

Tower of Tom

Generated by Doxygen 1.9.4

1 Hierarchical Index	1
1.1 Class Hierarchy	1
2 Class Index	3
2.1 Class List	3
3 File Index	5
3.1 File List	5
4 Class Documentation	7
4.1 Character Class Reference	8
4.1.1 Member Function Documentation	10
4.1.1.1 collided()	10
4.1.1.2 damageable()	11
4.1.1.3 isDead()	11
4.1.1.4 move()	11
4.1.1.5 normalize()	12
4.1.1.6 takeDamage()	12
4.2 Devil Class Reference	13
4.2.1 Member Function Documentation	15
4.2.1.1 collided()	15
4.2.1.2 move()	15
4.2.1.3 takeDamage()	15
4.3 Door Class Reference	16
4.3.1 Member Function Documentation	18
4.3.1.1 setLocation()	18
4.3.1.2 takeDamage()	18
4.4 Game Class Reference	19
4.4.1 Member Function Documentation	20
4.4.1.1 update()	20
4.5 GameObject Class Reference	21
4.5.1 Member Function Documentation	23
4.5.1.1 bounds()	23
4.5.1.2 draw()	23
4.5.1.3 setTexture()	24
4.5.1.4 takeDamage()	24
4.6 Map Class Reference	25
4.6.1 Member Function Documentation	27
4.6.1.1 addCharacter()	27
4.6.1.2 addProp()	27
4.6.1.3 collided()	27
4.6.1.4 draw()	28
4.6.1.5 setMaze()	28

4.6.1.6 update()	28
4.7 Minion Class Reference	29
4.7.1 Member Function Documentation	31
4.7.1.1 collided()	31
4.7.1.2 move()	31
4.7.1.3 takeDamage()	32
4.8 Player Class Reference	33
4.8.1 Member Function Documentation	35
4.8.1.1 attacking()	35
4.8.1.2 attackPoint()	36
4.8.1.3 collided()	36
4.8.1.4 control()	36
4.8.1.5 draw()	36
4.8.1.6 hpString()	37
4.8.1.7 move()	37
4.8.1.8 setLocation()	37
4.8.1.9 takeDamage()	38
4.9 Textures Class Reference	38
4.10 Wall Class Reference	39
4.10.1 Member Function Documentation	41
4.10.1.1 takeDamage()	41
5 File Documentation	43
5.1 Character.h	43
5.2 Devil.h	43
5.3 Door.h	44
5.4 Game.h	44
5.5 GameObject.h	44
5.6 Map.h	44
5.7 Minion.h	45
5.8 Player.h	45
5.9 Textures.h	46
5.10 Wall.h	46
Index	47

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

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

Game	19
GameObject	21
Character	8
Devil	13
Minion	29
Player	33
Door	16
Wall	39
Map	25
Textures	38

Chapter 2

Class Index

2.1 Class List

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

Character	8
Devil	13
Door	16
Game	19
GameObject	21
Map	25
Minion	29
Player	33
Textures	38
Wall	39

