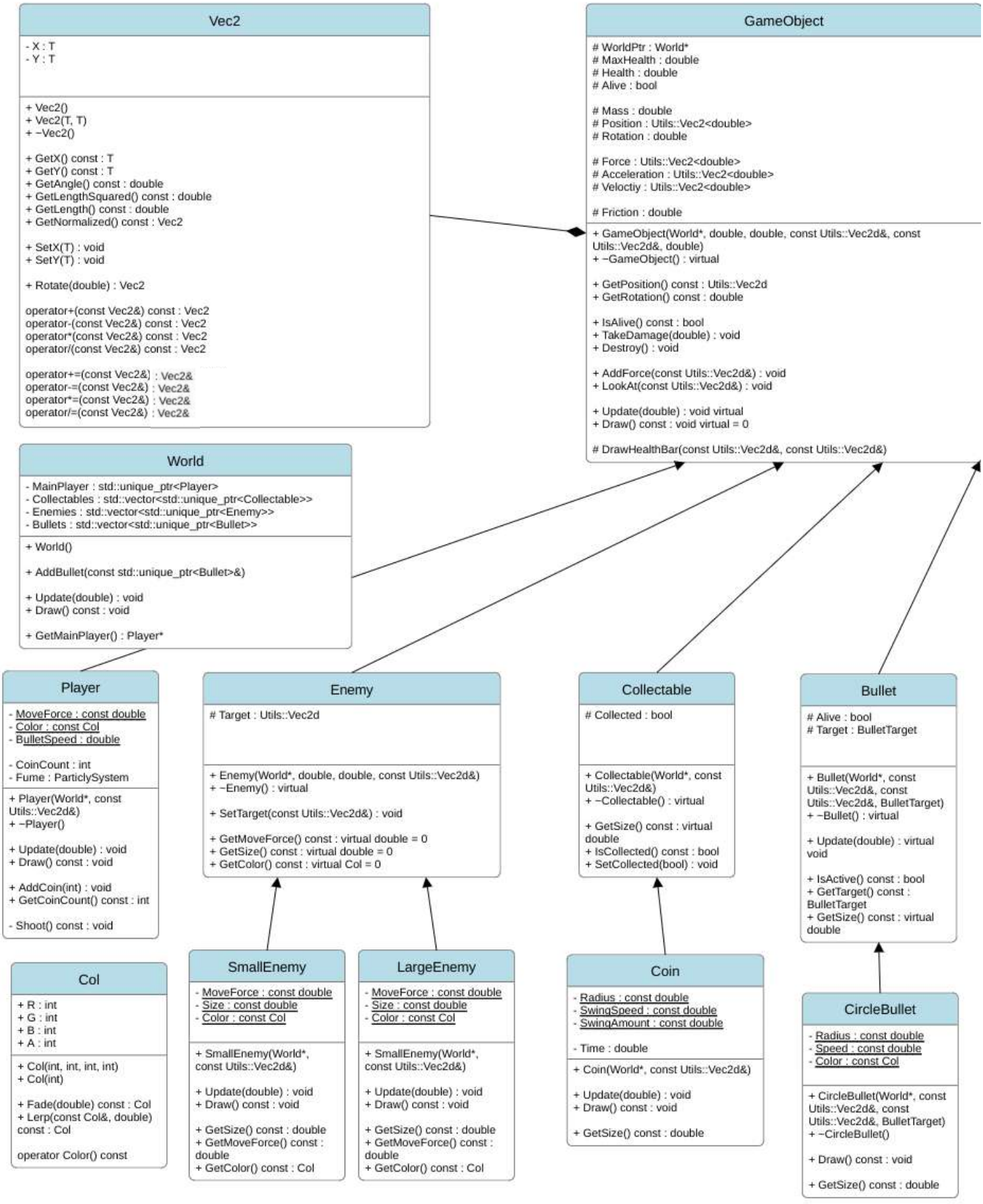
A játék lényege röviden

Egy két dimenziós (grafikus), lövöldözős játékot készítettem.
A játékos célja, hogy minél több ideig éljen, és minél több coin-t szedjen fel. A játék során ellenfelek közelednek a játékos felé, minél több coin-ja van, annál több.
Az ellenfelek úgy tudnak sebezni, hogy nekimennek a játékosnak. A különböző méretű ellenfeleknek eltérő mennyiségű hp-ja van. A felhasználónak menekülnie kell az ellenfelek elől, és le kell lőnie őket. Ha 0 alá csökken a hp-ja, akkor meghal és vége a játéknak.

A játékost a WASD billentyűkkel lehet irányítani, illetve az egérrel lehet forogni és lőni (a játékos az egér felé néz és arra is lő)
Minden akciót valamilyen vizuális visszajelzés kísér (pl. ha felvesz egy coin-t, nő egy számláló, ha lelő egy ellefelet, az eltűnik).


Vec2
- X : T
- Y : T
+ Vec2()
+ Vec2(T, T)
+ ~Vec2()
+ GetX() const : T
+ GetY() const : T
+ GetAngle() const : double
+ GetLengthSquared() const : double
+ GetLength() const : double
+ GetNormalized() const : Vec2
+ SetX(T) : void
+ SetY(T) : void
+ Rotate(double) : Vec2
operator+(const Vec2&) const : Vec2
operator-(const Vec2&) const : Vec2
operator*(const Vec2&) const : Vec2
operator/(const Vec2&) const : Vec2
operator+=(const Vec2&) : Vec2&
operator-=(const Vec2&) : Vec2&
operator*=(const Vec2&) : Vec2&
operator/=(const Vec2&) : Vec2&
GameObject
WorldPtr : World*
MaxHealth : double
Health : double
Alive : bool
Mass : double
Position : Utils::Vec2<double>
Rotation : double
Force : Utils::Vec2<double>
Acceleration : Utils::Vec2<double>
Velocity : Utils::Vec2<double>
Friction : double
+ GameObject(World*, double, double, const Utils::Vec2d&, const Utils::Vec2d&, double)
+ ~GameObject() : virtual
+ GetPosition() const : Utils::Vec2d
+ GetRotation() const : double
+ IsAlive() const : bool
+ TakeDamage(double) : void
+ Destroy() : void
+ AddForce(const Utils::Vec2d&) : void
+ LookAt(const Utils::Vec2d&) : void
+ Update(double) : void virtual
+ Draw() const : void virtual = 0
# DrawHealthBar(const Utils::Vec2d&, const Utils::Vec2d&)
World
- MainPlayer : std::unique_ptr<Player>
- Collectables : std::vector<std::unique_ptr<Collectable>>
- Enemies : std::vector<std::unique_ptr<Enemy>>
- Bullets : std::vector<std::unique_ptr<Bullet>>
+ World()
+ AddBullet(const std::unique_ptr<Bullet>&)
+ Update(double) : void
+ Draw() const : void
+ GetMainPlayer() : Player*
Player
- MoveForce : const double
- Color : const Col
- BulletSpeed : double
- CoinCount : int
- Fume : ParticlySystem
+ Player(World*, const Utils::Vec2d&)
+ ~Player()
+ Update(double) : void
+ Draw() const : void
+ AddCoin(int) : void
+ GetCoinCount() const : int
- Shoot() const : void
Enemy
Target : Utils::Vec2d
+ Enemy(World*, double, double, const Utils::Vec2d&)
+ ~Enemy() : virtual
+ SetTarget(const Utils::Vec2d&) : void
+ GetMoveForce() const : virtual double = 0
+ GetSize() const : virtual double = 0
+ GetColor() const : virtual Col = 0
Collectable
Collected : bool
+ Collectable(World*, const Utils::Vec2d&)
+ ~Collectable() : virtual
+ GetSize() const : virtual double
+ IsCollected() const : bool
+ SetCollected(bool) : void
Bullet
Alive : bool
Target : BulletTarget
+ Bullet(World*, const Utils::Vec2d&, const Utils::Vec2d&, BulletTarget)
+ ~Bullet() : virtual
+ Update(double) : virtual void
+ IsActive() const : bool
+ GetTarget() const : BulletTarget
+ GetSize() const : virtual double
Col
+ R : int
+ G : int
+ B : int
+ A : int
+ Col(int, int, int, int)
+ Col(int)
+ Fade(double) const : Col
+ Lerp(const Col&, double) const : Col
operator Color() const
SmallEnemy
- MoveForce : const double
- Size : const double
- Color : const Col
+ SmallEnemy(World*, const Utils::Vec2d&)
+ Update(double) : void
+ Draw() const : void
+ GetSize() const : double
+ GetMoveForce() const : double
+ GetColor() const : Col
LargeEnemy
- MoveForce : const double
- Size : const double
- Color : const Col
+ SmallEnemy(World*, const Utils::Vec2d&)
+ Update(double) : void
+ Draw() const : void
+ GetSize() const : double
+ GetMoveForce() const : double
+ GetColor() const : Col
Coin
- Radius : const double
- SwingSpeed : const double
- SwingAmount : const double
- Time : double
+ Coin(World*, const Utils::Vec2d&)
+ Update(double) : void
+ Draw() const : void
+ GetSize() const : double
CircleBullet
- Radius : const double
- Speed : const double
- Color : const Col
+ CircleBullet(World*, const Utils::Vec2d&, const Utils::Vec2d&, BulletTarget)
+ ~CircleBullet()
+ Draw() const : void
+ GetSize() const : double
ParticleSystem
- Particle : std::vector<Particle>
+ ParticleSystem()
+ ~ParticleSystem()
+ Emit(const Utils::Vec2d&, const Utils::Vec2d&, double, double, const Col&, const Col&, double, double, int)
+ Update(double) : void
+ Draw() const : void
Particle
+ Position : Utils::Vec2d
+ Velocity : Utils::Vec2d
+ Age : double
+ LifeTime : double
+ StartColor : Col
+ EndColor : Col
+ StartSize : double
+ EndSize : double
+ Particle(const Utils::Vec2d&, const Utils::Vec2d&, double, const Col&, const Col&, double, double)
+ Update(double) : void
+ Draw() const : void

A program felépítése a következő:

-A main file-ban megnyitok egy ablakot Raylibbel, utána létrehozok egy World-öt és amíg nyitva van az ablak, meghívom az Update és Draw metódusait.

-A világ tartalmazza a játékost, az összes Coin-t, Enemy-t és Bullet-ot.

-A játékos figyeli a billentyűzetet és az egeret a mozgáshoz, az Enemy-k pedig automatikusan mozognak a játékos felé. A Coin-ok és a Bullet-ok szintén maguk mozognak.

-Minden objektum, többek között azok is, amiket felsoroltam a GameObject osztályból származik. A GameObject osztály pedig tartalmaz egy World*-t, ami az őt tartalmazó osztályra mutat. Ez azért szükséges, mert ha a játékos lő, akkor létrehoz egy Bullet-ot, és hozzá kell adnia a World-höz, amiben van. Lehet, hogy lenne ennél jobb megoldás is, de azért nincs vészesen sok objektum, szóval egy pointer talán nem nagy gond (viszont nagyon megkönnyíti a dolgom).

-A World.Update update-el mindent (mozgat), ezen felül ellenőrzi a következő ütközéseket:
Player - Collectable, Player - Bullet (ilyen egyébként nincs végül), Enemy - Bullet, Player - Enemy
Ha ütközés történt, törli a megfelelő objektumokat.

-A ParticleSystem egy helyen jelenik meg, méghozzá a játékos hoz létre egy példányt, és amikor mozog, meghívja az Emit metódust, ami kibocsát néhány Particle-t. Természetesen a játékos Update metódusa meghívja a ParticleSystem Update metódusát, az pedig a Particle-ök Update metódusait, így egy idő után a Particle-ök eltűnnek (a ParticleSystem törli a öreg Particle-öket)

-A különböző méretű ellenfelek valóban csak a méretükben és a sebzésükben térnek el (de egy kicsit máshogy is vannak rajzolva), így tényleg valamennyire felesleges az öröklődés, de eredetileg terveztem ennél nagyobb eltérést is

-Nagyjából mindenhez írtam egy rövid tesztet, amelyek GTest-et használnak. A Raylib függőséget sajnos nem nagyon lehet kiküszöbölni, mert a program számos része - még a nem grafikus részek is - használnak Raylib-es függvényeket, és építenek arra, hogy az ablak inicializálva van. Például amikor új ellenfeleket hoz létre a világ, akkor lekérdezi az ablak méretét. (Ha nem lenne inicializálva az ablak, ezt nem tudná megtenni, de használok például ütközés ellenörző függvényeket, illetve Raylib struktúrákat is használok stb.)

