char *vertexSource, const char *fragmentSource, const char *geometrySource =
25         nullptr); // note: geometry source code is optional
26     // utility functions
27     void SetFloat (const char *name, float value, bool useShader = false);
28     void SetInteger (const char *name, int value, bool useShader = false);
29     void SetVector2f (const char *name, float x, float y, bool useShader = false);
30     void SetVector2f (const char *name, const glm::vec2 &value, bool useShader = false);
31     void SetVector3f (const char *name, float x, float y, float z, bool useShader = false);
32     void SetVector3f (const char *name, const glm::vec3 &value, bool useShader = false);
33     void SetVector4f (const char *name, float x, float y, float z, float w, bool useShader = false);
34     void SetVector4f (const char *name, const glm::vec4 &value, bool useShader = false);
35     void SetMatrix4 (const char *name, const glm::mat4 &matrix, bool useShader = false);
36 private:
37     // checks if compilation or linking failed and if so, print the error logs
38     void checkCompileErrors(unsigned int object, std::string type);
39 };
40 #endif

```

## 6.9 include/shader\_s.h File Reference

```

#include <glad/glad.h>
#include <string>
#include <fstream>
#include <sstream>
#include <iostream>

```

### Classes

- class [Shader](#)

## 6.10 shader\_s.h

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

```

1 #ifndef SHADER_H
2 #define SHADER_H
3
4 #include <glad/glad.h>
5
6 #include <string>
7 #include <fstream>
8 #include <sstream>
9 #include <iostream>
10
11 class Shader
12 {
13 public:
14     unsigned int ID;
15     // constructor generates the shader on the fly
16     // -----
17     Shader(const char* vertexPath, const char* fragmentPath)
18     {
19         // 1. retrieve the vertex/fragment source code from filePath
20         std::string vertexCode;
21         std::string fragmentCode;
22         std::ifstream vShaderFile;
23         std::ifstream fShaderFile;
24         // ensure ifstream objects can throw exceptions:
25         vShaderFile.exceptions (std::ifstream::failbit | std::ifstream::badbit);
26         fShaderFile.exceptions (std::ifstream::failbit | std::ifstream::badbit);
27         try

```

```

28     {
29         // open files
30         vShaderFile.open(vertexPath);
31         fShaderFile.open(fragmentPath);
32         std::stringstream vShaderStream, fShaderStream;
33         // read file's buffer contents into streams
34         vShaderStream << vShaderFile.rdbuf();
35         fShaderStream << fShaderFile.rdbuf();
36         // close file handlers
37         vShaderFile.close();
38         fShaderFile.close();
39         // convert stream into string
40         vertexCode = vShaderStream.str();
41         fragmentCode = fShaderStream.str();
42     }
43     catch (std::ifstream::failure& e)
44     {
45         std::cout << "ERROR::SHADER::FILE_NOT_SUCCESFULLY_READ: " << e.what() << std::endl;
46     }
47     const char* vShaderCode = vertexCode.c_str();
48     const char * fShaderCode = fragmentCode.c_str();
49     // 2. compile shaders
50     unsigned int vertex, fragment;
51     // vertex shader
52     vertex = glCreateShader(GL_VERTEX_SHADER);
53     glShaderSource(vertex, 1, &vShaderCode, NULL);
54     glCompileShader(vertex);
55     checkCompileErrors(vertex, "VERTEX");
56     // fragment Shader
57     fragment = glCreateShader(GL_FRAGMENT_SHADER);
58     glShaderSource(fragment, 1, &fShaderCode, NULL);
59     glCompileShader(fragment);
60     checkCompileErrors(fragment, "FRAGMENT");
61     // shader Program
62     ID = glCreateProgram();
63     glAttachShader(ID, vertex);
64     glAttachShader(ID, fragment);
65     glLinkProgram(ID);
66     checkCompileErrors(ID, "PROGRAM");
67     // delete the shaders as they're linked into our program now and no longer necessary
68     glDeleteShader(vertex);
69     glDeleteShader(fragment);
70 }
71 // activate the shader
72 // -----
73 void use()
74 {
75     glUseProgram(ID);
76 }
77 // utility uniform functions
78 // -----
79 void setBool(const std::string &name, bool value) const
80 {
81     glUniform1i(glGetUniformLocation(ID, name.c_str()), (int)value);
82 }
83 // -----
84 void setInt(const std::string &name, int value) const
85 {
86     glUniform1i(glGetUniformLocation(ID, name.c_str()), value);
87 }
88 // -----
89 void setFloat(const std::string &name, float value) const
90 {
91     glUniform1f(glGetUniformLocation(ID, name.c_str()), value);
92 }
93
94 private:
95     // utility function for checking shader compilation/linking errors.
96     // -----
97     void checkCompileErrors(unsigned int shader, std::string type)
98     {
99         int success;
100         char infoLog[1024];
101         if (type != "PROGRAM")
102         {
103             glGetShaderiv(shader, GL_COMPILE_STATUS, &success);
104             if (!success)
105             {
106                 glGetShaderInfoLog(shader, 1024, NULL, infoLog);
107                 std::cout << "ERROR::SHADER_COMPILATION_ERROR of type: " << type << "\n" << infoLog << "\n --
----- " << std::endl;
108             }
109         }
110         else
111         {
112             glGetProgramiv(shader, GL_LINK_STATUS, &success);
113             if (!success)

```

