

## Arcade

Generated by Doxygen 1.9.1



<b>1 Hierarchical Index</b>	<b>1</b>
1.1 Class Hierarchy	1
<b>2 Class Index</b>	<b>3</b>
2.1 Class List	3
<b>3 File Index</b>	<b>5</b>
3.1 File List	5
<b>4 Class Documentation</b>	<b>7</b>
4.1 Cell Class Reference	7
4.1.1 Constructor & Destructor Documentation	8
4.1.1.1 Cell() [1/2]	8
4.1.1.2 Cell() [2/2]	8
4.1.1.3 ~Cell()	8
4.1.2 Member Function Documentation	8
4.1.2.1 checkCollision()	9
4.1.2.2 getPos()	9
4.1.2.3 getRotation()	9
4.1.2.4 getX()	9
4.1.2.5 getY()	9
4.1.2.6 setPos() [1/2]	9
4.1.2.7 setPos() [2/2]	9
4.1.2.8 setRotation()	10
4.1.3 Member Data Documentation	10
4.1.3.1 _pos	10
4.2 core Class Reference	10
4.2.1 Detailed Description	12
4.2.2 Constructor & Destructor Documentation	13
4.2.2.1 core() [1/2]	13
4.2.2.2 core() [2/2]	13
4.2.2.3 ~core()	13
4.2.3 Member Function Documentation	13
4.2.3.1 check_time()	13
4.2.3.2 handleInputConfig()	14
4.2.3.3 handleInputGame()	14
4.2.3.4 handleInputMenu()	14
4.2.3.5 initConfig()	14
4.2.3.6 initGame()	15
4.2.3.7 initMenu()	15
4.2.3.8 loadGame()	15
4.2.3.9 loadGraphical()	15
4.2.3.10 loadLibrariesFromDirectory()	15

4.2.3.11 loop()	16
4.2.3.12 loopConfig()	16
4.2.3.13 loopEnd()	16
4.2.3.14 loopGame()	16
4.2.3.15 loopHighScore()	16
4.2.3.16 loopMenu()	17
4.2.3.17 switchGraphics()	17
4.2.3.18 writeConfig()	17
4.2.4 Member Data Documentation	17
4.2.4.1 _clock_end	17
4.2.4.2 _clock_end_fixed	17
4.2.4.3 _clock_start	18
4.2.4.4 _clock_start_fixed	18
4.2.4.5 _configMenuChoice	18
4.2.4.6 _currentGameLibraryPath	18
4.2.4.7 _currentGraphicLibrary	18
4.2.4.8 _game	18
4.2.4.9 _gameLibraries	19
4.2.4.10 _graphical	19
4.2.4.11 _graphicLibraries	19
4.2.4.12 _key	19
4.2.4.13 _menustate	19
4.2.4.14 _optionsConfig	19
4.2.4.15 _optionsMenu	20
4.2.4.16 _state	20
4.2.4.17 destroyGame	20
4.2.4.18 destroyGraphic	20
4.3 Food Class Reference	21
4.3.1 Constructor & Destructor Documentation	21
4.3.1.1 Food() [1/2]	22
4.3.1.2 Food() [2/2]	22
4.3.2 Member Function Documentation	22
4.3.2.1 respawn()	22
4.4 HighScore Class Reference	22
4.4.1 Member Function Documentation	22
4.4.1.1 getLogin()	23
4.4.1.2 getTop10Reverse()	23
4.4.1.3 readScore()	23
4.4.1.4 writeScore()	23
4.5 IGames Class Reference	23
4.5.1 Detailed Description	24
4.5.2 Constructor & Destructor Documentation	24

4.5.2.1 ~IGames()	24
4.5.3 Member Function Documentation	25
4.5.3.1 collisionFood()	25
4.5.3.2 collisionMap()	26
4.5.3.3 getElements()	26
4.5.3.4 getScore()	26
4.5.3.5 getSpeed()	27
4.5.3.6 handleCollision()	27
4.5.3.7 handleInput()	27
4.5.3.8 init()	27
4.5.3.9 loadAssets()	28
4.5.3.10 loadTexts()	28
4.5.3.11 parse_map()	28
4.5.3.12 readConfig()	28
4.5.3.13 readScore()	28
4.5.3.14 update()	28
4.5.3.15 updateTexts()	29
4.5.3.16 writeScore()	29
4.6 IGraphic Class Reference	29
4.6.1 Detailed Description	30
4.6.2 Constructor & Destructor Documentation	30
4.6.2.1 ~IGraphic()	30
4.6.3 Member Function Documentation	30
4.6.3.1 debug_print()	31
4.6.3.2 deleteAssets()	31
4.6.3.3 display_asset()	31
4.6.3.4 display_high_score()	31
4.6.3.5 display_menu()	32
4.6.3.6 display_text()	32
4.6.3.7 displayConfig()	32
4.6.3.8 get_index_tuple()	33
4.6.3.9 get_key()	33
4.6.3.10 load_assets()	33
4.6.3.11 load_config()	34
4.6.3.12 load_menu()	34
4.6.3.13 load_text()	34
4.7 Map Class Reference	35
4.7.1 Constructor & Destructor Documentation	35
4.7.1.1 Map() [1/2]	35
4.7.1.2 Map() [2/2]	35
4.7.1.3 ~Map()	35
4.7.2 Member Function Documentation	35

4.7.2.1 getMap()	36
4.7.2.2 getPosition()	36
4.7.2.3 getPositionsVector()	36
4.7.3 Member Data Documentation	36
4.7.3.1 _map	36
4.8 Ncurses Class Reference	36
4.8.1 Constructor & Destructor Documentation	38
4.8.1.1 Ncurses()	38
4.8.1.2 ~Ncurses()	38
4.8.2 Member Function Documentation	38
4.8.2.1 debug_print()	38
4.8.2.2 deleteAssets()	38
4.8.2.3 display() [1/4]	39
4.8.2.4 display() [2/4]	39
4.8.2.5 display() [3/4]	39
4.8.2.6 display() [4/4]	39
4.8.2.7 display_asset()	39
4.8.2.8 display_high_score()	40
4.8.2.9 display_menu()	40
4.8.2.10 display_separation_x()	40
4.8.2.11 display_separation_y()	40
4.8.2.12 display_text()	41
4.8.2.13 displayConfig()	41
4.8.2.14 get_index_tuple()	41
4.8.2.15 get_key()	42
4.8.2.16 get_screen_x()	42
4.8.2.17 get_screen_y()	42
4.8.2.18 init()	42
4.8.2.19 load_assets()	42
4.8.2.20 load_config()	43
4.8.2.21 load_menu()	43
4.8.2.22 load_text()	43
4.8.2.23 window_clear()	43
4.8.2.24 window_refresh()	44
4.9 Nibbler Class Reference	44
4.9.1 Detailed Description	46
4.9.2 Constructor & Destructor Documentation	46
4.9.2.1 Nibbler() [1/2]	46
4.9.2.2 Nibbler() [2/2]	46
4.9.2.3 ~Nibbler()	47
4.9.3 Member Function Documentation	47
4.9.3.1 addTime()	47

4.9.3.2 autoMove()	47
4.9.3.3 collisionFood()	47
4.9.3.4 collisionMap()	48
4.9.3.5 getElements()	48
4.9.3.6 getIndexString()	48
4.9.3.7 getLogin()	49
4.9.3.8 getRemainingTime()	49
4.9.3.9 getScore()	49
4.9.3.10 getSpeed()	49
4.9.3.11 handleCollision()	50
4.9.3.12 handleInput()	50
4.9.3.13 init()	50
4.9.3.14 initTexts()	50
4.9.3.15 loadAssets()	50
4.9.3.16 loadTexts()	51
4.9.3.17 parse_map()	51
4.9.3.18 readConfig()	51
4.9.3.19 readScore()	51
4.9.3.20 TimeOut()	51
4.9.3.21 update()	52
4.9.3.22 updateTexts()	52
4.9.3.23 writeScore()	52
4.9.4 Member Data Documentation	52
4.9.4.1 _clock_end	52
4.9.4.2 _clock_start	52
4.9.4.3 _configData	53
4.9.4.4 _food	53
4.9.4.5 _highScore	53
4.9.4.6 _highscore	53
4.9.4.7 _level	53
4.9.4.8 _map	53
4.9.4.9 _score	53
4.9.4.10 _skinMultiplier	53
4.9.4.11 _snake	54
4.9.4.12 _speed	54
4.9.4.13 _string	54
4.9.4.14 HighScoreIsSet	54
4.10 PlayerS Class Reference	54
4.10.1 Constructor & Destructor Documentation	55
4.10.1.1 PlayerS() [1/2]	55
4.10.1.2 PlayerS() [2/2]	55
4.10.1.3 ~PlayerS()	55

4.10.2 Member Function Documentation	55
4.10.2.1 checkCollision()	55
4.10.2.2 checkEatHimself()	55
4.10.2.3 getBodyPos()	55
4.10.2.4 getHeadPos()	56
4.10.2.5 getRotation()	56
4.10.2.6 getSize()	56
4.10.2.7 getSnakePosVector()	56
4.10.2.8 getSnakeTailPosVector()	56
4.10.2.9 getTailPos()	56
4.10.2.10 grow()	56
4.10.2.11 move()	56
4.10.2.12 setRotation()	57
4.11 Position Class Reference	57
4.11.1 Detailed Description	58
4.11.2 Constructor & Destructor Documentation	58
4.11.2.1 Position() [1/3]	58
4.11.2.2 Position() [2/3]	58
4.11.2.3 Position() [3/3]	58
4.11.2.4 ~Position()	59
4.11.3 Member Function Documentation	59
4.11.3.1 getPos()	59
4.11.3.2 getPosPair()	59
4.11.3.3 getRotation()	59
4.11.3.4 getX()	60
4.11.3.5 getY()	60
4.11.3.6 setRotation()	60
4.11.3.7 setX()	60
4.11.3.8 setY()	60
4.11.4 Member Data Documentation	61
4.11.4.1 _rotation	61
4.11.4.2 _x	61
4.11.4.3 _y	61
4.12 Sdl2 Class Reference	62
4.12.1 Constructor & Destructor Documentation	63
4.12.1.1 Sdl2()	63
4.12.1.2 ~Sdl2()	63
4.12.2 Member Function Documentation	63
4.12.2.1 debug_print()	63
4.12.2.2 display_asset()	63
4.12.2.3 display_menu()	64
4.12.2.4 display_text()	64



4.12.2.5 get_index_tuple()	64
4.12.2.6 get_key()	65
4.12.2.7 init()	65
4.12.2.8 load_assets()	65
4.12.2.9 load_menu()	65
4.12.2.10 load_text()	66
4.12.3 Member Data Documentation	66
4.12.3.1 _active	66
4.12.3.2 _assets	66
4.12.3.3 _event	66
4.12.3.4 _menu	66
4.12.3.5 _renderer	67
4.12.3.6 _string	67
4.12.3.7 _window	67
4.13 SfmI Class Reference	67
4.13.1 Constructor & Destructor Documentation	69
4.13.1.1 SfmI()	69
4.13.1.2 ~SfmI()	69
4.13.2 Member Function Documentation	69
4.13.2.1 debug_print()	69
4.13.2.2 deleteAssets()	70
4.13.2.3 display_asset()	70
4.13.2.4 display_high_score()	70
4.13.2.5 display_menu()	70
4.13.2.6 display_text()	71
4.13.2.7 displayConfig()	71
4.13.2.8 get_index_tuple()	71
4.13.2.9 get_key()	72
4.13.2.10 getConfigData()	72
4.13.2.11 init()	72
4.13.2.12 isConfirmSoundPlaying()	73
4.13.2.13 load_assets()	73
4.13.2.14 load_background_menu()	73
4.13.2.15 load_config()	73
4.13.2.16 load_menu()	74
4.13.2.17 load_text()	74
4.13.2.18 reverseMusic()	74
4.13.3 Member Data Documentation	75
4.13.3.1 _active	75
4.13.3.2 _assets	75
4.13.3.3 _configData	75
4.13.3.4 _event	75

