

Me want cookie!

Generated by Doxygen 1.9.0

1 Hierarchical Index	1
1.1 Class Hierarchy	1
2 Class Index	3
2.1 Class List	3
3 Class Documentation	5
3.1 BadCookie Class Reference	5
3.1.1 Detailed Description	5
3.1.2 Constructor & Destructor Documentation	5
3.1.2.1 BadCookie()	6
3.2 Carrot Class Reference	6
3.2.1 Detailed Description	6
3.2.2 Constructor & Destructor Documentation	6
3.2.2.1 Carrot()	7
3.2.2.2 ~Carrot()	7
3.3 Chili Class Reference	7
3.3.1 Detailed Description	8
3.3.2 Constructor & Destructor Documentation	8
3.3.2.1 Chili()	8
3.3.2.2 ~Chili()	8
3.3.3 Member Function Documentation	8
3.3.3.1 collision()	8
3.4 Cookie Class Reference	9
3.4.1 Detailed Description	9
3.4.2 Constructor & Destructor Documentation	9
3.4.2.1 Cookie()	9
3.4.2.2 ~Cookie()	9
3.4.3 Member Function Documentation	10
3.4.3.1 update()	10
3.5 Enemy Class Reference	10
3.5.1 Detailed Description	10
3.5.2 Constructor & Destructor Documentation	11
3.5.2.1 Enemy()	11
3.5.2.2 ~Enemy()	11
3.5.3 Member Function Documentation	11
3.5.3.1 move()	11
3.5.3.2 update()	11
3.6 Entity Class Reference	12
3.6.1 Detailed Description	12
3.6.2 Constructor & Destructor Documentation	12
3.6.2.1 Entity()	12
3.6.2.2 ~Entity()	13

3.6.3 Member Function Documentation	13
3.6.3.1 move()	13
3.6.3.2 update()	13
3.6.4 Member Data Documentation	13
3.6.4.1 move_direction	13
3.6.4.2 movement	14
3.7 Game Class Reference	14
3.7.1 Detailed Description	14
3.7.2 Constructor & Destructor Documentation	14
3.7.2.1 Game()	14
3.7.3 Member Function Documentation	14
3.7.3.1 run()	15
3.8 GoodCookie Class Reference	15
3.8.1 Detailed Description	15
3.8.2 Constructor & Destructor Documentation	15
3.8.2.1 GoodCookie()	16
3.9 Menu Class Reference	16
3.9.1 Detailed Description	16
3.9.2 Constructor & Destructor Documentation	16
3.9.2.1 Menu()	16
3.9.3 Member Function Documentation	16
3.9.3.1 getPressedItem()	17
3.9.3.2 run()	17
3.10 Object Class Reference	17
3.10.1 Constructor & Destructor Documentation	18
3.10.1.1 Object()	18
3.10.1.2 ~Object()	18
3.10.2 Member Function Documentation	18
3.10.2.1 collision()	18
3.10.2.2 draw()	18
3.10.2.3 getsprite()	19
3.10.2.4 update()	19
3.10.3 Member Data Documentation	19
3.10.3.1 sprite	19
3.10.3.2 texture	19
3.11 Player Class Reference	19
3.11.1 Detailed Description	20
3.11.2 Constructor & Destructor Documentation	20
3.11.2.1 Player()	20
3.11.3 Member Function Documentation	20
3.11.3.1 change_coordinates()	20
3.11.3.2 collision_with_wall()	20

3.11.3.3 <code>get_lives()</code>	21
3.11.3.4 <code>get_points()</code>	21
3.11.3.5 <code>input()</code>	21
3.11.3.6 <code>move()</code>	21
3.11.3.7 <code>set_lives()</code>	21
3.11.3.8 <code>set_points()</code>	21
3.11.3.9 <code>set_speed()</code>	22
3.11.3.10 <code>update()</code>	22
3.12 Wall Class Reference	22
3.12.1 Detailed Description	22
3.12.2 Constructor & Destructor Documentation	23
3.12.2.1 <code>Wall()</code>	23
3.12.2.2 <code>~Wall()</code>	23
3.12.3 Member Function Documentation	23
3.12.3.1 <code>update()</code>	23
3.13 World Class Reference	23
3.13.1 Detailed Description	24
3.13.2 Constructor & Destructor Documentation	24
3.13.2.1 <code>World()</code>	24
3.13.2.2 <code>~World()</code>	24
3.13.3 Member Function Documentation	24
3.13.3.1 <code>remove_object()</code>	24
3.13.4 Member Data Documentation	25
3.13.4.1 <code>entities</code>	25
3.13.4.2 <code>myPlayer</code>	25
3.13.4.3 <code>walls</code>	25
3.13.4.4 <code>win</code>	25
Index	27