-Memóriaszivárgás nincs, ezt az AddressSanitizer segítségével ellenőríztem.

-A CMakeLists.txt-t végül teljesen újraírtam, egy libet hozok létre a kódból, majd ezt linkelem magához a játékhoz, és egy teszt programhoz is. Az assets is átkerült a gyökérbe.

-Ajánlom a shell script-ek használatát (run_game.sh, run_test.sh), ezek lefordítják és le is futtatják az adott programot.

-A program forráskódja fönt van GitHubon: https://github.com/mandliors/Prog2_NHF

Tesztelési dokumentáció

Minden kulcsfontosságú részhez írtam egy rövid tesztet:

-Tesztelem a Raylib inicializálását (ez feltétele annak, hogy a többi teszt hibamentesen lefusson (részletesbb magyarázat a terv.pdf-ben található)
-A Vec2d osztály konstruktorát és néhány metódusát tesztelem
-Létrehozok egy világot, ami létrehoz egy játékost, majd lerakok egy coin-t a játékos fölé. Meghívom a World.Update-et, ekkorra a játékosnak fel kellett szednie a coin-t, és pont ezt tesztelem
-Kilövök egy lövedéket a játékos irányába, update-elem a világot addig, amíg oda nem ér a lövedék, és ellenőrzöm, hogy meghalt-e a játékos
-Majd létrehozok egy ellenfelet, és ugyanúgy járok el, mint a lövedékkel (azaz megvárom, amíg nekimegy a játékosnak, és megnézem, hogy meghalt-e)
-Két Enemy-Bullet collision-t vizsgálok (az egyik igaz, a másik hamis)
-A Col osztály konstruktorait és a Fade, illetve a Lerp metódusokat is tesztelem
-Írtam egy tesztet a ParticleSystem-hez is, ami a Particle-ök számát vizsgálja (kezdetben 3, 1 másodperc múlva pedig 0)

A teszt indításához a run_test.sh script futtatását javaslom, ami lefordítja a teszt programot és futtatja is.

Prog2_NHF

v1.0

Generated by Doxygen 1.10.0

1 Namespace Index 1

1.1 Namespace List . . . 1

2 Hierarchical Index 3

2.1 Class Hierarchy . . . 3

3 Class Index 5

3.1 Class List . . . 5

4 File Index 7

4.1 File List . . . 7

5 Namespace Documentation 9

5.1 AssetManager Namespace Reference . . . 9

5.1.1 Enumeration Type Documentation . . . 9

5.1.1.1 TextureType . . . 9

5.1.2 Function Documentation . . . 9

5.1.2.1 GetTexture() . . . 9

5.1.2.2 LoadAssets() . . . 10

5.1.2.3 UnloadAssets() . . . 10

5.2 Utils Namespace Reference . . . 10

5.2.1 Typedef Documentation . . . 10

5.2.1.1 Vec2d . . . 10

5.2.1.2 Vec2i . . . 10

5.2.2 Function Documentation . . . 10

5.2.2.1 operator<<() . . . 10

5.2.2.2 Vector2ToVec2() . . . 11

6 Class Documentation 13

6.1 Bullet Class Reference . . . 13

6.1.1 Member Enumeration Documentation . . . 13

6.1.1.1 BulletTarget . . . 13

6.1.2 Constructor & Destructor Documentation . . . 14

6.1.2.1 Bullet() . . . 14

6.1.2.2 ~Bullet() . . . 14

6.1.3 Member Function Documentation . . . 14

6.1.3.1 GetSize() . . . 14

6.1.3.2 GetTarget() . . . 14

6.1.3.3 Update() . . . 14

6.1.4 Member Data Documentation . . . 15

6.1.4.1 Target . . . 15

6.2 CircleBullet Class Reference . . . 15

6.2.1 Constructor & Destructor Documentation . . . 15

6.2.1.1 CircleBullet() . . . 15

6.2.2 Member Function Documentation . . . 16
6.2.2.1 Draw() . . . 16
6.2.2.2 GetSize() . . . 16
6.3 Coin Class Reference . . . 16
6.3.1 Constructor & Destructor Documentation . . . 17
6.3.1.1 Coin() . . . 17
6.3.2 Member Function Documentation . . . 17
6.3.2.1 Draw() . . . 17
6.3.2.2 GetSize() . . . 17
6.3.2.3 Update() . . . 18
6.4 Col Struct Reference . . . 18
6.4.1 Constructor & Destructor Documentation . . . 18
6.4.1.1 Col() [1/2] . . . 18
6.4.1.2 Col() [2/2] . . . 19
6.4.2 Member Function Documentation . . . 19
6.4.2.1 Fade() . . . 19
6.4.2.2 Lerp() . . . 19
6.4.2.3 operator Color() . . . 19
6.4.3 Member Data Documentation . . . 20
6.4.3.1 A . . . 20
6.4.3.2 B . . . 20
6.4.3.3 G . . . 20
6.4.3.4 R . . . 20
6.5 Collectable Class Reference . . . 20
6.5.1 Constructor & Destructor Documentation . . . 21
6.5.1.1 Collectable() . . . 21
6.5.1.2 ~Collectable() . . . 21
6.5.2 Member Function Documentation . . . 21
6.5.2.1 GetSize() . . . 21
6.5.2.2 IsCollected() . . . 22
6.5.2.3 SetCollected() . . . 22
6.5.3 Member Data Documentation . . . 22
6.5.3.1 Collected . . . 22
6.6 Enemy Class Reference . . . 22
6.6.1 Constructor & Destructor Documentation . . . 23
6.6.1.1 Enemy() . . . 23
6.6.1.2 ~Enemy() . . . 23
6.6.2 Member Function Documentation . . . 23
6.6.2.1 GetColor() . . . 23
6.6.2.2 GetMoveForce() . . . 24
6.6.2.3 GetSize() . . . 24
6.6.2.4 SetTarget() . . . 24

6.6.3 Member Data Documentation . . . 24
6.6.3.1 Target . . . 24
6.7 GameObject Class Reference . . . 25
6.7.1 Constructor & Destructor Documentation . . . 25
6.7.1.1 GameObject() . . . 25
6.7.1.2 ~GameObject() . . . 25
6.7.2 Member Function Documentation . . . 25
6.7.2.1 AddForce() . . . 25
6.7.2.2 Destroy() . . . 27
6.7.2.3 Draw() . . . 27
6.7.2.4 DrawHealthBar() . . . 27
6.7.2.5 GetPosition() . . . 27
6.7.2.6 GetRotation() . . . 27
6.7.2.7 IsAlive() . . . 28
6.7.2.8 LookAt() . . . 28
6.7.2.9 TakeDamage() . . . 29
6.7.2.10 Update() . . . 29
6.7.3 Member Data Documentation . . . 29
6.7.3.1 Acceleration . . . 29
6.7.3.2 Alive . . . 29
6.7.3.3 Force . . . 29
6.7.3.4 Friction . . . 30
6.7.3.5 Health . . . 30
6.7.3.6 Mass . . . 30
6.7.3.7 MaxHealth . . . 30
6.7.3.8 Position . . . 30
6.7.3.9 Rotation . . . 30
6.7.3.10 Velocity . . . 30
6.7.3.11 WorldPtr . . . 30
6.8 LargeEnemy Class Reference . . . 30
6.8.1 Constructor & Destructor Documentation . . . 31
6.8.1.1 LargeEnemy() . . . 31
6.8.2 Member Function Documentation . . . 31
6.8.2.1 Draw() . . . 31
6.8.2.2 GetColor() . . . 32
6.8.2.3 GetMoveForce() . . . 32
6.8.2.4 GetSize() . . . 32
6.8.2.5 Update() . . . 32
6.9 ParticleSystem Class Reference . . . 33
6.9.1 Constructor & Destructor Documentation . . . 33
6.9.1.1 ParticleSystem() . . . 33
6.9.1.2 ~ParticleSystem() . . . 33

6.9.2 Member Function Documentation . . . 33
6.9.2.1 Draw() . . . 33
6.9.2.2 Emit() . . . 33
6.9.2.3 GetParticleCount() . . . 34
6.9.2.4 Update() . . . 34
6.10 Player Class Reference . . . 34
6.10.1 Constructor & Destructor Documentation . . . 35
6.10.1.1 Player() . . . 35
6.10.1.2 ∼Player() . . . 35
6.10.2 Member Function Documentation . . . 35
6.10.2.1 AddCoin() . . . 35
6.10.2.2 Draw() . . . 36
6.10.2.3 GetCoinCount() . . . 36
6.10.2.4 Update() . . . 36
6.11 SmallEnemy Class Reference . . . 36
6.11.1 Constructor & Destructor Documentation . . . 37
6.11.1.1 SmallEnemy() . . . 37
6.11.2 Member Function Documentation . . . 37
6.11.2.1 Draw() . . . 37
6.11.2.2 GetColor() . . . 37
6.11.2.3 GetMoveForce() . . . 38
6.11.2.4 GetSize() . . . 38
6.11.2.5 Update() . . . 38
6.12 Utils::Vec2< T > Class Template Reference . . . 38
6.12.1 Constructor & Destructor Documentation . . . 39
6.12.1.1 Vec2() [1/2] . . . 39
6.12.1.2 Vec2() [2/2] . . . 39
6.12.1.3 ∼Vec2() . . . 40
6.12.2 Member Function Documentation . . . 40
6.12.2.1 Down() . . . 40
6.12.2.2 FromAngle() . . . 40
6.12.2.3 GetAngle() . . . 40
6.12.2.4 GetLength() . . . 41
6.12.2.5 GetLengthSquared() . . . 41
6.12.2.6 GetNormalized() . . . 41
6.12.2.7 GetX() . . . 41
6.12.2.8 GetY() . . . 42
6.12.2.9 Left() . . . 42
6.12.2.10 One() . . . 42
6.12.2.11 operator Vector2() . . . 42
6.12.2.12 operator∗() [1/2] . . . 42
6.12.2.13 operator∗() [2/2] . . . 43

6.12.2.14 operator*=() [1/2] . . . 43
6.12.2.15 operator*=() [2/2] . . . 43
6.12.2.16 operator+() . . . 44
6.12.2.17 operator+=() . . . 44
6.12.2.18 operator-() [1/2] . . . 44
6.12.2.19 operator-() [2/2] . . . 45
6.12.2.20 operator-=() . . . 45
6.12.2.21 operator/() . . . 45
6.12.2.22 operator/=() [1/2] . . . 46
6.12.2.23 operator/=() [2/2] . . . 46
6.12.2.24 operator^() . . . 46
6.12.2.25 Right() . . . 47
6.12.2.26 Rotate() . . . 47
6.12.2.27 SetX() . . . 47
6.12.2.28 SetY() . . . 47
6.12.2.29 Up() . . . 48
6.12.2.30 Zero() . . . 48
6.13 World Class Reference . . . 48
6.13.1 Constructor & Destructor Documentation . . . 48
6.13.1.1 World() . . . 48
6.13.2 Member Function Documentation . . . 48
6.13.2.1 AddBullet() . . . 48
6.13.2.2 AddCollectable() . . . 49
6.13.2.3 AddEnemy() . . . 49
6.13.2.4 Draw() . . . 49
6.13.2.5 GetMainPlayer() . . . 49
6.13.2.6 Update() . . . 49

7 File Documentation **51**

7.1 src/AssetManager/AssetManager.cpp File Reference . . . 51
7.2 src/AssetManager/AssetManager.h File Reference . . . 51
7.3 AssetManager.h . . . 51
7.4 src/Bullets/Bullet.cpp File Reference . . . 52
7.5 src/Bullets/Bullet.h File Reference . . . 52
7.6 Bullet.h . . . 52
7.7 src/Bullets/CircleBullet.cpp File Reference . . . 52
7.8 src/Bullets/CircleBullet.h File Reference . . . 52
7.9 CircleBullet.h . . . 52
7.10 src/Collectables/Coin.cpp File Reference . . . 53
7.11 src/Collectables/Coin.h File Reference . . . 53
7.12 Coin.h . . . 53
7.13 src/Collectables/Collectable.h File Reference . . . 53

