Tower of Tom

Generated by Doxygen 1.9.4

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

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

Chapter 2

Class Index

2.1 Class List

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

Chapter 3

File Index

3.1 File List

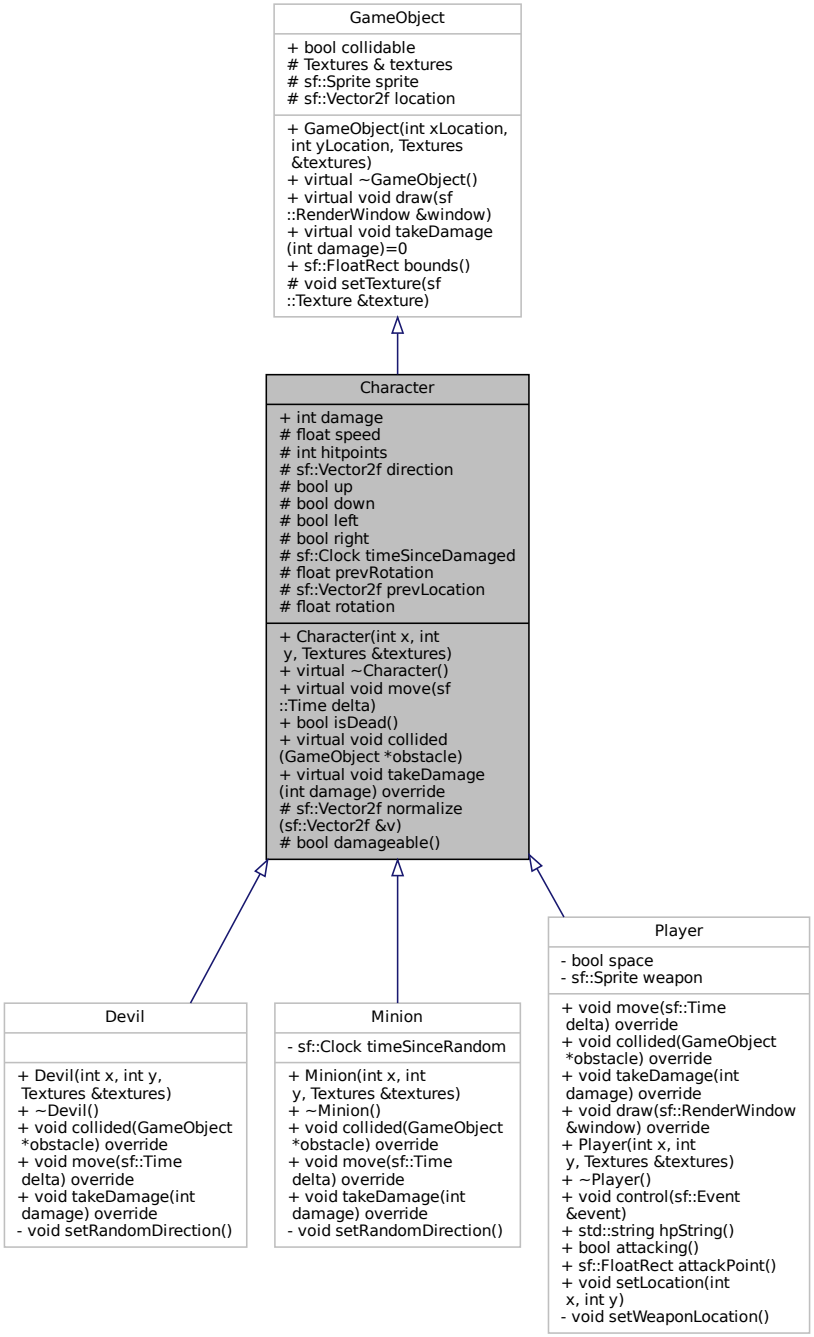
Here is a list of all documented files with brief descriptions:

Chapter 4

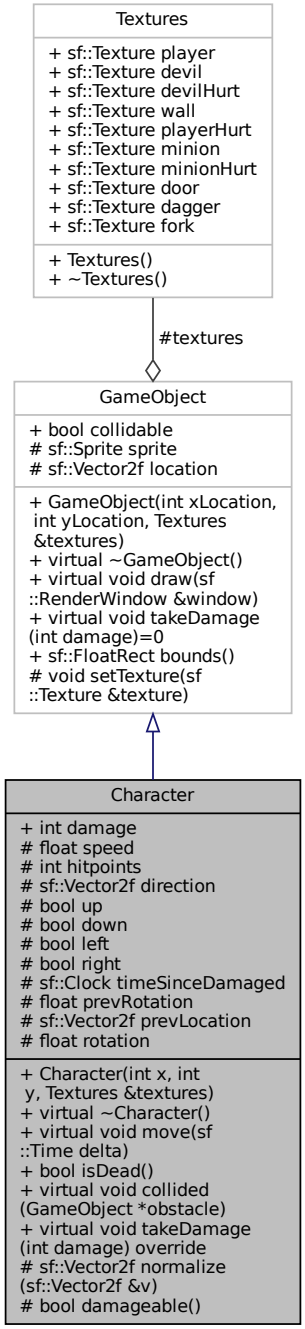
Class Documentation

4.1 Character Class Reference

Inheritance diagram for Character:



Collaboration diagram for Character:



Public Member Functions

- **Character** (int x, int y, Textures &textures)
- virtual void move (sf::Time delta)

  Moves the character based on speed in the direction that is set.
- bool isDead ()

  If the character is dead or alive.

- virtual void collided (GameObject *obstacle)
  sets what happens when the character collides with another object
- virtual void takeDamage (int damage) override
  Lowers the hitpoints of the object.

