CPP dungeon crawler

Generated by Doxygen 1.9.8

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

These can be installed by running
`sudo apt install libsdl2-2.0-0 libsdl2-image-2.0-0 libsdl2-ttf-2.0-0 libsd2-mixer-2.0-0`

1.1.0.2 Compiling

Run `make` in the project root.
`make run` starts the game.

1.2 Playing the game

After `make run` the game starts instantly. The game quits if the player dies or completes the game.

1.2.0.1 Controls

- WASD keys: Move around
- Arrow keys: Shoot
- E: Interact with items (switch weapons, pick up healing potions, enter the next room)

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

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

Chapter 3

Class Index

3.1 Class List

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

Chapter 4

File Index

4.1 File List

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

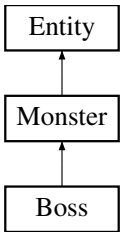
Chapter 5

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 **dropWeapon** (std::list< Weapon * > &)
- virtual void **dropPotion** (std::list< HealingPotion * > &)

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

- Item * **item_**
- 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**

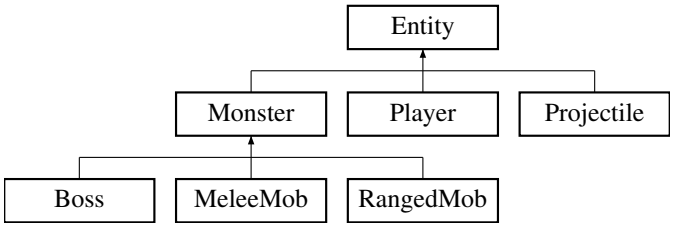
The documentation for this struct was generated from the following file:

- entity.hpp

5.3 Entity Class Reference

```
#include <entity.hpp>
```

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_**

5.3.1 Detailed Description

Base class for entities in the game

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.
- HealingPotion * **scanPotions** (Renderer &)

  Checks if potions 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_**
- int **mob_attack_delay_**

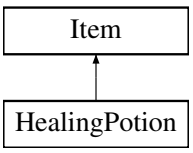
The documentation for this class was generated from the following files:

- game.hpp
- game.cpp

5.5 HealingPotion Class Reference

`#include <consumables.hpp>`

Inheritance diagram for HealingPotion:



Public Member Functions

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

Public Member Functions inherited from Item

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

Additional Inherited Members

Public Attributes inherited from Item

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

5.5.1 Detailed Description

Class for healing potion

Inherits Item base class Potions are dropped by melee monsters

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

- consumables.hpp
- consumables.cpp

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**

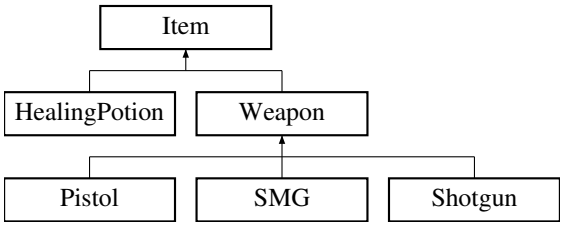
The documentation for this struct was generated from the following file:

- input.hpp

5.10 Item Class Reference

`#include <item.hpp>`

Inheritance diagram for Item:



Public Member Functions

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

Public Attributes

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

5.10.1 Detailed Description

Base class for all types of items found in the game

Inherited by Weapon and HealingPotion classes

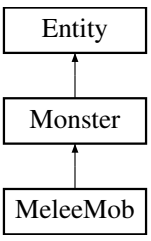
The documentation for this class was generated from the following file:

- item.hpp

5.11 MeleeMob Class Reference

`#include <monster.hpp>`

Inheritance diagram for MeleeMob:



Public Member Functions

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

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 **dropWeapon** (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

- HealingPotion ∗ **potion_**
- int **attack_ticks_**
- int **attack_cooldown_**

Public Attributes inherited from Monster

- Item ∗ **item_**
- 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 dropPotion()

```
void MeleeMob::dropPotion (
            std::list< HealingPotion * > & potions )  [virtual]
```

Reimplemented from Monster.

5.11.2.3 setMove()

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

Reimplemented from Monster.