7.14 Collectable.h . . . 54
7.15 src/Collisions/Collisions.cpp File Reference . . . 54
7.15.1 Function Documentation . . . 54
7.15.1.1 CheckCollision() [1/4] . . . 54
7.15.1.2 CheckCollision() [2/4] . . . 54
7.15.1.3 CheckCollision() [3/4] . . . 55
7.15.1.4 CheckCollision() [4/4] . . . 55
7.16 src/Collisions/Collisions.h File Reference . . . 56
7.16.1 Function Documentation . . . 56
7.16.1.1 CheckCollision() [1/4] . . . 56
7.16.1.2 CheckCollision() [2/4] . . . 56
7.16.1.3 CheckCollision() [3/4] . . . 56
7.16.1.4 CheckCollision() [4/4] . . . 57
7.17 Collisions.h . . . 57
7.18 src/Color/Color.cpp File Reference . . . 58
7.19 src/Color/Color.h File Reference . . . 58
7.20 Color.h . . . 58
7.21 src/Enemies/Enemy.h File Reference . . . 58
7.22 Enemy.h . . . 58
7.23 src/Enemies/LargeEnemy.cpp File Reference . . . 59
7.24 src/Enemies/LargeEnemy.h File Reference . . . 59
7.25 LargeEnemy.h . . . 59
7.26 src/Enemies/SmallEnemy.cpp File Reference . . . 59
7.27 src/Enemies/SmallEnemy.h File Reference . . . 59
7.28 SmallEnemy.h . . . 59
7.29 src/GameObjects/GameObject.cpp File Reference . . . 60
7.30 src/GameObjects/GameObject.h File Reference . . . 60
7.31 GameObject.h . . . 60
7.32 src/GameObjects/Player.cpp File Reference . . . 61
7.33 src/GameObjects/Player.h File Reference . . . 61
7.34 Player.h . . . 61
7.35 src/main.cpp File Reference . . . 61
7.35.1 Function Documentation . . . 61
7.35.1.1 main() . . . 61
7.36 src/ParticleSystem/ParticleSystem.cpp File Reference . . . 61
7.37 src/ParticleSystem/ParticleSystem.h File Reference . . . 61
7.38 ParticleSystem.h . . . 62
7.39 src/Utils/Vec2.cpp File Reference . . . 62
7.40 src/Utils/Vec2.h File Reference . . . 62
7.41 Vec2.h . . . 63
7.42 src/World/World.cpp File Reference . . . 64
7.43 src/World/World.h File Reference . . . 64

7.44 World.h 64

Index **67**

Chapter 1

Namespace Index

1.1 Namespace List

Here is a list of all namespaces with brief descriptions:

AssetManager 9
Utils 10

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

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

Col 18
GameObject 25
  Bullet 13
    CircleBullet 15
  Collectable 20
    Coin 16
  Enemy 22
    LargeEnemy 30
    SmallEnemy 36
  Player 34
ParticleSystem 33
Utils::Vec2< T > 38
World 48

Chapter 3

Class Index

3.1 Class List

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

Bullet 13
CircleBullet 15
Coin 16
Col 18
Collectable 20
Enemy 22
GameObject 25
LargeEnemy 30
ParticleSystem 33
Player 34
SmallEnemy 36
Utils::Vec2< T > 38
World 48

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

src/main.cpp . . . 61
src/AssetManager/AssetManager.cpp . . . 51
src/AssetManager/AssetManager.h . . . 51
src/Bullets/Bullet.cpp . . . 52
src/Bullets/Bullet.h . . . 52
src/Bullets/CircleBullet.cpp . . . 52
src/Bullets/CircleBullet.h . . . 52
src/Collectables/Coin.cpp . . . 53
src/Collectables/Coin.h . . . 53
src/Collectables/Collectable.h . . . 53
src/Collisions/Collisions.cpp . . . 54
src/Collisions/Collisions.h . . . 56
src/Color/Color.cpp . . . 58
src/Color/Color.h . . . 58
src/Enemies/Enemy.h . . . 58
src/Enemies/LargeEnemy.cpp . . . 59
src/Enemies/LargeEnemy.h . . . 59
src/Enemies/SmallEnemy.cpp . . . 59
src/Enemies/SmallEnemy.h . . . 59
src/GameObjects/GameObject.cpp . . . 60
src/GameObjects/GameObject.h . . . 60
src/GameObjects/Player.cpp . . . 61
src/GameObjects/Player.h . . . 61
src/ParticleSystem/ParticleSystem.cpp . . . 61
src/ParticleSystem/ParticleSystem.h . . . 61
src/Utils/Vec2.cpp . . . 62
src/Utils/Vec2.h . . . 62
src/World/World.cpp . . . 64
src/World/World.h . . . 64

Chapter 5

Namespace Documentation

5.1 AssetManager Namespace Reference

Functions

- Texture2D * GetTexture (TextureType asset)

5.1.1 Enumeration Type Documentation

5.1.1.1 TextureType

```
enum class AssetManager::TextureType  [strong]
```

Enum class for texture type.

Enumerator

SMALL_ENEMY	
LARGE_ENEMY	
ASSET_COUNT	

5.1.2 Function Documentation

5.1.2.1 GetTexture()

```
Texture2D * AssetManager::GetTexture (
            TextureType asset )
```

Returns the texture of the asset.

Parameters

asset	The texture type

Returns

Texture2D* The texture of the asset ("AssetManager keeps ownership")

5.1.2.2 LoadAssets()

```
void AssetManager::LoadAssets ( )
```

Loads all the necessary images.

5.1.2.3 UnloadAssets()

```
void AssetManager::UnloadAssets ( )
```

Unloads all the images loaded previously.

5.2 Utils Namespace Reference

Classes

- class Vec2

Functions

- template<typename T >
  std::ostream & operator<< (std::ostream &os, const Vec2< T > &vec)

5.2.1 Typedef Documentation

5.2.1.1 Vec2d

```
typedef Vec2<double> Utils::Vec2d
```

5.2.1.2 Vec2i

```
typedef Vec2<int> Utils::Vec2i
```

5.2.2 Function Documentation

5.2.2.1 operator<<()

```
template<typename T >
std::ostream & Utils::operator<< (
            std::ostream & os,
            const Vec2< T > & vec )
```

Inserts a vector into an output stream.

Parameters

os	The output stream
vec	The vector

Returns

std::ostream& The modified output stream

5.2.2.2 Vector2ToVec2()

```
Vec2d Utils::Vector2ToVec2 (
            const Vector2 & vec )
```

Chapter 6

Class Documentation

6.1 Bullet Class Reference

`#include <Bullet.h>`

Public Member Functions

- Bullet (World *worldPtr, const Utils::Vec2d &position, const Utils::Vec2d &velocity, BulletTarget target)
- virtual void Update (double dt) override
- BulletTarget GetTarget () const
- virtual double GetSize () const =0

Public Member Functions inherited from GameObject

- GameObject (World *worldPtr, double health=0.0, double mass=0.0, const Utils::Vec2d &position=Utils::Vec2d::Zero(), double friction=0.00)
- Utils::Vec2d GetPosition () const
- double GetRotation () const
- bool IsAlive () const
- void TakeDamage (double damage)
- void AddForce (const Utils::Vec2d &force)
- void LookAt (const Utils::Vec2d &target)

6.1.1 Member Enumeration Documentation

6.1.1.1 BulletTarget

`enum class Bullet::BulletTarget [strong]`

Enum class for bullet target type (the bullet only damages the target type)

Enumerator

PLAYER	
ENEMY	

6.1.2 Constructor & Destructor Documentation

6.1.2.1 Bullet()

```
Bullet::Bullet (
            World * worldPtr,
            const Utils::Vec2d & position,
            const Utils::Vec2d & velocity,
            BulletTarget target )  [inline]
```

Constructs a new Bullet object.

Parameters

worldPtr	The world that owns the object
position	The position of the object
velocity	The velocity of the object
target	The target type of the object

6.1.2.2 ~Bullet()

```
virtual Bullet::~Bullet ( )  [virtual], [default]
```

Destroy the Bullet object (virtual needed for inheritance)

6.1.3 Member Function Documentation

6.1.3.1 GetSize()

```
virtual double Bullet::GetSize ( ) const  [pure virtual]
```

Returns the size of the bullet.

Returns

double The size of the bullet

Implemented in CircleBullet.

6.1.3.2 GetTarget()

```
BulletTarget Bullet::GetTarget ( ) const  [inline]
```

Returns the target type of the bullet.

Returns

BulletTarget The target type of the bullet

6.1.3.3 Update()

```
void Bullet::Update (
            double dt )  [override], [virtual]
```

Updates the bullet object.

Parameters

dt	The delta time

Reimplemented from GameObject.

6.1.4 Member Data Documentation

6.1.4.1 Target

```
BulletTarget Bullet::Target  [protected]
```

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

- src/Bullets/Bullet.h
- src/Bullets/Bullet.cpp

6.2 CircleBullet Class Reference

```
#include <CircleBullet.h>
```

Public Member Functions

- CircleBullet (World *worldPtr, const Utils::Vec2d &position, const Utils::Vec2d &direction, BulletTarget target)
- double GetSize () const override

Public Member Functions inherited from Bullet

- Bullet (World *worldPtr, const Utils::Vec2d &position, const Utils::Vec2d &velocity, BulletTarget target)
- virtual void Update (double dt) override
- BulletTarget GetTarget () const

Public Member Functions inherited from GameObject

- GameObject (World *worldPtr, double health=0.0, double mass=0.0, const Utils::Vec2d &position=Utils::Vec2d::Zero(), double friction=0.00)
- Utils::Vec2d GetPosition () const
- double GetRotation () const
- bool IsAlive () const
- void TakeDamage (double damage)
- void AddForce (const Utils::Vec2d &force)
- void LookAt (const Utils::Vec2d &target)

6.2.1 Constructor & Destructor Documentation

6.2.1.1 CircleBullet()

```
CircleBullet::CircleBullet (
            World * worldPtr,
            const Utils::Vec2d & position,
            const Utils::Vec2d & direction,
            BulletTarget target )  [inline]
```

Constructs a new CircleBullet object.

Parameters

worldPtr	The world that owns the object
position	The position of the object
direction	The direction of the object
target	The target type of the object

6.2.2 Member Function Documentation

6.2.2.1 Draw()

```
void CircleBullet::Draw ( ) const  [override], [virtual]
```

Draws the circle bullet object.

Implements GameObject.

6.2.2.2 GetSize()

```
double CircleBullet::GetSize ( ) const  [inline], [override], [virtual]
```

Returns the size of the circle bullet.

Returns

double The size of the circle bullet

Implements Bullet.

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

- src/Bullets/CircleBullet.h
- src/Bullets/CircleBullet.cpp

6.3 Coin Class Reference

```
#include <Coin.h>
```

Public Member Functions

- Coin (World *worldPtr, const Utils::Vec2d &position)
- void Update (double dt) override
- double GetSize () const override

Public Member Functions inherited from Collectable

- Collectable (World *worldPtr, const Utils::Vec2d &position=Utils::Vec2d::Zero())
- bool IsCollected () const
- void SetCollected (bool collected)

Public Member Functions inherited from GameObject

- GameObject (World *worldPtr, double health=0.0, double mass=0.0, const Utils::Vec2d &position=Utils::Vec2d::Zero(), double friction=0.00)
- Utils::Vec2d GetPosition () const
- double GetRotation () const
- bool IsAlive () const
- void TakeDamage (double damage)
- void AddForce (const Utils::Vec2d &force)
- void LookAt (const Utils::Vec2d &target)

6.3.1 Constructor & Destructor Documentation

6.3.1.1 Coin()

```
Coin::Coin (
            World * worldPtr,
            const Utils::Vec2d & position )  [inline]
```

Constructs a new Coin object.

Parameters

worldPtr	The world that owns the object
position	The position of the object

6.3.2 Member Function Documentation

6.3.2.1 Draw()

```
void Coin::Draw ( ) const  [override], [virtual]
```

Draws the coin object.

Implements GameObject.

6.3.2.2 GetSize()

```
double Coin::GetSize ( ) const  [inline], [override], [virtual]
```

Returns the size of the coin.

Returns

double The size of the coin

Implements Collectable.

6.3.2.3 Update()

```
void Coin::Update (
            double dt )  [inline], [override], [virtual]
```

Updates the coin object.