Public Attributes

- int **damage**
  the amount of damage the character does when attacking

Protected Member Functions

- sf::Vector2f normalize (sf::Vector2f &v)
  Normalizes the direction so the character doesn't move faster diagonally.
- bool damageable ()
  if the character can be damaged or not

Protected Attributes

- float **speed**
  movment speed of the character in pixels per second
- int **hitpoints** {1}
  number of hitpoints of the character
- sf::Vector2f **direction** {0,0}
  the direction the character is moving in
- bool **up** {false}
  helping attribute for direction of movement
- bool **down** {false}
  helping attribute for direction of movement
- bool **left** {false}
  helping attribute for direction of movement
- bool **right** {false}
  helping attribute for direction of movement
- sf::Clock **timeSinceDamaged**
  The time since the character took any damage.
- float **prevRotation** {0}
  the previous rotation before the current one
- sf::Vector2f **prevLocation** {0,0}
  the previous location before the current one
- float **rotation** {0}
  the angular rotation of the character

4.1.1 Member Function Documentation

4.1.1.1 collided()

```
void Character::collided (
            GameObject * obstacle )  [virtual]
```

sets what happens when the character collides with another object

Parameters

obstacle	the object that the character has collided with

Reimplemented in Devil, Minion, and Player.

4.1.1.2 damageable()

```
bool Character::damageable ( )  [protected]
```

if the character can be damaged or not

Returns

true can be damaged

false can not be damaged

4.1.1.3 isDead()

```
bool Character::isDead ( )
```

If the character is dead or alive.

Returns

true Character is dead

false Character is alive

4.1.1.4 move()

```
void Character::move (
            sf::Time delta )  [virtual]
```

Moves the character based on speed in the direction that is set.

Parameters

delta	time since last gameloop

Reimplemented in Devil, Minion, and Player.

4.1.1.5 normalize()

```
sf::Vector2f Character::normalize (
            sf::Vector2f & v )  [protected]
```

Normalizes the direction so the character doesn't move faster diagonally.

Parameters

v	the currect direction

Returns

sf::Vector2f the normalized direction

4.1.1.6 takeDamage()

```
void Character::takeDamage (
            int damage )  [override], [virtual]
```

Lowers the hitpoints of the object.

Parameters

damage	amount of hitpoints to deduct

Implements GameObject.

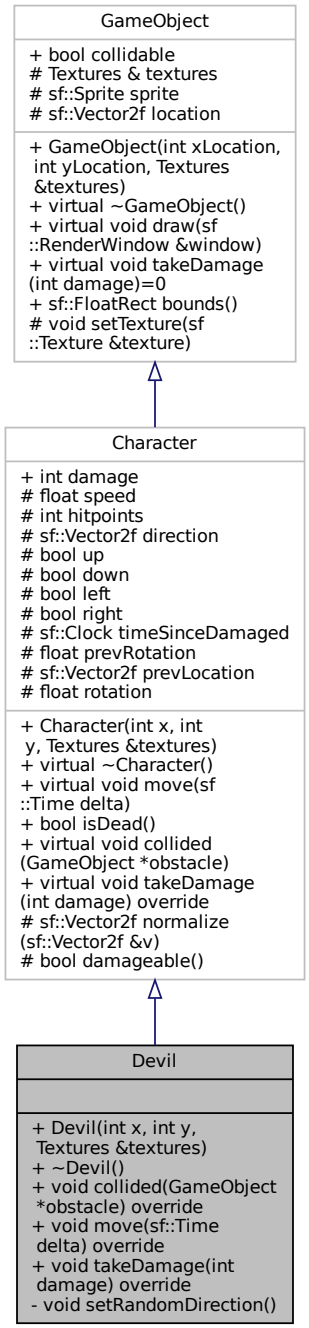
Reimplemented in Devil, Minion, and Player.

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

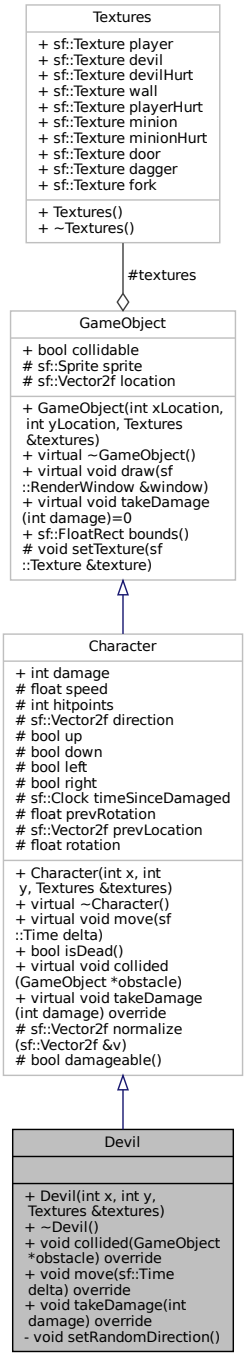
- Game/Character.h
- Game/Character.cc

4.2 Devil Class Reference

Inheritance diagram for Devil:



Collaboration diagram for Devil:



Public Member Functions

- **Devil** (int x, int y, Textures &textures)
- void collided (GameObject ∗obstacle) override

  sets what happens when the character collides with another object
- void move (sf::Time delta) override

  Moves the character based on speed in the direction that is set.

- void takeDamage (int damage) override

  Lowers the hitpoints of the object.

Private Member Functions

- void **setRandomDirection** ()

  Sets a random direction to be moved in.

Additional Inherited Members

4.2.1 Member Function Documentation

4.2.1.1 collided()

```
void Devil::collided (
            GameObject * obstacle )  [override], [virtual]
```

sets what happens when the character collides with another object

Parameters

obstacle	the object that the character has collided with

Reimplemented from Character.

4.2.1.2 move()

```
void Devil::move (
            sf::Time delta )  [override], [virtual]
```

Moves the character based on speed in the direction that is set.

Parameters

delta	time since last gameloop

Reimplemented from Character.

4.2.1.3 takeDamage()

```
void Devil::takeDamage (
```

```
int damage )  [override], [virtual]
```

Lowers the hitpoints of the object.