Chapter 3

File Index

3.1 File List

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

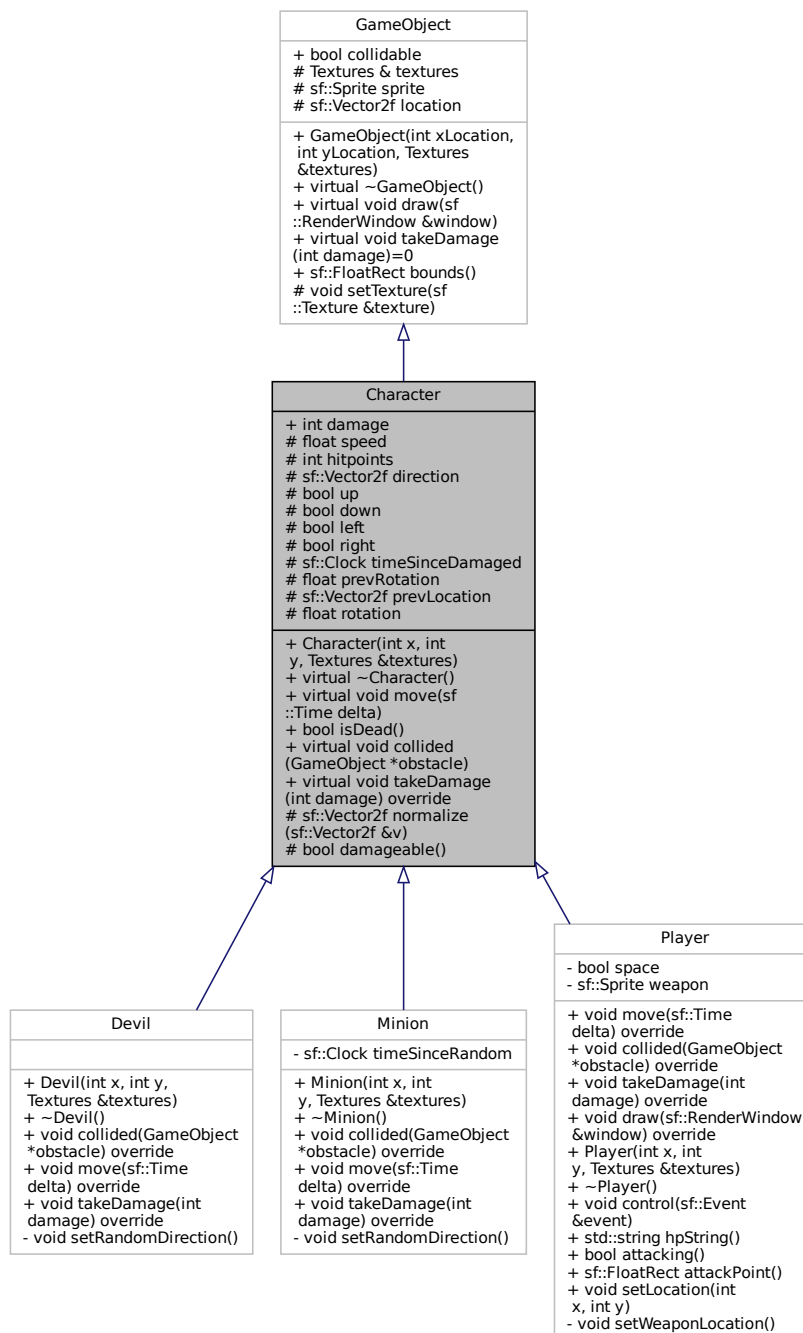
Game/ Character.h	43
Game/ Devil.h	43
Game/ Door.h	44
Game/ Game.h	44
Game/ GameObject.h	44
Game/ Map.h	44
Game/ Minion.h	45
Game/ Player.h	45
Game/ Textures.h	46
Game/ Wall.h	46

Chapter 4

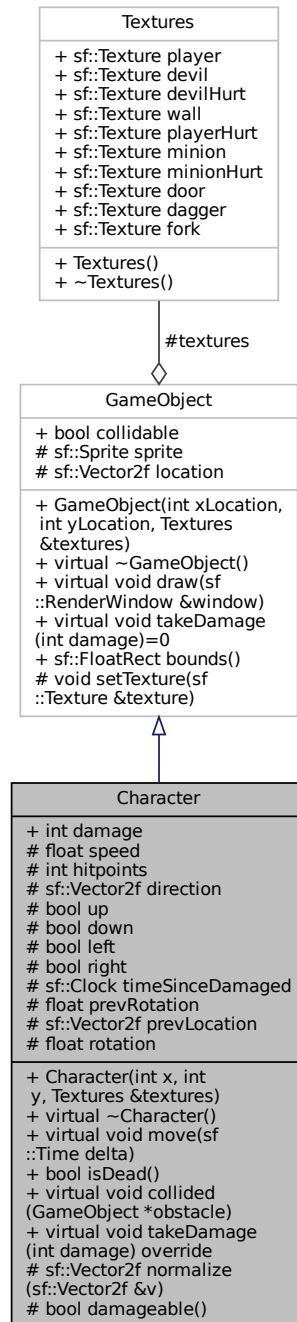
Class Documentation

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**
movement 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

<i>obstacle</i>	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

<i>delta</i>	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

<i>v</i>	the current 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

<i>damage</i>	amount of hitpoints to deduct
---------------	-------------------------------

Implements [GameObject](#).

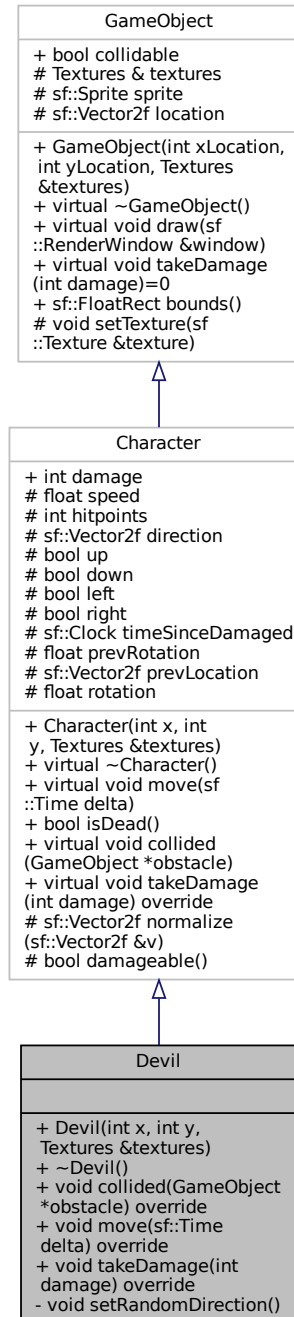
Reimplemented in [Devil](#), [Minion](#), and [Player](#).