```

114         {
115             glGetProgramInfoLog(shader, 1024, NULL, infoLog);
116             std::cout << "ERROR::PROGRAM_LINKING_ERROR of type: " << type << "\n" << infoLog << "\n --
-----" << std::endl;
117         }
118     }
119 }
120 };
121 #endif

```

## 6.11 include/sprite\_renderer.h File Reference

```

#include <glad/glad.h>
#include <glm/glm.hpp>
#include <glm/gtc/matrix_transform.hpp>
#include "texture.h"
#include "shader.h"

```

### Classes

- class [SpriteRenderer](#)

## 6.12 sprite\_renderer.h

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

```

1 #ifndef SPRITE_RENDERER_H
2 #define SPRITE_RENDERER_H
3
4 #include <glad/glad.h>
5 #include <glm/glm.hpp>
6 #include <glm/gtc/matrix_transform.hpp>
7
8 #include "texture.h"
9 #include "shader.h"
10
11
12 class SpriteRenderer
13 {
14 public:
15     // Constructor (inits shaders/shapes)
16     SpriteRenderer(const Shader &shader);
17     // Destructor
18     ~SpriteRenderer();
19     // Renders a defined quad textured with given sprite
20     void DrawSprite(const Texture2D &texture, glm::vec2 position, glm::vec2 size = glm::vec2(10.0f,
10.0f), float rotate = 0.0f, glm::vec3 color = glm::vec3(1.0f));
21 private:
22     // Render state
23     Shader shader;
24     unsigned int quadVAO;
25     // Initializes and configures the quad's buffer and vertex attributes
26     void initRenderData();
27 };
28
29 #endif

```

## 6.13 include/texture.h File Reference

```

#include <glad/glad.h>

```

## Classes

- class [Texture2D](#)

## 6.14 texture.h

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

```
1 #ifndef TEXTURE_H
2 #define TEXTURE_H
3
4 #include <glad/glad.h>
5
6 // Texture2D is able to store and configure a texture in OpenGL.
7 // It also hosts utility functions for easy management.
8 class Texture2D
9 {
10 public:
11     // holds the ID of the texture object, used for all texture operations to reference to this particular
12     // texture
13     unsigned int ID;
14     // texture image dimensions
15     unsigned int Width, Height; // width and height of loaded image in pixels
16     // texture Format
17     unsigned int Internal_Format; // format of texture object
18     unsigned int Image_Format; // format of loaded image
19     // texture configuration
20     unsigned int Wrap_S; // wrapping mode on S axis
21     unsigned int Wrap_T; // wrapping mode on T axis
22     unsigned int Filter_Min; // filtering mode if texture pixels < screen pixels
23     unsigned int Filter_Max; // filtering mode if texture pixels > screen pixels
24     // constructor (sets default texture modes)
25     Texture2D();
26     // generates texture from image data
27     void Generate(unsigned int width, unsigned int height, unsigned char* data);
28     // binds the texture as the current active GL_TEXTURE_2D texture object
29     void Bind() const;
30 };
31 #endif
```

## 6.15 README.md File Reference

## 6.16 src/game.cpp File Reference