Parameters

damage	amount of hitpoints to deduct

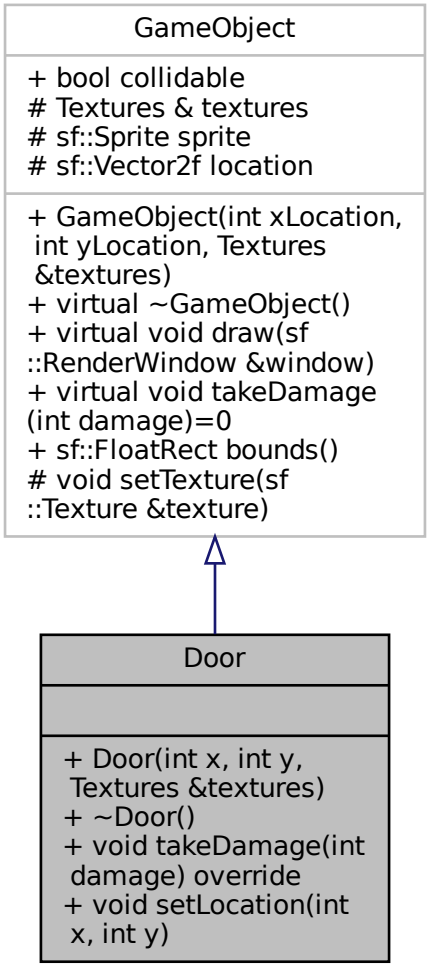
Reimplemented from Character.

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

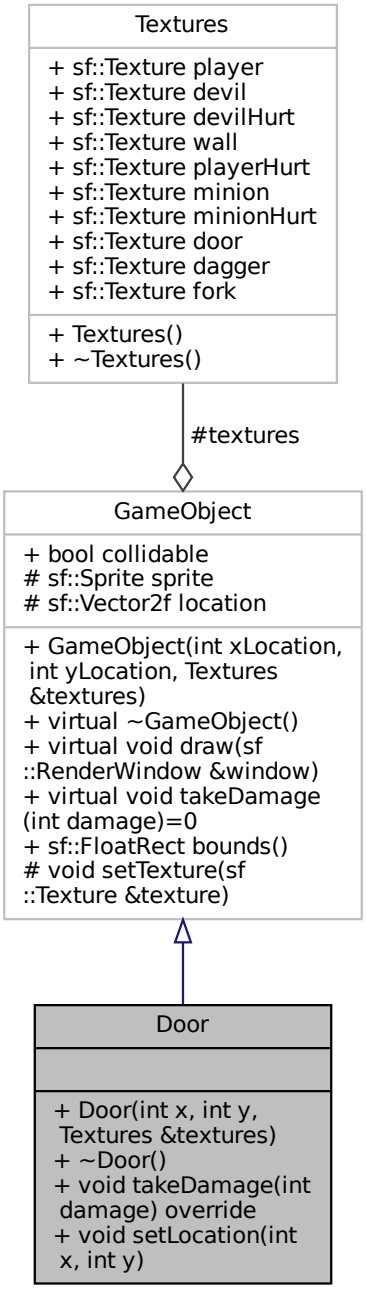
- Game/Devil.h
- Game/Devil.cc

4.3 Door Class Reference

Inheritance diagram for Door:



Collaboration diagram for Door:



Public Member Functions

- **Door** (int x, int y, Textures &textures)
- void takeDamage (int damage) override

  Lowers the hitpoints of the object.
- void setLocation (int x, int y)

  Set the Location of the door.

Additional Inherited Members

4.3.1 Member Function Documentation

4.3.1.1 setLocation()

```
void Door::setLocation (
            int x,
            int y )
```

Set the Location of the door.

Parameters

x	x value of location
y	y value of location

4.3.1.2 takeDamage()

```
void Door::takeDamage (
            int damage )  [override], [virtual]
```

Lowers the hitpoints of the object.

Parameters

damage	amount of hitpoints to deduct

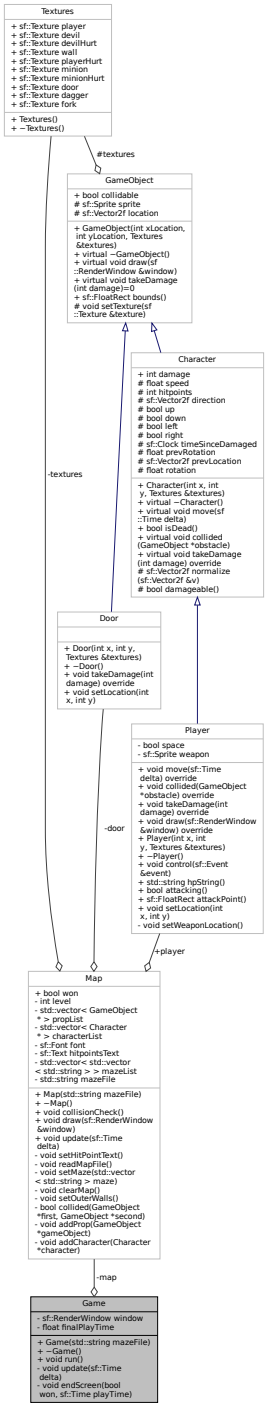
Implements GameObject.

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

- Game/Door.h
- Game/Door.cc

4.4 Game Class Reference

Collaboration diagram for Game:



Public Member Functions

- **Game** (std::string mazeFile)
- void **run** ()

  Runs the game.

Private Member Functions

- void update (sf::Time delta)

  updates everything about the game from movements to damage and beyond.
- void **endScreen** (bool won, sf::Time playTime)

Private Attributes

- sf::RenderWindow **window** {sf::VideoMode(1024, 576), "Tower of Tom"}

  The window that the game is displayed in.
- Map **map**

  The object that represents the games map.
- float **finalPlayTime** {0}

4.4.1 Member Function Documentation

4.4.1.1 update()

```
void Game::update (
            sf::Time delta )  [private]
```

updates everything about the game from movements to damage and beyond.

