CPP dungeon crawler

Generated by Doxygen 1.9.8

1 Game name 1

1.1 Build instructions . . . 1

1.1.0.1 Prerequisites . . . 1

1.1.0.2 Compiling . . . 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 Class Documentation 9

5.1 Boss Class Reference . . . 9

5.1.1 Detailed Description . . . 10

5.1.2 Member Function Documentation . . . 11

5.1.2.1 attack() . . . 11

5.1.2.2 getAttackSound() . . . 11

5.1.2.3 setMove() . . . 11

5.2 Coordinate Struct Reference . . . 11

5.3 Entity Class Reference . . . 11

5.4 Game Class Reference . . . 12

5.5 HealingPotion Class Reference . . . 13

5.6 Hud Class Reference . . . 14

5.7 Input Class Reference . . . 14

5.7.1 Detailed Description . . . 14

5.8 InputMapping Struct Reference . . . 15

5.9 InputState Struct Reference . . . 15

5.10 Item Class Reference . . . 15

5.11 MeleeMob Class Reference . . . 16

5.11.1 Detailed Description . . . 17

5.11.2 Member Function Documentation . . . 18

5.11.2.1 attack() . . . 18

5.11.2.2 setMove() . . . 18

5.12 Monster Class Reference . . . 18

5.12.1 Detailed Description . . . 19

5.13 Pistol Class Reference . . . 19

5.13.1 Detailed Description . . . 21

5.14 Player Class Reference . . . 21

5.15 Projectile Class Reference . . . 22

5.16 RangedMob Class Reference . . . 23

5.16.1 Detailed Description . . . 24

5.16.2 Member Function Documentation . . . 25
5.16.2.1 attack() . . . 25
5.16.2.2 dropItem() . . . 25
5.16.2.3 getAttackSound() . . . 25
5.16.2.4 setMove() . . . 25
5.17 Renderer Class Reference . . . 26
5.18 Room Class Reference . . . 26
5.19 RoomTemplate Struct Reference . . . 27
5.20 Shotgun Class Reference . . . 27
5.20.1 Detailed Description . . . 28
5.20.2 Member Function Documentation . . . 29
5.20.2.1 shoot() . . . 29
5.20.2.2 toString() . . . 29
5.21 SMG Class Reference . . . 29
5.21.1 Detailed Description . . . 30
5.22 SoundSet Struct Reference . . . 31
5.23 Weapon Class Reference . . . 31
5.23.1 Detailed Description . . . 32
5.23.2 Member Function Documentation . . . 32
5.23.2.1 shoot() . . . 32

6 File Documentation **33**

6.1 consumables.hpp . . . 33
6.2 entity.hpp . . . 33
6.3 game.hpp . . . 34
6.4 hud.hpp . . . 34
6.5 input.hpp . . . 35
6.6 item.hpp . . . 35
6.7 monster.hpp . . . 36
6.8 player.hpp . . . 37
6.9 renderer.hpp . . . 38
6.10 room.hpp . . . 38
6.11 weapon.hpp . . . 39

Index **41**

Chapter 1

Game name

This is a rogue-like dungeon crawler top-down shooter written in c++. In the game, the player has to get through a set of randomized rooms with increasingly difficult random monsters with random weapons until they reach the final boss. Defeating the boss ends the game. The game is reset upon player death.

1.1 Build instructions

1.1.0.1 Prerequisites

Compiling the project requires gcc and make.

The following libraries also to be installed on your system:

- sdl2
- sdl2_image
- sdl2_ttf
- sdl2_mixer

1.1.0.2 Compiling

Run `make` in the project root.

`make run` starts the game.

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

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

- Coordinate . . . 11
- Entity . . . 11
  - Monster . . . 18
    - Boss . . . 9
    - MeleeMob . . . 16
    - RangedMob . . . 23
  - Player . . . 21
  - Projectile . . . 22
- Game . . . 12
- Hud . . . 14
- Input . . . 14
- InputMapping . . . 15
- InputState . . . 15
- Item . . . 15
  - HealingPotion . . . 13
  - Weapon . . . 31
    - Pistol . . . 19
    - SMG . . . 29
    - Shotgun . . . 27
- Renderer . . . 26
- Room . . . 26
- RoomTemplate . . . 27
- SoundSet . . . 31

Chapter 3

Class Index

3.1 Class List

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

Boss . . . 9
Coordinate . . . 11
Entity . . . 11
Game . . . 12
HealingPotion . . . 13
Hud . . . 14
Input
    A class to hande inputs . . . 14
InputMapping . . . 15
InputState . . . 15
Item . . . 15
MeleeMob . . . 16
Monster . . . 18
Pistol
    Pistol class. High damage, slow firerate . . . 19
Player . . . 21
Projectile . . . 22
RangedMob . . . 23
Renderer . . . 26
Room . . . 26
RoomTemplate . . . 27
Shotgun
    Shotgun class. Multiple low damage projectiles, low firerate . . . 27
SMG
    SMG class. Low damage, high firerate . . . 29
SoundSet . . . 31
Weapon
    Base class for weapon . . . 31

Chapter 4

File Index

4.1 File List

Here is a list of all documented files with brief descriptions:

consumables.hpp 33
entity.hpp 33
game.hpp 34
hud.hpp 34
input.hpp 35
item.hpp 35
monster.hpp 36
player.hpp 37
renderer.hpp 38
room.hpp 38
weapon.hpp 39

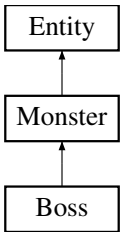
Chapter 5

Class Documentation

5.1 Boss Class Reference

`#include <monster.hpp>`

Inheritance diagram for Boss:



Public Member Functions

- **Boss** (int)
- bool attack (Player &, std::list< Projectile > &)
- void setMove (Player &)
- Mix_Chunk * getAttackSound ()