4.13.3.5 _menu	75
4.13.3.6 _music	76
4.13.3.7 _optionsText	76
4.13.3.8 _sound_menu	76
4.13.3.9 _string	76
4.13.3.10 _window	76
4.14 Snake Class Reference	77
4.14.1 Constructor & Destructor Documentation	78
4.14.1.1 Snake() [1/2]	78
4.14.1.2 Snake() [2/2]	79
4.14.1.3 ~Snake()	79
4.14.2 Member Function Documentation	79
4.14.2.1 collisionFood()	79
4.14.2.2 collisionMap()	79
4.14.2.3 getElements()	80
4.14.2.4 getLogin()	80
4.14.2.5 getScore()	80
4.14.2.6 getSpeed()	80
4.14.2.7 handleCollision()	81
4.14.2.8 handleInput()	81
4.14.2.9 init()	82
4.14.2.10 initTexts()	82
4.14.2.11 loadAssets()	82
4.14.2.12 loadTexts()	82
4.14.2.13 parse_map()	82
4.14.2.14 readConfig()	83
4.14.2.15 readScore()	83
4.14.2.16 update()	83
4.14.2.17 updateTexts()	83
4.14.2.18 writeScore()	84
4.14.3 Member Data Documentation	84
4.14.3.1 _configData	84
4.14.3.2 _food	84
4.14.3.3 _highScore	84
4.14.3.4 _highscore	84
4.14.3.5 _level	84
4.14.3.6 _map	85
4.14.3.7 _score	85
4.14.3.8 _skinMultiplier	85
4.14.3.9 _snake	85
4.14.3.10 _speed	85
4.14.3.11 _string	85

4.14.3.12 HighScoresSet . . . . .	85
<b>5 File Documentation . . . . .</b>	<b>87</b>
5.1 shared/Position.cpp File Reference . . . . .	87
5.2 shared/Position.hpp File Reference . . . . .	87
5.2.1 Detailed Description . . . . .	88
5.2.2 Enumeration Type Documentation . . . . .	88
5.2.2.1 direction_t . . . . .	88
5.3 src/core.cpp File Reference . . . . .	89
5.3.1 Function Documentation . . . . .	89
5.3.1.1 main() . . . . .	89
5.4 src/core.hpp File Reference . . . . .	90
5.4.1 Typedef Documentation . . . . .	91
5.4.1.1 GetIdFuncPtr . . . . .	91
5.4.1.2 keys_t . . . . .	91
5.4.1.3 LibGameDeletePtr . . . . .	91
5.4.1.4 LibGameFuncPtr . . . . .	91
5.4.1.5 LibGraphicDeletePtr . . . . .	91
5.4.1.6 LibGraphicFuncPtr . . . . .	92
5.4.1.7 state_t . . . . .	92
5.4.2 Enumeration Type Documentation . . . . .	92
5.4.2.1 keys_e . . . . .	92
5.4.2.2 state_e . . . . .	92
5.5 src_game/IGames.hpp File Reference . . . . .	93
5.6 src_game/Nibbler/Nibbler_game.cpp File Reference . . . . .	94
5.6.1 Function Documentation . . . . .	94
5.6.1.1 create() . . . . .	94
5.6.1.2 createGameInstance() . . . . .	94
5.6.1.3 destroy() . . . . .	94
5.6.1.4 getName() . . . . .	95
5.7 src_game/Nibbler/Nibbler_game.hpp File Reference . . . . .	95
5.8 src_game/shared/Cell.cpp File Reference . . . . .	96
5.9 src_game/shared/Cell.hpp File Reference . . . . .	96
5.10 src_game/shared/Food.cpp File Reference . . . . .	97
5.11 src_game/shared/Food.hpp File Reference . . . . .	98
5.12 src_game/shared/HighScore.cpp File Reference . . . . .	99
5.12.1 Function Documentation . . . . .	99
5.12.1.1 extractScore() . . . . .	99
5.13 src_game/shared/HighScore.hpp File Reference . . . . .	99
5.14 src_game/shared/Map.cpp File Reference . . . . .	100
5.15 src_game/shared/Map.hpp File Reference . . . . .	101
5.16 src_game/shared/PlayerS.cpp File Reference . . . . .	102

5.17 src_game/shared/PlayerS.hpp File Reference . . . . .	102
5.18 src_game/Snake/Snake_game.cpp File Reference . . . . .	103
5.18.1 Function Documentation . . . . .	104
5.18.1.1 create() . . . . .	104
5.18.1.2 createGameInstance() . . . . .	104
5.18.1.3 destroy() . . . . .	104
5.18.1.4 getName() . . . . .	104
5.19 src_game/Snake/Snake_game.hpp File Reference . . . . .	104
5.20 src_graphic/IGraphic.hpp File Reference . . . . .	105
5.20.1 Detailed Description . . . . .	106
5.21 src_graphic/src_ncurses/basic.cpp File Reference . . . . .	106
5.21.1 Macro Definition Documentation . . . . .	107
5.21.1.1 OFFSET_X . . . . .	107
5.21.1.2 OFFSET_Y . . . . .	107
5.21.1.3 RECT_SIZE . . . . .	108
5.21.2 Function Documentation . . . . .	108
5.21.2.1 createGraphicInstance() . . . . .	108
5.21.2.2 destroy() . . . . .	108
5.21.2.3 getName() . . . . .	108
5.22 src_graphic/src_sdl2/basic.cpp File Reference . . . . .	108
5.22.1 Function Documentation . . . . .	109
5.22.1.1 create() . . . . .	109
5.22.1.2 getName() . . . . .	109
5.22.1.3 sdl2_load_assets() . . . . .	109
5.23 src_graphic/src_sfml/basic.cpp File Reference . . . . .	109
5.23.1 Macro Definition Documentation . . . . .	110
5.23.1.1 CD_SIZE . . . . .	110
5.23.1.2 OFFSET_X . . . . .	110
5.23.1.3 OFFSET_Y . . . . .	110
5.23.1.4 RECT_SIZE . . . . .	110
5.23.2 Function Documentation . . . . .	110
5.23.2.1 createGraphicInstance() . . . . .	110
5.23.2.2 destroy() . . . . .	111
5.23.2.3 getName() . . . . .	111
5.24 src_graphic/src_ncurses/basic.hpp File Reference . . . . .	111
5.25 src_graphic/src_sdl2/basic.hpp File Reference . . . . .	112
5.26 src_graphic/src_sfml/basic.hpp File Reference . . . . .	113

# Chapter 1

## Hierarchical Index

### 1.1 Class Hierarchy

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

Cell . . . . .	7
Food . . . . .	21
core . . . . .	10
HighScore . . . . .	22
IGames . . . . .	23
Nibbler . . . . .	44
Snake . . . . .	77
IGraphic . . . . .	29
Ncurses . . . . .	36
Sdl2 . . . . .	62
Sfml . . . . .	67
Map . . . . .	35
PlayerS . . . . .	54
Position . . . . .	57



## Chapter 2

# Class Index

### 2.1 Class List

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

<a href="#">Cell</a>	7
<a href="#">core</a>	
Main Core class for the application	10
<a href="#">Food</a>	21
<a href="#">HighScore</a>	22
<a href="#">IGames</a>	
Interface for handling games	23
<a href="#">IGraphic</a>	
Interface for handling game assets	29
<a href="#">Map</a>	35
<a href="#">Ncurses</a>	36
<a href="#">Nibbler</a>	
<a href="#">Nibbler</a> class used to manage the game <a href="#">Nibbler</a>	44
<a href="#">PlayerS</a>	54
<a href="#">Position</a>	
<a href="#">Position</a> class used to store the position of an object	57
<a href="#">Sdl2</a>	62
<a href="#">Sfml</a>	67
<a href="#">Snake</a>	77





## Chapter 3

# File Index

### 3.1 File List

Here is a list of all files with brief descriptions:

shared/ <a href="#">Position.cpp</a> . . . . .	87
shared/ <a href="#">Position.hpp</a>	
<a href="#">Position</a> class declaration used to store the position of an object . . . . .	87
src/ <a href="#">core.cpp</a> . . . . .	89
src/ <a href="#">core.hpp</a> . . . . .	90
src_game/ <a href="#">IGames.hpp</a> . . . . .	93
src_game/Nibbler/ <a href="#">Nibbler_game.cpp</a> . . . . .	94
src_game/Nibbler/ <a href="#">Nibbler_game.hpp</a> . . . . .	95
src_game/shared/ <a href="#">Cell.cpp</a> . . . . .	96
src_game/shared/ <a href="#">Cell.hpp</a> . . . . .	96
src_game/shared/ <a href="#">Food.cpp</a> . . . . .	97
src_game/shared/ <a href="#">Food.hpp</a> . . . . .	98
src_game/shared/ <a href="#">HighScore.cpp</a> . . . . .	99
src_game/shared/ <a href="#">HighScore.hpp</a> . . . . .	99
src_game/shared/ <a href="#">Map.cpp</a> . . . . .	100
src_game/shared/ <a href="#">Map.hpp</a> . . . . .	101
src_game/shared/ <a href="#">PlayerS.cpp</a> . . . . .	102
src_game/shared/ <a href="#">PlayerS.hpp</a> . . . . .	102
src_game/Snake/ <a href="#">Snake_game.cpp</a> . . . . .	103
src_game/Snake/ <a href="#">Snake_game.hpp</a> . . . . .	104
src_graphic/ <a href="#">IGraphic.hpp</a>	
Header file for the <a href="#">IGraphic</a> interface . . . . .	105
src_graphic/src_ncurses/ <a href="#">basic.cpp</a> . . . . .	106
src_graphic/src_ncurses/ <a href="#">basic.hpp</a> . . . . .	111
src_graphic/src_sdl2/ <a href="#">basic.cpp</a> . . . . .	108
src_graphic/src_sdl2/ <a href="#">basic.hpp</a> . . . . .	112
src_graphic/src_sfml/ <a href="#">basic.cpp</a> . . . . .	109
src_graphic/src_sfml/ <a href="#">basic.hpp</a> . . . . .	113



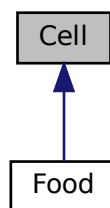
## Chapter 4

# Class Documentation

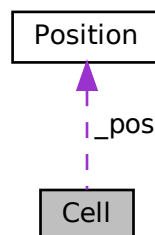
### 4.1 Cell Class Reference

```
#include <Cell.hpp>
```

Inheritance diagram for Cell:



Collaboration diagram for Cell:



## Public Member Functions

- [Cell](#) ()
- [Cell](#) (int x, int y)
- [~Cell](#) ()
- void [setPos](#) (int x, int y)
- void [setPos](#) ([Position](#) pos)
- [Position](#) [getPos](#) ()
- int [getX](#) ()
- int [getY](#) ()
- [direction\\_t](#) [getRotation](#) ()
- bool [checkCollision](#) ([Position](#) pos)
- void [setRotation](#) ([direction\\_t](#) dir)

## Protected Attributes

- [Position](#) [\\_pos](#)