Parameters

dt	The delta time

Reimplemented from GameObject.

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

- src/Collectables/Coin.h
- src/Collectables/Coin.cpp

6.4 Col Struct Reference

```
#include <Color.h>
```

Public Member Functions

- Col (int r, int g, int b, int a=255)
- Col (int gray)
- Col Fade (double t) const
- Col Lerp (const Col &other, double t) const
- operator Color () const

6.4.1 Constructor & Destructor Documentation

6.4.1.1 Col() [1/2]

```
Col::Col (
            int r,
            int g,
            int b,
            int a = 255 )
```

Constructs a new Col object.

Parameters

r	The red value
g	The green value
b	The blue value
a	The alpha value

6.4.1.2 Col() [2/2]

```
Col::Col (
            int gray )
```

Constructs a new grayscale Col object (alpha is 255)

Parameters

gray	The grayscale value

6.4.2 Member Function Documentation

6.4.2.1 Fade()

```
Col Col::Fade (
            double t ) const
```

Fades the color by a certain amount (changes the alpha value)

Parameters

t	The amount to fade by (0.0 - 1.0)

Returns

Col The faded color

6.4.2.2 Lerp()

```
Col Col::Lerp (
            const Col & other,
            double t ) const
```

Linearly interpolates between two colors.

Parameters

other	The other color
t	The interpolation value

Returns

Col The interpolated color

6.4.2.3 operator Color()

```
Col::operator Color ( ) const
```

Converts the color to a Color object (Raylib color)

Returns

Color The Color object

6.4.3 Member Data Documentation

6.4.3.1 A

```
int Col::A
```

6.4.3.2 B

```
int Col::B
```

6.4.3.3 G

```
int Col::G
```

6.4.3.4 R

```
int Col::R
```

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

- src/Color/Color.h
- src/Color/Color.cpp

6.5 Collectable Class Reference

```
#include <Collectable.h>
```

Public Member Functions

- Collectable (World *worldPtr, const Utils::Vec2d &position=Utils::Vec2d::Zero())
- virtual double GetSize () const =0
- bool IsCollected () const
- void SetCollected (bool collected)

Public Member Functions inherited from GameObject

- GameObject (World *worldPtr, double health=0.0, double mass=0.0, const Utils::Vec2d &position=Utils::Vec2d::Zero(), double friction=0.00)
- Utils::Vec2d GetPosition () const
- double GetRotation () const
- bool IsAlive () const
- void TakeDamage (double damage)
- void AddForce (const Utils::Vec2d &force)
- void LookAt (const Utils::Vec2d &target)
- virtual void Update (double dt)

6.5.1 Constructor & Destructor Documentation

6.5.1.1 Collectable()

```
Collectable::Collectable (
            World * worldPtr,
            const Utils::Vec2d & position = Utils::Vec2d::Zero() )  [inline]
```

Constructs a new Collectable object.

Parameters

worldPtr	The world that owns the object
position	The position of the object

6.5.1.2 ~Collectable()

```
virtual Collectable::~Collectable ( )  [virtual], [default]
```

Destroys the Collectable object (virtual needed for inheritance)

6.5.2 Member Function Documentation

6.5.2.1 GetSize()

```
virtual double Collectable::GetSize ( ) const  [pure virtual]
```

Returns the size of the object.

Returns

double The size of the object

Implemented in Coin.

6.5.2.2 IsCollected()

```
bool Collectable::IsCollected ( ) const  [inline]
```

Returns whether the object is collected.

Returns

bool True if the object is collected, false otherwise

6.5.2.3 SetCollected()

```
void Collectable::SetCollected (
            bool collected )  [inline]
```

Sets whether the object is collected.

Parameters

collected	True if the object is collected, false otherwise

6.5.3 Member Data Documentation

6.5.3.1 Collected

```
bool Collectable::Collected  [protected]
```

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

- src/Collectables/Collectable.h

6.6 Enemy Class Reference

```
#include <Enemy.h>
```

Public Member Functions

- Enemy (World *worldPtr, double health=100.0, double mass=1.0, const Utils::Vec2d &position=Utils::Vec2d::Zero())
- void SetTarget (const Utils::Vec2d &target)
- virtual double GetMoveForce () const =0
- virtual double GetSize () const =0
- virtual Col GetColor () const =0

Public Member Functions inherited from GameObject

- GameObject (World *worldPtr, double health=0.0, double mass=0.0, const Utils::Vec2d &position=Utils::Vec2d::Zero(), double friction=0.00)
- Utils::Vec2d GetPosition () const
- double GetRotation () const
- bool IsAlive () const
- void TakeDamage (double damage)
- void AddForce (const Utils::Vec2d &force)
- void LookAt (const Utils::Vec2d &target)
- virtual void Update (double dt)

6.6.1 Constructor & Destructor Documentation

6.6.1.1 Enemy()

```
Enemy::Enemy (
            World * worldPtr,
            double health = 100.0,
            double mass = 1.0,
            const Utils::Vec2d & position = Utils::Vec2d::Zero() )  [inline]
```

Constructs a new Enemy object.

Parameters

worldPtr	The world that owns the object
health	Max health of the object
mass	Mass of the object
position	Position of the object

6.6.1.2 ~Enemy()

```
virtual Enemy::~Enemy ( )  [virtual], [default]
```

Destroys the Enemy object (virtual needed for inheritance)

6.6.2 Member Function Documentation

6.6.2.1 GetColor()

```
virtual Col Enemy::GetColor ( ) const  [pure virtual]
```

Returns the color of the object (needed because of inheritance)

Returns

Col The color of the object

Implemented in LargeEnemy, and SmallEnemy.

6.6.2.2 GetMoveForce()

```
virtual double Enemy::GetMoveForce ( ) const  [pure virtual]
```

Returns the move force of the object (needed because of inheritance)

Returns

double The move force of the object

Implemented in LargeEnemy, and SmallEnemy.

6.6.2.3 GetSize()

```
virtual double Enemy::GetSize ( ) const  [pure virtual]
```

Returns the size of the object (needed because of inheritance)

Returns

double The size of the object

Implemented in LargeEnemy, and SmallEnemy.

6.6.2.4 SetTarget()

```
void Enemy::SetTarget (
            const Utils::Vec2d & target )  [inline]
```

Sets the target of the enemy (where it will move towards)

Parameters

target	The target to set

6.6.3 Member Data Documentation

6.6.3.1 Target

```
Utils::Vec2d Enemy::Target  [protected]
```

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

- src/Enemies/Enemy.h

6.7 GameObject Class Reference

`#include <GameObject.h>`

Public Member Functions

- GameObject (World *worldPtr, double health=0.0, double mass=0.0, const Utils::Vec2d &position=Utils::Vec2d::Zero(), double friction=0.00)
- Utils::Vec2d GetPosition () const
- double GetRotation () const
- bool IsAlive () const
- void TakeDamage (double damage)
- void AddForce (const Utils::Vec2d &force)
- void LookAt (const Utils::Vec2d &target)
- virtual void Update (double dt)

6.7.1 Constructor & Destructor Documentation

6.7.1.1 GameObject()

```
GameObject::GameObject (
            World * worldPtr,
            double health = 0.0,
            double mass = 0.0,
            const Utils::Vec2d & position = Utils::Vec2d::Zero(),
            double friction = 0.00 )  [inline]
```

Construct a new GameObject.

Parameters

worldPtr	The world that owns the object
health	Max health of the object
mass	Mass of the object
position	Position of the object
friction	Friction of the object

6.7.1.2 ~GameObject()

```
virtual GameObject::~GameObject ( )  [virtual], [default]
```

Destroy the Game Object object (virtual needed for inheritance)

6.7.2 Member Function Documentation

6.7.2.1 AddForce()

```
void GameObject::AddForce (
            const Utils::Vec2d & force )  [inline]
```

Adds force to the object.

Parameters

force	The force to add

6.7.2.2 Destroy()

```
void GameObject::Destroy ( )  [inline]
```

Destroys the object.

6.7.2.3 Draw()

```
virtual void GameObject::Draw ( ) const  [pure virtual]
```

Draws the object.

Implemented in CircleBullet, Coin, LargeEnemy, SmallEnemy, and Player.

6.7.2.4 DrawHealthBar()

```
void GameObject::DrawHealthBar (
            const Utils::Vec2d & position,
            const Utils::Vec2d & size ) const  [protected]
```

6.7.2.5 GetPosition()

```
Utils::Vec2d GameObject::GetPosition ( ) const  [inline]
```

Returns the position of the object.

Returns

Utils::Vec2d The position of the object

6.7.2.6 GetRotation()

```
double GameObject::GetRotation ( ) const  [inline]
```

Returns the rotation of the object.

Returns

double The rotation of the object

6.7.2.7 IsAlive()

```
bool GameObject::IsAlive ( ) const  [inline]
```

Returns whether the object is alive.

Returns

bool Whether the object is alive

6.7.2.8 LookAt()

```
void GameObject::LookAt (
            const Utils::Vec2d & target )
```

Rotates the object to look at a target.

Parameters

target	The target to look at

6.7.2.9 TakeDamage()

```
void GameObject::TakeDamage (
            double damage )  [inline]
```

Deals damage to the object (decreses health, destroys if health is <= 0)

Parameters

damage	The amount of damage to deal

6.7.2.10 Update()

```
void GameObject::Update (
            double dt )  [virtual]
```

Updates the object (using euler integration)

Parameters

dt	The delta time

Reimplemented in Bullet, Coin, LargeEnemy, SmallEnemy, and Player.

6.7.3 Member Data Documentation

6.7.3.1 Acceleration

```
Utils::Vec2d GameObject::Acceleration  [protected]
```

6.7.3.2 Alive

```
bool GameObject::Alive  [protected]
```

6.7.3.3 Force

```
Utils::Vec2d GameObject::Force  [protected]
```

6.7.3.4 Friction

```
double GameObject::Friction  [protected]
```

6.7.3.5 Health

```
double GameObject::Health  [protected]
```

6.7.3.6 Mass

```
double GameObject::Mass  [protected]
```

6.7.3.7 MaxHealth

```
double GameObject::MaxHealth  [protected]
```

6.7.3.8 Position

```
Utils::Vec2d GameObject::Position  [protected]
```

6.7.3.9 Rotation

```
double GameObject::Rotation  [protected]
```

6.7.3.10 Velocity

```
Utils::Vec2d GameObject::Velocity  [protected]
```

6.7.3.11 WorldPtr

```
World* GameObject::WorldPtr  [protected]
```

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

- src/GameObjects/GameObject.h
- src/GameObjects/GameObject.cpp

6.8 LargeEnemy Class Reference

```
#include <LargeEnemy.h>
```

Public Member Functions

- LargeEnemy (World *worldPtr, const Utils::Vec2d &position=Utils::Vec2d::Zero())
- void Update (double dt) override
- double GetSize () const override
- double GetMoveForce () const override
- Col GetColor () const override

Public Member Functions inherited from Enemy

- Enemy (World *worldPtr, double health=100.0, double mass=1.0, const Utils::Vec2d &position=Utils::Vec2d::Zero())
- void SetTarget (const Utils::Vec2d &target)

Public Member Functions inherited from GameObject

- GameObject (World *worldPtr, double health=0.0, double mass=0.0, const Utils::Vec2d &position=Utils::Vec2d::Zero(), double friction=0.00)
- Utils::Vec2d GetPosition () const
- double GetRotation () const
- bool IsAlive () const
- void TakeDamage (double damage)
- void AddForce (const Utils::Vec2d &force)
- void LookAt (const Utils::Vec2d &target)

6.8.1 Constructor & Destructor Documentation

6.8.1.1 LargeEnemy()

```
LargeEnemy::LargeEnemy (
            World * worldPtr,
            const Utils::Vec2d & position = Utils::Vec2d::Zero() )  [inline]
```

Constructs a new SmallEnemy object.

Parameters

worldPtr	The world that owns the object
position	The position of the object

6.8.2 Member Function Documentation

6.8.2.1 Draw()

```
void LargeEnemy::Draw ( ) const  [override], [virtual]
```

Draws the small enemy object.

Implements GameObject.

6.8.2.2 GetColor()

```
Col LargeEnemy::GetColor ( ) const  [inline], [override], [virtual]
```

Returns the color of the object.

Returns

Col The color of the object

Implements Enemy.