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

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

Game	14
Menu	16
Object	17
Cookie	9
BadCookie	5
GoodCookie	15
Entity	12
Enemy	10
Carrot	6
Chili	7
Player	19
Wall	22
World	23

Chapter 2

Class Index

2.1 Class List

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

BadCookie	5
Carrot	6
Chili	7
Cookie	9
Enemy	10
Entity	12
Game	14
GoodCookie	15
Menu	16
Object	17
Player	19
Wall	22
World	23

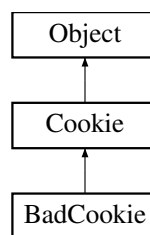
Chapter 3

Class Documentation

3.1 BadCookie Class Reference

```
#include <cookie.h>
```

Inheritance diagram for BadCookie:



Public Member Functions

- [BadCookie](#) (float x, float y)

Additional Inherited Members

3.1.1 Detailed Description

Defines the [BadCookie](#).

3.1.2 Constructor & Destructor Documentation

3.1.2.1 BadCookie()

```
BadCookie::BadCookie (
    float x,
    float y )
```

Constructor that takes coordinates as parameters.

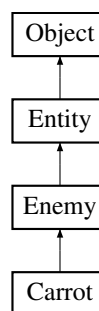
The documentation for this class was generated from the following files:

- cookie.h
- cookie.cc

3.2 Carrot Class Reference

```
#include <enemy.h>
```

Inheritance diagram for Carrot:



Public Member Functions

- [Carrot](#) (float x, float y, int dir_1, int dir_2, int dir_3, int dir_4, float dist)
- [~Carrot](#) ()=default

Additional Inherited Members

3.2.1 Detailed Description

Defines the carrot.

3.2.2 Constructor & Destructor Documentation

3.2.2.1 Carrot()

```
Carrot::Carrot (
    float x,
    float y,
    int dir_1,
    int dir_2,
    int dir_3,
    int dir_4,
    float dist )
```

Constructor that takes the carrots position, route and distance as parameters.

3.2.2.2 ~Carrot()

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

Destructor set to default.

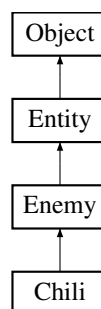
The documentation for this class was generated from the following files:

- enemy.h
- enemy.cc

3.3 Chili Class Reference

```
#include <enemy.h>
```

Inheritance diagram for Chili:



Public Member Functions

- [Chili](#) (float x, float y, int dir_1, int dir_2, int dir_3, int dir_4, float dist)
- [~Chili](#) ()=default
- bool [collision](#) ([Object](#) const &object) const override

Additional Inherited Members

3.3.1 Detailed Description

Defines the chili.

3.3.2 Constructor & Destructor Documentation

3.3.2.1 Chili()

```
Chili::Chili (
    float x,
    float y,
    int dir_1,
    int dir_2,
    int dir_3,
    int dir_4,
    float dist )
```

Constructor that takes the chilis position, route and distance as parameters.

3.3.2.2 ~Chili()

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

Destructor set to default.

3.3.3 Member Function Documentation

3.3.3.1 collision()

```
bool Chili::collision (
    Object const & object ) const [override], [virtual]
```

Checks if the player is within a certain range of the chili.

Reimplemented from [Object](#).

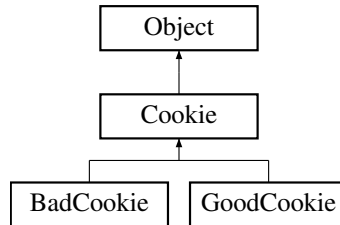
The documentation for this class was generated from the following files:

- enemy.h
- enemy.cc

3.4 Cookie Class Reference

```
#include <cookie.h>
```

Inheritance diagram for Cookie:



Public Member Functions

- [Cookie](#) (std::string, float x, float y, int points)
- [~Cookie](#) ()=default
- void [update](#) ([Player](#) &object, sf::Time time)

Additional Inherited Members

3.4.1 Detailed Description

Defines the cookies

3.4.2 Constructor & Destructor Documentation

3.4.2.1 Cookie()

```
Cookie::Cookie (
    std::string image,
    float x,
    float y,
    int points )
```

Constructor that takes the texture, coordinates and the amount of points the cookie will give to the player.