Parameters

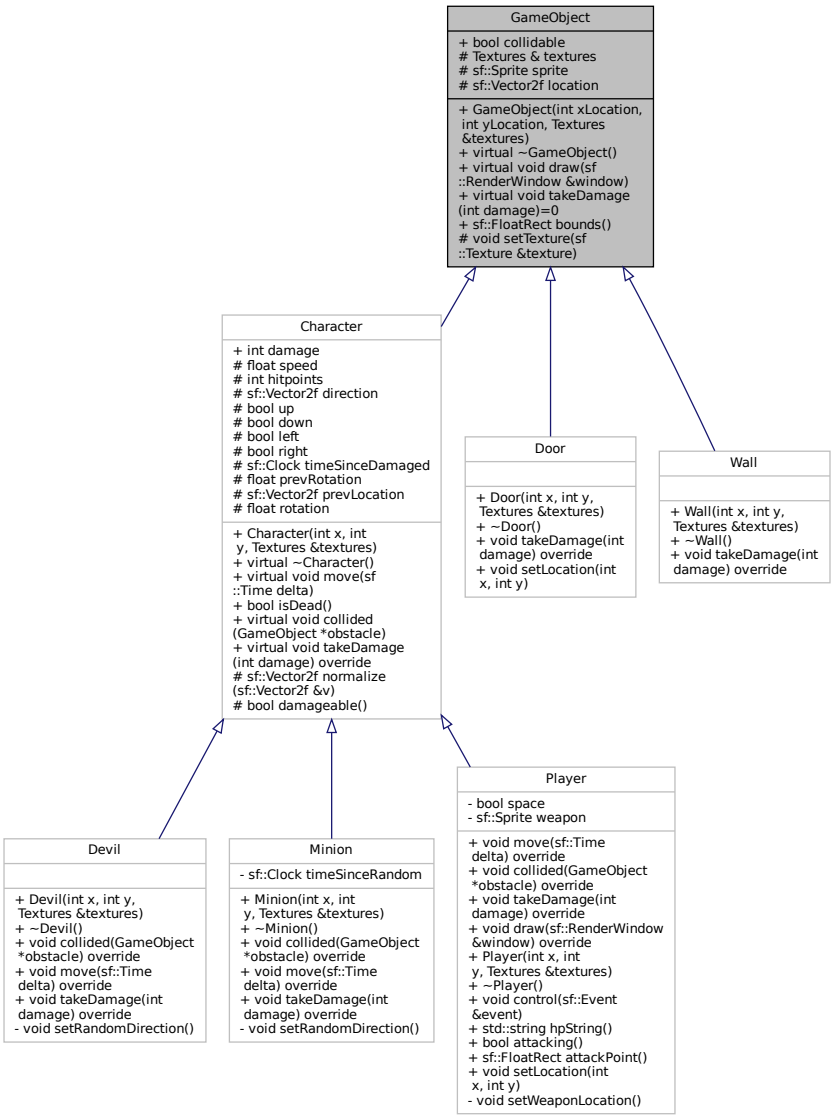
delta	The diffrents in time since last update cycle.

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

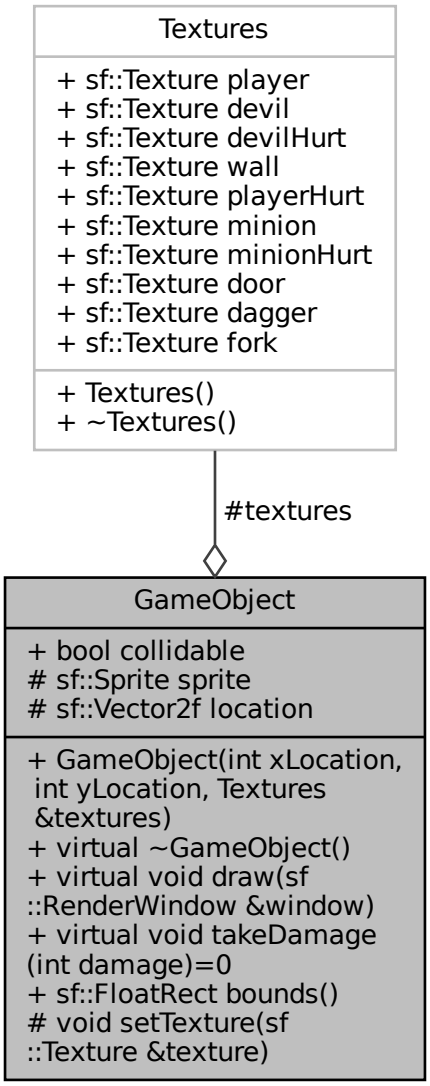
- Game/Game.h
- Game/Game.cc

4.5 GameObject Class Reference

Inheritance diagram for GameObject:



Collaboration diagram for GameObject:



Public Member Functions

- **GameObject** (int xLocation, int yLocation, Textures &textures)
- virtual void draw (sf::RenderWindow &window)

  Renders the object on the screen.
- virtual void takeDamage (int damage)=0

  Lowers the hitpoints of the object.
- sf::FloatRect bounds ()

  returns the objects hitbox

Public Attributes

- bool **collidable** {false}

  if the object is collidable or not

Protected Member Functions

- void setTexture (sf::Texture &texture)

  Set the Texture of the Sprite.

Protected Attributes

- Textures & **textures**

  Refrence to Map::textures.
- sf::Sprite **sprite**

  Sprite representing object.
- sf::Vector2f **location** {0,0}

  the objects location on the map

4.5.1 Member Function Documentation

4.5.1.1 bounds()

```
sf::FloatRect GameObject::bounds ( )
```

returns the objects hitbox

Returns

sf::FloatRect SMFL hitbox used for collision detection

4.5.1.2 draw()

```
void GameObject::draw (
            sf::RenderWindow & window )  [virtual]
```

Renders the object on the screen.

Parameters

window	Refrence to the game window

Reimplemented in Player.

4.5.1.3 setTexture()

```
void GameObject::setTexture (
            sf::Texture & texture )  [protected]
```

Set the Texture of the Sprite.

Parameters

texture	SFML texture

4.5.1.4 takeDamage()

```
virtual void GameObject::takeDamage (
            int damage )  [pure virtual]
```

Lowers the hitpoints of the object.