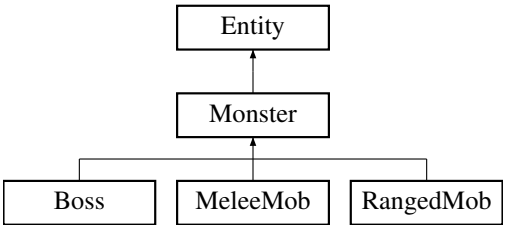
The documentation for this class was generated from the following files:

- 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 **dropWeapon** (std::list< Weapon * > &)
- virtual void **dropPotion** (std::list< HealingPotion * > &)
- 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

- Item * **item_**
- 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:

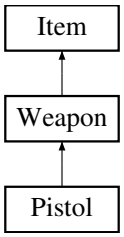
- monster.hpp
- monster.cpp

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** ()
- std::string **getName** ()

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.

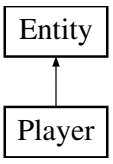
The documentation for this class was generated from the following files:

- weapon.hpp
- weapon.cpp

5.14 Player Class Reference

`#include <player.hpp>`

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 **healMax** ()
- 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_**

5.14.1 Detailed Description

Class for player

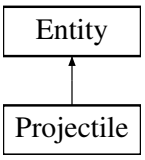
The documentation for this class was generated from the following files:

- player.hpp
- player.cpp

5.15 Projectile Class Reference

`#include <weapon.hpp>`

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_**

5.15.1 Detailed Description

Class for projectiles shot by weapons

Inherits Entity base class

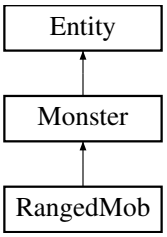
The documentation for this class was generated from the following file:

- 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 dropWeapon (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** ()
- virtual void **dropPotion** (std::list< HealingPotion ∗ > &)

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

- Item * **item_**
- 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 dropWeapon()

```
void RangedMob::dropWeapon (
            std::list< Weapon * > & weapons )  [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

`#include <room.hpp>`

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< HealingPotion * > **potions_**
- std::list< Weapon * > **weapons_**

5.18.1 Detailed Description

Class for rooms in the game

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:

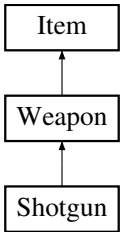
- room.hpp

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** ()
- std::string **getName** ()

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:

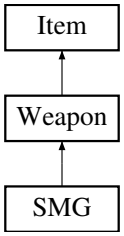
- weapon.hpp
- weapon.cpp

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** ()
- std::string **getName** ()

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:

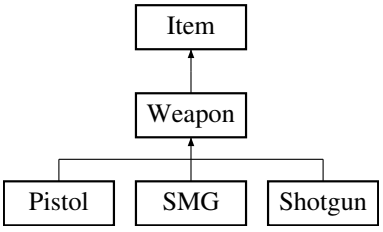
- entity.hpp

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** ()
- std::string **getName** ()

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 #include "renderer.hpp"
00007
00014 class HealingPotion: public Item {
00015 public:
00016     HealingPotion(std::string name, int size, int healing) : Item(name, size) {
00017         healing_ = healing;
00018     }
00019     int getHealing() {
00020         return healing_;
00021     }
00022     std::string toString();
00023
00024 private:
00025     int healing_;
00026 };
00027
00028 // Non-member functions
00029 HealingPotion* genPotion(Renderer&, int);
00030
00031 #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
00024 class Entity {
00025
00026 public:
00027     int x_;
00028     int y_;
00029     int size_x_;
00030     int size_y_;
```

```
00031     int speed_;
00032     float direction_;
00033     SDL_Texture *texture_;
00034 
00035     Entity(int, int, int, int);
00036     void move();
00037     Coordinate newPos();
00038     Coordinate center();
00039     Coordinate newCenter();
00040     bool collidesWith(Entity&);
00041 
00042 };
00043 
00044 
00045 #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 "input.hpp"
00008 #include "player.hpp"
00009 #include "room.hpp"
00010 #include "renderer.hpp"
00011 #include "hud.hpp"
00012 
00013 class Game {
00014 
00015 public:
00016 
00017     Game();
00018     void movePlayer(InputState&);
00019     void spawnProjectile(int, int, int, int, int, float, SDL_Texture*);
00021     void moveProjectiles(Renderer&);
00023     void moveMonsters(Renderer&);
00024     void parseInput(Renderer&);
00026     int tick(Renderer&);
00028     void render(Renderer&);
00030     void changeRoom(Renderer&);
00032     void calcOffset(Renderer&);
00034     void scanNear(Renderer&);
00035     void menuTick(Renderer&);
00036     void menuRender(Renderer&);
00038     Weapon* scanWeapons(Renderer&);
00040     HealingPotion* scanPotions(Renderer&);
00041 
00042 
00043     std::list<Room> room_templates_;
00044     Room *room_;
00045     Room *room1_;
00046     bool running_;
00047     Input input_;
00048     int x_offset_;
00049     int y_offset_;
00050     std::string infoText;
00051     int game_level_;
00052     Hud hud_;
00053     Player player_;
00054     std::list<Projectile> projectiles_;
00055     Weapon* displayWeapon_;
00056     bool paused_;
00057     int mob_attack_delay_; // delay to prevent spawn kill
00058 
00059 };
00060 
00061 
00062 #endif
00063 
00064 
```