```
#include "game.h"
#include "resource_manager.h"
#include "sprite_renderer.h"
#include "game_object.h"
```

## Typedefs

- typedef std::tuple< bool, [Direction](#), glm::vec2 > [Collision](#)

## Enumerations

- enum [Direction](#) { [UP](#) , [RIGHT](#) , [DOWN](#) , [LEFT](#) }

## Functions

- `const glm::vec2 INITIAL_BALL_VELOCITY` (100.0f, -350.0f)
- `const float PLAYER_VELOCITY` (500.0f)
- `const glm::vec2 PLAYER_SIZE` (100.0f, 200.0f)
- `bool CheckCollision` (`GameObject` &one, `GameObject` &two)
- `Collision CheckCollision` (`BallObject` &one, `GameObject` &two)
- `Direction VectorDirection` (`glm::vec2` closest)

## Variables

- `SpriteRenderer * Renderer`
- `const float BALL_RADIUS` = 25.0f
- `BallObject * Ball`
- `GameObject * Player1`
- `GameObject * Player2`

### 6.16.1 Typedef Documentation

#### 6.16.1.1 Collision

```
typedef std::tuple<bool, Direction, glm::vec2> Collision
```

### 6.16.2 Enumeration Type Documentation

#### 6.16.2.1 Direction

```
enum Direction
```

Enumerator

UP	
RIGHT	
DOWN	
LEFT	

### 6.16.3 Function Documentation

#### 6.16.3.1 CheckCollision() [1/2]

```
Collision CheckCollision (
    BallObject & one,
    GameObject & two )
```

#### 6.16.3.2 CheckCollision() [2/2]

```
bool CheckCollision (
    GameObject & one,
    GameObject & two )
```

#### 6.16.3.3 INITIAL\_BALL\_VELOCITY()

```
const glm::vec2 INITIAL_BALL_VELOCITY (
    100.  0f,
    -350. 0f )
```

#### 6.16.3.4 PLAYER\_SIZE()

```
const glm::vec2 PLAYER_SIZE (
    100.  0f,
    200. 0f )
```

#### 6.16.3.5 PLAYER\_VELOCITY()

```
const float PLAYER_VELOCITY (
    500.  0f )
```

#### 6.16.3.6 VectorDirection()

```
Direction VectorDirection (
    glm::vec2 closest )
```

### 6.16.4 Variable Documentation



#### 6.16.4.1 Ball

```
BallObject* Ball
```

#### 6.16.4.2 BALL\_RADIUS

```
const float BALL_RADIUS = 25.0f
```

#### 6.16.4.3 Player1

```
GameObject* Player1
```

#### 6.16.4.4 Player2

```
GameObject* Player2
```

#### 6.16.4.5 Renderer

```
SpriteRenderer* Renderer
```

### 6.17 src/game\_object.cpp File Reference

```
#include "game_object.h"
```

### 6.18 src/main.cpp File Reference

```
#include <glad/glad.h>
#include <GLFW/glfw3.h>
#include "game.h"
#include "resource_manager.h"
#include <iostream>
```

## Functions

- void [framebuffer\\_size\\_callback](#) (GLFWwindow \*window, int width, int height)
- void [key\\_callback](#) (GLFWwindow \*window, int key, int scancode, int action, int mode)
- int [main](#) (int argc, char \*argv[])

## Variables

- const unsigned int [SCREEN\\_WIDTH](#) = 800
- const unsigned int [SCREEN\\_HEIGHT](#) = 600
- [Game Headsoccer](#) ([SCREEN\\_WIDTH](#), [SCREEN\\_HEIGHT](#))

## 6.18.1 Function Documentation

### 6.18.1.1 framebuffer\_size\_callback()

```
void framebuffer_size_callback (
    GLFWwindow * window,
    int width,
    int height )
```

### 6.18.1.2 key\_callback()

```
void key_callback (
    GLFWwindow * window,
    int key,
    int scancode,
    int action,
    int mode )
```

### 6.18.1.3 main()

```
int main (
    int argc,
    char * argv[] )
```

## 6.18.2 Variable Documentation

### 6.18.2.1 Headsoccer

```
Game Headsoccer(SCREEN_WIDTH, SCREEN_HEIGHT) (  
    SCREEN_WIDTH ,  
    SCREEN_HEIGHT )
```

### 6.18.2.2 SCREEN\_HEIGHT