Public Member Functions inherited from Monster

- **Monster** (const std::string &, int, int, int, int, int, int, int)
- int **GetHP** ()
- int **GetDMG** ()
- void **TakeDMG** (int)
- bool **isAlive** ()
- std::string **getName** ()
- virtual void **dropItem** (std::list< Weapon * > &)

Public Member Functions inherited from Entity

- **Entity** (int, int, int, int)
- void **move** ()
- Coordinate **newPos** ()
- Coordinate **center** ()
- Coordinate **newCenter** ()
- bool **collidesWith** (Entity &)

Public Attributes

- int **attack_pattern_**
- Weapon * **weapon_**
- int **attack_ticks_**
- int **attack_cooldown_**
- int **optimal_distance_**

Public Attributes inherited from Monster

- SoundSet **sounds_**

Public Attributes inherited from Entity

- int **x_**
- int **y_**
- int **size_x_**
- int **size_y_**
- int **speed_**
- float **direction_**
- SDL_Texture * **texture_**

Additional Inherited Members

Protected Attributes inherited from Monster

- bool **alive_**
- int **hp_**
- int **dmg_**
- int **max_speed_**
- std::string **name_**

5.1.1 Detailed Description

Final boss

- Combines melee and ranged attack patterns

5.1.2 Member Function Documentation

5.1.2.1 attack()

```
bool Boss::attack (
            Player & p,
            std::list< Projectile > & projectiles )  [virtual]
```

Reimplemented from Monster.

5.1.2.2 getAttackSound()

```
Mix_Chunk * Boss::getAttackSound ( )  [virtual]
```

Reimplemented from Monster.

5.1.2.3 setMove()

```
void Boss::setMove (
            Player & p )  [virtual]
```

Reimplemented from Monster.

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

- monster.hpp
- monster.cpp

5.2 Coordinate Struct Reference

Public Attributes

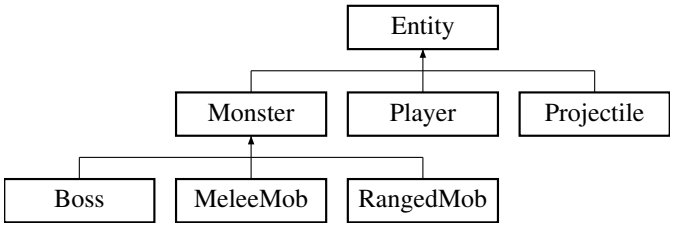
- int **x**
- int **y**

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

- entity.hpp

5.3 Entity Class Reference

Inheritance diagram for Entity:



Public Member Functions

- **Entity** (int, int, int, int)
- void **move** ()
- Coordinate **newPos** ()
- Coordinate **center** ()
- Coordinate **newCenter** ()
- bool **collidesWith** (Entity &)

Public Attributes

- int **x_**
- int **y_**
- int **size_x_**
- int **size_y_**
- int **speed_**
- float **direction_**
- SDL_Texture * **texture_**

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

- entity.hpp
- entity.cpp

5.4 Game Class Reference

Public Member Functions

- void **movePlayer** (InputState &)
- void **spawnProjectile** (int, int, int, int, int, float, SDL_Texture *)
- void **moveProjectiles** (Renderer &)

  All projectiles move. Deal damage to entities that are hit.
- void **moveMonsters** (Renderer &)

  All monsters move according to their movement pattern.
- void **parseInput** (Renderer &)
- int **tick** (Renderer &)

  A standard game cycle.
- void **render** (Renderer &)

  Renders the game.
- void **changeRoom** (Renderer &)

  Creates a new room.
- void **calcOffset** (Renderer &)

  Calculates camera offset.
- void **scanNear** (Renderer &)

  Scans nearby objects to display in info text.
- void **menuTick** (Renderer &)
- void **menuRender** (Renderer &)
- Weapon * **scanWeapons** (Renderer &)

  Checks if weapons are nearby to pick up.

Public Attributes

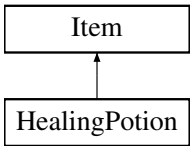
- std::list< Room > **room_templates_**
- Room * **room_**
- Room * **room1_**
- bool **running_**
- Input **input_**
- int **x_offset_**
- int **y_offset_**
- std::string **infoText**
- int **game_level_**
- Hud **hud_**
- Player **player_**
- std::list< Projectile > **projectiles_**
- Weapon * **displayWeapon_**
- bool **paused_**

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

- game.hpp
- game.cpp

5.5 HealingPotion Class Reference

Inheritance diagram for HealingPotion:



Public Member Functions

- **HealingPotion** (std::string name, int size, int healing)
- int **getHealing** ()

Public Member Functions inherited from Item

- **Item** (std::string name, int size)
- void **equip** ()
- int **getSize** ()

Additional Inherited Members

Public Attributes inherited from Item

- SDL_Texture * **texture_**
- int **x_**
- int **y_**

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

- consumables.hpp

5.6 Hud Class Reference

Public Member Functions

- **Hud** (int hudposx, int hudposy)
- void **drawInfo** (Renderer &, int level, int health, int maxHp, int room)

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

- hud.hpp
- hud.cpp

5.7 Input Class Reference

A class to hande inputs.

`#include <input.hpp>`

Public Member Functions

- int **scan** ()
- void **keyDown** (SDL_KeyboardEvent *)
- void **keyUp** (SDL_KeyboardEvent *)
- void **resetInput** ()
- void **resetInteract** ()
- InputState **getState** ()

5.7.1 Detailed Description

A class to hande inputs.