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
00012 class Item {
00013
00014 public:
00015     Item(std::string name, int size) {
00016         name_ = name;
00017         size_ = size;
00018         x_ = 0;
00019         y_ = 0;
00020     }
00021
00022     void equip() {
00023         equipped_ = true;
00024     }
00025     int getSize() {
00026         return size_;
00027     }
00028     std::string getName() {
00029         return name_;
00030     }
00031
00032     SDL_Texture *texture_;
00033
00034     // For world location
00035     int x_;
00036     int y_;
00037
00038 private:
00039     std::string name_;
00040     bool equipped_;
00041     int size_;
00042
00043 };
00044
00045 #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 "consumables.hpp"
00009 #include <SDL2/SDL_mixer.h>
00010 #include <SDL2/SDL_render.h>
00011 #include <string>
00012
00017 class Monster: public Entity {
00018 public:
00019     Monster(const std::string&, int, int, int, int, int, int, int);
00020     ~Monster();
00021
00022     int GetHP();
00023     int GetDMG();
00024     void TakeDMG(int);
00025     bool isAlive();
00026     std::string getName();
00027     Item* item_;
00028
00029     virtual void setMove(Player&);
00030     virtual bool attack(Player&, std::list<Projectile>&);
00031     virtual void dropWeapon(std::list<Weapon*>&);
00032     virtual void dropPotion(std::list<HealingPotion*>&);
00033     virtual Mix_Chunk* getAttackSound();
00034
00035     SoundSet sounds_;
00036
00037 protected:
00038     bool alive_;
00039     int hp_;
00040     int dmg_;
00041     int max_speed_;
00042     std::string name_;
00043
00044 };
00045
```