Parameters

damage	amount of hitpoints to deduct

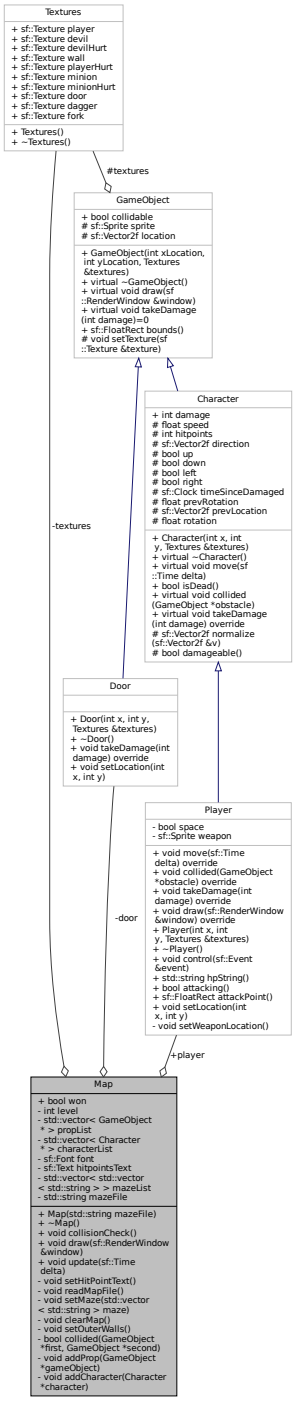
Implemented in Character, Devil, Door, Minion, Player, and Wall.

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

- Game/GameObject.h
- Game/GameObject.cc

4.6 Map Class Reference

Collaboration diagram for Map:



Public Member Functions

- **Map** (std::string mazeFile)
- void **collisionCheck** ()

Checks if any object has collided with any other object.

- void draw (sf::RenderWindow &window)

  Renders all objects to the screen.
- void update (sf::Time delta)

  updates all objects and thier attributes

Public Attributes

- bool **won** {false}

  When the game ends this bool determines if we won or lost.
- Player * **player**

  Reference to the player.

Private Member Functions

- void **setHitPointText** ()

  Setup the Hit Point Text object.
- void **readMapFile** ()

  Reads data from the map file.
- void setMaze (std::vector< std::string > maze)

  Setup the Maze/level.
- void **clearMap** ()

  clears the map of all objects
- void **setOuterWalls** ()

  Set the Outer Walls along window edges.
- bool collided (GameObject *first, GameObject *second)

  Checks if two objects have collided.
- void addProp (GameObject *gameObject)

  adds a gameobject to propList vector
- void addCharacter (Character *character)

  Adds a charatect to characterList vector.

Private Attributes

- int **level** {0}

  The map level.
- std::vector< GameObject * > **propList**

  vector of all non-Character gameobjects
- std::vector< Character * > **characterList**

  vector of all non-Player characters
- Textures **textures**

  Object that handles SFML Textures.
- sf::Font **font**

  The font for the HP-Text.
- sf::Text **hitpointsText**

  SFML Text for shoeing the players hitpoints.
- std::vector< std::vector< std::string > > **mazeList**

  Object for holding all levels of the game.
- Door * **door**

  Refrence to the exit/next level door.
- std::string **mazeFile**

  The path to the file of levels.

4.6.1 Member Function Documentation

4.6.1.1 addCharacter()

```
void Map::addCharacter (
            Character * character )  [private]
```

Adds a charatect to characterList vector.

Parameters

character	any character

4.6.1.2 addProp()

```
void Map::addProp (
            GameObject * gameObject )  [private]
```

adds a gameobject to propList vector

Parameters

gameObject	any game object

4.6.1.3 collided()

```
bool Map::collided (
            GameObject * first,
            GameObject * second )  [private]
```

Checks if two objects have collided.

Parameters

first	any game object
second	any game object

Returns

true The object have collided

false The object have not collided

4.6.1.4 draw()

```
void Map::draw (
            sf::RenderWindow & window )
```

Renders all objects to the screen.

Parameters

window	Refrence to the game window

4.6.1.5 setMaze()

```
void Map::setMaze (
            std::vector< std::string > maze )  [private]
```

Setup the Maze/level.

Parameters

maze	the maze as read from file

4.6.1.6 update()

```
void Map::update (
            sf::Time delta )
```

updates all objects and thier attributes

Parameters

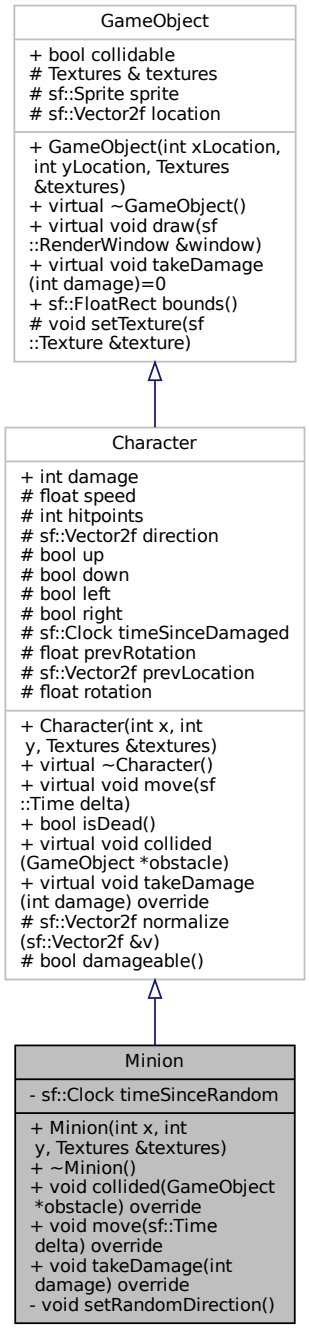
delta	time diffrence between each gameloop