3.4.2.2 ~Cookie()

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

Destructor set to default.

3.4.3 Member Function Documentation

3.4.3.1 update()

```
void Cookie::update (
    Player & object,
    sf::Time time ) [virtual]
```

Updates the player in the game loop when it has collided with a cookie.

Implements [Object](#).

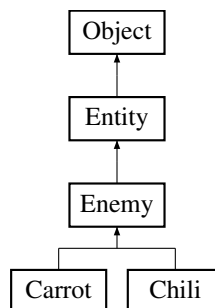
The documentation for this class was generated from the following files:

- cookie.h
- cookie.cc

3.5 Enemy Class Reference

```
#include <enemy.h>
```

Inheritance diagram for Enemy:



Public Member Functions

- [Enemy](#) (std::string, float, float, int, int, int, int, float, int)
- [~Enemy](#) ()=default
- void [move](#) (sf::Time)
- void [update](#) ([Player](#) &object, sf::Time time)

Additional Inherited Members

3.5.1 Detailed Description

Defines the games enemies.

3.5.2 Constructor & Destructor Documentation

3.5.2.1 Enemy()

```
Enemy::Enemy (
    std::string image,
    float x,
    float y,
    int dir_1,
    int dir_2,
    int dir_3,
    int dir_4,
    float dist,
    int lives )
```

Constructor that takes the texture, coordinates and route of the enemy as well as how far it walks and how many lives it takes.

3.5.2.2 ~Enemy()

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

The enemys destructor set to default.

3.5.3 Member Function Documentation

3.5.3.1 move()

```
void Enemy::move (
    sf::Time time ) [virtual]
```

Moves the enemy according to it's route.

Implements [Entity](#).

3.5.3.2 update()

```
void Enemy::update (
    Player & object,
    sf::Time time ) [virtual]
```

Updates enemy and player in the game loop.

Implements [Entity](#).

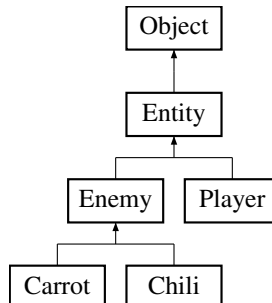
The documentation for this class was generated from the following files:

- enemy.h
- enemy.cc

3.6 Entity Class Reference

```
#include <entity.h>
```

Inheritance diagram for Entity:



Public Member Functions

- `Entity` (std::string text, float w, float h, float x, float y)
- `~Entity` ()=default
- virtual void `move` (sf::Time)=0
- virtual void `update` (Player &object, sf::Time time)=0

Protected Attributes

- int `move_direction` {}
- sf::Time `movement` {}

3.6.1 Detailed Description

Defines the entities in the game.

3.6.2 Constructor & Destructor Documentation

3.6.2.1 Entity()

```
Entity::Entity (
    std::string text,
    float w,
    float h,
    float x,
    float y )
```

Constructor that takes 4 floats as parameter that represent the size and placement of the entity. The constructor also takes the texture that the sprite will be set to.

3.6.2.2 ~Entity()

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

Default destructor.

3.6.3 Member Function Documentation

3.6.3.1 move()

```
virtual void Entity::move (
    sf::Time ) [pure virtual]
```

Defines how the entities move in the game.

Implemented in [Player](#), and [Enemy](#).

3.6.3.2 update()

```
virtual void Entity::update (
    Player & object,
    sf::Time time ) [pure virtual]
```

Defines how the object and player will be updated in the game loop.

Implements [Object](#).

Implemented in [Player](#), and [Enemy](#).

3.6.4 Member Data Documentation

3.6.4.1 move_direction

```
int Entity::move_direction {} [protected]
```

Private variable that contains an integer that represent the direction that the entity is moving.

3.6.4.2 movement

```
sf::Time Entity::movement {} [protected]
```

Private variable that is used to check how the entity should moves.

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

- entity.h
- entity.cc

3.7 Game Class Reference

```
#include <game.h>
```

Public Member Functions

- [Game](#) (std::string const &level)
- void [run](#) (sf::RenderWindow &window)

3.7.1 Detailed Description

Runs and updates the game.