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

- input.hpp
- input.cpp

5.8 InputMapping Struct Reference

Public Attributes

- uint32_t **up**
- uint32_t **down**
- uint32_t **left**
- uint32_t **right**
- uint32_t **attack**
- uint32_t **interact**
- uint32_t **menu**
- uint32_t **attackUp**
- uint32_t **attackDown**
- uint32_t **attackLeft**
- uint32_t **attackRight**

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

- input.hpp

5.9 InputState Struct Reference

Public Attributes

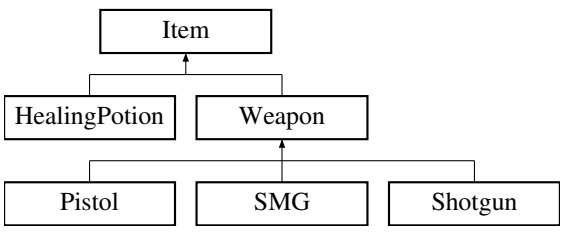
- bool **up**
- bool **down**
- bool **left**
- bool **right**
- bool **attack**
- bool **interact**
- bool **menu**
- bool **attackUp**
- bool **attackDown**
- bool **attackLeft**
- bool **attackRight**

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

- input.hpp

5.10 Item Class Reference

Inheritance diagram for Item:



Public Member Functions

- **Item** (std::string name, int size)
- void **equip** ()
- int **getSize** ()

Public Attributes

- SDL_Texture * **texture_**
- int **x_**
- int **y_**

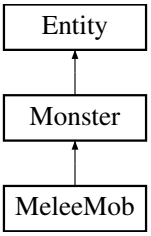
The documentation for this class was generated from the following file:

- item.hpp

5.11 MeleeMob Class Reference

`#include <monster.hpp>`

Inheritance diagram for MeleeMob:



Public Member Functions

- **MeleeMob** (int, int, int, int, int)
- bool attack (Player &, std::list< Projectile > &)
- void setMove (Player &)

Public Member Functions inherited from Monster

- **Monster** (const std::string &, int, int, int, int, int, int, int)
- int **GetHP** ()
- int **GetDMG** ()
- void **TakeDMG** (int)
- bool **isAlive** ()
- std::string **getName** ()
- virtual void **dropItem** (std::list< Weapon * > &)
- virtual Mix_Chunk * **getAttackSound** ()

Public Member Functions inherited from Entity

- **Entity** (int, int, int, int)
- void **move** ()
- Coordinate **newPos** ()
- Coordinate **center** ()
- Coordinate **newCenter** ()
- bool **collidesWith** (Entity &)

Public Attributes

- int **attack_ticks_**
- int **attack_cooldown_**

Public Attributes inherited from Monster

- SoundSet **sounds_**

Public Attributes inherited from Entity

- int **x_**
- int **y_**
- int **size_x_**
- int **size_y_**
- int **speed_**
- float **direction_**
- SDL_Texture * **texture_**

Additional Inherited Members

Protected Attributes inherited from Monster

- bool **alive_**
- int **hp_**
- int **dmg_**
- int **max_speed_**
- std::string **name_**

5.11.1 Detailed Description

Basic melee monster

- Moves towards the player
- Deals melee damage when close

5.11.2 Member Function Documentation

5.11.2.1 attack()

```
bool MeleeMob::attack (
            Player & p,
            std::list< Projectile > &  )  [virtual]
```

Reimplemented from Monster.

5.11.2.2 setMove()

```
void MeleeMob::setMove (
            Player & p )  [virtual]
```

Reimplemented from Monster.

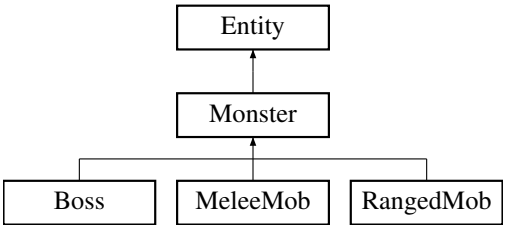
The documentation for this class was generated from the following files:

- monster.hpp
- monster.cpp

5.12 Monster Class Reference

```
#include <monster.hpp>
```

Inheritance diagram for Monster:



Public Member Functions

- **Monster** (const std::string &, int, int, int, int, int, int, int)
- int **GetHP** ()
- int **GetDMG** ()
- void **TakeDMG** (int)
- bool **isAlive** ()
- std::string **getName** ()
- virtual void **setMove** (Player &)
- virtual bool **attack** (Player &, std::list< Projectile > &)
- virtual void **dropItem** (std::list< Weapon * > &)
- virtual Mix_Chunk * **getAttackSound** ()

Public Member Functions inherited from Entity

- **Entity** (int, int, int, int)
- void **move** ()
- Coordinate **newPos** ()
- Coordinate **center** ()
- Coordinate **newCenter** ()
- bool **collidesWith** (Entity &)

Public Attributes

- SoundSet **sounds_**

Public Attributes inherited from Entity

- int **x_**
- int **y_**
- int **size_x_**
- int **size_y_**
- int **speed_**
- float **direction_**
- SDL_Texture * **texture_**

Protected Attributes

- bool **alive_**
- int **hp_**
- int **dmg_**
- int **max_speed_**
- std::string **name_**

5.12.1 Detailed Description

Base class for monster

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

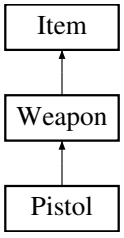
- monster.hpp
- monster.cpp

5.13 Pistol Class Reference

Pistol class. High damage, slow firerate.

`#include <weapon.hpp>`

Inheritance diagram for Pistol:



Public Member Functions

- **Pistol** (const std::string &name, int size, int dmg, int pspeed, int firerate)

Public Member Functions inherited from Weapon

- **Weapon** (const std::string &name, int size, int dmg, int pspeed, int firerate)
- int **getDmg** ()
- int **getProjectileSpeed** ()
- int **getFirerate** ()
- std::string **getName** ()
- virtual void shoot (std::list< Projectile > &projectiles, Entity source, int dmg, float direction, bool damage_↩ monsters)

  Creates a projectile and inserts it into projectiles list.
- virtual std::string **toString** ()

Public Member Functions inherited from Item

- **Item** (std::string name, int size)
- void **equip** ()
- int **getSize** ()

Additional Inherited Members

Public Attributes inherited from Weapon

- SDL_Texture * **texture_**
- SDL_Texture * **projectile_texture_**
- Mix_Chunk * **sound_**

Public Attributes inherited from Item

- SDL_Texture * **texture_**
- int **x_**
- int **y_**

Protected Attributes inherited from Weapon

- int **dmg_**
- int **projectile_speed_**
- int **firerate_**
- int **projectile_size_x_** = 10
- int **projectile_size_y_** = 10
- std::string **name_**

5.13.1 Detailed Description

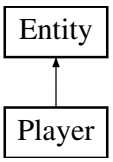
Pistol class. High damage, slow firerate.