### 4.1.1 Constructor & Destructor Documentation

#### 4.1.1.1 [Cell\(\)](#) [1/2]

```
Cell::Cell ( )
```

#### 4.1.1.2 [Cell\(\)](#) [2/2]

```
Cell::Cell (
    int x,
    int y )
```

#### 4.1.1.3 [~Cell\(\)](#)

```
Cell::~~Cell ( )
```

### 4.1.2 Member Function Documentation

#### 4.1.2.1 checkCollision()

```
bool Cell::checkCollision (
    Position pos )
```

#### 4.1.2.2 getPos()

```
Position Cell::getPos ( )
```

#### 4.1.2.3 getRotation()

```
direction_t Cell::getRotation ( )
```

#### 4.1.2.4 getX()

```
int Cell::getX ( )
```

#### 4.1.2.5 getY()

```
int Cell::getY ( )
```

#### 4.1.2.6 setPos() [1/2]

```
void Cell::setPos (
    int x,
    int y )
```

#### 4.1.2.7 setPos() [2/2]

```
void Cell::setPos (
    Position pos )
```

#### 4.1.2.8 setRotation()

```
void Cell::setRotation (
    direction_t dir )
```

### 4.1.3 Member Data Documentation

#### 4.1.3.1 \_pos

```
Position Cell::_pos [protected]
```

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

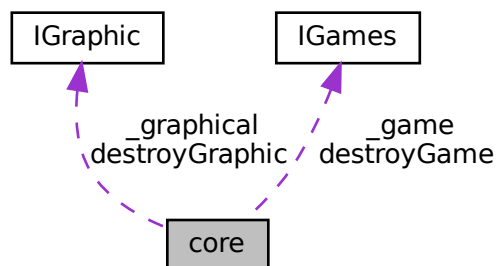
- [src\\_game/shared/Cell.hpp](#)
- [src\\_game/shared/Cell.cpp](#)

## 4.2 core Class Reference

Main Core class for the application.

```
#include <core.hpp>
```

Collaboration diagram for core:



## Public Member Functions

- `core ()`  
*Initial.*
- `core (const std::string &libGraphical)`  
*Constructor with graphical library.*
- `~core ()`  
*Default destructor.*
- `void loop ()`  
*Main loop of the application.*
- `void loopGame ()`  
*Loop for the game state.*
- `void loopMenu ()`  
*Loop for the menu state.*
- `void loadGame (const std::string &libGame)`  
*Load a game library.*
- `void loopEnd ()`  
*Loop for the end state.*
- `void loadGraphical (const std::string &libGraphical)`  
*Load a graphical library.*
- `void handleInputConfig ()`  
*Handle input for the config state.*
- `void loopHighScore ()`  
*Loop for the high score state.*
- `void initGame ()`  
*Init the game state.*
- `void initMenu ()`  
*Init the menu state.*
- `void handleInputMenu ()`  
*Handle input for the menu state.*
- `bool check_time (double time, std::chrono::time_point< std::chrono::high_resolution_clock > *clock_start, std::chrono::time_point< std::chrono::high_resolution_clock > *clock_end)`  
*Check if the time is elapsed.*
- `void writeConfig ()`  
*Write the config file.*
- `void initConfig ()`  
*Init the config state.*
- `void loopConfig ()`  
*Loop for the config state.*
- `void switchGraphics (const std::string &libGraphic)`  
*Switch the graphical library.*
- `void loadLibrariesFromDirectory (const std::string &dirPath)`  
*Load all the libraries from a directory.*
- `void handleInputGame ()`  
*Handle input for the game state.*

## Protected Attributes

- [IGames](#) \* [\\_game](#)  
*Game library pointer.*
- [IGraphic](#) \* [\\_graphical](#)  
*Graphical library pointer.*
- `int` [\\_key](#) = 0  
*int for the key pressed to be handled. we use sfml values so all the libraries graphical need to be compatible with sfml by being converted.*
- `int` [\\_optionsMenu](#) = 0  
*\_optionsMenu is used to know which menu option is selected. 0 will launch nibbler, 1 will launch snake, 2 will launch the config menu.*
- `int` [\\_menustate](#) = 0  
*\_menustate is used to know in which state of the confirmation menu we are. 0 = wait for input 1 = is loading, the launch can still be stopped 2 = is loaded so it launches the game.*
- `int` [\\_state](#) = [UNDEFINED](#)  
*\_state is used to know in which state of the application we are. 0 = undefined 1 = menu 2 = game 3 = high score 4 = config 5 = end*
- `std::vector< std::tuple< std::string, std::vector< size_t >, size_t > >` [\\_optionsConfig](#)  
*\_optionsConfig is used to know the possible values for the config menu. 0 = res \t 0 = 800x600 \t 1 = 1280x1024 \t 2 = 1920x1080 1 = skin \t 0 = default \t 1 = classic \t 2 = retro \t 3 = rainbow \t 4 = classic variant \t 5 = retro variant \t 6 = cave variant*
- `size_t` [\\_configMenuChoice](#) = 0  
*\_configMenuChoice is used to know which option is selected in the config menu. 0 = res 1 = skin 2 = quit*
- [LibGraphicDeletePtr](#) [destroyGraphic](#)  
*destroyGraphic is used to destroy the graphical library.*
- [LibGameDeletePtr](#) [destroyGame](#)  
*destroyGame is used to destroy the game library.*
- `std::vector< std::string >` [\\_gameLibraries](#)  
*\_gameLibraries is used to store the game libraries in a vector.*
- `std::vector< std::string >` [\\_graphicLibraries](#)  
*\_graphicLibraries is used to store the graphical libraries in a vector.*
- `std::string` [\\_currentGraphicLibrary](#)  
*\_currentGraphicLibrary is used to store the current graphical library.*
- `std::string` [\\_currentGameLibraryPath](#)  
*\_currentGameLibrary is used to store the current game library.*
- `std::chrono::time_point< std::chrono::high_resolution_clock >` [\\_clock\\_start](#)  
*\_clock\_start is used to store the start time with [\\_clock\\_end](#) to calculate the time elapsed. with the `_game->speed` it will be used to know if the game needs to be updated.*
- `std::chrono::time_point< std::chrono::high_resolution_clock >` [\\_clock\\_end](#)  
*\_clock\_end is used to store the end time with [\\_clock\\_start](#) to calculate the time elapsed. with the `_game->speed` it will be used to know if the game needs to be updated.*
- `std::chrono::time_point< std::chrono::high_resolution_clock >` [\\_clock\\_start\\_fixed](#)  
*\_clock\_start\_fixed is used to store the start time with [\\_clock\\_end\\_fixed](#) to calculate the time elapsed. its based with a 60fps so it will be used to know when some dynamic resource such as the text, timer etc... need to be updated.*
- `std::chrono::time_point< std::chrono::high_resolution_clock >` [\\_clock\\_end\\_fixed](#)  
*\_clock\_end\_fixed is used to store the end time with [\\_clock\\_start\\_fixed](#) to calculate the time elapsed. its based with a 60fps so it will be used to know when some dynamic resource such as the text, timer etc... need to be updated.*

### 4.2.1 Detailed Description

Main Core class for the application.



## 4.2.2 Constructor & Destructor Documentation

### 4.2.2.1 core() [1/2]

```
core::core ( )
```

Initial.

Default constructor.

### 4.2.2.2 core() [2/2]

```
core::core (
    const std::string & libGraphical )
```

Constructor with graphical library.

#### Parameters

<i>libGraphical</i>	Graphical library to load.
---------------------	----------------------------

#### Returns

core

### 4.2.2.3 ~core()

```
core::~~core ( )
```

Default destructor.

## 4.2.3 Member Function Documentation

### 4.2.3.1 check\_time()

```
bool core::check_time (
    double time,
    std::chrono::time_point< std::chrono::high_resolution_clock > * clock_start,
    std::chrono::time_point< std::chrono::high_resolution_clock > * clock_end )
```

Check if the time is elapsed.

**Parameters**

<i>time</i>	Time to check.
<i>clock_start</i>	Start time.
<i>clock_end</i>	End time.

**Returns**

bool True if the time is elapsed, false otherwise.

**4.2.3.2 handleInputConfig()**

```
void core::handleInputConfig ( )
```

Handle input for the config state.

**4.2.3.3 handleInputGame()**

```
void core::handleInputGame ( )
```

Handle input for the game state.

**4.2.3.4 handleInputMenu()**

```
void core::handleInputMenu ( )
```

Handle input for the menu state.

**4.2.3.5 initConfig()**

```
void core::initConfig ( )
```

Init the config state.

#### 4.2.3.6 initGame()

```
void core::initGame ( )
```

Init the game state.

#### 4.2.3.7 initMenu()

```
void core::initMenu ( )
```

Init the menu state.

#### 4.2.3.8 loadGame()

```
void core::loadGame (
    const std::string & libGame )
```

Load a game library.

##### Parameters

<i>libGame</i>	Game library to load.
----------------	-----------------------

#### 4.2.3.9 loadGraphical()

```
void core::loadGraphical (
    const std::string & libGraphical )
```

Load a graphical library.

##### Parameters

<i>libGraphical</i>	Graphical library to load.
---------------------	----------------------------

#### 4.2.3.10 loadLibrariesFromDirectory()

```
core::loadLibrariesFromDirectory (
    const std::string & dirPath )
```

Load all the libraries from a directory.

**Parameters**

<i>dirPath</i>	Directory to load the libraries from.
----------------	---------------------------------------

**Returns**

[\\_gameLibraries](#) and [\\_graphicLibraries](#) are filled.

**4.2.3.11 loop()**

```
void core::loop ( )
```

Main loop of the application.

**4.2.3.12 loopConfig()**

```
void core::loopConfig ( )
```

Loop for the config state.

**4.2.3.13 loopEnd()**

```
void core::loopEnd ( )
```

Loop for the end state.

**4.2.3.14 loopGame()**

```
void core::loopGame ( )
```

Loop for the game state.

**4.2.3.15 loopHighScore()**

```
void core::loopHighScore ( )
```

Loop for the high score state.

#### 4.2.3.16 loopMenu()

```
void core::loopMenu ( )
```

Loop for the menu state.

#### 4.2.3.17 switchGraphics()

```
void core::switchGraphics (
    const std::string & libGraphic )
```

Switch the graphical library.

##### Parameters

<i>libGraphic</i>	Graphical library to load.
-------------------	----------------------------

#### 4.2.3.18 writeConfig()

```
void core::writeConfig ( )
```

Write the config file.

### 4.2.4 Member Data Documentation

#### 4.2.4.1 \_clock\_end

```
std::chrono::time_point<std::chrono::high_resolution_clock> core::_clock_end [protected]
```

`_clock_end` is used to store the end time with `_clock_start` to calculate the time elapsed. with the `_game->speed` it will be used to know if the game needs to be updated.

#### 4.2.4.2 \_clock\_end\_fixed

```
std::chrono::time_point<std::chrono::high_resolution_clock> core::_clock_end_fixed [protected]
```

`_clock_end_fixed` is used to store the end time with `_clock_start_fixed` to calculate the time elapsed. its based with a 60fps so it will be used to know when some dynamic ressource such as the text, timer etc... need to be updated.

#### 4.2.4.3 `_clock_start`

```
std::chrono::time_point<std::chrono::high_resolution_clock> core::_clock_start [protected]
```

`_clock_start` is used to store the start time with `_clock_end` to calculate the time elapsed. with the `_game->speed` it will be used to know if the game needs to be updated.

#### 4.2.4.4 `_clock_start_fixed`

```
std::chrono::time_point<std::chrono::high_resolution_clock> core::_clock_start_fixed [protected]
```

`_clock_start_fixed` is used to store the start time with `_clock_end_fixed` to calculate the time elapsed. its based with a 60fps so it will be used to know when some dynamic ressource such as the text, timer etc... need to be updated.