6.8.2.3 GetMoveForce()

```
double LargeEnemy::GetMoveForce ( ) const  [inline], [override], [virtual]
```

Returns the move force of the object.

Returns

double The move force of the object

Implements Enemy.

6.8.2.4 GetSize()

```
double LargeEnemy::GetSize ( ) const  [inline], [override], [virtual]
```

Returns the size of the object.

Returns

double The size of the object

Implements Enemy.

6.8.2.5 Update()

```
void LargeEnemy::Update (
            double dt )  [override], [virtual]
```

Updates the small enemy object.

Parameters

dt	The delta time

Reimplemented from GameObject.

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

- src/Enemies/LargeEnemy.h
- src/Enemies/LargeEnemy.cpp

6.9 ParticleSystem Class Reference

`#include <ParticleSystem.h>`

Public Member Functions

- size_t GetParticleCount () const
- void Emit (const Utils::Vec2d &position, const Utils::Vec2d &direction, double randomAngle, double lifetime, const Col &startColor, const Col &endColor, double startSize, double endSize, size_t count)
- void Update (double dt)

6.9.1 Constructor & Destructor Documentation

6.9.1.1 ParticleSystem()

```
ParticleSystem::ParticleSystem ( )
```

Constructs a new Particle System object.

6.9.1.2 ~ParticleSystem()

```
ParticleSystem::~ParticleSystem ( )  [default]
```

Destroys the Particle System object.

6.9.2 Member Function Documentation

6.9.2.1 Draw()

```
void ParticleSystem::Draw ( ) const
```

Draws the particles in the system.

6.9.2.2 Emit()

```
void ParticleSystem::Emit (
            const Utils::Vec2d & position,
            const Utils::Vec2d & direction,
            double randomAngle,
            double lifetime,
            const Col & startColor,
            const Col & endColor,
            double startSize,
            double endSize,
            size_t count )
```

Emits particles with given parameters.

Parameters

position	The position to emit the particles from
direction	The direction the particles will move
randomAngle	The max angle the direction will be randomized by
lifetime	The lifetime of the particles in seconds
startColor	The starting color of the particles
endColor	The ending color of the particles
startSize	The starting size of the particles
endSize	The ending size of the particles
count	The number of particles to emit

6.9.2.3 GetParticleCount()

```
size_t ParticleSystem::GetParticleCount ( ) const  [inline]
```

Returns the number of particles in the system.

Returns

size_t The number of particles in the system

6.9.2.4 Update()

```
void ParticleSystem::Update (
            double dt )
```

Updates the particles in the system.

Parameters

dt	The delta time

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

- src/ParticleSystem/ParticleSystem.h
- src/ParticleSystem/ParticleSystem.cpp

6.10 Player Class Reference

```
#include <Player.h>
```

Public Member Functions

- Player (World *worldPtr, const Utils::Vec2d &position=Utils::Vec2d::Zero())
- void Update (double dt) override
- void AddCoin (size_t count=1)
- size_t GetCoinCount () const

Public Member Functions inherited from GameObject

- GameObject (World *worldPtr, double health=0.0, double mass=0.0, const Utils::Vec2d &position=Utils::Vec2d::Zero(), double friction=0.00)
- Utils::Vec2d GetPosition () const
- double GetRotation () const
- bool IsAlive () const
- void TakeDamage (double damage)
- void AddForce (const Utils::Vec2d &force)
- void LookAt (const Utils::Vec2d &target)

6.10.1 Constructor & Destructor Documentation

6.10.1.1 Player()

```
Player::Player (
            World * worldPtr,
            const Utils::Vec2d & position = Utils::Vec2d::Zero() )  [inline]
```

Constructs a new Player object.

Parameters

worldPtr	The world that owns the object
position	The position of the object

6.10.1.2 ~Player()

```
Player::~Player ( )  [default]
```

Destroys the Player object.

6.10.2 Member Function Documentation

6.10.2.1 AddCoin()

```
void Player::AddCoin (
            size_t count = 1 )  [inline]
```

Adds coins to the player.

Parameters

count	The number of coins to add

6.10.2.2 Draw()

```
void Player::Draw ( ) const  [override], [virtual]
```

Draws the player object.

Implements GameObject.

6.10.2.3 GetCoinCount()

```
size_t Player::GetCoinCount ( ) const  [inline]
```

Returns the coin count of the player.

Returns

size_t The coin count of the player

6.10.2.4 Update()

```
void Player::Update (
            double dt )  [override], [virtual]
```

Updates the player object.

Parameters

dt	The delta time

Reimplemented from GameObject.

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

- src/GameObjects/Player.h
- src/GameObjects/Player.cpp

6.11 SmallEnemy Class Reference

```
#include <SmallEnemy.h>
```

Public Member Functions

- SmallEnemy (World *worldPtr, const Utils::Vec2d &position=Utils::Vec2d::Zero())
- void Update (double dt) override
- double GetSize () const override
- double GetMoveForce () const override
- Col GetColor () const override

Public Member Functions inherited from Enemy

- Enemy (World *worldPtr, double health=100.0, double mass=1.0, const Utils::Vec2d &position=Utils::Vec2d::Zero())
- void SetTarget (const Utils::Vec2d &target)

Public Member Functions inherited from GameObject

- GameObject (World *worldPtr, double health=0.0, double mass=0.0, const Utils::Vec2d &position=Utils::Vec2d::Zero(), double friction=0.00)
- Utils::Vec2d GetPosition () const
- double GetRotation () const
- bool IsAlive () const
- void TakeDamage (double damage)
- void AddForce (const Utils::Vec2d &force)
- void LookAt (const Utils::Vec2d &target)

6.11.1 Constructor & Destructor Documentation

6.11.1.1 SmallEnemy()

```
SmallEnemy::SmallEnemy (
            World * worldPtr,
            const Utils::Vec2d & position = Utils::Vec2d::Zero() )  [inline]
```

Constructs a new SmallEnemy object.

Parameters

worldPtr	The world that owns the object
position	The position of the object

6.11.2 Member Function Documentation

6.11.2.1 Draw()

```
void SmallEnemy::Draw ( ) const  [override], [virtual]
```

Draws the small enemy object.

Implements GameObject.

6.11.2.2 GetColor()

```
Col SmallEnemy::GetColor ( ) const  [inline], [override], [virtual]
```

Returns the color of the object.

Returns

Col The color of the object

Implements Enemy.

6.11.2.3 GetMoveForce()

```
double SmallEnemy::GetMoveForce ( ) const  [inline], [override], [virtual]
```

Returns the move force of the object.

Returns

double The move force of the object

Implements Enemy.

6.11.2.4 GetSize()

```
double SmallEnemy::GetSize ( ) const  [inline], [override], [virtual]
```

Returns the size of the object.

Returns

double The size of the object

Implements Enemy.

6.11.2.5 Update()

```
void SmallEnemy::Update (
            double dt )  [override], [virtual]
```

Updates the small enemy object.

Parameters

dt	The delta time

Reimplemented from GameObject.

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

- src/Enemies/SmallEnemy.h
- src/Enemies/SmallEnemy.cpp

6.12 Utils::Vec2< T > Class Template Reference

```
#include <Vec2.h>
```

Public Member Functions

- Vec2 (T x, T y)
- T GetX () const
- T GetY () const
- double GetAngle () const
- double GetLengthSquared () const
- double GetLength () const
- Vec2 GetNormalized () const
- void SetX (T x)
- void SetY (T y)
- Vec2 Rotate (double angle) const
- Vec2 operator+ (const Vec2 &other) const
- Vec2 operator- (const Vec2 &other) const
- T operator∗ (const Vec2 &other) const
- T operator^ (const Vec2 &other) const
- Vec2 operator∗ (const T &other) const
- Vec2 operator/ (const T &other) const
- Vec2 operator- () const
- Vec2 & operator+= (const Vec2 &other)
- Vec2 & operator-= (const Vec2 &other)
- Vec2 & operator∗= (const Vec2 &other)
- Vec2 & operator/= (const Vec2 &other)
- Vec2 & operator∗= (const T &other)
- Vec2 & operator/= (const T &other)

Static Public Member Functions

- static Vec2 FromAngle (double angle)
- static Vec2 Zero ()
- static Vec2 One ()
- static Vec2 Up ()
- static Vec2 Down ()
- static Vec2 Left ()
- static Vec2 Right ()

6.12.1 Constructor & Destructor Documentation

6.12.1.1 Vec2() [1/2]

```
template<typename T >
Utils::Vec2< T >::Vec2 ( )  [inline]
```

Constructs a new Vec2 object with a given type (default values are 0)

6.12.1.2 Vec2() [2/2]

```
template<typename T >
Utils::Vec2< T >::Vec2 (
            T x,
            T y )  [inline]
```

Constructs a new Vec2 object with a given type and given values.

Parameters

x	The x value
y	The y value

6.12.1.3 ~Vec2()

```
template<typename T >
Utils::Vec2< T >::~Vec2 ( )  [default]
```

6.12.2 Member Function Documentation

6.12.2.1 Down()

```
template<typename T >
static Vec2 Utils::Vec2< T >::Down ( )  [inline], [static]
```

Returns a unit vector pointing down.

Returns

Vec2 The down vector

6.12.2.2 FromAngle()

```
template<typename T >
static Vec2 Utils::Vec2< T >::FromAngle (
            double angle )  [inline], [static]
```

Creates a vector from an angle.

Parameters

angle	The angle

Returns

Vec2 The vector from the angle

6.12.2.3 GetAngle()

```
template<typename T >
double Utils::Vec2< T >::GetAngle ( ) const  [inline]
```

Returns the angle of the vector.

Returns

double The angle of the vector

6.12.2.4 GetLength()

```
template<typename T >
double Utils::Vec2< T >::GetLength ( ) const  [inline]
```

Returns the length of the vector.

Returns

double The length of the vector

6.12.2.5 GetLengthSquared()

```
template<typename T >
double Utils::Vec2< T >::GetLengthSquared ( ) const  [inline]
```

Returns the squared length of the vector.

Returns

double The squared length of the vector

6.12.2.6 GetNormalized()

```
template<typename T >
Vec2 Utils::Vec2< T >::GetNormalized ( ) const  [inline]
```

Returns the normalized vector.

Returns

Vec2 The normalized vector

6.12.2.7 GetX()

```
template<typename T >
T Utils::Vec2< T >::GetX ( ) const  [inline]
```

Returns the x value.

Returns

T The x value

6.12.2.8 GetY()

```
template<typename T >
T Utils::Vec2< T >::GetY ( ) const  [inline]
```

Returns the y value.

Returns

T The y value

6.12.2.9 Left()

```
template<typename T >
static Vec2 Utils::Vec2< T >::Left ( )  [inline], [static]
```

Returns a unit vector pointing left.

Returns

Vec2 The left vector

6.12.2.10 One()

```
template<typename T >
static Vec2 Utils::Vec2< T >::One ( )  [inline], [static]
```

Returns a one vector.

Returns

Vec2 The one vector

6.12.2.11 operator Vector2()

```
template<typename T >
Utils::Vec2< T >::operator Vector2 ( ) const  [inline]
```

6.12.2.12 operator*() [1/2]

```
template<typename T >
Vec2 Utils::Vec2< T >::operator* (
            const T & other ) const  [inline]
```

Returns the element-wise product of two vectors.

Parameters

other	The other vector

Returns

Vec2 The product of the two vectors

6.12.2.13 operator*() [2/2]

```
template<typename T >
T Utils::Vec2< T >::operator* (
            const Vec2< T > & other ) const  [inline]
```

Returns the dot product of two vectors.

Parameters

other	The other vector

Returns

T The dot product of the two vectors

6.12.2.14 operator*=() [1/2]

```
template<typename T >
Vec2 & Utils::Vec2< T >::operator*= (
            const T & other )  [inline]
```

Multiplies the vector with a scalar.

Parameters

other	The scalar

Returns

Vec2& The current vector

6.12.2.15 operator*=() [2/2]

```
template<typename T >
Vec2 & Utils::Vec2< T >::operator*= (
            const Vec2< T > & other )  [inline]
```

Multiplies another vector with the current vector.

Parameters

other	The other vector

Returns

Vec2& The current vector

6.12.2.16 operator+()

```
template<typename T >
Vec2 Utils::Vec2< T >::operator+ (
            const Vec2< T > & other ) const  [inline]
```

Returns the sum of two vectors.