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

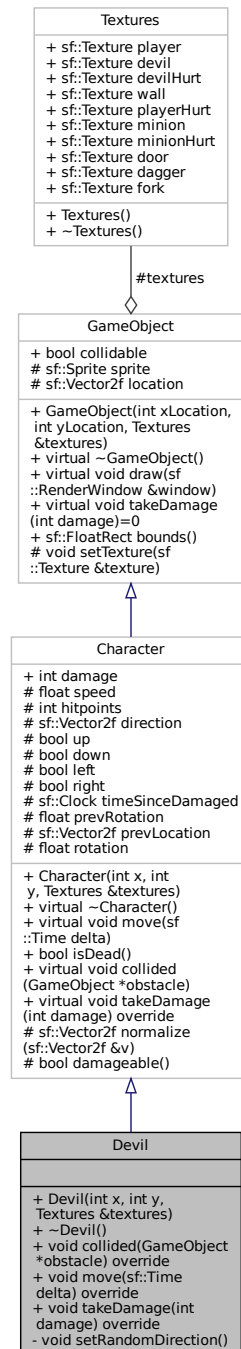
- Game/Character.h
- Game/Character.cc

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

<i>obstacle</i>	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

<i>delta</i>	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

<i>damage</i>	amount of hitpoints to deduct
---------------	-------------------------------

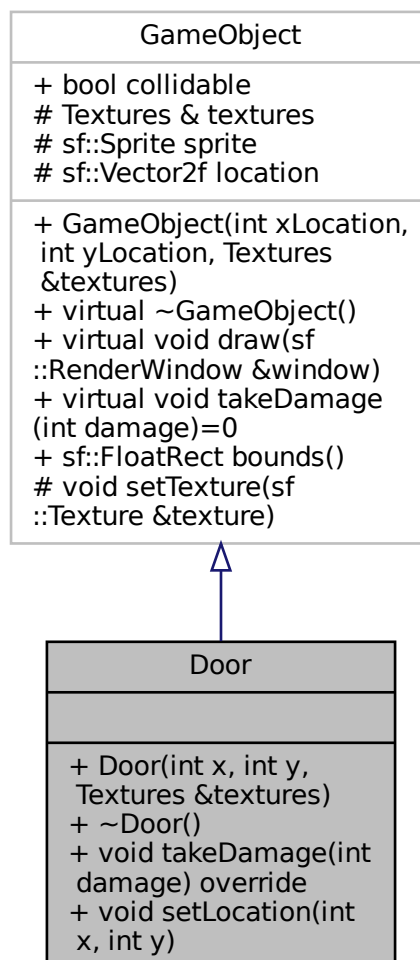
Reimplemented from [Character](#).

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

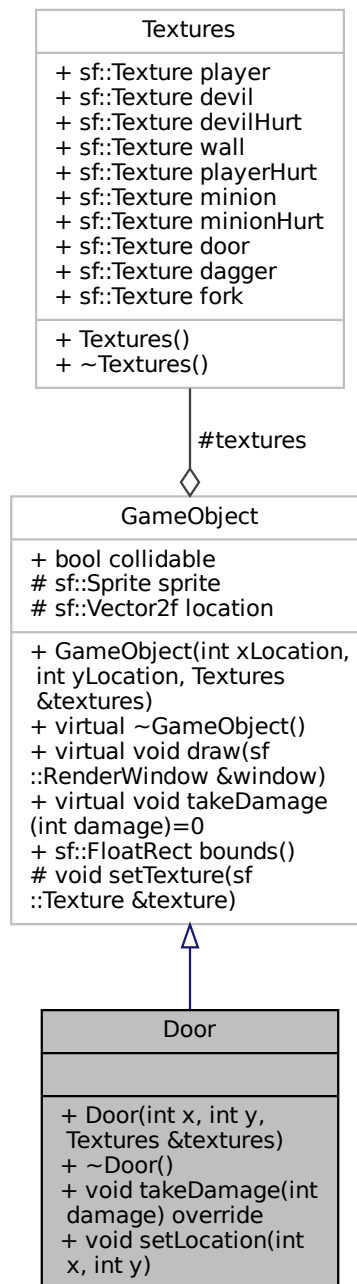
- Game/Devil.h
- Game/Devil.cc

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
Lowens 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

<i>x</i>	x value of location
<i>y</i>	y value of location

4.3.1.2 takeDamage()

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

Lowers the hitpoints of the object.