```
const unsigned int SCREEN_HEIGHT = 600
```

### 6.18.2.3 SCREEN\_WIDTH

```
const unsigned int SCREEN_WIDTH = 800
```

## 6.19 src/resource\_manager.cpp File Reference

```
#include "resource_manager.h"  
#include <iostream>  
#include <sstream>  
#include <fstream>  
#include "stb_image.h"
```

## 6.20 src/shader.cpp File Reference

```
#include "shader.h"  
#include <iostream>
```

## 6.21 src/sprite\_renderer.cpp File Reference

```
#include "sprite_renderer.h"
```

## 6.22 src/texture.cpp File Reference

```
#include <iostream>  
#include "texture.h"
```



# Index

- ~Game
  - Game, [11](#)
- ~SpriteRenderer
  - SpriteRenderer, [21](#)
- Ball
  - game.cpp, [34](#)
- BALL\_RADIUS
  - game.cpp, [35](#)
- BallObject, [9](#)
  - BallObject, [9](#)
  - Move, [10](#)
  - Radius, [10](#)
  - Reset, [10](#)
  - Stuck, [10](#)
- Bind
  - Texture2D, [23](#)
- CheckCollision
  - game.cpp, [33](#), [34](#)
- Clear
  - ResourceManager, [16](#)
- Collision
  - game.cpp, [33](#)
- Color
  - GameObject, [14](#)
- Compile
  - Shader, [18](#)
- Destroyed
  - GameObject, [14](#)
- Direction
  - game.cpp, [33](#)
- DoCollisions
  - Game, [11](#)
- DOWN
  - game.cpp, [33](#)
- Draw
  - GameObject, [14](#)
- DrawSprite
  - SpriteRenderer, [21](#)
- Filter\_Max
  - Texture2D, [23](#)
- Filter\_Min
  - Texture2D, [23](#)
- framebuffer\_size\_callback
  - main.cpp, [36](#)
- Game, [11](#)
  - ~Game, [11](#)
- DoCollisions, [11](#)
- Game, [11](#)
- Height, [12](#)
- Init, [11](#)
- Keys, [12](#)
- ProcessInput, [12](#)
- Render, [12](#)
- ResetPlayers, [12](#)
- State, [12](#)
- Update, [12](#)
- Width, [13](#)
- game.cpp
  - Ball, [34](#)
  - BALL\_RADIUS, [35](#)
  - CheckCollision, [33](#), [34](#)
  - Collision, [33](#)
  - Direction, [33](#)
  - DOWN, [33](#)
  - INITIAL\_BALL\_VELOCITY, [34](#)
  - LEFT, [33](#)
  - Player1, [35](#)
  - Player2, [35](#)
  - PLAYER\_SIZE, [34](#)
  - PLAYER\_VELOCITY, [34](#)
  - Renderer, [35](#)
  - RIGHT, [33](#)
  - UP, [33](#)
  - VectorDirection, [34](#)
- game.h
  - GAME\_ACTIVE, [25](#)
  - GAME\_MENU, [25](#)
  - GAME\_WIN, [25](#)
  - GameState, [25](#)
- GAME\_ACTIVE
  - game.h, [25](#)
- GAME\_MENU
  - game.h, [25](#)
- GAME\_WIN
  - game.h, [25](#)
- GameObject, [13](#)
  - Color, [14](#)
  - Destroyed, [14](#)
  - Draw, [14](#)
  - GameObject, [13](#), [14](#)
  - IsSolid, [14](#)
  - Position, [14](#)
  - Rotation, [15](#)
  - Size, [15](#)
  - Sprite, [15](#)

- Velocity, 15
- GameState
  - game.h, 25
- Generate
  - Texture2D, 23
- GetShader
  - ResourceManager, 16
- GetTexture
  - ResourceManager, 16
- Headsoccer
  - main.cpp, 36
- Height
  - Game, 12
  - Texture2D, 23
- ID
  - Shader, 21
  - Texture2D, 23
- Image\_Format
  - Texture2D, 23
- include/game.h, 25, 26
- include/game\_object.h, 26
- include/resource\_manager.h, 27
- include/shader.h, 28