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

- weapon.hpp
- weapon.cpp

5.14 Player Class Reference

Inheritance diagram for Player:



Public Member Functions

- **Player** (const std::string &, int, int)
- const std::string **GetName** () const
- int **GetHP** ()
- int **GetXP** ()
- int **GetDMG** ()
- int **GetMaxSpeed** ()
- int **GetLevel** ()
- int **getMaxHp** ()
- void **Heal** (int)
- void **TakeDMG** (int)
- void **UpdateXP** (int)
- void **UpdateDMG** (int)
- void **setMove** (InputState &)
- bool **attack** (InputState &, std::list< Projectile > &)
- float **getAttackDirection** ()
- bool **gainXP** (int)
- bool **isAlive** ()
- void **equipWeapon** (Weapon *, Renderer &r)

Public Member Functions inherited from Entity

- **Entity** (int, int, int, int)
- void **move** ()
- Coordinate **newPos** ()
- Coordinate **center** ()
- Coordinate **newCenter** ()
- bool **collidesWith** (Entity &)

Public Attributes

- Weapon * **weapon_**
- SDL_Texture * **texture_front_**
- SDL_Texture * **texture_right_**
- SDL_Texture * **texture_left_**
- int **shoot_ticks_**
- SoundSet **sounds_**

Public Attributes inherited from Entity

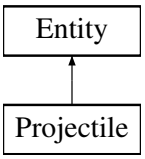
- int **x_**
- int **y_**
- int **size_x_**
- int **size_y_**
- int **speed_**
- float **direction_**
- SDL_Texture * **texture_**

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

- player.hpp
- player.cpp

5.15 Projectile Class Reference

Inheritance diagram for Projectile:



Public Member Functions

- **Projectile** (int x, int y, int size_x, int size_y, int dmg, float direction, int speed)

Public Member Functions inherited from Entity

- **Entity** (int, int, int, int)
- void **move** ()
- Coordinate **newPos** ()
- Coordinate **center** ()
- Coordinate **newCenter** ()
- bool **collidesWith** (Entity &)

Public Attributes

- int **dmg_**
- bool **damage_monsters_**

Public Attributes inherited from Entity

- int **x_**
- int **y_**
- int **size_x_**
- int **size_y_**
- int **speed_**
- float **direction_**
- SDL_Texture * **texture_**

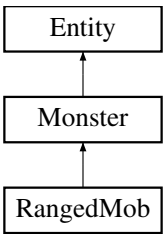
The documentation for this class was generated from the following file:

- weapon.hpp

5.16 RangedMob Class Reference

`#include <monster.hpp>`

Inheritance diagram for RangedMob:



Public Member Functions

- **RangedMob** (int, int, int, int, int, Weapon *)
- bool attack (Player &, std::list< Projectile > &)
- void setMove (Player &)
- void dropItem (std::list< Weapon * > &)
- Mix_Chunk * getAttackSound ()

Public Member Functions inherited from Monster

- **Monster** (const std::string &, int, int, int, int, int, int, int)
- int **GetHP** ()
- int **GetDMG** ()
- void **TakeDMG** (int)
- bool **isAlive** ()
- std::string **getName** ()

Public Member Functions inherited from Entity

- **Entity** (int, int, int, int)
- void **move** ()
- Coordinate **newPos** ()
- Coordinate **center** ()
- Coordinate **newCenter** ()
- bool **collidesWith** (Entity &)

Public Attributes

- Weapon * **weapon_**
- int **attack_ticks_**
- int **optimal_distance_**

Public Attributes inherited from Monster

- SoundSet **sounds_**

Public Attributes inherited from Entity

- int **x_**
- int **y_**
- int **size_x_**
- int **size_y_**
- int **speed_**
- float **direction_**
- SDL_Texture * **texture_**

Additional Inherited Members

Protected Attributes inherited from Monster

- bool **alive_**
- int **hp_**
- int **dmg_**
- int **max_speed_**
- std::string **name_**

5.16.1 Detailed Description

Basic ranged monster

- Tries to keep at optimal_distance_ from the player
- Deals ranged damage

5.16.2 Member Function Documentation

5.16.2.1 attack()

```
bool RangedMob::attack (
            Player & p,
            std::list< Projectile > & projectiles )  [virtual]
```

Reimplemented from Monster.

5.16.2.2 dropItem()

```
void RangedMob::dropItem (
            std::list< Weapon * > & items )  [virtual]
```

Reimplemented from Monster.

5.16.2.3 getAttackSound()

```
Mix_Chunk * RangedMob::getAttackSound ( )  [virtual]
```

Reimplemented from Monster.

5.16.2.4 setMove()

```
void RangedMob::setMove (
            Player & p )  [virtual]
```

Reimplemented from Monster.

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

- monster.hpp
- monster.cpp

5.17 Renderer Class Reference

Public Member Functions

- **Renderer** (int, int, uint32_t, uint32_t)

  Create renderer. Takes in window width and height and flags.
- void **initSDL** ()

  initializes SDL
- void **prepareScene** ()
- void **presentScene** ()
- SDL_Texture * **loadTexture** (const char *)

  Loads a texture into memory. Takes path to texture file.
- void **drawTexture** (SDL_Texture *, int, int, double, SDL_RendererFlip)

  Draws a texture on the screen.
- void **destroy** ()

  Deinitializes SDL.
- void **set_flags** (uint32_t, uint32_t)
- int **getWinWidth** ()
- int **getWinHeight** ()
- void **draw_text** (const char *str, int x, int y, SDL_Color={255, 255, 255})
- TTF_Font * **GetFont** ()
- void **renderText** (SDL_Surface *text, int x, int y)

  Renders text on the screen.
- SDL_Surface * **InitText** (char *str)
- void **playSound** (Mix_Chunk *, int)

  Plays back a sound.
- Mix_Chunk * **loadSound** (const char *)

  Loads a sound into memory. Takes path to sound file.