Parameters

<i>damage</i>	amount of hitpoints to deduct
---------------	-------------------------------

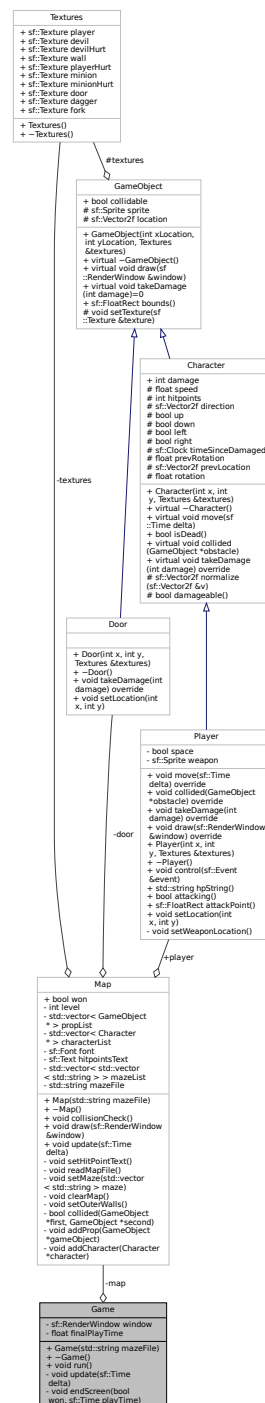
Implements [GameObject](#).

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

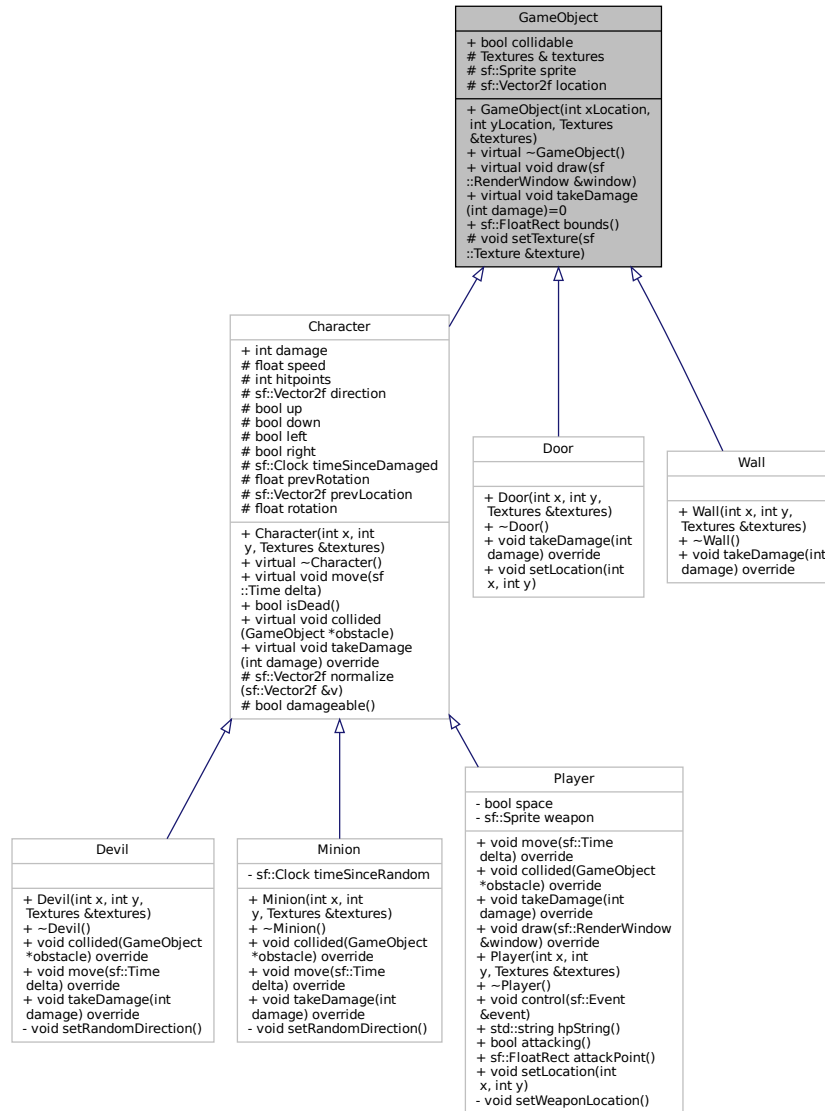
<i>delta</i>	The diffrents in time since last update cycle.
--------------	--