#### 4.2.4.5 `_configMenuChoice`

```
size_t core::_configMenuChoice = 0 [protected]
```

`_configMenuChoice` is used to know which option is selected in the config menu. 0 = res 1 = skin 2 = quit

#### 4.2.4.6 `_currentGameLibraryPath`

```
std::string core::_currentGameLibraryPath [protected]
```

`_currentGameLibrary` is used to store the current game library.

#### 4.2.4.7 `_currentGraphicLibrary`

```
std::string core::_currentGraphicLibrary [protected]
```

`_currentGraphicLibrary` is used to store the current graphical library.

#### 4.2.4.8 `_game`

```
IGames* core::_game [protected]
```

Game library pointer.

#### 4.2.4.9 `_gameLibraries`

```
std::vector<std::string> core::_gameLibraries [protected]
```

`_gameLibraries` is used to store the game libraries in a vector.

#### 4.2.4.10 `_graphical`

```
IGraphic* core::_graphical [protected]
```

Graphical library pointer.

#### 4.2.4.11 `_graphicLibraries`

```
std::vector<std::string> core::_graphicLibraries [protected]
```

`_graphicLibraries` is used to store the graphical libraries in a vector.

#### 4.2.4.12 `_key`

```
int core::_key = 0 [protected]
```

int for the key pressed to be handled. we use sfml values so all the libraries graphical need to be compatible with sfml by being converted.

#### 4.2.4.13 `_menustate`

```
int core::_menustate = 0 [protected]
```

`_menustate` is used to know in which state of the confirmation menu we are. 0 = wait for input 1 = is loading, the launch can still be stopped 2 = is loaded so it launches the game.

#### 4.2.4.14 `_optionsConfig`

```
std::vector<std::tuple<std::string, std::vector<size_t>, size_t> > core::_optionsConfig  
[protected]
```

`_optionsConfig` is used to know the possible values for the config menu. 0 = res \t 0 = 800x600 \t 1 = 1280x1024 \t 2 = 1920x1080 1 = skin \t 0 = default \t 1 = classic \t 2 = retro \t 3 = rainbow \t 4 = classic variant \t 5 = retro variant \t 6 = cave variant

#### 4.2.4.15 `_optionsMenu`

```
int core::_optionsMenu = 0 [protected]
```

`_optionsMenu` is used to know which menu option is selected. 0 will launch nibbler, 1 will launch snake, 2 will launch the config menu.

#### 4.2.4.16 `_state`

```
int core::_state = UNDEFINED [protected]
```

`_state` is used to know in which state of the application we are. 0 = undefined 1 = menu 2 = game 3 = high score 4 = config 5 = end

#### 4.2.4.17 `destroyGame`

```
LibGameDeletePtr core::destroyGame [protected]
```

`destroyGame` is used to destroy the game library.

#### 4.2.4.18 `destroyGraphic`

```
LibGraphicDeletePtr core::destroyGraphic [protected]
```

`destroyGraphic` is used to destroy the graphical library.

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

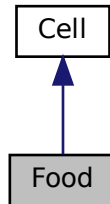
- [src/core.hpp](#)
- [src/core.cpp](#)



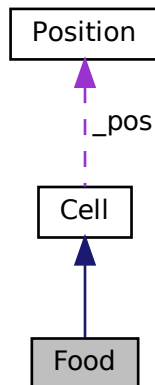
## 4.3 Food Class Reference

```
#include <Food.hpp>
```

Inheritance diagram for Food:



Collaboration diagram for Food:



### Public Member Functions

- [Food](#) ()
- [Food](#) (int x, int y)
- std::pair< int, int > [respawn](#) (std::vector< [Position](#) > mapPos, std::vector< [Position](#) > pos\_snake, std::vector< [Position](#) > possible\_pos)

### Additional Inherited Members

#### 4.3.1 Constructor & Destructor Documentation

#### 4.3.1.1 Food() [1/2]

```
Food::Food ( )
```

#### 4.3.1.2 Food() [2/2]

```
Food::Food (
    int x,
    int y )
```

### 4.3.2 Member Function Documentation

#### 4.3.2.1 respawn()

```
std::pair< int, int > Food::respawn (
    std::vector< Position > mapPos,
    std::vector< Position > pos_snake,
    std::vector< Position > possible_pos )
```

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

- src\_game/shared/[Food.hpp](#)
- src\_game/shared/[Food.cpp](#)

## 4.4 HighScore Class Reference

```
#include <HighScore.hpp>
```

### Public Member Functions

- std::vector< std::string > [writeScore](#) (std::string gameName, size\_t \_score)
- std::vector< std::string > [getTop10Reverse](#) (const std::vector< std::string > &lines, std::string gameName)
- std::string [getLogin](#) ()
- std::vector< std::string > [readScore](#) (std::string gameName)

#### 4.4.1 Member Function Documentation

#### 4.4.1.1 getLogin()

```
std::string HighScore::getLogin ( )
```

#### 4.4.1.2 getTop10Reverse()

```
std::vector< std::string > HighScore::getTop10Reverse (
    const std::vector< std::string > & lines,
    std::string gameName )
```

#### 4.4.1.3 readScore()

```
std::vector< std::string > HighScore::readScore (
    std::string gameName )
```

#### 4.4.1.4 writeScore()

```
std::vector< std::string > HighScore::writeScore (
    std::string gameName,
    size_t _score )
```

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

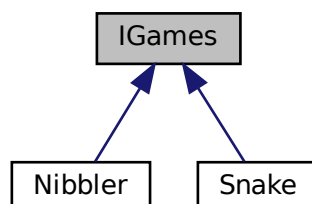
- [src\\_game/shared/HighScore.hpp](#)
- [src\\_game/shared/HighScore.cpp](#)

## 4.5 IGames Class Reference

Interface for handling games.

```
#include <IGames.hpp>
```

Inheritance diagram for IGames:



## Public Member Functions

- virtual `~IGames()`=default
- virtual void `init()`=0  
*Initialize the game.*
- virtual bool `update` (int key)=0  
*Update the game based on the key pressed.*
- virtual `std::vector< std::tuple< char, std::string, int > >` `loadAssets()`=0
- virtual `std::vector< std::tuple< char, Position > >` `getElements()`=0
- virtual `std::tuple< std::vector< Position >, std::vector< Position > >` `parse_map` (std::string map\_name)=0
- virtual double `getScore()`=0  
*Get the score of the game.*
- virtual `size_t` `getSpeed()`=0  
*Get the speed of the game.*
- virtual void `handleCollision()`=0  
*Handle the collision of the game.*
- virtual void `handleInput` (int key)=0  
*Handle the input of the game.*
- virtual bool `collisionMap` (Position pos)=0  
*Check if the position is a wall.*
- virtual void `collisionFood()`=0  
*Check if the position is a food.*
- virtual `std::vector< std::tuple< std::string, Position > >` `loadTexts()`=0
- virtual void `updateTexts()`=0  
*Update the texts of the game.*
- virtual `std::vector< std::string >` `writeScore()`=0  
*Write the score of the game.*
- virtual `std::vector< std::string >` `readScore()`=0  
*Read the score of the game.*
- virtual void `readConfig()`=0

### 4.5.1 Detailed Description

Interface for handling games.

This class is an interface for handling games in the arcade program. such as the snake, the food, the map, the score, the speed, the collision, the input, the texts, the score, the highscore, the time

### 4.5.2 Constructor & Destructor Documentation

#### 4.5.2.1 ~IGames()

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

### 4.5.3 Member Function Documentation

#### 4.5.3.1 collisionFood()

```
bool IGames::collisionFood ( ) [pure virtual]
```

Check if the position is a food.

**Parameters**

<i>pos</i>	The position to check
------------	-----------------------

**Returns**

true if the position is a food, false if not

Implemented in [Snake](#), and [Nibbler](#).

**4.5.3.2 collisionMap()**

```
bool IGames::collisionMap (
    Position pos ) [pure virtual]
```

Check if the position is a wall.

**Parameters**

<i>pos</i>	The position to check
------------	-----------------------

**Returns**

true if the position is a wall, false if not

Implemented in [Snake](#), and [Nibbler](#).

**4.5.3.3 getElements()**

```
virtual std::vector<std::tuple<char, Position> > IGames::getElements ( ) [pure virtual]
```

Implemented in [Snake](#), and [Nibbler](#).

**4.5.3.4 getScore()**

```
double IGames::getScore ( ) [pure virtual]
```

Get the score of the game.

**Returns**

A double containing the score

Implemented in [Snake](#), and [Nibbler](#).

#### 4.5.3.5 getSpeed()

```
size_t IGames::getSpeed ( ) [pure virtual]
```

Get the speed of the game.

##### Returns

A `size_t` containing the speed

Implemented in [Snake](#), and [Nibbler](#).

#### 4.5.3.6 handleCollision()

```
void IGames::handleCollision ( ) [pure virtual]
```

Handle the collision of the game.

Implemented in [Snake](#), and [Nibbler](#).

#### 4.5.3.7 handleInput()

```
void IGames::handleInput (
    int key ) [pure virtual]
```

Handle the input of the game.

##### Parameters

<i>key</i>	The key pressed
------------	-----------------

Implemented in [Snake](#), and [Nibbler](#).

#### 4.5.3.8 init()

```
void IGames::init ( ) [pure virtual]
```

Initialize the game.

This function is used to initialize the game

Implemented in [Snake](#), and [Nibbler](#).

#### 4.5.3.9 loadAssets()

```
virtual std::vector<std::tuple<char, std::string, int> > IGames::loadAssets ( ) [pure virtual]
```

Implemented in [Snake](#), and [Nibbler](#).

#### 4.5.3.10 loadTexts()

```
virtual std::vector<std::tuple<std::string, Position> > IGames::loadTexts ( ) [pure virtual]
```

Implemented in [Snake](#), and [Nibbler](#).

#### 4.5.3.11 parse\_map()

```
virtual std::tuple<std::vector<Position>, std::vector<Position> > IGames::parse_map (
    std::string map_name ) [pure virtual]
```

Implemented in [Snake](#), and [Nibbler](#).

#### 4.5.3.12 readConfig()

```
virtual void IGames::readConfig ( ) [pure virtual]
```

Implemented in [Snake](#), and [Nibbler](#).

#### 4.5.3.13 readScore()

```
std::vector< std::string > IGames::readScore ( ) [pure virtual]
```

Read the score of the game.

##### Returns

A vector of string containing the score

Implemented in [Snake](#), and [Nibbler](#).

#### 4.5.3.14 update()

```
bool IGames::update (
    int key ) [pure virtual]
```

Update the game based on the key pressed.



**Parameters**

<i>key</i>	The key pressed for handling the input later
------------	--

**Returns**

true if the game is still running, false if the game is over

Implemented in [Snake](#), and [Nibbler](#).

**4.5.3.15 updateTexts()**

```
void IGames::updateTexts ( ) [pure virtual]
```

Update the texts of the game.

Implemented in [Snake](#), and [Nibbler](#).

**4.5.3.16 writeScore()**

```
std::vector< std::string > IGames::writeScore ( ) [pure virtual]
```

Write the score of the game.

**Returns**

A vector of string containing the score

Implemented in [Snake](#), and [Nibbler](#).