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

- renderer.hpp
- renderer.cpp

5.18 Room Class Reference

Public Member Functions

- **Room** (const std::string &, int, int, SDL_Texture *, SDL_Texture *)
- void **addRandomMonsters** (Renderer &, int, int)
- void **addRandomItems** (Renderer &r, int level, int amount)
- void **addAdvanceDoor** ()
- void **addItem** (Item *)

Public Attributes

- std::string **name_**
- SDL_Texture * **texture_**
- SDL_Texture * **advanceDoor_**
- int **advanceDoorX_**
- int **advanceDoorY_**
- int **width_**
- int **height_**
- std::list< Monster * > **monsters_**
- std::list< Item * > **items_**
- std::list< Weapon * > **weapons_**

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

- room.hpp
- room.cpp

5.19 RoomTemplate Struct Reference

Public Attributes

- std::string **name**
- std::string **texture_location**
- int **width**
- int **height**
- int **mobs_min**
- int **mobs_max**

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

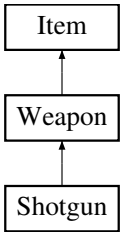
- room.hpp

5.20 Shotgun Class Reference

Shotgun class. Multiple low damage projectiles, low firerate.

`#include <weapon.hpp>`

Inheritance diagram for Shotgun:



Public Member Functions

- **Shotgun** (const std::string &name, int size, int dmg, int pspeed, int firerate, int pellets, float spread)
- void shoot (std::list< Projectile > &projectiles, Entity source, int dmg, float direction, bool damage_monsters)

  Creates a projectile and inserts it into projectiles list.
- int **getPellets** ()
- float **getSpread** ()
- std::string toString ()

Public Member Functions inherited from Weapon

- **Weapon** (const std::string &name, int size, int dmg, int pspeed, int firerate)
- int **getDmg** ()
- int **getProjectileSpeed** ()
- int **getFirerate** ()
- std::string **getName** ()

Public Member Functions inherited from Item

- **Item** (std::string name, int size)
- void **equip** ()
- int **getSize** ()

Additional Inherited Members

Public Attributes inherited from Weapon

- SDL_Texture * **texture_**
- SDL_Texture * **projectile_texture_**
- Mix_Chunk * **sound_**

Public Attributes inherited from Item

- SDL_Texture * **texture_**
- int **x_**
- int **y_**

Protected Attributes inherited from Weapon

- int **dmg_**
- int **projectile_speed_**
- int **firerate_**
- int **projectile_size_x_** = 10
- int **projectile_size_y_** = 10
- std::string **name_**

5.20.1 Detailed Description

Shotgun class. Multiple low damage projectiles, low firerate.

5.20.2 Member Function Documentation

5.20.2.1 shoot()

```
void Shotgun::shoot (
            std::list< Projectile > & projectiles,
            Entity source,
            int dmg,
            float direction,
            bool damage_monsters )  [virtual]
```

Creates a projectile and inserts it into projectiles list.

Reimplemented from Weapon.

5.20.2.2 toString()

```
std::string Shotgun::toString ( )  [virtual]
```

Reimplemented from Weapon.

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

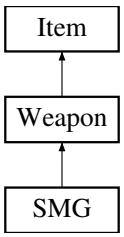
- weapon.hpp
- weapon.cpp

5.21 SMG Class Reference

SMG class. Low damage, high firerate.

```
#include <weapon.hpp>
```

Inheritance diagram for SMG:



Public Member Functions

- **SMG** (const std::string &name, int size, int dmg, int pspeed, int firerate)

Public Member Functions inherited from Weapon

- **Weapon** (const std::string &name, int size, int dmg, int pspeed, int firerate)
- int **getDmg** ()
- int **getProjectileSpeed** ()
- int **getFirerate** ()
- std::string **getName** ()
- virtual void shoot (std::list< Projectile > &projectiles, Entity source, int dmg, float direction, bool damage_↩ monsters)

  Creates a projectile and inserts it into projectiles list.
- virtual std::string **toString** ()

Public Member Functions inherited from Item

- **Item** (std::string name, int size)
- void **equip** ()
- int **getSize** ()

Additional Inherited Members

Public Attributes inherited from Weapon

- SDL_Texture * **texture_**
- SDL_Texture * **projectile_texture_**
- Mix_Chunk * **sound_**

Public Attributes inherited from Item

- SDL_Texture * **texture_**
- int **x_**
- int **y_**

Protected Attributes inherited from Weapon

- int **dmg_**
- int **projectile_speed_**
- int **firerate_**
- int **projectile_size_x_** = 10
- int **projectile_size_y_** = 10
- std::string **name_**

5.21.1 Detailed Description

SMG class. Low damage, high firerate.

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

- weapon.hpp

5.22 SoundSet Struct Reference

Public Attributes

- Mix_Chunk * **attack_**
- Mix_Chunk * **hit_**
- Mix_Chunk * **death_**
- Mix_Chunk * **taunt_**

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

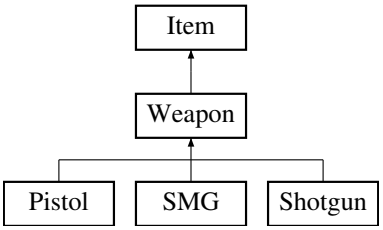
- entity.hpp

5.23 Weapon Class Reference

Base class for weapon.