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

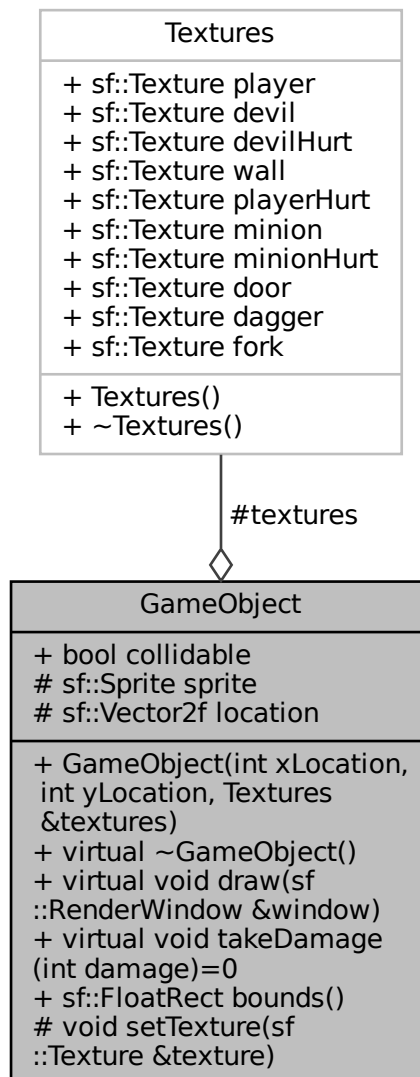
- Game/Game.h
- Game/Game.cc

4.5 GameObject Class Reference

Inheritance diagram for GameObject:



Collaboration diagram for GameObject:



Public Member Functions

- **GameObject** (int xLocation, int yLocation, [Textures](#) &textures)
Creates a new game object with the given location and textures.
- virtual void [draw](#) (sf::RenderWindow &window)
Renders the object on the screen.
- virtual void [takeDamage](#) (int damage)=0
Lowens 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

<i>window</i>	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

<i>texture</i>	SFML texture
----------------	--------------

4.5.1.4 takeDamage()

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

Lowers the hitpoints of the object.

Parameters

<i>damage</i>	amount of hitpoints to deduct
---------------	-------------------------------

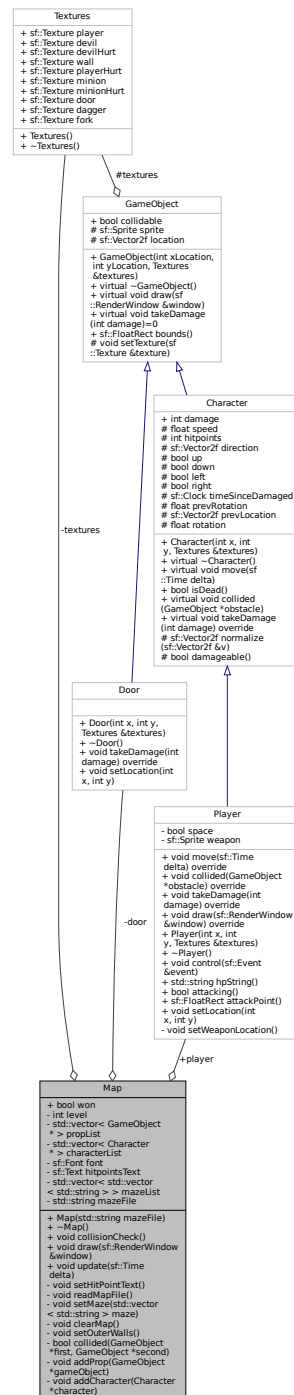
Implemented in [Character](#), [Devil](#), [Door](#), [Minion](#), [Player](#), and [Wall](#).

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

<i>character</i>	any character
------------------	---------------

4.6.1.2 addProp()

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

adds a gameobject to propList vector

Parameters

<i>gameObject</i>	any game object
-------------------	-----------------

4.6.1.3 collided()

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

Checks if two objects have collided.

Parameters

<i>first</i>	any game object
<i>second</i>	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

<i>window</i>	Reference to the game window
---------------	------------------------------

4.6.1.5 setMaze()

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

Setup the Maze/level.

Parameters

<i>maze</i>	the maze as read from file
-------------	----------------------------

4.6.1.6 update()