3.7.2 Constructor & Destructor Documentation

3.7.2.1 Game()

```
Game::Game (
    std::string const & level )
```

The constructor takes the name of the file that contains the level the game will run as parameter.

3.7.3 Member Function Documentation

3.7.3.1 run()

```
void Game::run (
    sf::RenderWindow & window )
```

Runs the game loop.

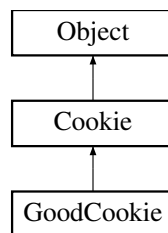
The documentation for this class was generated from the following files:

- game.h
- game.cc

3.8 GoodCookie Class Reference

```
#include <cookie.h>
```

Inheritance diagram for GoodCookie:



Public Member Functions

- [GoodCookie](#) (float x, float y)

Additional Inherited Members

3.8.1 Detailed Description

Defines the [GoodCookie](#).

3.8.2 Constructor & Destructor Documentation

3.8.2.1 GoodCookie()

```
GoodCookie::GoodCookie (
    float x,
    float y )
```

Constructor that takes coordinates as parameters.

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

- cookie.h
- cookie.cc

3.9 Menu Class Reference

```
#include <Menu.h>
```

Public Member Functions

- [Menu](#) (float width, float height)
- int [getPressedItem](#) ()
- int [run](#) (sf::RenderWindow &window)

3.9.1 Detailed Description

Handles showing the menu and selecting options on screen.

3.9.2 Constructor & Destructor Documentation

3.9.2.1 Menu()

```
Menu::Menu (
    float width,
    float height )
```

The constructor takes the width and height to know how to center the menu.

3.9.3 Member Function Documentation

3.9.3.1 getPressedItem()

```
int Menu::getPressedItem ( )
```

Returns the ItemIndex variable to find which button was pressed.

3.9.3.2 run()

```
int Menu::run (
    sf::RenderWindow & window )
```

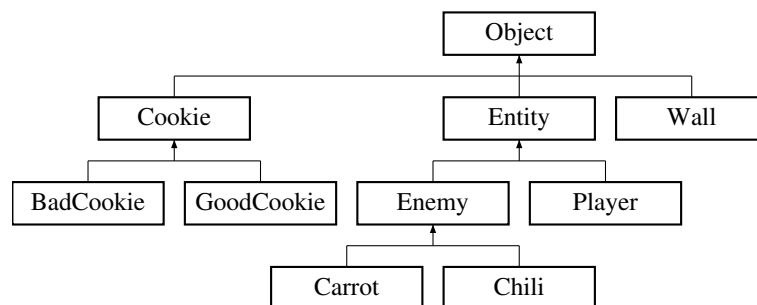
Function that runs the menu loop and checks if a button has been pressed.

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

- Menu.h
- Menu.cc

3.10 Object Class Reference

Inheritance diagram for Object:



Public Member Functions

- [Object](#) (std::string text, float w, float h, float x, float y)
- virtual [~Object](#) ()=default
- void [draw](#) (sf::RenderWindow &window) const
- virtual void [update](#) ([Player](#) &object, sf::Time time)=0
- virtual bool [collision](#) ([Object](#) const &other) const
- sf::Sprite [getsprite](#) () const

Protected Attributes

- sf::Sprite [sprite](#)
- sf::Texture [texture](#)

3.10.1 Constructor & Destructor Documentation

3.10.1.1 Object()

```
Object::Object (
    std::string text,
    float w,
    float h,
    float x,
    float y )
```

Constructor that takes texture, the scale and position as parameters for the objects sprite.

3.10.1.2 ~Object()

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

Default destructor.

3.10.2 Member Function Documentation

3.10.2.1 collision()

```
bool Object::collision (
    Object const & other ) const [virtual]
```

Defines when tow objects have collided with eachother.

Reimplemented in [Chili](#).

3.10.2.2 draw()

```
void Object::draw (
    sf::RenderWindow & window ) const
```

Draws the saved sprite to a RenderWindow.

3.10.2.3 getsprite()

```
sf::Sprite Object::getsprite ( ) const
```

Returns the saved sprite.

3.10.2.4 update()

```
virtual void Object::update (
    Player & object,
    sf::Time time ) [pure virtual]
```

Defines how the player will be updated in the game loop.