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

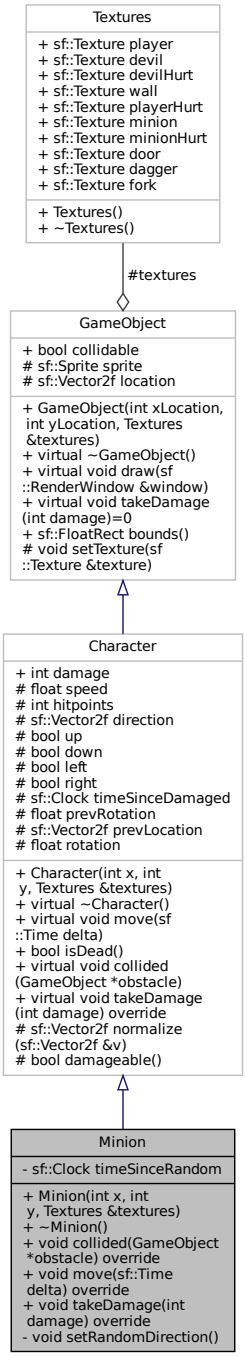
- Game/Map.h
- Game/Map.cc

4.7 Minion Class Reference

Inheritance diagram for Minion:



Collaboration diagram for Minion:



Public Member Functions

- **Minion** (int x, int y, Textures &textures)
- void collided (GameObject ∗obstacle) override

  sets what happens when the character collides with another object
- void move (sf::Time delta) override

  Moves the character based on speed in the direction that is set.

- void takeDamage (int damage) override

  Lowers the hitpoints of the object.

Private Member Functions

- void **setRandomDirection** ()

  Sets a random direction to be moved in.

Private Attributes

- sf::Clock **timeSinceRandom**

  time since setRandomDirection was called

Additional Inherited Members

4.7.1 Member Function Documentation

4.7.1.1 collided()

```
void Minion::collided (
            GameObject * obstacle )  [override], [virtual]
```

sets what happens when the character collides with another object

Parameters

obstacle	the object that the character has collided with

Reimplemented from Character.

4.7.1.2 move()

```
void Minion::move (
            sf::Time delta )  [override], [virtual]
```

Moves the character based on speed in the direction that is set.

Parameters

delta	time since last gameloop

Reimplemented from Character.

4.7.1.3 takeDamage()

```
void Minion::takeDamage (
            int damage )  [override], [virtual]
```

Lowers the hitpoints of the object.

Parameters

damage	amount of hitpoints to deduct

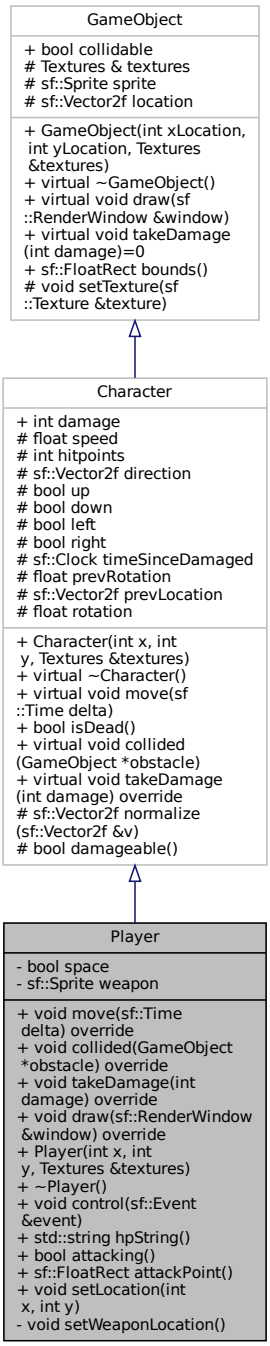
Reimplemented from Character.

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

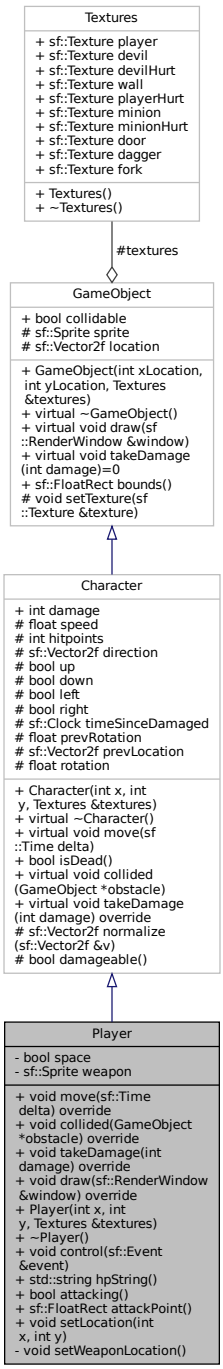
- Game/Minion.h
- Game/Minion.cc

4.8 Player Class Reference

Inheritance diagram for Player:



Collaboration diagram for Player:



Public Member Functions

- void move (sf::Time delta) override

  Moves the character based on speed in the direction that is set.
- void collided (GameObject *obstacle) override

  sets what happens when the character collides with another object
- void takeDamage (int damage) override

Lowers the hitpoints of the object.

- void draw (sf::RenderWindow &window) override

  Renders the object on the screen.

- **Player** (int x, int y, Textures &textures)
- void control (sf::Event &event)

  handles keyboard input

- std::string hpString ()

  Gets the players hitpoints as a string.

- bool attacking ()

  If the player is attacking or not.

- sf::FloatRect attackPoint ()

  returns the hitbox of the players weapon

- void setLocation (int x, int y)

  Set the Location of the player.

Private Member Functions

- void **setWeaponLocation** ()

  Set the Weapons location based on players location.

Private Attributes

- bool **space** {false}

  To see if spacebar is pressed.

- sf::Sprite **weapon**

  the sprite representing the players weapon

Additional Inherited Members

4.8.1 Member Function Documentation

4.8.1.1 attacking()

```
bool Player::attacking ( )
```

If the player is attacking or not.

Returns

true player is attacking

false player is not attacking

4.8.1.2 attackPoint()

```
sf::FloatRect Player::attackPoint ( )
```

returns the hitbox of the players weapon

Returns

sf::FloatRect the weapons hitbox

4.8.1.3 collided()

```
void Player::collided (
            GameObject * obstacle )  [override], [virtual]
```

sets what happens when the character collides with another object

Parameters

obstacle	the object that the character has collided with

Reimplemented from Character.