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

updates all objects and thier attributes

Parameters

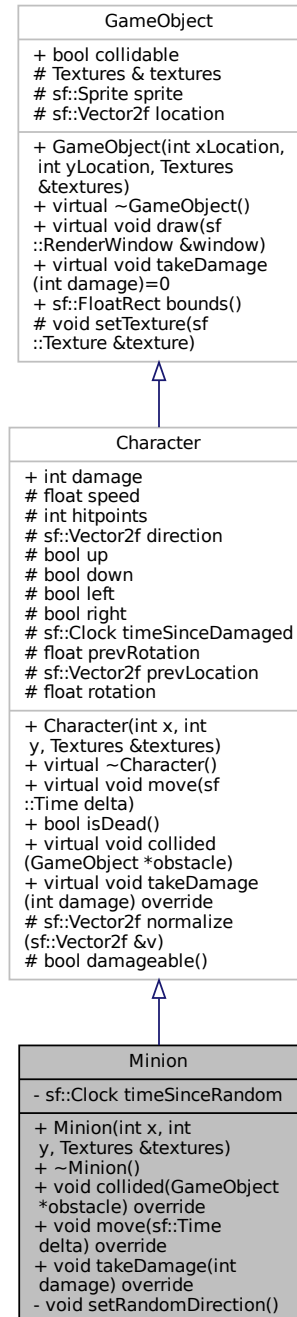
<i>delta</i>	time diffrence between each gameloop
--------------	--------------------------------------

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

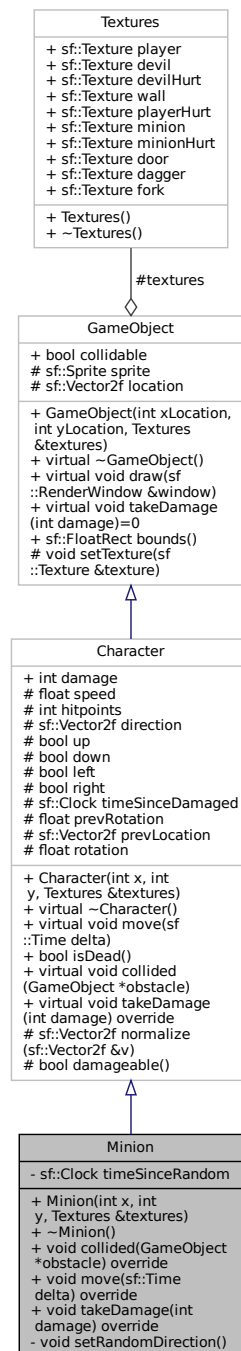
- Game/Map.h
- Game/Map.cc

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

<i>obstacle</i>	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

<i>delta</i>	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

<i>damage</i>	amount of hitpoints to deduct
---------------	-------------------------------

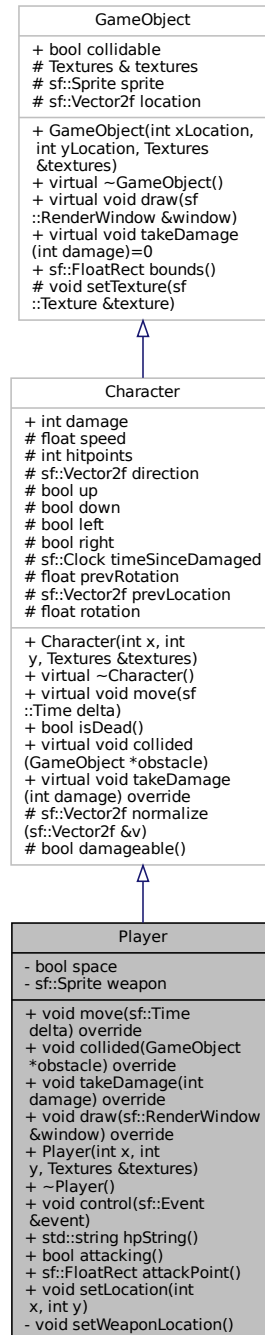
Reimplemented from [Character](#).

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

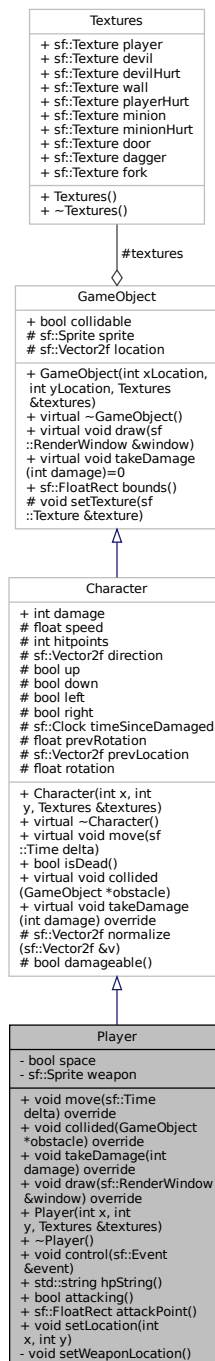
- Game/Minion.h
- Game/Minion.cc

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

<i>obstacle</i>	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

<i>event</i>	event containing keyboard input
--------------	---------------------------------

4.8.1.5 draw()

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

Renders the object on the screen.

Parameters

<i>window</i>	Reference 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

<i>delta</i>	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

<i>x</i>	x value of location
<i>y</i>	y value of location

4.8.1.9 takeDamage()

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

Lowers the hitpoints of the object.