Implemented in [Entity](#), [Wall](#), [Player](#), [Enemy](#), and [Cookie](#).

3.10.3 Member Data Documentation

3.10.3.1 sprite

```
sf::Sprite Object::sprite [protected]
```

Private variable that holds the sprite that represents the object.

3.10.3.2 texture

```
sf::Texture Object::texture [protected]
```

Private variable that contains the texture that the sprite has.

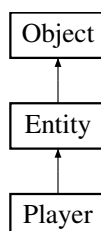
The documentation for this class was generated from the following files:

- [object.h](#)
- [object.cc](#)

3.11 Player Class Reference

```
#include <player.h>
```

Inheritance diagram for Player:



Public Member Functions

- [Player](#) ()
- void [input](#) (int, sf::Time)
- void [move](#) (sf::Time)
- void [update](#) ([Player](#) &object, sf::Time time)
- void [collision_with_wall](#) ()
- void [set_points](#) (int new_points)
- int [get_points](#) () const
- void [set_lives](#) (int new_lives)
- int [get_lives](#) () const
- void [set_speed](#) (float)
- void [change_coordinates](#) (float, float)

Additional Inherited Members

3.11.1 Detailed Description

Defines the player.

3.11.2 Constructor & Destructor Documentation

3.11.2.1 Player()

```
Player::Player ( )
```

The constructor does not take any parameters.

3.11.3 Member Function Documentation

3.11.3.1 change_coordinates()

```
void Player::change_coordinates (
    float x,
    float y )
```

Changes players coordinates when initialized in world.

3.11.3.2 collision_with_wall()

```
void Player::collision_with_wall ( )
```

Checks if player collides with wall and change direction.

3.11.3.3 `get_lives()`

```
int Player::get_lives ( ) const
```

Retrieves the amount of lives the player has.

3.11.3.4 `get_points()`

```
int Player::get_points ( ) const
```

Retrieves the amount of points the player has.

3.11.3.5 `input()`

```
void Player::input (
    int direction,
    sf::Time time )
```

Takes the input from game and initializes movement.

3.11.3.6 `move()`

```
void Player::move (
    sf::Time time ) [virtual]
```

Checks if player can move in said direction and then execute.

Implements [Entity](#).

3.11.3.7 `set_lives()`

```
void Player::set_lives (
    int new_lives )
```

Changes the lives variable.

3.11.3.8 `set_points()`

```
void Player::set_points (
    int new_points )
```

Changes the points variable.

3.11.3.9 set_speed()

```
void Player::set_speed (
    float new_speed )
```

Changes the speed variable.

3.11.3.10 update()

```
void Player::update (
    Player & object,
    sf::Time time ) [inline], [virtual]
```

Players update does not do anything.

Implements [Entity](#).

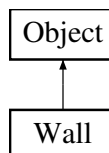
The documentation for this class was generated from the following files:

- player.h
- player.cc

3.12 Wall Class Reference

```
#include <wall.h>
```

Inheritance diagram for Wall:



Public Member Functions

- [Wall](#) (float, float, float, float)
- [~Wall](#) ()=default
- void [update](#) ([Player](#) &object, sf::Time time)

Additional Inherited Members

3.12.1 Detailed Description

Defines the walls in the game

3.12.2 Constructor & Destructor Documentation

3.12.2.1 Wall()

```
Wall::Wall (
    float h,
    float w,
    float x,
    float y )
```

Constructor that takes 4 floats that represent the size and placement of the wall.

3.12.2.2 ~Wall()

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

Destructor set to default.

3.12.3 Member Function Documentation

3.12.3.1 update()

```
void Wall::update (
    Player & object,
    sf::Time time ) [inline], [virtual]
```

Walls update does not do anything.

Implements [Object](#).

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

- wall.h
- wall.cc

3.13 World Class Reference

```
#include <world.h>
```

Public Member Functions

- [World](#) (std::string levelname)
- [~World](#) ()
- void [remove_object](#) ([Object](#) &object)

Public Attributes

- std::vector< [Object](#) * > [walls](#)
- std::vector< [Object](#) * > [entities](#)
- [Player](#) [myPlayer](#)
- sf::Sprite [win](#) {}

3.13.1 Detailed Description

Holds all the game objects, and reads them from a file.