```
00053 class MeleeMob: public Monster {
00054
00055 public:
00056     MeleeMob(int, int, int, int, int, HealingPotion*);
00057
00058     bool attack(Player&, std::list<Projectile>&);
00059     void setMove(Player&);
00060     void dropPotion(std::list<HealingPotion*>&);
00061
00062     HealingPotion* potion_;
00063     int attack_ticks_;
00064     int attack_cooldown_;
00065 };
00066
00067
00075 class RangedMob: public Monster {
00076
00077 public:
00078     RangedMob(int, int, int, int, int, Weapon*);
00079
00080     bool attack(Player&, std::list<Projectile>&);
00081     void setMove(Player&);
00082     void dropWeapon(std::list<Weapon*>&);
00083     Mix_Chunk* getAttackSound();
00084
00085     Weapon* weapon_;
00086     int attack_ticks_;
00087     int optimal_distance_;
00088
00089 };
00090
00097 class Boss: public Monster {
00098
00099 public:
00100     Boss(int);
00101
00102     bool attack(Player&, std::list<Projectile>&);
00103     void setMove(Player&);
00104     Mix_Chunk* getAttackSound();
00105
00106     // attack pattern 0 = melee, 1 = ranged
00107     int attack_pattern_;
00108     Weapon* weapon_;
00109     int attack_ticks_;
00110     int attack_cooldown_;
00111     int optimal_distance_;
00112
00113 };
00114
00116 enum MonsterType {
00117     MeleeMobType,
00118     RangedMobType
00119 };
00120
00122 Monster* genRandomMob(Renderer&, int, int, int);
00123
00124 #endif
00125
```

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
00015 class Player: public Entity {
00016 public:
00017     Player(const std::string&, int, int);
00018
00019     const std::string GetName() const;
00020     int GetHP();
00021     int GetXP();
00022     int GetDMG();
00023     int GetMaxSpeed();
00024     int GetLevel();
00025     int getMaxHp();
00026     void healMax();
```

```
00027     void Heal(int);
00028     void TakeDMG(int);
00029     void UpdateXP(int);
00030     void UpdateDMG(int);
00031     void setMove(InputState&);
00032     bool attack(InputState&, std::list<Projectile>&);
00033     float getAttackDirection();
00034     bool gainXP(int);
00035     bool isAlive();
00036     void equipWeapon(Weapon*, Renderer& r);
00037 
00038 
00039     Weapon *weapon_;
00040     SDL_Texture *texture_front_;
00041     SDL_Texture *texture_right_;
00042     SDL_Texture *texture_left_;
00043     int shoot_ticks_;
00044     SoundSet sounds_;
00045 
00046 private:
00047     bool alive_;
00048     std::string name_;
00049     int hp_;
00050     int max_hp_;
00051     int dmg_;
00052     int xp_;
00053     int max_speed_;
00054     std::list<std::string> inventory_;      // string should be changed to Item when there is a class
      for it
00055     int level_;
00056     int xp_to_Level_up_;
00057     float attack_direction_;
00058 
00059 };
00060 
00061 
00062 #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
00014 class Room {
00015 public:
00016     Room(const std::string&, int, int, SDL_Texture*, SDL_Texture*);
00017     ~Room();
00018
00019     void addRandomMonsters(Renderer&, int, int);
00020     void addRandomItems(Renderer& r, int level, int amount);
00021     void addAdvanceDoor();
00022     void addItem(Item*);
00023
00024     std::string name_;
00025     SDL_Texture *texture_;
00026     SDL_Texture *advanceDoor_;
00027     int advanceDoorX_;
00028     int advanceDoorY_;
00029     int width_;
00030     int height_;
00031     std::list<Monster*> monsters_;
00032     std::list<HealingPotion*> potions_;
00033     std::list<Weapon*> weapons_;
00034
00035 };
00036
00037 typedef struct {
00038     std::string name;
00039     std::string texture_location;
00040     int width;
00041     int height;
00042     int mobs_min;
00043     int mobs_max;
00044 } RoomTemplate;
00045
00046
00047 Room* genRoom(Renderer&, int);
00048 Room* genBossRoom(Renderer&, int);
00049
00050 #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
00018 class Projectile: public Entity {
00019 public:
00020     Projectile(int x, int y, int size_x, int size_y, int dmg, float direction, int speed):
00021     Entity(x, y, size_x, size_y) {
00022         dmg_ = dmg;
00023         direction_ = direction;
00024         speed_ = speed;
00025     }
00026
00027     int dmg_;
00028     bool damage_monsters_;
00029 };
00030
```

```
00032 class Weapon: public Item {
00033
00034 public:
00035     Weapon(const std::string& name, int size, int dmg, int pspeed, int firerate);
00036     int getDmg();
00037     int getProjectileSpeed();
00038     int getFirerate();
00039     std::string getName();
00040
00042     virtual void shoot(std::list<Projectile>& projectiles, Entity source, int dmg, float direction,
      bool damage_monsters);
00043     virtual std::string toString();
00044
00045     SDL_Texture* texture_;
00046     SDL_Texture* projectile_texture_;
00047     Mix_Chunk* sound_;
00048
00049 protected:
00050     int dmg_;
00051     int projectile_speed_;
00052     int firerate_;  // Rounds per second
00053     int projectile_size_x_ = 10;
00054     int projectile_size_y_ = 10;
00055     std::string name_;
00056 };
00057
00059 class Pistol: public Weapon {
00060 public:
00061     Pistol(const std::string& name, int size, int dmg, int pspeed, int firerate);
00062
00063 };
00064
00066 class SMG: public Weapon {
00067 public:
00068     SMG(const std::string& name, int size, int dmg, int pspeed, int firerate);
00069
00070 };
00071
00073 class Shotgun: public Weapon {
00074 public:
00075     Shotgun(const std::string& name, int size, int dmg, int pspeed, int firerate, int pellets, float
      spread);
00076     void shoot(std::list<Projectile>& projectiles, Entity source, int dmg, float direction, bool
      damage_monsters);
00077     int getPellets();
00078     float getSpread();
00079     std::string toString();
00080
00081 private:
00082     int pellets_;
00083     float spread_;
00084 };
00085
00087 enum GunType {
00088     PistolType,
00089     SMGType,
00090     ShotgunType
00091 };
00092
00093 Weapon* genRandomWeapon(Renderer&, int);
00094
00095 #endif
```

Index