Parameters

other	The other vector

Returns

Vec2 The sum of the two vectors

6.12.2.17 operator+=()

```
template<typename T >
Vec2 & Utils::Vec2< T >::operator+= (
            const Vec2< T > & other )  [inline]
```

Adds another vector to the current vector.

Parameters

other	The other vector

Returns

Vec2& The current vector

6.12.2.18 operator-() [1/2]

```
template<typename T >
Vec2 Utils::Vec2< T >::operator- ( ) const  [inline]
```

Returns the negation of the vector.

Returns

Vec2 The negation of the vector

6.12.2.19 operator-() [2/2]

```
template<typename T >
Vec2 Utils::Vec2< T >::operator- (
            const Vec2< T > & other ) const  [inline]
```

Returns the difference of two vectors.

Parameters

other	The other vector

Returns

Vec2 The difference of the two vectors

6.12.2.20 operator-=()

```
template<typename T >
Vec2 & Utils::Vec2< T >::operator-= (
            const Vec2< T > & other )  [inline]
```

Subtracts another vector from the current vector.

Parameters

other	The other vector

Returns

Vec2& The current vector

6.12.2.21 operator/()

```
template<typename T >
Vec2 Utils::Vec2< T >::operator/ (
            const T & other ) const  [inline]
```

Returns the element-wise division of two vectors.

Parameters

other	The other vector

Returns

Vec2 The division of the two vectors

6.12.2.22 operator/=() [1/2]

```
template<typename T >
Vec2 & Utils::Vec2< T >::operator/= (
            const T & other )  [inline]
```

Divides the vector with a scalar.

Parameters

other	The scalar

Returns

Vec2& The current vector

6.12.2.23 operator/=() [2/2]

```
template<typename T >
Vec2 & Utils::Vec2< T >::operator/= (
            const Vec2< T > & other )  [inline]
```

Divides another vector with the current vector.

Parameters

other	The other vector

Returns

Vec2& The current vector

6.12.2.24 operator^()

```
template<typename T >
T Utils::Vec2< T >::operator^ (
            const Vec2< T > & other ) const  [inline]
```

Returns the cross product of two vectors.

Parameters

other	The other vector

Returns

T The cross product of the two vectors

6.12.2.25 Right()

```
template<typename T >
static Vec2 Utils::Vec2< T >::Right ( )  [inline], [static]
```

Returns a unit vector pointing right.

Returns

Vec2 The right vector

6.12.2.26 Rotate()

```
template<typename T >
Vec2 Utils::Vec2< T >::Rotate (
            double angle ) const  [inline]
```

Rotates the vector by a given angle.

Parameters

angle	The angle to rotate the vector by

Returns

Vec2 The rotated vector

6.12.2.27 SetX()

```
template<typename T >
void Utils::Vec2< T >::SetX (
            T x )  [inline]
```

Sets the x value.

Parameters

x	The new x value

6.12.2.28 SetY()

```
template<typename T >
void Utils::Vec2< T >::SetY (
            T y )  [inline]
```

Sets the y value.

Parameters

y	The new y value

6.12.2.29 Up()

```
template<typename T >
static Vec2 Utils::Vec2< T >::Up ( )  [inline], [static]
```

Returns a unit vector pointing up.

Returns

Vec2 The up vector

6.12.2.30 Zero()

```
template<typename T >
static Vec2 Utils::Vec2< T >::Zero ( )  [inline], [static]
```

Returns a zero vector.

Returns

Vec2 The zero vector

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

- src/Utils/Vec2.h

6.13 World Class Reference

```
#include <World.h>
```

Public Member Functions

- void AddCollectable (std::unique_ptr< Collectable > collectable)
- void AddEnemy (std::unique_ptr< Enemy > enemy)
- void AddBullet (std::unique_ptr< Bullet > bullet)
- void Update (double dt)
- Player * GetMainPlayer ()

6.13.1 Constructor & Destructor Documentation

6.13.1.1 World()

```
World::World ( )
```

Constructs a new World object.

6.13.2 Member Function Documentation

6.13.2.1 AddBullet()

```
void World::AddBullet (
            std::unique_ptr< Bullet > bullet )  [inline]
```

Adds a bullet to the world (takes ownership)

Parameters

bullet	The bullet to add

6.13.2.2 AddCollectable()

```
void World::AddCollectable (
            std::unique_ptr< Collectable > collectable )  [inline]
```

Adds a collectable to the world (takes ownership)

Parameters

collectable	The collectable to add

6.13.2.3 AddEnemy()

```
void World::AddEnemy (
            std::unique_ptr< Enemy > enemy )  [inline]
```

Adds an enemy to the world (takes ownership)

Parameters

enemy	The enemy to add

6.13.2.4 Draw()

```
void World::Draw ( ) const
```

Calls the Draw function of all objects in the world and shows FPS.

6.13.2.5 GetMainPlayer()

```
Player * World::GetMainPlayer ( )  [inline]
```

Returns the main player.

Returns

Player* The main player (the world keeps ownership of the player object)

6.13.2.6 Update()

```
void World::Update (
            double dt )
```

Generates objects, calls the Update function of all objects in the world.

Parameters

dt	The delta time

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

- src/World/World.h
- src/World/World.cpp

Chapter 7

File Documentation

7.1 src/AssetManager/AssetManager.cpp File Reference

Namespaces

- namespace AssetManager

Functions

- Texture2D * AssetManager::GetTexture (TextureType asset)

7.2 src/AssetManager/AssetManager.h File Reference

Namespaces

- namespace AssetManager

Functions

- Texture2D * AssetManager::GetTexture (TextureType asset)

7.3 AssetManager.h

Go to the documentation of this file.

```
00001 #ifndef CPORTA
00002
00003 #ifndef ASSET_MANAGER_H
00004 #define ASSET_MANAGER_H
00005
00006 #include <raylib.h>
00007
00008 namespace AssetManager {
00009
00014     enum class TextureType
00015     {
00016         SMALL_ENEMY = 0, LARGE_ENEMY, ASSET_COUNT
00017     };
00018
00022     void LoadAssets();
00026     void UnloadAssets();
00027
00034     Texture2D* GetTexture(TextureType asset);
00035 }
00036
00037
00038 #endif
00039
00040 #endif
```

7.4 src/Bullets/Bullet.cpp File Reference

7.5 src/Bullets/Bullet.h File Reference

Classes

- class Bullet

7.6 Bullet.h

Go to the documentation of this file.

```
00001 #ifndef BULLET_H
00002 #define BULLET_H
00003
00004 #include "GameObjects/GameObject.h"
00005 #include "Utils/Vec2.h"
00006
00007 class Bullet : public GameObject
00008 {
00009 public:
00013     enum class BulletTarget { PLAYER = 0, ENEMY };
00014
00015 public:
00024     Bullet(World* worldPtr, const Utils::Vec2d& position, const Utils::Vec2d& velocity, BulletTarget
      target)
00025         : GameObject(worldPtr, 0.0, 0.5, position, 0.0), Target(target) { Velocity = velocity; }
00029     virtual ~Bullet() = default;
00030
00036     virtual void Update(double dt) override;
00037     //virtual void Draw() const override; // this is a pure virtual function in GameObject
00038
00044     BulletTarget GetTarget() const { return Target; }
00050     virtual double GetSize() const = 0;
00051
00052 protected:
00053     BulletTarget Target;
00054 };
00055
00056 #endif
```

7.7 src/Bullets/CircleBullet.cpp File Reference

7.8 src/Bullets/CircleBullet.h File Reference

Classes

- class CircleBullet

7.9 CircleBullet.h

Go to the documentation of this file.

```
00001 #ifndef CIRCLEBULLET_H
00002 #define CIRCLEBULLET_H
00003
00004 #include "Bullet.h"
00005 #include "Color/Color.h"
00006
00007 class CircleBullet : public Bullet
00008 {
```

```
00009 public:
00018     CircleBullet(World* worldPtr, const Utils::Vec2d& position, const Utils::Vec2d& direction,
      BulletTarget target)
00019         : Bullet(worldPtr, position, direction.GetNormalized() * Speed, target) { }
00020
00021     // void Update(double dt) override; // no need to override
00025     void Draw() const override;
00026
00032     double GetSize() const override { return Radius; }
00033
00034 private:
00035     static const double Radius;
00036     static const double Speed;
00037     static const Col Color;
00038 };
00039
00040 #endif
```

7.10 src/Collectables/Coin.cpp File Reference

7.11 src/Collectables/Coin.h File Reference

Classes

- class Coin

7.12 Coin.h

Go to the documentation of this file.

```
00001 #ifndef COIN_H
00002 #define COIN_H
00003
00004 #include "Collectable.h"
00005
00006 class Coin : public Collectable
00007 {
00008 public:
00015     Coin(World* worldPtr, const Utils::Vec2d& position) : Collectable(worldPtr, position), Time(0.0) {
      }
00016
00022     void Update(double dt) override { Time += dt; }
00026     void Draw() const override;
00027
00033     double GetSize() const override { return Radius * 2.0f; }
00034
00035 private:
00036     static const double Radius;
00037     static const double SwingSpeed;
00038     static const double SwingAmount;
00039
00040 private:
00041     double Time; // keep track of time for the swing animation
00042 };
00043
00044 #endif
```

7.13 src/Collectables/Collectable.h File Reference

Classes

- class Collectable

7.14 Collectable.h

Go to the documentation of this file.

```
00001 #ifndef COLLECTABLE_H
00002 #define COLLECTABLE_H
00003
00004 #include "GameObjects/GameObject.h"
00005
00006 class Collectable : public GameObject
00007 {
00008 public:
00015     Collectable(World* worldPtr, const Utils::Vec2d& position = Utils::Vec2d::Zero())
00016         : GameObject(worldPtr, 0.0, 0.0, position, 0.0), Collected(false) { }
00020     virtual ~Collectable() = default;
00021
00027     virtual double GetSize() const = 0;
00033     bool IsCollected() const { return Collected; }
00039     void SetCollected(bool collected) { Collected = collected; }
00040
00041 protected:
00042     bool Collected;
00043 };
00044
00045
00046 #endif
```

7.15 src/Collisions/Collisions.cpp File Reference

Functions

- bool CheckCollision (const Player &player, const Collectable &collectable)
- bool CheckCollision (const Player &player, const Bullet &bullet)
- bool CheckCollision (const Player &player, const Enemy &enemy)
- bool CheckCollision (const Enemy &enemy, const Bullet &bullet)

7.15.1 Function Documentation

7.15.1.1 CheckCollision() [1/4]

```
bool CheckCollision (
            const Enemy & enemy,
            const Bullet & bullet )
```

Checks if the enemy collides with the bullet (AABB collision detection)

Parameters

enemy	The enemy object
bullet	The enemy object

Returns

bool True if the enemy collides with the bullet, false otherwise

7.15.1.2 CheckCollision() [2/4]

```
bool CheckCollision (
```

```
            const Player & player,
            const Bullet & bullet )
```

Checks if the player collides with the bullet (AABB collision detection)

Parameters

player	The player object
bullet	The bullet object

Returns

bool True if the player collides with the bullet, false otherwise

7.15.1.3 CheckCollision() [3/4]

```
bool CheckCollision (
            const Player & player,
            const Collectable & collectable )
```

Checks if the player collides with the collectable (AABB collision detection)

Parameters

player	The player object
collectable	The collectable object

Returns

bool True if the player collides with the collectable, false otherwise

7.15.1.4 CheckCollision() [4/4]

```
bool CheckCollision (
            const Player & player,
            const Enemy & enemy )
```

Checks if the player collides with the enemy (AABB collision detection)

Parameters

player	The player object
enemy	The enemy object

Returns

bool True if the player collides with the enemy, false otherwise

7.16 src/Collisions/Collisions.h File Reference

Functions

- bool CheckCollision (const Player &player, const Collectable &collectable)
- bool CheckCollision (const Player &player, const Bullet &bullet)
- bool CheckCollision (const Player &player, const Enemy &enemy)
- bool CheckCollision (const Enemy &enemy, const Bullet &bullet)

7.16.1 Function Documentation

7.16.1.1 CheckCollision() [1/4]

```
bool CheckCollision (
            const Enemy & enemy,
            const Bullet & bullet )
```

Checks if the enemy collides with the bullet (AABB collision detection)