Parameters

<i>damage</i>	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

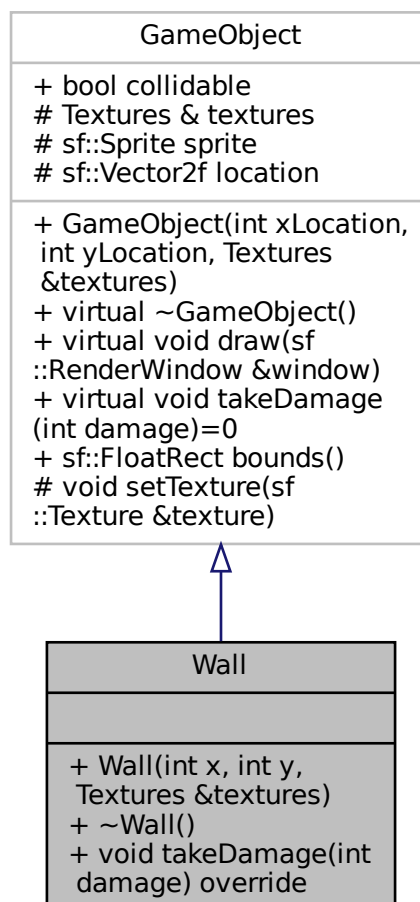
- sf::Texture **player**
- sf::Texture **devil**
- sf::Texture **devilHurt**
- sf::Texture **wall**
- sf::Texture **playerHurt**
- sf::Texture **minion**
- sf::Texture **minionHurt**
- sf::Texture **door**
- sf::Texture **dagger**
- sf::Texture **fork**

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

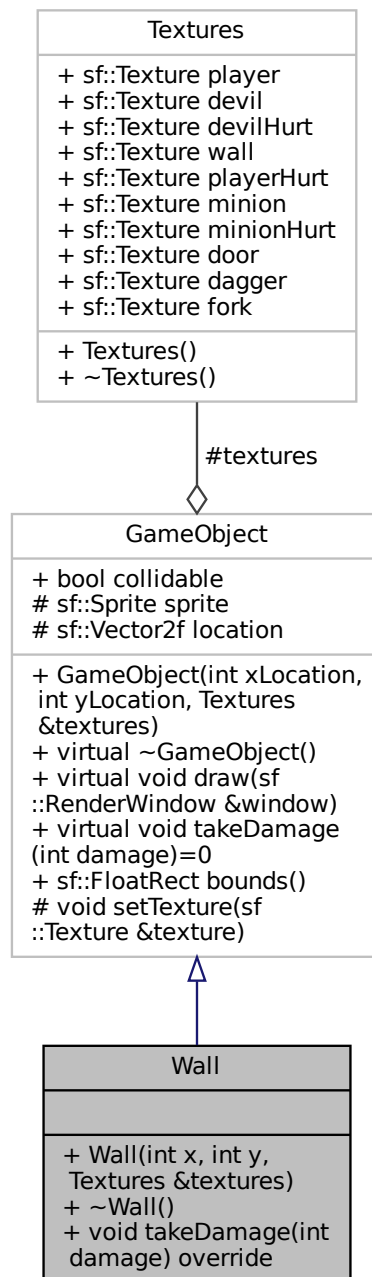
- Game/Textures.h
- Game/Textures.cc

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

<i>damage</i>	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:
9         float speed;
10        int hitpoints{1};
11        sf::Vector2f direction{0,0};
12        bool up{false};
13        bool down{false};
14        bool left{false};
15        bool right{false};
16        sf::Clock timeSinceDamaged;
17        float prevRotation{0};
18        sf::Vector2f prevLocation{0,0};
19        float rotation{0};
20    public:
21        int damage;
22    // Members
23    protected:
24        sf::Vector2f normalize(sf::Vector2f &v);
25        bool damageable();
26    public:
27        Character(int x, int y, Textures &textures);
28        virtual ~Character();
29        virtual void move(sf::Time delta);
30        bool isDead();
31        virtual void collided(GameObject *obstacle);
32        virtual void takeDamage(int damage) override;
33    };
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
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:
10        void setRandomDirection();
11    public:
12        Devil(int x, int y, Textures &textures);
13        ~Devil();
14        void collided(GameObject *obstacle) override;
15        void move(sf::Time delta) override;
16        void takeDamage(int damage) override;
17    };
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
```

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:
18     int level{0};
19     std::vector<GameObject *> propList;
20     std::vector<Character *> characterList;
21     Textures textures;
22     sf::Font font;
23     sf::Text hitpointsText;
24     std::vector<std::vector<std::string>> mazeList;
25     Door *door;
26     std::string mazeFile;
27 public:
28     bool won{false};
29     Player *player;
30 // Members
31 private:
32     void setHitPointText();
33     void readMapFile();
34     void setMaze(std::vector<std::string> maze);
35     void clearMap();
36     void setOuterWalls();
37     bool collided(GameObject *first, GameObject *second);
38     void addProp(GameObject *gameObject);
39     void addCharacter(Character *character);
40 public:
41     Map(std::string mazeFile);
42     ~Map();
43     void collisionCheck();
44     void draw(sf::RenderWindow &window);
45     void update(sf::Time delta);
46 };