4.8.1.4 control()

```
void Player::control (
            sf::Event & event )
```

handles keyboard input

Parameters

event	event containing keyboard input

4.8.1.5 draw()

```
void Player::draw (
            sf::RenderWindow & window )  [override], [virtual]
```

Renders the object on the screen.

Parameters

window	Refrence to the game window

Reimplemented from GameObject.

4.8.1.6 hpString()

```
std::string Player::hpString ( )
```

Gets the players hitpoints as a string.

Returns

std::string players hitpoints

4.8.1.7 move()

```
void Player::move (
            sf::Time delta )  [override], [virtual]
```

Moves the character based on speed in the direction that is set.

Parameters

delta	time since last gameloop

Reimplemented from Character.

4.8.1.8 setLocation()

```
void Player::setLocation (
            int x,
            int y )
```

Set the Location of the player.

Parameters

x	x value of location
y	y value of location

4.8.1.9 takeDamage()

```
void Player::takeDamage (
            int damage )  [override], [virtual]
```

Lowers the hitpoints of the object.

Parameters

damage	amount of hitpoints to deduct

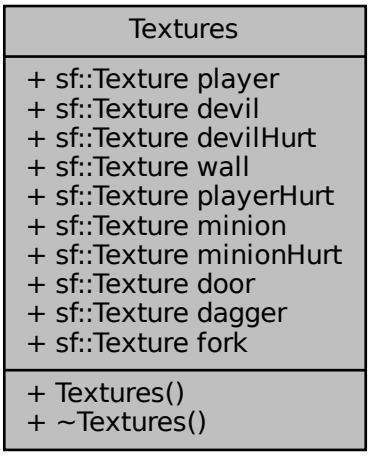
Reimplemented from Character.

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

- Game/Player.h
- Game/Player.cc

4.9 Textures Class Reference

Collaboration diagram for Textures:



Public Attributes

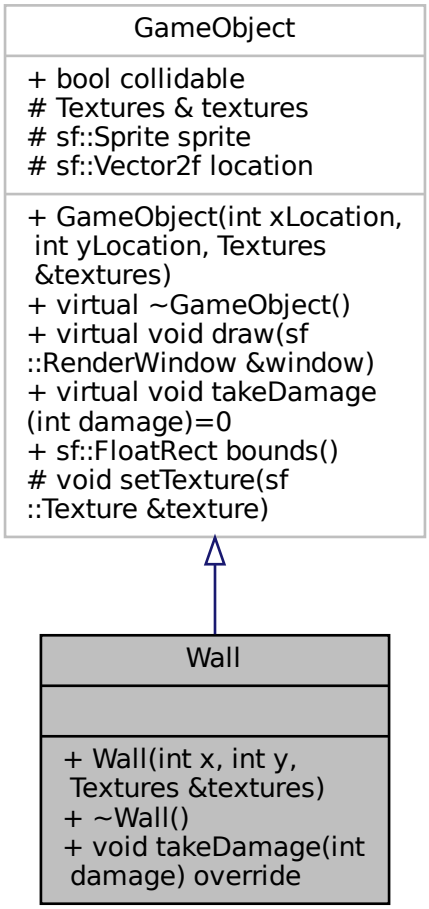
- sf::Texture **player**
- sf::Texture **devil**
- sf::Texture **devilHurt**
- sf::Texture **wall**
- sf::Texture **playerHurt**
- sf::Texture **minion**
- sf::Texture **minionHurt**
- sf::Texture **door**
- sf::Texture **dagger**
- sf::Texture **fork**

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

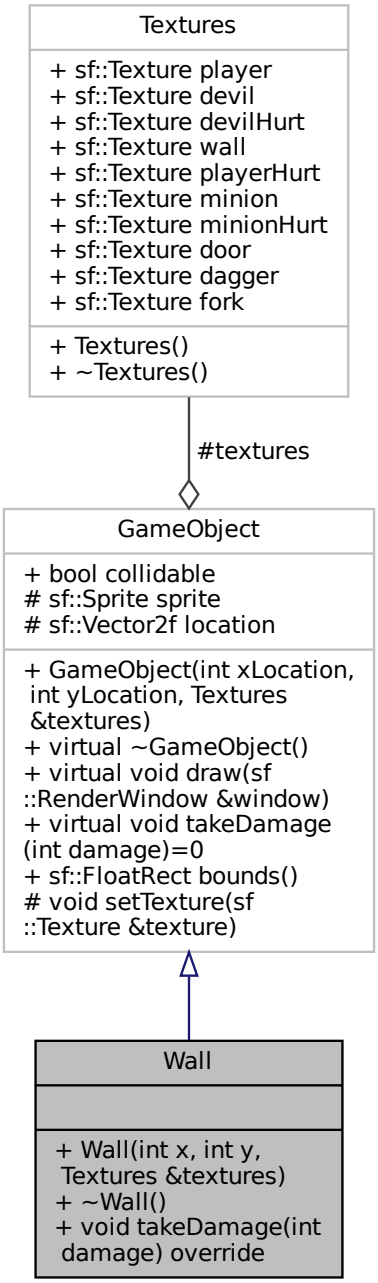
- Game/Textures.h
- Game/Textures.cc

4.10 Wall Class Reference

Inheritance diagram for Wall:



Collaboration diagram for Wall:



Public Member Functions

- **Wall** (int x, int y, Textures &textures)
- void takeDamage (int damage) override

  Lowers the hitpoints of the object.

Additional Inherited Members

4.10.1 Member Function Documentation

4.10.1.1 takeDamage()

```
void Wall::takeDamage (
            int damage )  [override], [virtual]
```

Lowers the hitpoints of the object.

Parameters

damage	amount of hitpoints to deduct

Implements GameObject.