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

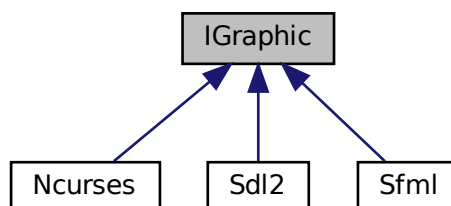
- [src\\_game/IGames.hpp](#)

## 4.6 IGraphic Class Reference

Interface for handling game assets.

```
#include <IGraphic.hpp>
```

Inheritance diagram for IGraphic:



## Public Member Functions

- virtual `~IGraphic()`=default
- virtual void `display_text` (std::vector< std::tuple< std::string, [Position](#) >> text\_to\_display)=0  
*Display text objects.*
- virtual int `get_index_tuple` (char name)=0  
*Get the index of a tuple by identifier character.*
- virtual int `get_key` ()=0  
*Get the pressed key code.*
- virtual void `deleteAssets` ()=0  
*Delete all loaded assets.*
- virtual void `load_text` (std::vector< std::tuple< std::string, [Position](#) >> text\_toload)=0  
*Load text objects.*
- virtual int `display_menu` (int choice, int isRunning)=0  
*Display the menu.*
- virtual void `load_menu` (std::vector< std::string > &menu\_toload)=0  
*Load the menu.*
- virtual void `display_high_score` (std::vector< std::string > high\_score)=0  
*Display the high scores.*
- virtual void `displayConfig` (size\_t \_menuChoice, std::vector< std::tuple< std::string, std::vector< size\_t >, size\_t >> \_options)=0  
*Display the configuration menu.*
- virtual void `load_config` (std::vector< std::tuple< std::string, std::vector< size\_t >, size\_t >> \_options)=0  
*Load the configuration menu.*
- virtual void `debug_print` ()=0  
*Debug function to print the loaded assets.*
- virtual void `load_assets` (std::vector< std::tuple< char, std::string, int >> asset\_toload)=0  
*Load assets from a vector of tuples.*
- virtual void `display_asset` (std::vector< std::tuple< char, [Position](#) >> object\_to\_display)=0  
*Display assets from a vector of tuples.*

### 4.6.1 Detailed Description

Interface for handling game assets.

[IGraphic](#) provides an interface for loading, displaying, and managing game assets such as images, text, and menus.

### 4.6.2 Constructor & Destructor Documentation

#### 4.6.2.1 ~IGraphic()

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

### 4.6.3 Member Function Documentation

#### 4.6.3.1 debug\_print()

```
void IGraphic::debug_print ( ) [pure virtual]
```

Debug function to print the loaded assets.

Implemented in [Sfml](#), [Sdl2](#), and [Ncurses](#).

#### 4.6.3.2 deleteAssets()

```
virtual void IGraphic::deleteAssets ( ) [pure virtual]
```

Delete all loaded assets.

Implemented in [Sfml](#), and [Ncurses](#).

#### 4.6.3.3 display\_asset()

```
void IGraphic::display_asset (
    std::vector< std::tuple< char, Position >> object_to_display ) [pure virtual]
```

Display assets from a vector of tuples.

##### Parameters

<i>object_to_display</i>	Vector of tuples containing (identifier character, position)
--------------------------	--

Implemented in [Sfml](#), [Sdl2](#), and [Ncurses](#).

#### 4.6.3.4 display\_high\_score()

```
virtual void IGraphic::display_high_score (
    std::vector< std::string > high_score ) [pure virtual]
```

Display the high scores.

##### Parameters

<i>high_score</i>	Vector of high scores as strings
-------------------	----------------------------------

Implemented in [Sfml](#), and [Ncurses](#).

#### 4.6.3.5 display\_menu()

```
virtual int IGraphic::display_menu (
    int choice,
    int isRunning ) [pure virtual]
```

Display the menu.

##### Parameters

<i>choice</i>	Menu choice index
<i>isRunning</i>	Menu running state

##### Returns

Menu status

Implemented in [Sfml](#), and [Ncurses](#).

#### 4.6.3.6 display\_text()

```
virtual void IGraphic::display_text (
    std::vector< std::tuple< std::string, Position >> text_to_display ) [pure virtual]
```

Display text objects.

##### Parameters

<i>text_to_display</i>	Vector of tuples containing (text string, position)
------------------------	---

Implemented in [Sfml](#), [Sdl2](#), and [Ncurses](#).

#### 4.6.3.7 displayConfig()

```
virtual void IGraphic::displayConfig (
    size_t _menuChoice,
    std::vector< std::tuple< std::string, std::vector< size_t >, size_t >> _options
) [pure virtual]
```

Display the configuration menu.

##### Parameters

<i>_menuChoice</i>	Selected menu choice
<i>_options</i>	Vector of tuples containing (option name, option values, selected value)

Implemented in [Sfml](#), and [Ncurses](#).

#### 4.6.3.8 get\_index\_tuple()

```
virtual int IGraphic::get_index_tuple (
    char name ) [pure virtual]
```

Get the index of a tuple by identifier character.

##### Parameters

<i>name</i>	Identifier character of the tuple
-------------	-----------------------------------

##### Returns

Index of the tuple

Implemented in [Sfml](#), [Sdl2](#), and [Ncurses](#).

#### 4.6.3.9 get\_key()

```
virtual int IGraphic::get_key ( ) [pure virtual]
```

Get the pressed key code.

##### Returns

Key code of the pressed key

Implemented in [Sfml](#), [Sdl2](#), and [Ncurses](#).

#### 4.6.3.10 load\_assets()

```
void IGraphic::load_assets (
    std::vector< std::tuple< char, std::string, int >> asset_toload ) [pure virtual]
```

Load assets from a vector of tuples.

##### Parameters

<i>asset_toload</i>	Vector of tuples containing (identifier character, asset path, asset type)
---------------------	--

Implemented in [Sfml](#), [Sdl2](#), and [Ncurses](#).

#### 4.6.3.11 load\_config()

```
virtual void IGraphic::load_config (
    std::vector< std::tuple< std::string, std::vector< size_t >, size_t >> _options
) [pure virtual]
```

Load the configuration menu.

##### Parameters

<i>_options</i>	Vector of tuples containing (option name, option values, selected value)
-----------------	--

Implemented in [Sfml](#), and [Ncurses](#).

#### 4.6.3.12 load\_menu()

```
virtual void IGraphic::load_menu (
    std::vector< std::string > & menu_toload ) [pure virtual]
```

Load the menu.

##### Parameters

<i>menu_toload</i>	Vector of menu items as strings
--------------------	---------------------------------

Implemented in [Sfml](#), [Sdl2](#), and [Ncurses](#).

#### 4.6.3.13 load\_text()

```
virtual void IGraphic::load_text (
    std::vector< std::tuple< std::string, Position >> text_toload ) [pure virtual]
```

Load text objects.

##### Parameters

<i>text_toload</i>	Vector of tuples containing (text string, position)
--------------------	---

Implemented in [Sfml](#), [Sdl2](#), and [Ncurses](#).

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

- [src\\_graphic/IGraphic.hpp](#)

## 4.7 Map Class Reference

```
#include <Map.hpp>
```

### Public Member Functions

- [Map](#) ()
- [Map](#) (std::vector< [Position](#) > &positions)
- [~Map](#) ()
- std::vector< [Cell](#) > [getMap](#) ()
- [Position](#) [getPosition](#) (int index)
- std::vector< [Position](#) > [getPositionsVector](#) ()

### Protected Attributes

- std::vector< [Cell](#) > [\\_map](#)

### 4.7.1 Constructor & Destructor Documentation

#### 4.7.1.1 Map() [1/2]

```
Map::Map ( )
```

#### 4.7.1.2 Map() [2/2]

```
Map::Map (
    std::vector< Position > & positions )
```

#### 4.7.1.3 ~Map()

```
Map::~Map ( )
```

### 4.7.2 Member Function Documentation

#### 4.7.2.1 getMap()

```
std::vector< Cell > Map::getMap ( )
```

#### 4.7.2.2 getPosition()

```
Position Map::getPosition (
    int index )
```

#### 4.7.2.3 getPositionsVector()

```
std::vector< Position > Map::getPositionsVector ( )
```

### 4.7.3 Member Data Documentation

#### 4.7.3.1 \_map

```
std::vector<Cell> Map::_map [protected]
```

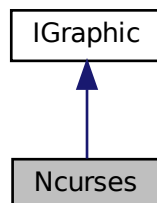
The documentation for this class was generated from the following files:

- [src\\_game/shared/Map.hpp](#)
- [src\\_game/shared/Map.cpp](#)

## 4.8 Ncurses Class Reference

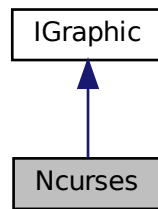
```
#include <basic.hpp>
```

Inheritance diagram for Ncurses:





Collaboration diagram for Ncurses:



## Public Member Functions

- `Ncurses` ()
- `~Ncurses` ()
- virtual void `init` ()
- virtual void `load_assets` (std::vector< std::tuple< char, std::string, int >> asset\_toload)  
*Load assets from a vector of tuples.*
- virtual void `display_asset` (std::vector< std::tuple< char, `Position` >> object\_to\_display)  
*Display assets from a vector of tuples.*
- virtual void `debug_print` ()  
*Debug function to print the loaded assets.*
- virtual int `get_index_tuple` (char name)  
*Get the index of a tuple by identifier character.*
- virtual int `get_key` ()  
*Get the pressed key code.*
- virtual void `display_text` (std::vector< std::tuple< std::string, `Position` >> text\_to\_display)  
*Display text objects.*
- virtual void `load_text` (std::vector< std::tuple< std::string, `Position` >> text\_toload)  
*Load text objects.*
- virtual void `deleteAssets` ()  
*Delete all loaded assets.*
- virtual int `display_menu` (int choice, int isRunning)  
*Display the menu.*
- virtual void `load_menu` (std::vector< std::string > &menu\_toload)  
*Load the menu.*
- virtual void `display_high_score` (std::vector< std::string > high\_score)  
*Display the high scores.*
- void `display` (int y, int x, std::string str)
- void `display` (int y, int x, char c)
- void `display` (`Position` pos, std::string str)
- void `display` (`Position` pos, char c)
- int `get_screen_y` ()
- int `get_screen_x` ()
- void `window_refresh` ()
- void `window_clear` ()
- void `display_separation_x` (int nb\_separation)

- void [display\\_separation\\_y](#) (int nb\_separation)
- virtual void [displayConfig](#) (size\_t \_menuChoice, std::vector< std::tuple< std::string, std::vector< size\_t >, size\_t >> \_options)  
*Display the configuration menu.*
- virtual void [load\\_config](#) (std::vector< std::tuple< std::string, std::vector< size\_t >, size\_t >> \_options)  
*Load the configuration menu.*

## 4.8.1 Constructor & Destructor Documentation

### 4.8.1.1 Ncurses()

```
Ncurses::Ncurses ( )
```

### 4.8.1.2 ~Ncurses()

```
Ncurses::~~Ncurses ( )
```

## 4.8.2 Member Function Documentation

### 4.8.2.1 debug\_print()

```
void Ncurses::debug_print ( ) [virtual]
```

Debug function to print the loaded assets.

Implements [IGraphic](#).

### 4.8.2.2 deleteAssets()

```
void Ncurses::deleteAssets ( ) [virtual]
```

Delete all loaded assets.

Implements [IGraphic](#).

#### 4.8.2.3 display() [1/4]

```
void Ncurses::display (
    int y,
    int x,
    char c )
```

#### 4.8.2.4 display() [2/4]

```
void Ncurses::display (
    int y,
    int x,
    std::string str )
```

#### 4.8.2.5 display() [3/4]

```
void Ncurses::display (
    Position pos,
    char c )
```

#### 4.8.2.6 display() [4/4]

```
void Ncurses::display (
    Position pos,
    std::string str )
```

#### 4.8.2.7 display\_asset()

```
void Ncurses::display_asset (
    std::vector< std::tuple< char, Position >> object_to_display ) [virtual]
```

Display assets from a vector of tuples.