- include/shader\_s.h, 29
- include/sprite\_renderer.h, 31
- include/texture.h, 31, 32
- Init
  - Game, 11
- INITIAL\_BALL\_VELOCITY
  - game.cpp, 34
- Internal\_Format
  - Texture2D, 24
- IsSolid
  - GameObject, 14
- key\_callback
  - main.cpp, 36
- Keys
  - Game, 12
- LEFT
  - game.cpp, 33
- LoadShader
  - ResourceManager, 16
- LoadTexture
  - ResourceManager, 16
- main
  - main.cpp, 36
- main.cpp
  - framebuffer\_size\_callback, 36
  - Headsoccer, 36
  - key\_callback, 36
  - main, 36
  - SCREEN\_HEIGHT, 37
  - SCREEN\_WIDTH, 37
- Move
  - BallObject, 10
- Player1
  - game.cpp, 35
- Player2
  - game.cpp, 35
- PLAYER\_SIZE
  - game.cpp, 34
- PLAYER\_VELOCITY
  - game.cpp, 34
- Position
  - GameObject, 14
- ProcessInput
  - Game, 12
- Radius
  - BallObject, 10
- README.md, 32
- Render
  - Game, 12
- Renderer
  - game.cpp, 35
- Reset
  - BallObject, 10
- ResetPlayers
  - Game, 12
- ResourceManager, 15
  - Clear, 16
  - GetShader, 16
  - GetTexture, 16
  - LoadShader, 16
  - LoadTexture, 16
  - Shaders, 17
  - Textures, 17
- RIGHT
  - game.cpp, 33
- Rotation
  - GameObject, 15
- SCREEN\_HEIGHT
  - main.cpp, 37
- SCREEN\_WIDTH
  - main.cpp, 37
- setBool
  - Shader, 18
- SetFloat
  - Shader, 18
- setFloat
  - Shader, 18
- setInt
  - Shader, 19
- SetInteger
  - Shader, 19
- SetMatrix4
  - Shader, 19
- SetVector2f
  - Shader, 19
- SetVector3f
  - Shader, 19, 20

- SetVector4f
  - Shader, [20](#)
- Shader, [17](#)
  - Compile, [18](#)
  - ID, [21](#)
  - setBool, [18](#)
  - SetFloat, [18](#)
  - setFloat, [18](#)
  - setInt, [19](#)
  - SetInteger, [19](#)
  - SetMatrix4, [19](#)
  - SetVector2f, [19](#)
  - SetVector3f, [19](#), [20](#)
  - SetVector4f, [20](#)
  - Shader, [18](#)
  - Use, [20](#)
  - use, [20](#)
- Shaders
  - ResourceManager, [17](#)
- Size
  - GameObject, [15](#)
- Sprite
  - GameObject, [15](#)
- SpriteRenderer, [21](#)
  - ~SpriteRenderer, [21](#)
  - DrawSprite, [21](#)
  - SpriteRenderer, [21](#)
- src/game.cpp, [32](#)
- src/game\_object.cpp, [35](#)
- src/main.cpp, [35](#)
- src/resource\_manager.cpp, [37](#)
- src/shader.cpp, [37](#)
- src/sprite\_renderer.cpp, [37](#)
- src/texture.cpp, [37](#)
- State
  - Game, [12](#)
- Stuck
  - BallObject, [10](#)
- Texture2D, [22](#)
  - Bind, [23](#)
  - Filter\_Max, [23](#)
  - Filter\_Min, [23](#)
  - Generate, [23](#)
  - Height, [23](#)
  - ID, [23](#)
  - Image\_Format, [23](#)
  - Internal\_Format, [24](#)
  - Texture2D, [22](#)
  - Width, [24](#)
  - Wrap\_S, [24](#)
  - Wrap\_T, [24](#)
- Textures
  - ResourceManager, [17](#)
- UP
  - game.cpp, [33](#)
- Update
  - Game, [12](#)
- Use
  - Shader, [20](#)
- use
  - Shader, [20](#)
- VectorDirection
  - game.cpp, [34](#)
- Velocity
  - GameObject, [15](#)
- Width
  - Game, [13](#)
  - Texture2D, [24](#)
- Wrap\_S
  - Texture2D, [24](#)
- Wrap\_T
  - Texture2D, [24](#)