`#include <weapon.hpp>`

Inheritance diagram for Weapon:



Public Member Functions

- **Weapon** (const std::string &name, int size, int dmg, int pspeed, int firerate)
- int **getDmg** ()
- int **getProjectileSpeed** ()
- int **getFirerate** ()
- std::string **getName** ()
- virtual void shoot (std::list< Projectile > &projectiles, Entity source, int dmg, float direction, bool damage_↩ monsters)

  Creates a projectile and inserts it into projectiles list.
- virtual std::string **toString** ()

Public Member Functions inherited from Item

- **Item** (std::string name, int size)
- void **equip** ()
- int **getSize** ()

Public Attributes

- SDL_Texture * **texture_**
- SDL_Texture * **projectile_texture_**
- Mix_Chunk * **sound_**

Public Attributes inherited from Item

- SDL_Texture * **texture_**
- int **x_**
- int **y_**

Protected Attributes

- int **dmg_**
- int **projectile_speed_**
- int **firerate_**
- int **projectile_size_x_** = 10
- int **projectile_size_y_** = 10
- std::string **name_**

5.23.1 Detailed Description

Base class for weapon.

5.23.2 Member Function Documentation

5.23.2.1 shoot()

```
void Weapon::shoot (
            std::list< Projectile > & projectiles,
            Entity source,
            int dmg,
            float direction,
            bool damage_monsters )  [virtual]
```

Creates a projectile and inserts it into projectiles list.

Reimplemented in Shotgun.

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

- weapon.hpp
- weapon.cpp

Chapter 6

File Documentation

6.1 consumables.hpp

```
00001 #ifndef CONSUMABLES
00002 #define CONSUMABLES
00003
00004 #include <string>
00005 #include "item.hpp"
00006
00007 class HealingPotion: public Item {
00008 public:
00009     HealingPotion(std::string name, int size, int healing) : Item(name, size) {
00010         healing_ = healing;
00011     }
00012     int getHealing() {
00013         return healing_;
00014     }
00015
00016 private:
00017     int healing_;
00018 };
00019
00020 #endif
```

6.2 entity.hpp

```
00001 #ifndef ENTITY
00002 #define ENTITY
00003
00004 #include <SDL2/SDL.h>
00005 #include <SDL2/SDL_mixer.h>
00006
00007 typedef struct {
00008     int x;
00009     int y;
00010 } Coordinate;
00011
00012
00013 typedef struct {
00014     Mix_Chunk* attack_;
00015     Mix_Chunk* hit_;
00016     Mix_Chunk* death_;
00017     Mix_Chunk* taunt_;
00018 } SoundSet;
00019
00020
00021 class Entity {
00022
00023 public:
00024     int x_;
00025     int y_;
00026     int size_x_;
00027     int size_y_;
00028     int speed_;
00029     float direction_;
00030     SDL_Texture *texture_;
00031
```

```
00032     Entity(int, int, int, int);
00033     void move();
00034     Coordinate newPos();
00035     Coordinate center();
00036     Coordinate newCenter();
00037     bool collidesWith(Entity&);
00038
00039 };
00040
00041
00042 #endif
```

6.3 game.hpp

```
00001 #ifndef GAME
00002 #define GAME
00003
00004 #include <SDL2/SDL.h>
00005 #include <SDL2/SDL_render.h>
00006 #include <list>
00007 #include "entity.hpp"
00008 #include "input.hpp"
00009 #include "player.hpp"
00010 #include "room.hpp"
00011 #include "renderer.hpp"
00012 #include "hud.hpp"
00013
00014 class Game {
00015
00016 public:
00017
00018     Game();
00019     void movePlayer(InputState&);
00020     void spawnProjectile(int, int, int, int, int, float, SDL_Texture*);
00022     void moveProjectiles(Renderer&);
00024     void moveMonsters(Renderer&);
00025     void parseInput(Renderer&);
00027     int tick(Renderer&);
00029     void render(Renderer&);
00031     void changeRoom(Renderer&);
00033     void calcOffset(Renderer&);
00035     void scanNear(Renderer&);
00036     void menuTick(Renderer&);
00037     void menuRender(Renderer&);
00039     Weapon* scanWeapons(Renderer&);
00040
00041
00042     std::list<Room> room_templates_;
00043     Room *room_;
00044     Room *room1_;
00045     bool running_;
00046     Input input_;
00047     int x_offset_;
00048     int y_offset_;
00049     std::string infoText;
00050     int game_level_;
00051     Hud hud_;
00052     Player player_;
00053     std::list<Projectile> projectiles_;
00054     Weapon* displayWeapon_;
00055     bool paused_;
00056
00057 };
00058
00059
00060 #endif
00061
00062
```

6.4 hud.hpp

```
00001 #ifndef HUD
00002 #define HUD
00003
00004 #include "renderer.hpp"
00005
00006 class Hud{
00007 public:
00008     Hud(int hudposx, int hudposy);
```

```
00009     void drawInfo(Renderer&, int level, int health, int maxHp, int room);
00010
00011 private:
00012     int hudPosX;
00013     int hudPosY;
00014
00015 };
00016
00017
00018
00019
00020
00021
00022
00023 #endif
```

6.5 input.hpp

```
00001 #ifndef INPUT
00002 #define INPUT
00003
00004 #include <SDL2/SDL.h>
00005
00006 typedef struct {
00007     bool up;
00008     bool down;
00009     bool left;
00010     bool right;
00011     bool attack;
00012     bool interact;
00013     bool menu;
00014     bool attackUp;
00015     bool attackDown;
00016     bool attackLeft;
00017     bool attackRight;
00018 } InputState;
00019
00020 typedef struct {
00021     uint32_t up;
00022     uint32_t down;
00023     uint32_t left;
00024     uint32_t right;
00025     uint32_t attack;
00026     uint32_t interact;
00027     uint32_t menu;
00028     uint32_t attackUp;
00029     uint32_t attackDown;
00030     uint32_t attackLeft;
00031     uint32_t attackRight;
00032 } InputMapping;
00033
00034
00036 class Input {
00037 public:
00038
00039     Input();
00040     int scan();
00041     void keyDown(SDL_KeyboardEvent*);
00042     void keyUp(SDL_KeyboardEvent*);
00043     void resetInput();
00044     void resetInteract();
00045     InputState getState();
00046
00047 private:
00048
00049     InputState state_;
00050     InputMapping mapping_;
00051
00052 };
00053
00054 #endif
```