##### Parameters

<i>object_to_display</i>	Vector of tuples containing (identifier character, position)
--------------------------	--

Implements [IGraphic](#).

#### 4.8.2.8 display\_high\_score()

```
void Ncurses::display_high_score (
    std::vector< std::string > high_score ) [virtual]
```

Display the high scores.

##### Parameters

<i>high_score</i>	Vector of high scores as strings
-------------------	----------------------------------

Implements [IGraphic](#).

#### 4.8.2.9 display\_menu()

```
int Ncurses::display_menu (
    int choice,
    int isRunning ) [virtual]
```

Display the menu.

##### Parameters

<i>choice</i>	Menu choice index
<i>isRunning</i>	Menu running state

##### Returns

Menu status

Implements [IGraphic](#).

#### 4.8.2.10 display\_separation\_x()

```
void Ncurses::display_separation_x (
    int nb_separation )
```

#### 4.8.2.11 display\_separation\_y()

```
void Ncurses::display_separation_y (
    int nb_separation )
```

**4.8.2.12 display\_text()**

```
void Ncurses::display_text (
    std::vector< std::tuple< std::string, Position >> text_to_display ) [virtual]
```

Display text objects.

**Parameters**

<i>text_to_display</i>	Vector of tuples containing (text string, position)
------------------------	---

Implements [IGraphic](#).

**4.8.2.13 displayConfig()**

```
void Ncurses::displayConfig (
    size_t _menuChoice,
    std::vector< std::tuple< std::string, std::vector< size_t >, size_t >> _options
) [virtual]
```

Display the configuration menu.

**Parameters**

<i>_menuChoice</i>	Selected menu choice
<i>_options</i>	Vector of tuples containing (option name, option values, selected value)

Implements [IGraphic](#).

**4.8.2.14 get\_index\_tuple()**

```
int Ncurses::get_index_tuple (
    char name ) [virtual]
```

Get the index of a tuple by identifier character.

**Parameters**

<i>name</i>	Identifier character of the tuple
-------------	-----------------------------------

**Returns**

Index of the tuple

Implements [IGraphic](#).

#### 4.8.2.15 get\_key()

```
int Ncurses::get_key ( ) [virtual]
```

Get the pressed key code.

##### Returns

Key code of the pressed key

Implements [IGraphic](#).

#### 4.8.2.16 get\_screen\_x()

```
int Ncurses::get_screen_x ( )
```

#### 4.8.2.17 get\_screen\_y()

```
int Ncurses::get_screen_y ( )
```

#### 4.8.2.18 init()

```
void Ncurses::init ( ) [virtual]
```

#### 4.8.2.19 load\_assets()

```
void Ncurses::load_assets (
    std::vector< std::tuple< char, std::string, int >> asset_toload ) [virtual]
```

Load assets from a vector of tuples.

##### Parameters

<i>asset_toload</i>	Vector of tuples containing (identifier character, asset path, asset type)
---------------------	--

Implements [IGraphic](#).

#### 4.8.2.20 load\_config()

```
void Ncurses::load_config (
    std::vector< std::tuple< std::string, std::vector< size_t >, size_t >> _options
) [virtual]
```

Load the configuration menu.

##### Parameters

<i>_options</i>	Vector of tuples containing (option name, option values, selected value)
-----------------	--

Implements [IGraphic](#).

#### 4.8.2.21 load\_menu()

```
void Ncurses::load_menu (
    std::vector< std::string > & menu_toload ) [virtual]
```

Load the menu.

##### Parameters

<i>menu_toload</i>	Vector of menu items as strings
--------------------	---------------------------------

Implements [IGraphic](#).

#### 4.8.2.22 load\_text()

```
void Ncurses::load_text (
    std::vector< std::tuple< std::string, Position >> text_toload ) [virtual]
```

Load text objects.

##### Parameters

<i>text_toload</i>	Vector of tuples containing (text string, position)
--------------------	---

Implements [IGraphic](#).

#### 4.8.2.23 window\_clear()

```
void Ncurses::window_clear ( )
```

#### 4.8.2.24 window\_refresh()

```
void Ncurses::window_refresh ( )
```

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

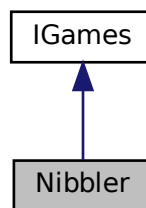
- [src\\_graphic/src\\_ncurses/basic.hpp](#)
- [src\\_graphic/src\\_ncurses/basic.cpp](#)

## 4.9 Nibbler Class Reference

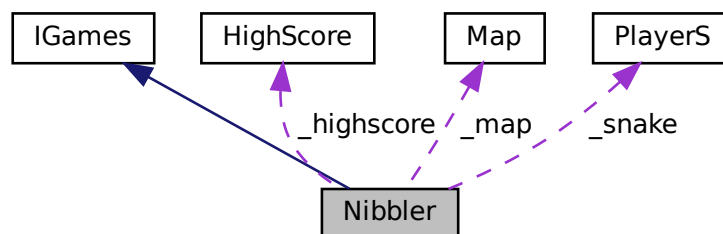
[Nibbler](#) class used to manage the game [Nibbler](#).

```
#include <Nibbler_game.hpp>
```

Inheritance diagram for Nibbler:



Collaboration diagram for Nibbler:





## Public Member Functions

- [Nibbler](#) ()  
*Construct.*
- [Nibbler](#) (int x, int y)  
*Construct.*
- [~Nibbler](#) ()  
*Destroy.*
- virtual void [init](#) ()  
*Init the game [Nibbler](#) (load the map, the snake, the food, the texts)*
- virtual void [initTexts](#) ()  
*Init the texts of the game [Nibbler](#) (score, highscore, level) [\\_string](#).*
- virtual bool [update](#) (int key)  
*Update the game [Nibbler](#) (update the snake, the food, the texts) based on the key pressed.*
- virtual std::vector< std::tuple< char, std::string, int > > [loadAssets](#) ()
- virtual std::vector< std::tuple< char, [Position](#) > > [getElements](#) ()
- virtual std::tuple< std::vector< [Position](#) >, std::vector< [Position](#) > > [parse\\_map](#) (std::string map\_name)
- virtual double [getScore](#) ()  
*Get the score of the game.*
- virtual size\_t [getSpeed](#) ()  
*Get the speed of the game.*
- virtual void [handleCollision](#) ()  
*Handle the collision of the game.*
- virtual void [handleInput](#) (int key)  
*Handle the input of the game.*
- virtual bool [collisionMap](#) ([Position](#) pos)  
*Check if the position is a wall.*
- virtual void [collisionFood](#) ()  
*Check if the position is a food.*
- virtual std::vector< std::tuple< std::string, [Position](#) > > [loadTexts](#) ()
- virtual void [updateTexts](#) ()  
*Update the texts of the game.*
- virtual std::vector< std::string > [writeScore](#) ()  
*Write the score of the game.*
- virtual std::vector< std::string > [readScore](#) ()  
*Read the score of the game.*
- virtual void [readConfig](#) ()
- double [getRemainingTime](#) ()  
*Get the remaining time of the game [Nibbler](#).*
- bool [TimeOut](#) ()  
*Check if the time is over or not [getRemainingTime\(\)](#)*
- int [getIndexString](#) (std::string)  
*Get the index of the string in the vector [\\_string](#).*
- void [addTime](#) (int time)  
*Add time to the game [Nibbler](#) (used to increase the time when the player eats a food)*
- std::string [getLogin](#) ()  
*Get the login of the player.*
- void [autoMove](#) ([Position](#) &pos)  
*Move the snake automatically if the player doesn't press any key it will check if the snake can move to the left, right, up or down.*

## Protected Attributes

- bool [HighScoreIsSet](#) = false
- size\_t [\\_skinMultiplier](#) = 0
- size\_t [\\_speed](#) = 250
- size\_t [\\_score](#) = 0
- size\_t [\\_highScore](#) = 0
- size\_t [\\_level](#) = 1
- std::vector< [Food](#) \* > [\\_food](#)
- Map \* [\\_map](#)
- [PlayerS](#) \* [\\_snake](#)
- std::vector< std::tuple< std::string, [Position](#) > > [\\_string](#)
- std::chrono::time\_point< std::chrono::high\_resolution\_clock > [\\_clock\\_start](#)
- std::chrono::time\_point< std::chrono::high\_resolution\_clock > [\\_clock\\_end](#)
- [HighScore](#) \* [\\_highscore](#)
- std::vector< std::pair< std::string, std::string > > [\\_configData](#)

### 4.9.1 Detailed Description

[Nibbler](#) class used to manage the game [Nibbler](#).

### 4.9.2 Constructor & Destructor Documentation

#### 4.9.2.1 [Nibbler\(\)](#) [1/2]

```
Nibbler::Nibbler ( )
```

Construct.

#### 4.9.2.2 [Nibbler\(\)](#) [2/2]

```
Nibbler::Nibbler (
    int x,
    int y )
```

Construct.

#### Parameters

<i>x</i>	X position
<i>y</i>	Y position

#### 4.9.2.3 ~Nibbler()

```
Nibbler::~Nibbler ( )
```

Destroy.

### 4.9.3 Member Function Documentation

#### 4.9.3.1 addTime()

```
void Nibbler::addTime (
    int time )
```

Add time to the game [Nibbler](#) (used to increase the time when the player eats a food)

##### Parameters

<i>time</i>	The time to add
-------------	-----------------

#### 4.9.3.2 autoMove()

```
void Nibbler::autoMove (
    Position & pos )
```

Move the snake automatically if the player doesn't press any key it will check if the snake can move to the left, right, up or down.

void [autoMove](#)([Position](#) &pos)

#### 4.9.3.3 collisionFood()

```
void Nibbler::collisionFood ( ) [virtual]
```

Check if the position is a food.

##### Parameters

<i>pos</i>	The position to check
------------	-----------------------

##### Returns

true if the position is a food, false if not

Implements [IGames](#).

#### 4.9.3.4 collisionMap()

```
bool Nibbler::collisionMap (
    Position pos ) [virtual]
```

Check if the position is a wall.

##### Parameters

<i>pos</i>	The position to check
------------	-----------------------

##### Returns

true if the position is a wall, false if not

Implements [IGames](#).

#### 4.9.3.5 getElements()

```
std::vector< std::tuple< char, Position > > Nibbler::getElements ( ) [virtual]
```

Implements [IGames](#).

#### 4.9.3.6 getIndexString()

```
int Nibbler::getIndexString (
    std::string str )
```

Get the index of the string in the vector [\\_string](#).

##### Parameters

<i>str</i>	The string to find
------------	--------------------

##### Returns

The index of the string in the vector

#### 4.9.3.7 getLogin()

```
std::string Nibbler::getLogin ( )
```

Get the login of the player.

##### Returns

The login of the player (used to save the highscore) [\\_highscore](#) it will use the name of the user session

#### 4.9.3.8 getRemainingTime()

```
double Nibbler::getRemainingTime ( )
```

Get the remaining time of the game [Nibbler](#).

##### Returns

The remaining time

#### 4.9.3.9 getScore()

```
double Nibbler::getScore ( ) [virtual]
```

Get the score of the game.

##### Returns

A double containing the score

Implements [IGames](#).

#### 4.9.3.10 getSpeed()

```
size_t Nibbler::getSpeed ( ) [virtual]
```

Get the speed of the game.

##### Returns

A `size_t` containing the speed

Implements [IGames](#).