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

- Game/Wall.h
- Game/Wall.cc

Chapter 5

File Documentation

5.1 Character.h

```
1 #pragma once
2
3 #include "GameObject.h"
4
5 class Character :  public GameObject
6 {
7 // Attributes
8 protected:
13      float speed;
18      int hitpoints{1};
23      sf::Vector2f direction{0,0};
28      bool up{false};
33      bool down{false};
38      bool left{false};
43      bool right{false};
48      sf::Clock timeSinceDamaged;
53      float prevRotation{0};
58      sf::Vector2f prevLocation{0,0};
63      float rotation{0};
64 public:
69      int damage;
70 // Members
71 protected:
78      sf::Vector2f normalize(sf::Vector2f &v);
85      bool damageable();
86 public:
87      Character(int x, int y, Textures &textures);
88      virtual ~Character();
94      virtual void move(sf::Time delta);
101      bool isDead();
107      virtual void collided(GameObject *obstacle);
108      virtual void takeDamage(int damage) override;
109 };
110
111
```

5.2 Devil.h

```
1 #pragma once
2
3 #include "Player.h"
4 #include "Character.h"
5 #include <cstdlib>
6
7 class Devil :  public Character
8 {
9 private:
14      void setRandomDirection();
15 public:
16      Devil(int x, int y, Textures &textures);
17      ~Devil();
18      void collided(GameObject *obstacle) override;
19      void move(sf::Time delta) override;
20      void takeDamage(int damage) override;
21 };
```

5.3 Door.h

```
1 #pragma once
2
3 #include "GameObject.h"
4
5 class Door :  public GameObject
6 {
7 private:
8
9 public:
10     Door(int x, int y, Textures &textures);
11     ~Door();
12     void takeDamage(int damage) override;
19     void setLocation(int x, int y);
20 };
21
22
```

5.4 Game.h

```
1 #include <SFML/Graphics.hpp>
2 #include <vector>
3 #include "GameObject.h"
4 #include "Map.h"
5
6 class Game
7 {
8 private:
13     sf::RenderWindow window{sf::VideoMode(1024, 576), "Tower of Tom"};
18     Map map;
24     void update(sf::Time delta);
25     void endScreen(bool won, sf::Time playTime);
26     float finalPlayTime{0};
27 public:
28     Game(std::string mazeFile);
29     ~Game();
35     void run();
36 };
```

5.5 GameObject.h

```
1 #pragma once
2
3 #include "Textures.h"
4 #include <SFML/Graphics.hpp>
5 #include <string>
6 #include <cmath>
7
8 class GameObject
9 {
10 // Attributes
11 protected:
16     Textures & textures;
21     sf::Sprite sprite;
26     sf::Vector2f location{0,0};
27
28 public:
33     bool collidable{false};
34
35 // Members
36 protected:
42     void setTexture(sf::Texture &texture);
43
44 public:
45     GameObject(int xLocation, int yLocation, Textures &textures);
46     virtual ~GameObject();
52     virtual void draw(sf::RenderWindow &window);
58     virtual void takeDamage(int damage) = 0;
64     sf::FloatRect bounds();
65 };
```

5.6 Map.h

```
1 #pragma once
```

```
2
3 #include <vector>
4 #include <fstream>
5 #include "GameObject.h"
6 #include "Player.h"
7 #include "Devil.h"
8 #include "Minion.h"
9 #include "Wall.h"
10 #include "Door.h"
11
12
13
14 class Map
15 {
16 // Attributes
17 private:
22     int level{0};
27     std::vector<GameObject *> propList;
32     std::vector<Character *> characterList;
37     Textures textures;
42     sf::Font font;
47     sf::Text hitpointsText;
52     std::vector<std::vector<std::string» mazeList;
57     Door *door;
62     std::string mazeFile;
63 public:
68     bool won{false};
73     Player *player;
74 // Members
75 private:
80     void setHitPointText();
85     void readMapFile();
91     void setMaze(std::vector<std::string> maze);
96     void clearMap();
101     void setOuterWalls();
110     bool collided(GameObject *first, GameObject *second);
116     void addProp(GameObject *gameObject);
122     void addCharacter(Character *character);
123 public:
124     Map(std::string mazeFile);
125     ~Map();
130     void collisionCheck();
136     void draw(sf::RenderWindow &window);
142     void update(sf::Time delta);
143 };
```

5.7 Minion.h

```
1 #pragma once
2 #include "Character.h"
3
4 class Minion :  public Character
5 {
6 private:
11     sf::Clock timeSinceRandom;
16     void setRandomDirection();
17 public:
18     Minion(int x, int y, Textures &textures);
19     ~Minion();
20     void collided(GameObject *obstacle) override;
21     void move(sf::Time delta) override;
22     void takeDamage(int damage) override;
23 };
24
```

5.8 Player.h

```
1 #pragma once
2
3 #include "Character.h"
4
5 class Player :  public Character
6 {
7 // Attributes
8 private:
13     bool space{false};
18     sf::Sprite weapon;
19 public:
20
```

```
21 // Members
22 private:
27     void setWeaponLocation();
28
29 public:
30     void move(sf::Time delta) override;
31     void collided(GameObject *obstacle) override;
32     void takeDamage(int damage) override;
33     void draw(sf::RenderWindow &window) override;
34
35     Player(int x, int y, Textures &textures);
36     ~Player();
42     void control(sf::Event &event);
48     std::string hpString();
55     bool attacking();
61     sf::FloatRect attackPoint();
68     void setLocation(int x, int y);
69 };
```

5.9 Textures.h

```
1 #pragma once
2 #include <SFML/Graphics.hpp>
3
4 class Textures
5 {
6 public:
7     sf::Texture player;
8     sf::Texture devil;
9     sf::Texture devilHurt;
10     sf::Texture wall;
11     sf::Texture playerHurt;
12     sf::Texture minion;
13     sf::Texture minionHurt;
14     sf::Texture door;
15     sf::Texture dagger;
16     sf::Texture fork;
17     Textures();
18     ~Textures();
19 };
```

5.10 Wall.h

```
1 #pragma once
2
3 #include "GameObject.h"
4
5 class Wall :  public GameObject
6 {
7 private:
8
9 public:
10     Wall(int x, int y, Textures &textures);
11     ~Wall();
12     void takeDamage(int damage) override;
13 };
14
15
```

Index