6.6 item.hpp

```
00001 #ifndef ITEM
00002 #define ITEM
00003
00004 #include <SDL2/SDL.h>
00005 #include <string>
```

```
00006
00007 class Item {
00008
00009 public:
00010     Item(std::string name, int size) {
00011         name_ = name;
00012         size_ = size;
00013         x_ = 0;
00014         y_ = 0;
00015     }
00016
00017     void equip() {
00018         equipped_ = true;
00019     }
00020     int getSize() {
00021         return size_;
00022     }
00023
00024     SDL_Texture *texture_;
00025
00026     // For world location
00027     int x_;
00028     int y_;
00029
00030 private:
00031     std::string name_;
00032     bool equipped_;
00033     int size_;
00034
00035 };
00036
00037 #endif
```

6.7 monster.hpp

```
00001 #ifndef MONSTER
00002 #define MONSTER
00003
00004 #include "entity.hpp"
00005 #include "player.hpp"
00006 #include "renderer.hpp"
00007 #include "weapon.hpp"
00008 #include <SDL2/SDL_mixer.h>
00009 #include <SDL2/SDL_render.h>
00010 #include <string>
00011
00016 class Monster: public Entity {
00017 public:
00018     Monster(const std::string&, int, int, int, int, int, int, int);
00019     ~Monster();
00020
00021     int GetHP();
00022     int GetDMG();
00023     void TakeDMG(int);
00024     bool isAlive();
00025     std::string getName();
00026
00027     virtual void setMove(Player&);
00028     virtual bool attack(Player&, std::list<Projectile>&);
00029     virtual void dropItem(std::list<Weapon*>&);
00030     virtual Mix_Chunk* getAttackSound();
00031
00032     SoundSet sounds_;
00033
00034 protected:
00035     bool alive_;
00036     int hp_;
00037     int dmg_;
00038     int max_speed_;
00039     std::string name_;
00040
00041 };
00042
00050 class MeleeMob: public Monster {
00051
00052 public:
00053     MeleeMob(int, int, int, int, int);
00054
00055     bool attack(Player&, std::list<Projectile>&);
00056     void setMove(Player&);
00057
00058     int attack_ticks_;
00059     int attack_cooldown_;
```

```
00060 };
00061
00062
00070 class RangedMob: public Monster {
00071
00072 public:
00073     RangedMob(int, int, int, int, int, Weapon*);
00074
00075     bool attack(Player&, std::list<Projectile>&);
00076     void setMove(Player&);
00077     void dropItem(std::list<Weapon*>&);
00078     Mix_Chunk* getAttackSound();
00079
00080     Weapon* weapon_;
00081     int attack_ticks_;
00082     int optimal_distance_;
00083
00084 };
00085
00092 class Boss: public Monster {
00093
00094 public:
00095     Boss(int);
00096
00097     bool attack(Player&, std::list<Projectile>&);
00098     void setMove(Player&);
00099     Mix_Chunk* getAttackSound();
00100
00101     // attack pattern 0 = melee, 1 = ranged
00102     int attack_pattern_;
00103     Weapon* weapon_;
00104     int attack_ticks_;
00105     int attack_cooldown_;
00106     int optimal_distance_;
00107
00108 };
00109
00111 enum MonsterType {
00112     MeleeMobType,
00113     RangedMobType
00114 };
00115
00117 Monster* genRandomMob(Renderer&, int, int, int);
00118
00119 #endif
00120
```

6.8 player.hpp

```
00001 #ifndef PLAYER
00002 #define PLAYER
00003
00004 #include <list>
00005 #include <string>
00006 #include <SDL2/SDL.h>
00007 #include "entity.hpp"
00008 #include "input.hpp"
00009 #include "weapon.hpp"
00010
00011 class Player: public Entity {
00012 public:
00013     Player(const std::string&, int, int);
00014
00015     const std::string GetName() const;
00016     int GetHP();
00017     int GetXP();
00018     int GetDMG();
00019     int GetMaxSpeed();
00020     int GetLevel();
00021     int getMaxHp();
00022
00023     void Heal(int);
00024     void TakeDMG(int);
00025     void UpdateXP(int);
00026     void UpdateDMG(int);
00027     void setMove(InputState&);
00028     bool attack(InputState&, std::list<Projectile>&);
00029     float getAttackDirection();
00030     bool gainXP(int);
00031     bool isAlive();
00032     void equipWeapon(Weapon*, Renderer& r);
00033
00034
```

```
00035     Weapon *weapon_;
00036     SDL_Texture *texture_front_;
00037     SDL_Texture *texture_right_;
00038     SDL_Texture *texture_left_;
00039     int shoot_ticks_;
00040     SoundSet sounds_;
00041
00042 private:
00043     bool alive_;
00044     std::string name_;
00045     int hp_;
00046     int max_hp_;
00047     int dmg_;
00048     int xp_;
00049     int max_speed_;
00050     std::list<std::string> inventory_;      // string should be changed to Item when there is a class
      for it
00051     int level_;
00052     int xp_to_Level_up_;
00053     float attack_direction_;
00054
00055 };
00056
00057
00058 #endif
```