3.13.2 Constructor & Destructor Documentation

3.13.2.1 World()

```
World::World (
    std::string levelname )
```

The constructor takes the name of the file that contains the level.

3.13.2.2 ~World()

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

Deletes all objects that are stored on the heap.

3.13.3 Member Function Documentation

3.13.3.1 remove_object()

```
void World::remove_object (
    Object & object )
```

Removes one object from the entities vector.

3.13.4 Member Data Documentation

3.13.4.1 entities

```
std::vector<Object*> World::entities
```

Public variable that contains entity objects.

3.13.4.2 myPlayer

```
Player World::myPlayer
```

Public variable an object of type player.

3.13.4.3 walls

```
std::vector<Object*> World::walls
```

Public variable that contains wall objects.

3.13.4.4 win

```
sf::Sprite World::win {}
```

A sprite that represent the goal for the player to reach.

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

- world.h
- world.cc

Index

- ~Carrot
 - Carrot, [7](#)
- ~Chili
 - Chili, [8](#)
- ~Cookie
 - Cookie, [9](#)
- ~Enemy
 - Enemy, [11](#)
- ~Entity
 - Entity, [12](#)
- ~Object
 - Object, [18](#)
- ~Wall
 - Wall, [23](#)
- ~World
 - World, [24](#)
- BadCookie, [5](#)
 - BadCookie, [5](#)
- Carrot, [6](#)
 - ~Carrot, [7](#)
 - Carrot, [6](#)
- change_coordinates
 - Player, [20](#)
- Chili, [7](#)
 - ~Chili, [8](#)
 - Chili, [8](#)
 - collision, [8](#)
- collision
 - Chili, [8](#)
 - Object, [18](#)
- collision_with_wall
 - Player, [20](#)
- Cookie, [9](#)
 - ~Cookie, [9](#)
 - Cookie, [9](#)
 - update, [10](#)
- draw
 - Object, [18](#)
- Enemy, [10](#)
 - ~Enemy, [11](#)
 - Enemy, [11](#)
 - move, [11](#)
 - update, [11](#)
- entities
 - World, [25](#)
- Entity, [12](#)
 - ~Entity, [12](#)
 - Entity, [12](#)
 - move, [13](#)
 - move_direction, [13](#)
 - movement, [13](#)
 - update, [13](#)
- Game, [14](#)
 - Game, [14](#)
 - run, [14](#)
- get_lives
 - Player, [20](#)
- get_points
 - Player, [21](#)
- getPressedItem
 - Menu, [16](#)
- getsprite
 - Object, [18](#)
- GoodCookie, [15](#)
 - GoodCookie, [15](#)
- input
 - Player, [21](#)
- Menu, [16](#)
 - getPressedItem, [16](#)
 - Menu, [16](#)
 - run, [17](#)
- move
 - Enemy, [11](#)
 - Entity, [13](#)
 - Player, [21](#)
- move_direction
 - Entity, [13](#)
- movement
 - Entity, [13](#)
- myPlayer
 - World, [25](#)
- Object, [17](#)
 - ~Object, [18](#)
 - collision, [18](#)
 - draw, [18](#)
 - getsprite, [18](#)
 - Object, [18](#)
 - sprite, [19](#)
 - texture, [19](#)
 - update, [19](#)
- Player, [19](#)
 - change_coordinates, [20](#)

- collision_with_wall, [20](#)
- get_lives, [20](#)
- get_points, [21](#)
- input, [21](#)
- move, [21](#)
- Player, [20](#)
- set_lives, [21](#)
- set_points, [21](#)
- set_speed, [21](#)
- update, [22](#)

remove_object

- World, [24](#)

run

- Game, [14](#)
- Menu, [17](#)

set_lives

- Player, [21](#)

set_points

- Player, [21](#)

set_speed

- Player, [21](#)

sprite

- Object, [19](#)

texture

- Object, [19](#)

update

- Cookie, [10](#)
- Enemy, [11](#)
- Entity, [13](#)
- Object, [19](#)
- Player, [22](#)
- Wall, [23](#)

Wall, [22](#)

- ~Wall, [23](#)
- update, [23](#)
- Wall, [23](#)

walls

- World, [25](#)

win

- World, [25](#)

World, [23](#)

- ~World, [24](#)
- entities, [25](#)
- myPlayer, [25](#)
- remove_object, [24](#)
- walls, [25](#)
- win, [25](#)
- World, [24](#)