```

5.7 Minion.h

```

1 #pragma once
2 #include "Character.h"
3
4 class Minion : public Character
5 {
6 private:
7     sf::Clock timeSinceRandom;
8     void setRandomDirection();
9 public:
10     Minion(int x, int y, Textures &textures);
11     ~Minion();
12     void collided(GameObject *obstacle) override;
13     void move(sf::Time delta) override;
14     void takeDamage(int damage) override;
15 };
16

```

5.8 Player.h

```

1 #pragma once
2
3 #include "Character.h"
4
5 class Player : public Character
6 {
7 // Attributes
8 private:
9     bool space{false};
10     sf::Sprite weapon;
11 public:
12

```

```
21 // Members
22 private:
23     void setWeaponLocation();
24
25 public:
26     void move(sf::Time delta) override;
27     void collided(GameObject *obstacle) override;
28     void takeDamage(int damage) override;
29     void draw(sf::RenderWindow &window) override;
30
31     Player(int x, int y, Textures &textures);
32     ~Player();
33     void control(sf::Event &event);
34     std::string hpString();
35     bool attacking();
36     sf::FloatRect attackPoint();
37     void setLocation(int x, int y);
38 };
```

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

- addCharacter
 - Map, [27](#)
- addProp
 - Map, [27](#)
- attacking
 - Player, [35](#)
- attackPoint
 - Player, [35](#)
- bounds
 - GameObject, [23](#)
- Character, [8](#)
 - collided, [10](#)
 - damageable, [11](#)
 - isDead, [11](#)
 - move, [11](#)
 - normalize, [11](#)
 - takeDamage, [12](#)
- collided
 - Character, [10](#)
 - Devil, [15](#)
 - Map, [27](#)
 - Minion, [31](#)
 - Player, [36](#)
- control
 - Player, [36](#)
- damageable
 - Character, [11](#)
- Devil, [13](#)
 - collided, [15](#)
 - move, [15](#)
 - takeDamage, [15](#)
- Door, [16](#)
 - setLocation, [18](#)
 - takeDamage, [18](#)
- draw
 - GameObject, [23](#)
 - Map, [28](#)
 - Player, [36](#)
- Game, [19](#)
 - update, [20](#)
- Game/Character.h, [43](#)
- Game/Devil.h, [43](#)
- Game/Door.h, [44](#)
- Game/Game.h, [44](#)
- Game/GameObject.h, [44](#)
- Game/Map.h, [44](#)
- Game/Minion.h, [45](#)
- Game/Player.h, [45](#)
- Game/Textures.h, [46](#)
- Game/Wall.h, [46](#)
- GameObject, [21](#)
 - bounds, [23](#)
 - draw, [23](#)
 - setTexture, [24](#)
 - takeDamage, [24](#)
- hpString
 - Player, [37](#)
- isDead
 - Character, [11](#)
- Map, [25](#)
 - addCharacter, [27](#)
 - addProp, [27](#)
 - collided, [27](#)
 - draw, [28](#)
 - setMaze, [28](#)
 - update, [28](#)
- Minion, [29](#)
 - collided, [31](#)
 - move, [31](#)
 - takeDamage, [32](#)
- move
 - Character, [11](#)
 - Devil, [15](#)
 - Minion, [31](#)
 - Player, [37](#)
- normalize
 - Character, [11](#)
- Player, [33](#)
 - attacking, [35](#)
 - attackPoint, [35](#)
 - collided, [36](#)
 - control, [36](#)
 - draw, [36](#)
 - hpString, [37](#)
 - move, [37](#)
 - setLocation, [37](#)
 - takeDamage, [38](#)
- setLocation
 - Door, [18](#)
 - Player, [37](#)
- setMaze

- Map, [28](#)
- setTexture
 - GameObject, [24](#)
- takeDamage
 - Character, [12](#)
 - Devil, [15](#)
 - Door, [18](#)
 - GameObject, [24](#)
 - Minion, [32](#)
 - Player, [38](#)
 - Wall, [41](#)
- Textures, [38](#)
- update
 - Game, [20](#)
 - Map, [28](#)
- Wall, [39](#)
 - takeDamage, [41](#)