6.9 renderer.hpp

```
00001 #ifndef RENDERER
00002 #define RENDERER
00003
00004 #include <SDL2/SDL.h>
00005 #include <SDL2/SDL_events.h>
00006 #include <SDL2/SDL_ttf.h>
00007 #include <SDL2/SDL_mixer.h>
00008
00009
00010 class Renderer {
00011 public:
00012
00014     Renderer(int, int, uint32_t, uint32_t);
00016     void initSDL();
00017     void prepareScene();
00018     void presentScene();
00020     SDL_Texture* loadTexture(const char*);
00022     void drawTexture(SDL_Texture*, int, int, double, SDL_RendererFlip);
00024     void destroy();
00025     void set_flags(uint32_t, uint32_t);
00026     int getWinWidth();
00027     int getWinHeight();
00028     void draw_text(const char* str, int x, int y, SDL_Color = {255,255,255});
00029     TTF_Font* GetFont();
00031     void renderText(SDL_Surface* text, int x, int y);
00032     SDL_Surface* InitText(char* str);
00034     void playSound(Mix_Chunk*, int);
00036     Mix_Chunk* loadSound(const char*);
00037
00038
00039 private:
00040
00041     SDL_Renderer* renderer_;
00042     SDL_Window* window_;
00043     uint32_t renderer_flags_;
00044     uint32_t window_flags_;
00045     TTF_Font *font_;
00046
00047     int width_;
00048     int height_;
00049
00050 };
00051
00052
00053 #endif
00054
```

6.10 room.hpp

```
00001 #ifndef ROOM
00002 #define ROOM
```

```
00003
00004 #include <list>
00005 #include <string>
00006 #include <SDL2/SDL.h>
00007 #include "monster.hpp"
00008 #include "renderer.hpp"
00009
00010
00011 class Room {
00012 public:
00013     Room(const std::string&, int, int, SDL_Texture*, SDL_Texture*);
00014     ~Room();
00015
00016     void addRandomMonsters(Renderer&, int, int);
00017     void addRandomItems(Renderer& r, int level, int amount);
00018     void addAdvanceDoor();
00019     void addItem(Item*);
00020
00021     std::string name_;
00022     SDL_Texture *texture_;
00023     SDL_Texture *advanceDoor_;
00024     int advanceDoorX_;
00025     int advanceDoorY_;
00026     int width_;
00027     int height_;
00028     std::list<Monster*> monsters_;
00029     std::list<Item*> items_;
00030     std::list<Weapon*> weapons_;
00031
00032 };
00033
00034 typedef struct {
00035     std::string name;
00036     std::string texture_location;
00037     int width;
00038     int height;
00039     int mobs_min;
00040     int mobs_max;
00041 } RoomTemplate;
00042
00043
00044 Room* genRoom(Renderer&, int);
00045 Room* genBossRoom(Renderer&, int);
00046
00047 #endif
```

6.11 weapon.hpp

```
00001 #ifndef WEAPON
00002 #define WEAPON
00003
00004 #include "item.hpp"
00005 #include "entity.hpp"
00006 #include "renderer.hpp"
00007 #include <SDL2/SDL_mixer.h>
00008 #include <list>
00009 #include <SDL2/SDL_render.h>
00010 #include <string>
00011
00012 class Projectile: public Entity {
00013 public:
00014     Projectile(int x, int y, int size_x, int size_y, int dmg, float direction, int speed):
00015     Entity(x, y, size_x, size_y) {
00016         dmg_ = dmg;
00017         direction_ = direction;
00018         speed_ = speed;
00019     }
00020
00021     int dmg_;
00022     bool damage_monsters_;
00023 };
00024
00026 class Weapon: public Item {
00027
00028 public:
00029     Weapon(const std::string& name, int size, int dmg, int pspeed, int firerate);
00030     int getDmg();
00031     int getProjectileSpeed();
00032     int getFirerate();
00033     std::string getName();
00034
00036     virtual void shoot(std::list<Projectile>& projectiles, Entity source, int dmg, float direction,
      bool damage_monsters);
```

```
00037     virtual std::string toString();
00038
00039     SDL_Texture* texture_;
00040     SDL_Texture* projectile_texture_;
00041     Mix_Chunk* sound_;
00042
00043 protected:
00044     int dmg_;
00045     int projectile_speed_;
00046     int firerate_;  // Rounds per second
00047     int projectile_size_x_ = 10;
00048     int projectile_size_y_ = 10;
00049     std::string name_;
00050 };
00051
00053 class Pistol: public Weapon {
00054 public:
00055     Pistol(const std::string& name, int size, int dmg, int pspeed, int firerate);
00056
00057 };
00058
00060 class SMG: public Weapon {
00061 public:
00062     SMG(const std::string& name, int size, int dmg, int pspeed, int firerate);
00063
00064 };
00065
00067 class Shotgun: public Weapon {
00068 public:
00069     Shotgun(const std::string& name, int size, int dmg, int pspeed, int firerate, int pellets, float
      spread);
00070     void shoot(std::list<Projectile>& projectiles, Entity source, int dmg, float direction, bool
      damage_monsters);
00071     int getPellets();
00072     float getSpread();
00073     std::string toString();
00074
00075 private:
00076     int pellets_;
00077     float spread_;
00078 };
00079
00081 enum GunType {
00082     PistolType,
00083     SMGType,
00084     ShotgunType
00085 };
00086
00087 Weapon* genRandomWeapon(Renderer&, int);
00088
00089 #endif
```

Index

attack
    Boss, 11
    MeleeMob, 18
    RangedMob, 25

Boss, 9
    attack, 11
    getAttackSound, 11
    setMove, 11

Coordinate, 11

dropItem
    RangedMob, 25

Entity, 11

Game, 12
Game name, 1
getAttackSound
    Boss, 11
    RangedMob, 25

HealingPotion, 13
Hud, 14

Input, 14
InputMapping, 15
InputState, 15
Item, 15

MeleeMob, 16
    attack, 18
    setMove, 18
Monster, 18

Pistol, 19
Player, 21
Projectile, 22

RangedMob, 23
    attack, 25
    dropItem, 25
    getAttackSound, 25
    setMove, 25
Renderer, 26
Room, 26
RoomTemplate, 27

setMove
    Boss, 11
    MeleeMob, 18
    RangedMob, 25
shoot
    Shotgun, 29
    Weapon, 32
Shotgun, 27
    shoot, 29
    toString, 29
SMG, 29
SoundSet, 31

toString
    Shotgun, 29

Weapon, 31
    shoot, 32