#### 4.9.3.11 handleCollision()

```
void Nibbler::handleCollision ( ) [virtual]
```

Handle the collision of the game.

Implements [IGames](#).

#### 4.9.3.12 handleInput()

```
void Nibbler::handleInput (
    int key ) [virtual]
```

Handle the input of the game.

##### Parameters

<i>key</i>	The key pressed
------------	-----------------

Implements [IGames](#).

#### 4.9.3.13 init()

```
void Nibbler::init ( ) [virtual]
```

Init the game [Nibbler](#) (load the map, the snake, the food, the texts)

Implements [IGames](#).

#### 4.9.3.14 initTexts()

```
void Nibbler::initTexts ( ) [virtual]
```

Init the texts of the game [Nibbler](#) (score, highscore, level) [\\_string](#).

#### 4.9.3.15 loadAssets()

```
std::vector< std::tuple< char, std::string, int > > Nibbler::loadAssets ( ) [virtual]
```

Implements [IGames](#).

#### 4.9.3.16 loadTexts()

```
std::vector< std::tuple< std::string, Position > > Nibbler::loadTexts ( ) [virtual]
```

Implements [IGames](#).

#### 4.9.3.17 parse\_map()

```
std::tuple< std::vector< Position >, std::vector< Position > > Nibbler::parse_map (
    std::string map_name ) [virtual]
```

Implements [IGames](#).

#### 4.9.3.18 readConfig()

```
void Nibbler::readConfig ( ) [virtual]
```

Implements [IGames](#).

#### 4.9.3.19 readScore()

```
std::vector< std::string > Nibbler::readScore ( ) [virtual]
```

Read the score of the game.

##### Returns

A vector of string containing the score

Implements [IGames](#).

#### 4.9.3.20 TimeOut()

```
bool Nibbler::TimeOut ( )
```

Check if the time is over or not [getRemainingTime\(\)](#)

##### Returns

True if the time is over, false otherwise

#### 4.9.3.21 update()

```
bool Nibbler::update (
    int key ) [virtual]
```

Update the game [Nibbler](#) (update the snake, the food, the texts) based on the key pressed.

Implements [IGames](#).

#### 4.9.3.22 updateTexts()

```
void Nibbler::updateTexts ( ) [virtual]
```

Update the texts of the game.

Implements [IGames](#).

#### 4.9.3.23 writeScore()

```
std::vector< std::string > Nibbler::writeScore ( ) [virtual]
```

Write the score of the game.

##### Returns

A vector of string containing the score

Implements [IGames](#).

### 4.9.4 Member Data Documentation

#### 4.9.4.1 \_clock\_end

```
std::chrono::time_point<std::chrono::high_resolution_clock> Nibbler::_clock_end [protected]
```

#### 4.9.4.2 \_clock\_start

```
std::chrono::time_point<std::chrono::high_resolution_clock> Nibbler::_clock_start [protected]
```



#### 4.9.4.3 `_configData`

```
std::vector<std::pair<std::string, std::string> > Nibbler::_configData [protected]
```

#### 4.9.4.4 `_food`

```
std::vector<Food*> Nibbler::_food [protected]
```

#### 4.9.4.5 `_highScore`

```
size_t Nibbler::_highScore = 0 [protected]
```

#### 4.9.4.6 `_highscore`

```
HighScore* Nibbler::_highscore [protected]
```

#### 4.9.4.7 `_level`

```
size_t Nibbler::_level = 1 [protected]
```

#### 4.9.4.8 `_map`

```
Map* Nibbler::_map [protected]
```

#### 4.9.4.9 `_score`

```
size_t Nibbler::_score = 0 [protected]
```

#### 4.9.4.10 `_skinMultiplier`

```
size_t Nibbler::_skinMultiplier = 0 [protected]
```

#### 4.9.4.11 `_snake`

```
PlayerS* Nibbler::_snake [protected]
```

#### 4.9.4.12 `_speed`

```
size_t Nibbler::_speed = 250 [protected]
```

#### 4.9.4.13 `_string`

```
std::vector<std::tuple<std::string, Position> > Nibbler::_string [protected]
```

#### 4.9.4.14 `HighScoreIsSet`

```
bool Nibbler::HighScoreIsSet = false [protected]
```

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

- [src\\_game/Nibbler/Nibbler\\_game.hpp](#)
- [src\\_game/Nibbler/Nibbler\\_game.cpp](#)

## 4.10 `PlayerS` Class Reference

```
#include <PlayerS.hpp>
```

### Public Member Functions

- [PlayerS](#) ()
- [PlayerS](#) (int x, int y)
- [~PlayerS](#) ()
- void [move](#) ([Position](#) pos)
- [Position](#) [getHeadPos](#) ()
- [Position](#) [getBodyPos](#) (int i)
- [Position](#) [getTailPos](#) ()
- [size\\_t](#) [getSize](#) ()
- void [grow](#) ()
- void [setRotation](#) ([direction\\_t](#) dir)
- [direction\\_t](#) [getRotation](#) ()
- bool [checkCollision](#) ([Position](#) pos)
- bool [checkEatHimself](#) ()
- [std::vector](#)< [Position](#) > [getSnakePosVector](#) ()
- [std::vector](#)< [Position](#) > [getSnakeTailPosVector](#) ()

## 4.10.1 Constructor & Destructor Documentation

### 4.10.1.1 PlayerS() [1/2]

```
PlayerS::PlayerS ( )
```

### 4.10.1.2 PlayerS() [2/2]

```
PlayerS::PlayerS (
    int x,
    int y )
```

### 4.10.1.3 ~PlayerS()

```
PlayerS::~~PlayerS ( )
```

## 4.10.2 Member Function Documentation

### 4.10.2.1 checkCollision()

```
bool PlayerS::checkCollision (
    Position pos )
```

### 4.10.2.2 checkEatHimself()

```
bool PlayerS::checkEatHimself ( )
```

### 4.10.2.3 getBodyPos()

```
Position PlayerS::getBodyPos (
    int i )
```

#### 4.10.2.4 getHeadPos()

```
Position PlayerS::getHeadPos ( )
```

#### 4.10.2.5 getRotation()

```
direction_t PlayerS::getRotation ( )
```

#### 4.10.2.6 getSize()

```
size_t PlayerS::getSize ( )
```

#### 4.10.2.7 getSnakePosVector()

```
std::vector< Position > PlayerS::getSnakePosVector ( )
```

#### 4.10.2.8 getSnakeTailPosVector()

```
std::vector< Position > PlayerS::getSnakeTailPosVector ( )
```

#### 4.10.2.9 getTailPos()

```
Position PlayerS::getTailPos ( )
```

#### 4.10.2.10 grow()

```
void PlayerS::grow ( )
```

#### 4.10.2.11 move()

```
void PlayerS::move (
    Position pos )
```

#### 4.10.2.12 setRotation()

```
void PlayerS::setRotation (
    direction_t dir )
```

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

- src\_game/shared/PlayerS.hpp
- src\_game/shared/PlayerS.cpp

## 4.11 Position Class Reference

[Position](#) class used to store the position of an object.

```
#include <Position.hpp>
```

### Public Member Functions

- [Position](#) ()  
*Construct.*
- [Position](#) (int x, int y)  
*Construct.*
- [Position](#) (int x, int y, [direction\\_t](#) dir)  
*Construct.*
- [~Position](#) ()  
*Destroy.*
- [std::tuple< int, int > getPos](#) ()  
*Get the position.*
- [std::pair< int, int > getPosPair](#) ()  
*Get the position.*
- void [setX](#) (int x)  
*Set the X position.*
- void [setY](#) (int y)  
*Set the Y position.*
- int [getX](#) ()
- int [getY](#) ()  
*Get the Y position.*
- [direction\\_t getRotation](#) ()  
*Get the direction.*
- void [setRotation](#) ([direction\\_t](#) dir)  
*Set the direction.*

### Public Attributes

- int [\\_x](#)  
*X position.*
- int [\\_y](#)  
*Y position.*
- [direction\\_t \\_rotation](#)  
*Direction.*

### 4.11.1 Detailed Description

[Position](#) class used to store the position of an object.

### 4.11.2 Constructor & Destructor Documentation

#### 4.11.2.1 Position() [1/3]

```
Position::Position ( )
```

Construct.

#### 4.11.2.2 Position() [2/3]

```
Position::Position (
    int x,
    int y )
```

Construct.

##### Parameters

<i>x</i>	X position
<i>y</i>	Y position

#### 4.11.2.3 Position() [3/3]

```
Position::Position (
    int x,
    int y,
    direction_t dir )
```

Construct.

##### Parameters

<i>x</i>	X position
<i>y</i>	Y position
<i>dir</i>	Direction of the object

#### 4.11.2.4 ~Position()

```
Position::~~Position ( )
```

Destroy.

### 4.11.3 Member Function Documentation

#### 4.11.3.1 getPos()

```
std::tuple< int, int > Position::getPos ( )
```

Get the position.

Returns

std::tuple<int, int> [Position](#)

#### 4.11.3.2 getPosPair()

```
std::pair< int, int > Position::getPosPair ( )
```

Get the position.

Returns

std::pair<int, int> [Position](#)

#### 4.11.3.3 getRotation()

```
direction_t Position::getRotation ( )
```

Get the direction.

Returns

direction\_t Direction

#### 4.11.3.4 getX()

```
int Position::getX ( )
```

#### 4.11.3.5 getY()

```
int Position::getY ( )
```

Get the Y position.

##### Returns

int Y position

#### 4.11.3.6 setRotation()

```
void Position::setRotation (
    direction_t dir )
```

Set the direction.

##### Parameters

<i>dir</i>	Direction
------------	-----------

#### 4.11.3.7 setX()

```
void Position::setX (
    int x )
```

Set the X position.

##### Parameters

<i>x</i>	X position
----------	------------

#### 4.11.3.8 setY()

```
void Position::setY (
    int y )
```



Set the Y position.

#### Parameters

<i>y</i>	Y position
----------	------------

### 4.11.4 Member Data Documentation

#### 4.11.4.1 `_rotation`

`direction_t` `Position::_rotation`

Direction.

#### 4.11.4.2 `_x`

`int` `Position::_x`

X position.

#### 4.11.4.3 `_y`

`int` `Position::_y`

Y position.

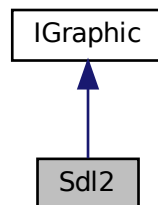
The documentation for this class was generated from the following files:

- [shared/Position.hpp](#)
- [shared/Position.cpp](#)

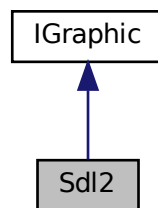
## 4.12 Sdl2 Class Reference

```
#include <basic.hpp>
```

Inheritance diagram for Sdl2:



Collaboration diagram for Sdl2:



### Public Member Functions

- [Sdl2](#) ()
- [~Sdl2](#) ()
- virtual void [init](#) ()
- virtual void [load\\_assets](#) (std::vector< std::tuple< char, std::string, int >> asset\_toload)  
*Load assets from a vector of tuples.*
- virtual void [display\\_asset](#) (std::vector< std::tuple< char, [Position](#) >> object\_to\_display)  
*Display assets from a vector of tuples.*
- virtual void [debug\\_print](#) ()  
*Debug function to print the loaded assets.*
- virtual int [get\\_index\\_tuple](#) (char name)  
*Get the index of a tuple by identifier character.*
- virtual int [get\\_key](#) ()  
*Get the pressed key code.*

- virtual void [display\\_text](#) (std::vector< std::tuple< std::string, [Position](#) >> text\_to\_display)  
*Display text objects.*
- virtual void [load\\_text](#) (std::vector< std::tuple< std::string, [Position](#) >> text\_toload)  
*Load text objects.*
- virtual void [display\\_menu](#) (int choice)
- virtual void [load\\_menu](#) (std::vector< std::string > &menu\_toload)  
*Load the menu.*

## Protected Attributes

- bool [\\_active](#)
- std::vector< std::tuple< char, SDL\_Rect, SDL\_Texture \* > > [\\_assets](#)
- std::vector< std::tuple< std::string, [Position](#), SDL\_Surface \* > > [\\_string](#)
- std::vector< std::tuple< std::string, SDL\_Rect, SDL\_Texture \* > > [\\_menu](#)
- SDL\_Window \* [\\_window](#)
- SDL\_Renderer \* [\\_renderer](#)
- SDL\_Event [\\_event](#)

## 4.12.1 Constructor & Destructor Documentation

### 4.12.1.1 Sdl2()

```
Sdl2::Sdl2 ( )
```

### 4.12.1.2 ~Sdl2()

```
Sdl2::~~Sdl2 ( )
```

## 4.12.2 Member Function Documentation

### 4.12.2.1 debug\_print()

```
virtual void Sdl2::debug_print ( ) [virtual]
```

Debug function to print the loaded assets.