Parameters

enemy	The enemy object
bullet	The enemy object

Returns

bool True if the enemy collides with the bullet, false otherwise

7.16.1.2 CheckCollision() [2/4]

```
bool CheckCollision (
            const Player & player,
            const Bullet & bullet )
```

Checks if the player collides with the bullet (AABB collision detection)

Parameters

player	The player object
bullet	The bullet object

Returns

bool True if the player collides with the bullet, false otherwise

7.16.1.3 CheckCollision() [3/4]

```
bool CheckCollision (
```

```
            const Player & player,
            const Collectable & collectable )
```

Checks if the player collides with the collectable (AABB collision detection)

Parameters

player	The player object
collectable	The collectable object

Returns

bool True if the player collides with the collectable, false otherwise

7.16.1.4 CheckCollision() [4/4]

```
bool CheckCollision (
            const Player & player,
            const Enemy & enemy )
```

Checks if the player collides with the enemy (AABB collision detection)

Parameters

player	The player object
enemy	The enemy object

Returns

bool True if the player collides with the enemy, false otherwise

7.17 Collisions.h

Go to the documentation of this file.

```
00001 #ifndef COLLISIONS_H
00002 #define COLLISIONS_H
00003
00004 #include "GameObjects/Player.h"
00005 #include "Collectables/Collectable.h"
00006 #include "Bullets/Bullet.h"
00007 #include "Enemies/Enemy.h"
00008
00016 bool CheckCollision(const Player& player, const Collectable& collectable);
00024 bool CheckCollision(const Player& player, const Bullet& bullet);
00032 bool CheckCollision(const Player& player, const Enemy& enemy);
00040 bool CheckCollision(const Enemy& enemy, const Bullet& bullet);
00041
00042 #endif
```

7.18 src/Color/Color.cpp File Reference

7.19 src/Color/Color.h File Reference

Classes

- struct Col

7.20 Color.h

Go to the documentation of this file.

```
00001 #ifndef COLOR_H
00002 #define COLOR_H
00003
00004 struct Color;
00005
00006 struct Col
00007 {
00016     Col(int r, int g, int b, int a = 255);
00022     Col(int gray);
00023
00030     Col Fade(double t) const;
00038     Col Lerp(const Col& other, double t) const;
00039
00045     operator Color() const;
00046
00047 public:
00048     int R, G, B, A;
00049 };
00050
00051 #endif
```

7.21 src/Enemies/Enemy.h File Reference

Classes

- class Enemy

7.22 Enemy.h

Go to the documentation of this file.

```
00001 #ifndef ENEMY_H
00002 #define ENEMY_H
00003
00004 #include "GameObjects/GameObject.h"
00005 #include "Color/Color.h"
00006
00007 class Enemy : public GameObject
00008 {
00009 public:
00018     Enemy(World* worldPtr, double health = 100.0, double mass = 1.0, const Utils::Vec2d& position =
      Utils::Vec2d::Zero())
00019         : GameObject(worldPtr, health, mass, position, 0.05), Target(position) { }
00023     virtual ~Enemy() = default;
00024
00030     void SetTarget(const Utils::Vec2d& target) { Target = target; }
00031
00037     virtual double GetMoveForce() const = 0;
00043     virtual double GetSize() const = 0;
00049     virtual Col GetColor() const = 0;
00050
00051 protected:
00052     Utils::Vec2d Target;
00053 };
00054
00055 #endif
```

7.23 src/Enemies/LargeEnemy.cpp File Reference

7.24 src/Enemies/LargeEnemy.h File Reference

Classes

- class LargeEnemy

7.25 LargeEnemy.h

Go to the documentation of this file.

```
00001 #ifndef LARGE_ENEMY_H
00002 #define LARGE_ENEMY_H
00003
00004 #include "Enemy.h"
00005
00006 class LargeEnemy : public Enemy
00007 {
00008 public:
00015     LargeEnemy(World* worldPtr, const Utils::Vec2d& position = Utils::Vec2d::Zero()) : Enemy(worldPtr,
      50.0, 1.0, position) { }
00016
00022     void Update(double dt) override;
00026     void Draw() const override;
00027
00033     double GetSize() const override { return Size; }
00039     double GetMoveForce() const override { return MoveForce; }
00045     Col GetColor() const override { return Color; }
00046
00047 private:
00048     static const double MoveForce;
00049     static const double Size;
00050     static const Col Color;
00051 };
00052
00053 #endif
```

7.26 src/Enemies/SmallEnemy.cpp File Reference

7.27 src/Enemies/SmallEnemy.h File Reference

Classes

- class SmallEnemy

7.28 SmallEnemy.h

Go to the documentation of this file.

```
00001 #ifndef SMALL_ENEMY_H
00002 #define SMALL_ENEMY_H
00003
00004 #include "Enemy.h"
00005
00006 class SmallEnemy : public Enemy
00007 {
00008 public:
00015     SmallEnemy(World* worldPtr, const Utils::Vec2d& position = Utils::Vec2d::Zero()) : Enemy(worldPtr,
      20.0, 1.0, position) { }
00016
```

```
00022     void Update(double dt) override;
00026     void Draw() const override;
00027
00033     double GetSize() const override { return Size; }
00039     double GetMoveForce() const override { return MoveForce; }
00045     Col GetColor() const override { return Color; }
00046
00047 private:
00048     static const double MoveForce;
00049     static const double Size;
00050     static const Col Color;
00051 };
00052
00053 #endif
```

7.29 src/GameObjects/GameObject.cpp File Reference

7.30 src/GameObjects/GameObject.h File Reference

Classes

- class GameObject

7.31 GameObject.h

Go to the documentation of this file.

```
00001 #ifndef GAMEOBJECT_H
00002 #define GAMEOBJECT_H
00003
00004 #include "Utils/Vec2.h"
00005
00006 class World; // forward declaration
00007 class GameObject
00008 {
00009 public:
00019     GameObject(World* worldPtr, double health = 0.0, double mass = 0.0, const Utils::Vec2d& position =
      Utils::Vec2d::Zero(), double friction = 0.00)
00020         : WorldPtr(worldPtr), MaxHealth(health), Health(health), Alive(true), Mass(mass),
      Position(position), Rotation(0.0), Force(Utils::Vec2d::Zero()), Acceleration(Utils::Vec2d::Zero()),
      Velocity(Utils::Vec2d::Zero()), Friction(friction) { }
00024     virtual ~GameObject() = default;
00025
00031     Utils::Vec2d GetPosition() const { return Position; }
00037     double GetRotation() const { return Rotation; }
00038
00044     bool IsAlive() const { return Alive; }
00050     void TakeDamage(double damage) { Health -= damage; if (Health <= 0.0) Destroy(); }
00054     void Destroy() { Alive = false; }
00055
00061     void AddForce(const Utils::Vec2d& force) { Force += force; }
00067     void LookAt(const Utils::Vec2d& target);
00068
00074     virtual void Update(double dt);
00078     virtual void Draw() const = 0;
00079
00080 protected:
00081     void DrawHealthBar(const Utils::Vec2d& position, const Utils::Vec2d& size) const;
00082
00083 protected:
00084     World* WorldPtr;
00085     double MaxHealth;
00086     double Health;
00087     bool Alive;
00088
00089     double Mass;
00090     Utils::Vec2d Position;
00091     double Rotation;
00092
00093     Utils::Vec2d Force;
00094     Utils::Vec2d Acceleration;
00095     Utils::Vec2d Velocity;
00096
00097     double Friction; // slows down the object (0.0 - 1.0)
00098 };
00099
00100 #endif
```

7.32 src/GameObjects/Player.cpp File Reference

7.33 src/GameObjects/Player.h File Reference

Classes

- class Player

7.34 Player.h

Go to the documentation of this file.

```
00001 #ifndef PLAYER_H
00002 #define PLAYER_H
00003
00004 #include "GameObject.h"
00005 #include "ParticleSystem/ParticleSystem.h"
00006
00007 class Player : public GameObject
00008 {
00009 public:
00016     Player(World* worldPtr, const Utils::Vec2d& position = Utils::Vec2d::Zero())
00017         : GameObject(worldPtr, 100.0, 1.0, position, 0.05), CoinCount(0), Fume() { }
00021     ~Player() = default;
00022
00028     void Update(double dt) override;
00032     void Draw() const override;
00033
00039     void AddCoin(size_t count = 1) { CoinCount += count; }
00045     size_t GetCoinCount() const { return CoinCount; }
00046
00047 private:
00048     void Shoot() const;
00049
00050 private:
00051     static const double MoveForce;
00052     static const Col Color;
00053     static const double BulletSpeed;
00054
00055 private:
00056     size_t CoinCount;
00057     ParticleSystem Fume;
00058 };
00059
00060
00061
00062 #endif
```

7.35 src/main.cpp File Reference

7.35.1 Function Documentation

7.35.1.1 main()

```
int main ( )
```

7.36 src/ParticleSystem/ParticleSystem.cpp File Reference

7.37 src/ParticleSystem/ParticleSystem.h File Reference

Classes

- class ParticleSystem

7.38 ParticleSystem.h

Go to the documentation of this file.

```
00001 #ifndef PARTICLESYSTEM_H
00002 #define PARTICLESYSTEM_H
00003
00004 #include "Utils/Vec2.h"
00005 #include "Color/Color.h"
00006
00007 #include <vector>
00008
00009 class ParticleSystem
00010 {
00011 public:
00015     ParticleSystem();
00019     ~ParticleSystem() = default;
00020
00026     size_t GetParticleCount() const { return Particles.size(); }
00040     void Emit(const Utils::Vec2d& position, const Utils::Vec2d& direction, double randomAngle, double
      lifetime, const Col& startColor, const Col& endColor, double startSize, double endSize, size_t count);
00041
00047     void Update(double dt);
00051     void Draw() const;
00052
00053 private:
00054     struct Particle
00055     {
00056         Particle(const Utils::Vec2d& position, const Utils::Vec2d& velocity, double lifetime, const
      Col& startColor, const Col& endColor, double startSize, double endSize);
00057
00058         void Update(double dt);
00059         void Draw() const;
00060
00061     public:
00062         Utils::Vec2d Position;
00063         Utils::Vec2d Velocity;
00064         double Age;
00065
00066         double Lifetime;
00067         Col StartColor, EndColor;
00068         double StartSize, EndSize;
00069     };
00070
00071 private:
00072     std::vector<Particle> Particles;
00073 };
00074
00075 #endif
```

7.39 src/Utils/Vec2.cpp File Reference

Namespaces

- namespace Utils

7.40 src/Utils/Vec2.h File Reference

Classes

- class Utils::Vec2< T >

Namespaces

- namespace Utils

Functions

- template<typename T >
  std::ostream & Utils::operator<< (std::ostream &os, const Vec2< T > &vec)

7.41 Vec2.h

Go to the documentation of this file.

```
00001 #ifndef VEC2_H
00002 #define VEC2_H
00003
00004 #include <raylib.h>
00005 #include <iostream>
00006 #include <math.h>
00007
00008 namespace Utils
00009 {
00010     template<typename T>
00011     class Vec2
00012     {
00013     public:
00017         Vec2() : X(0), Y(0) {}
00024         Vec2(T x, T y) : X(x), Y(y) {}
00025         ~Vec2() = default;
00026
00032         T GetX() const { return X; }
00038         T GetY() const { return Y; }
00044         double GetAngle() const { return atan2(Y, X) * RAD2DEG; }
00050         double GetLengthSquared() const { return X * X + Y * Y; }
00056         double GetLength() const { return sqrt(GetLengthSquared()); }
00062         Vec2 GetNormalized() const { return *this / GetLength(); }
00063
00069         void SetX(T x) { X = x; }
00075         void SetY(T y) { Y = y; }
00076
00083         Vec2 Rotate(double angle) const
00084         {
00085             double a = GetAngle() + angle;
00086             double l = GetLength();
00087             return Vec2(cos(a * DEG2RAD) * l, sin(a * DEG2RAD) * l);
00088         }
00089
00096         Vec2 operator+(const Vec2& other) const { return Vec2(X + other.X, Y + other.Y); }
00103         Vec2 operator-(const Vec2& other) const { return Vec2(X - other.X, Y - other.Y); }
00110         T operator*(const Vec2& other) const { return X * other.X + Y * other.Y; }
00117         T operator^(const Vec2& other) const { return X * other.Y - Y * other.X; }
00124         Vec2 operator*(const T& other) const { return Vec2(X * other, Y * other); }
00131         Vec2 operator/(const T& other) const { return Vec2(X / other, Y / other); }
00137         Vec2 operator-() const { return Vec2(-X, -Y); }
00138
00145         Vec2& operator+=(const Vec2& other)
00146         {
00147             X += other.X;
00148             Y += other.Y;
00149             return *this;
00150         }
00157         Vec2& operator-=(const Vec2& other)
00158         {
00159             X -= other.X;
00160             Y -= other.Y;
00161             return *this;
00162         }
00169         Vec2& operator*=(const Vec2& other)
00170         {
00171             X *= other.X;
00172             Y *= other.Y;
00173             return *this;
00174         }
00181         Vec2& operator/=(const Vec2& other)
00182         {
00183             X /= other.X;
00184             Y /= other.Y;
00185             return *this;
00186         }
00193         Vec2& operator*=(const T& other)
00194         {
00195             X *= other;
00196             Y *= other;
00197             return *this;
00198         }
```

```
00205         Vec2& operator/=(const T& other)
00206         {
00207             X /= other;
00208             Y /= other;
00209             return *this;
00210         }
00211         operator Vector2() const { return Vector2{ (float)X, (float)Y }; }
00212
00213     public:
00220         static Vec2 FromAngle(double angle) { return Vec2(cos(angle * DEG2RAD), sin(angle * DEG2RAD));
     }
00226         static Vec2 Zero() { return Vec2(0, 0); }
00232         static Vec2 One() { return Vec2(1, 1); }
00238         static Vec2 Up() { return Vec2(0, -1); }
00244         static Vec2 Down() { return Vec2(0, 1); }
00250         static Vec2 Left() { return Vec2(-1, 0); }
00256         static Vec2 Right() { return Vec2(1, 0); }
00257
00258     private:
00259         T X;
00260         T Y;
00261     };
00262
00270     template<typename T>
00271     std::ostream& operator«(std::ostream& os, const Vec2<T>& vec)
00272     {
00273         os « "(" « vec.GetX() « ", " « vec.GetY() « ")";
00274         return os;
00275     }
00276
00277     typedef Vec2<int> Vec2i;
00278     typedef Vec2<double> Vec2d;
00279
00280     Vec2d Vector2ToVec2(const Vector2& vec);
00281 }
00282
00283 #endif
```

7.42 src/World/World.cpp File Reference

7.43 src/World/World.h File Reference

Classes

- class World

7.44 World.h

Go to the documentation of this file.

```
00001 #ifndef WORLD_H
00002 #define WORLD_H
00003
00004 #include <vector>
00005 #include <memory>
00006
00007 #include "GameObjects/Player.h"
00008 #include "Collectables/Collectable.h"
00009 #include "Enemies/Enemy.h"
00010 #include "Bullets/Bullet.h"
00011
00012 class World
00013 {
00014 public:
00018     World();
00019
00025     void AddCollectable(std::unique_ptr<Collectable> collectable) {
      Collectables.push_back(std::move(collectable)); }
00031     void AddEnemy(std::unique_ptr<Enemy> enemy) { Enemies.push_back(std::move(enemy)); }
00037     void AddBullet(std::unique_ptr<Bullet> bullet) { Bullets.push_back(std::move(bullet)); }
00038
00044     void Update(double dt);
```

```
00048     void Draw() const;
00049
00055     Player* GetMainPlayer() { return MainPlayer.get(); }
00056
00057 private:
00058     std::unique_ptr<Player> MainPlayer;
00059     std::vector<std::unique_ptr<Collectable» Collectables;
00060     std::vector<std::unique_ptr<Enemy» Enemies;
00061     std::vector<std::unique_ptr<Bullet» Bullets;
00062 };
00063
00064 #endif
```

Index

~Bullet
Bullet, 14
~Collectable
Collectable, 21
~Enemy
Enemy, 23
~GameObject
GameObject, 25
~ParticleSystem
ParticleSystem, 33
~Player
Player, 35
~Vec2
Utils::Vec2< T >, 40

A
Col, 20
Acceleration
GameObject, 29
AddBullet
World, 48
AddCoin
Player, 35
AddCollectable
World, 49
AddEnemy
World, 49
AddForce
GameObject, 25
Alive
GameObject, 29
ASSET_COUNT
AssetManager, 9
AssetManager, 9
ASSET_COUNT, 9
GetTexture, 9
LARGE_ENEMY, 9
LoadAssets, 10
SMALL_ENEMY, 9
TextureType, 9
UnloadAssets, 10

B
Col, 20
Bullet, 13
~Bullet, 14
Bullet, 14
BulletTarget, 13
ENEMY, 13
GetSize, 14
GetTarget, 14
PLAYER, 13
Target, 15
Update, 14
BulletTarget
Bullet, 13

CheckCollision
Collisions.cpp, 54, 55
Collisions.h, 56, 57
CircleBullet, 15
CircleBullet, 15
Draw, 16
GetSize, 16
Coin, 16
Coin, 17
Draw, 17
GetSize, 17
Update, 17
Col, 18
A, 20
B, 20
Col, 18, 19
Fade, 19
G, 20
Lerp, 19
operator Color, 19
R, 20
Collectable, 20
~Collectable, 21
Collectable, 21
Collected, 22
GetSize, 21
IsCollected, 21
SetCollected, 22
Collected
Collectable, 22
Collisions.cpp
CheckCollision, 54, 55
Collisions.h
CheckCollision, 56, 57

Destroy
GameObject, 27
Down
Utils::Vec2< T >, 40
Draw
CircleBullet, 16
Coin, 17
GameObject, 27

LargeEnemy, 31
ParticleSystem, 33
Player, 35
SmallEnemy, 37
World, 49
DrawHealthBar
GameObject, 27

Emit
ParticleSystem, 33
ENEMY
Bullet, 13
Enemy, 22
~Enemy, 23
Enemy, 23
GetColor, 23
GetMoveForce, 23
GetSize, 24
SetTarget, 24
Target, 24

Fade
Col, 19
Force
GameObject, 29
Friction
GameObject, 29
FromAngle
Utils::Vec2< T >, 40

G
Col, 20
GameObject, 25
~GameObject, 25
Acceleration, 29
AddForce, 25
Alive, 29
Destroy, 27
Draw, 27
DrawHealthBar, 27
Force, 29
Friction, 29
GameObject, 25
GetPosition, 27
GetRotation, 27
Health, 30
IsAlive, 27
LookAt, 28
Mass, 30
MaxHealth, 30
Position, 30
Rotation, 30
TakeDamage, 29
Update, 29
Velocity, 30
WorldPtr, 30
GetAngle
Utils::Vec2< T >, 40
GetCoinCount
Player, 36
GetColor
Enemy, 23
LargeEnemy, 31
SmallEnemy, 37
GetLength
Utils::Vec2< T >, 41
GetLengthSquared
Utils::Vec2< T >, 41
GetMainPlayer
World, 49
GetMoveForce
Enemy, 23
LargeEnemy, 32
SmallEnemy, 37
GetNormalized
Utils::Vec2< T >, 41
GetParticleCount
ParticleSystem, 34
GetPosition
GameObject, 27
GetRotation
GameObject, 27
GetSize
Bullet, 14
CircleBullet, 16
Coin, 17
Collectable, 21
Enemy, 24
LargeEnemy, 32
SmallEnemy, 38
GetTarget
Bullet, 14
GetTexture
AssetManager, 9
GetX
Utils::Vec2< T >, 41
GetY
Utils::Vec2< T >, 41

Health
GameObject, 30

IsAlive
GameObject, 27
IsCollected
Collectable, 21

LARGE_ENEMY
AssetManager, 9
LargeEnemy, 30
Draw, 31
GetColor, 31
GetMoveForce, 32
GetSize, 32
LargeEnemy, 31
Update, 32
Left
Utils::Vec2< T >, 42

Lerp
Col, 19
LoadAssets
AssetManager, 10
LookAt
GameObject, 28

main
main.cpp, 61
main.cpp
main, 61
Mass
GameObject, 30
MaxHealth
GameObject, 30

One
Utils::Vec2< T >, 42
operator Color
Col, 19
operator Vector2
Utils::Vec2< T >, 42
operator<<
Utils, 10
operator+
Utils::Vec2< T >, 44
operator+=
Utils::Vec2< T >, 44
operator-
Utils::Vec2< T >, 44
operator-=
Utils::Vec2< T >, 45
operator/
Utils::Vec2< T >, 45
operator/=
Utils::Vec2< T >, 45, 46
operator*
Utils::Vec2< T >, 42, 43
operator*=
Utils::Vec2< T >, 43
operator^
Utils::Vec2< T >, 46

ParticleSystem, 33
~ParticleSystem, 33
Draw, 33
Emit, 33
GetParticleCount, 34
ParticleSystem, 33
Update, 34
PLAYER
Bullet, 13
Player, 34
~Player, 35
AddCoin, 35
Draw, 35
GetCoinCount, 36
Player, 35
Update, 36

Position
GameObject, 30

R
Col, 20
Right
Utils::Vec2< T >, 46
Rotate
Utils::Vec2< T >, 47
Rotation
GameObject, 30

SetCollected
Collectable, 22
SetTarget
Enemy, 24
SetX
Utils::Vec2< T >, 47
SetY
Utils::Vec2< T >, 47
SMALL_ENEMY
AssetManager, 9
SmallEnemy, 36
Draw, 37
GetColor, 37
GetMoveForce, 37
GetSize, 38
SmallEnemy, 37
Update, 38
src/AssetManager/AssetManager.cpp, 51
src/AssetManager/AssetManager.h, 51
src/Bullets/Bullet.cpp, 52
src/Bullets/Bullet.h, 52
src/Bullets/CircleBullet.cpp, 52
src/Bullets/CircleBullet.h, 52
src/Collectables/Coin.cpp, 53
src/Collectables/Coin.h, 53
src/Collectables/Collectable.h, 53, 54
src/Collisions/Collisions.cpp, 54
src/Collisions/Collisions.h, 56, 57
src/Color/Color.cpp, 58
src/Color/Color.h, 58
src/Enemies/Enemy.h, 58
src/Enemies/LargeEnemy.cpp, 59
src/Enemies/LargeEnemy.h, 59
src/Enemies/SmallEnemy.cpp, 59
src/Enemies/SmallEnemy.h, 59
src/GameObjects/GameObject.cpp, 60
src/GameObjects/GameObject.h, 60
src/GameObjects/Player.cpp, 61
src/GameObjects/Player.h, 61
src/main.cpp, 61
src/ParticleSystem/ParticleSystem.cpp, 61
src/ParticleSystem/ParticleSystem.h, 61, 62
src/Utils/Vec2.cpp, 62
src/Utils/Vec2.h, 62, 63
src/World/World.cpp, 64
src/World/World.h, 64

TakeDamage
GameObject, 29
Target
Bullet, 15
Enemy, 24
TextureType
AssetManager, 9

UnloadAssets
AssetManager, 10
Up
Utils::Vec2< T >, 48
Update
Bullet, 14
Coin, 17
GameObject, 29
LargeEnemy, 32
ParticleSystem, 34
Player, 36
SmallEnemy, 38
World, 49
Utils, 10
operator<<, 10
Vec2d, 10
Vec2i, 10
Vector2ToVec2, 11
Utils::Vec2< T >, 38
~Vec2, 40
Down, 40
FromAngle, 40
GetAngle, 40
GetLength, 41
GetLengthSquared, 41
GetNormalized, 41
GetX, 41
GetY, 41
Left, 42
One, 42
operator Vector2, 42
operator+, 44
operator+=, 44
operator-, 44
operator-=, 45
operator/, 45
operator/=, 45, 46
operator*, 42, 43
operator*=, 43
operator^, 46
Right, 46
Rotate, 47
SetX, 47
SetY, 47
Up, 48
Vec2, 39
Zero, 48

Vec2
Utils::Vec2< T >, 39
Vec2d
Utils, 10
Vec2i
Utils, 10
Vector2ToVec2
Utils, 11
Velocity
GameObject, 30

World, 48
AddBullet, 48
AddCollectable, 49
AddEnemy, 49
Draw, 49
GetMainPlayer, 49
Update, 49
World, 48
WorldPtr
GameObject, 30

Zero
Utils::Vec2< T >, 48