Implements [IGraphic](#).

### 4.12.2.2 display\_asset()

```
virtual void Sdl2::display_asset (
    std::vector< std::tuple< char, Position >> object_to_display ) [virtual]
```

Display assets from a vector of tuples.

**Parameters**

<i>object_to_display</i>	Vector of tuples containing (identifier character, position)
--------------------------	--

Implements [IGraphic](#).

**4.12.2.3 display\_menu()**

```
virtual void Sdl2::display_menu (
    int choice ) [virtual]
```

**4.12.2.4 display\_text()**

```
virtual void Sdl2::display_text (
    std::vector< std::tuple< std::string, Position >> text_to_display ) [virtual]
```

Display text objects.

**Parameters**

<i>text_to_display</i>	Vector of tuples containing (text string, position)
------------------------	---

Implements [IGraphic](#).

**4.12.2.5 get\_index\_tuple()**

```
int Sdl2::get_index_tuple (
    char name ) [virtual]
```

Get the index of a tuple by identifier character.

**Parameters**

<i>name</i>	Identifier character of the tuple
-------------	-----------------------------------

**Returns**

Index of the tuple

Implements [IGraphic](#).

#### 4.12.2.6 get\_key()

```
virtual int Sdl2::get_key ( ) [virtual]
```

Get the pressed key code.

##### Returns

Key code of the pressed key

Implements [IGraphic](#).

#### 4.12.2.7 init()

```
void Sdl2::init ( ) [virtual]
```

##### Initial value:

```
{  
    init()  
}
```

#### 4.12.2.8 load\_assets()

```
virtual void Sdl2::load_assets (  
    std::vector< std::tuple< char, std::string, int >> asset_toload ) [virtual]
```

Load assets from a vector of tuples.

##### Parameters

<i>asset_toload</i>	Vector of tuples containing (identifier character, asset path, asset type)
---------------------	--

Implements [IGraphic](#).

#### 4.12.2.9 load\_menu()

```
virtual void Sdl2::load_menu (  
    std::vector< std::string > & menu_toload ) [virtual]
```

Load the menu.

##### Parameters

<i>menu_toload</i>	Vector of menu items as strings
--------------------	---------------------------------

Implements [IGraphic](#).

#### 4.12.2.10 load\_text()

```
virtual void Sdl2::load_text (
    std::vector< std::tuple< std::string, Position >> text_toload ) [virtual]
```

Load text objects.

##### Parameters

<i>text_toload</i>	Vector of tuples containing (text string, position)
--------------------	---

Implements [IGraphic](#).

### 4.12.3 Member Data Documentation

#### 4.12.3.1 \_active

```
bool Sdl2::_active [protected]
```

#### 4.12.3.2 \_assets

```
std::vector<std::tuple<char, SDL_Rect, SDL_Texture*> > Sdl2::_assets [protected]
```

#### 4.12.3.3 \_event

```
SDL_Event Sdl2::_event [protected]
```

#### 4.12.3.4 \_menu

```
std::vector<std::tuple<std::string, SDL_Rect, SDL_Texture*> > Sdl2::_menu [protected]
```

#### 4.12.3.5 `_renderer`

```
SDL_Renderer* Sdl2::_renderer [protected]
```

#### 4.12.3.6 `_string`

```
std::vector<std::tuple<std::string, Position, SDL_Surface*> > Sdl2::_string [protected]
```

#### 4.12.3.7 `_window`

```
SDL_Window* Sdl2::_window [protected]
```

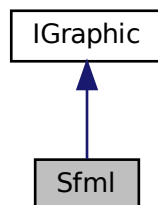
The documentation for this class was generated from the following files:

- [src\\_graphic/src\\_sdl2/basic.hpp](#)
- [src\\_graphic/src\\_sdl2/basic.cpp](#)

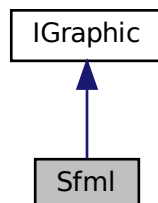
## 4.13 SfmI Class Reference

```
#include <basic.hpp>
```

Inheritance diagram for SfmI:



Collaboration diagram for SfmI:



## Public Member Functions

- [Sfml](#) ()
- [~Sfml](#) ()
- `std::pair< int, int >` [getConfigData](#) ()  
*Get the config data : resolution.*
- virtual void [init](#) ()  
*Initialize the window and the music.*
- virtual void [load\\_assets](#) (`std::vector< std::tuple< char, std::string, int >>` asset\_toload)  
*Load the assets of the game. it will store the assets in the vector of tuple see [\\_assets](#) for more details.*
- virtual void [display\\_asset](#) (`std::vector< std::tuple< char, Position >>` object\_to\_display)  
*Display the assets of the game. it will display the assets in the vector of tuple see [\\_assets](#) for more details. it will receive the position, and the identifier of the asset(character) to display.*
- virtual void [debug\\_print](#) ()  
*Print the assets of the game. it will print the assets in the vector of tuple see [\\_assets](#) for more details.*
- virtual int [get\\_index\\_tuple](#) (char name)  
*Get the index of the tuple in the vector of tuple see [\\_assets](#) for more details.*
- virtual int [get\\_key](#) ()  
*Get the key pressed by the user.*
- virtual void [display\\_text](#) (`std::vector< std::tuple< std::string, Position >>` text\_to\_display)  
*Display the text of the game. it will display the text in the vector of tuple see [\\_text](#).*
- virtual void [load\\_text](#) (`std::vector< std::tuple< std::string, Position >>` text\_toload)  
*Load the text of the game. it will store the text in the vector of tuple see [\\_text](#).*
- virtual int [display\\_menu](#) (int choice, int isRunning)  
*Display the menu of the game. it will display the menu in the vector of tuple see [\\_menu](#).*
- virtual void [load\\_menu](#) (`std::vector< std::string >` &menu\_toload)  
*Load the menu of the game. it will store the menu in the vector of tuple see [\\_menu](#).*
- virtual void [display\\_high\\_score](#) (`std::vector< std::string >` high\_score)  
*Display the high score of the game. it will display the high score in the vector of string received for the 10 best score.*
- virtual void [displayConfig](#) (`size_t` \_menuChoice, `std::vector< std::tuple< std::string, std::vector< size_t >, size_t >>` \_options)  
*Display the config of the game. it will display the config in the vector of tuple received. Highlighting the choice of the user.*
- virtual void [load\\_config](#) (`std::vector< std::tuple< std::string, std::vector< size_t >, size_t >>` \_options)  
*Load the config of the game. it will store the config in the vector of tuple received.*
- virtual void [deleteAssets](#) ()  
*Delete the assets of the game. it will delete the assets in the vector of tuple see [\\_assets](#).*
- void [load\\_background\\_menu](#) (`std::string` path)  
*Load the background of the menu. it will store the background in the vector of tuple see [\\_menu](#).*
- int [isConfirmSoundPlaying](#) ()  
*Check if the confirm sound is playing.*
- void [reverseMusic](#) (`std::string` name)  
*Stop the music and play the music in parameter.*

## Protected Attributes

- bool [\\_active](#)  
*Check if the game is active.*
- `std::vector< std::tuple< char, sf::RectangleShape *, sf::Texture >>` [\\_assets](#)  
*A vector of tuple containing the character of the asset, the rectangle shape of the asset and the texture of the asset.*
- `std::vector< std::tuple< std::string, Position, sf::Text * >>` [\\_string](#)



- A vector of tuple containing the string of the text, the position of the text and the text.*

  - `std::vector< std::tuple< std::string, sf::RectangleShape *, sf::Texture > > _menu`
- A vector of tuple containing the string of the menu, the rectangle shape of the menu and the texture of the menu.*

  - `std::vector< std::tuple< std::string, std::string, sf::Music * > > _music`
- A vector of tuple containing the name of the music, the path of the music and the music.*

  - `std::vector< std::tuple< std::string, std::string, sf::Sound *, sf::SoundBuffer * > > _sound_menu`
- A vector of tuple containing the name of the sound, the path of the sound, the sound and the sound buffer.*

  - `sf::RenderWindow _window`
- The window to render in SFML.*

  - `sf::Event _event`
- The event of the window.*

  - `std::vector< std::pair< std::string, std::string > > _configData`
- A vector of pair containing the name of the option and the value of the option.*

  - `std::vector< std::tuple< std::string, std::string, sf::Text * > > _optionsText`
- A vector of tuple containing the name of the option, the value of the option and the text.*

### 4.13.1 Constructor & Destructor Documentation

#### 4.13.1.1 Sfml()

```
Sfml::Sfml ( )
```

#### 4.13.1.2 ~Sfml()

```
Sfml::~Sfml ( )
```

### 4.13.2 Member Function Documentation

#### 4.13.2.1 debug\_print()

```
void Sfml::debug_print ( ) [virtual]
```

Print the assets of the game. it will print the assets in the vector of tuple see [\\_assets](#) for more details.

Implements [IGraphic](#).

#### 4.13.2.2 deleteAssets()

```
void Sfml::deleteAssets ( ) [virtual]
```

Delete the assets of the game. it will delete the assets in the vector of tuple see [\\_assets](#).

Implements [IGraphic](#).

#### 4.13.2.3 display\_asset()

```
void Sfml::display_asset (
    std::vector< std::tuple< char, Position >> object_to_display ) [virtual]
```

Display the assets of the game. it will display the assets in the vector of tuple see [\\_assets](#) for more details. it will receive the position, and the identifier of the asset(character) to display.

##### Parameters

<i>object_to_display</i>	A vector of tuple containing the character, the position of the asset
--------------------------	---

Implements [IGraphic](#).

#### 4.13.2.4 display\_high\_score()

```
void Sfml::display_high_score (
    std::vector< std::string > high_score ) [virtual]
```

Display the high score of the game. it will display the high score in the vector of string received for the 10 best score.

##### Parameters

<i>high_score</i>	A vector of string containing the high score.
-------------------	---

Implements [IGraphic](#).

#### 4.13.2.5 display\_menu()

```
int Sfml::display_menu (
    int choice,
    int isRunning ) [virtual]
```

Display the menu of the game. it will display the menu in the vector of tuple see [\\_menu](#).

## Parameters

<i>choice</i>	The choice of the user.
---------------	-------------------------

Implements [IGraphic](#).

**4.13.2.6 display\_text()**

```
void Sfml::display_text (
    std::vector< std::tuple< std::string, Position >> text_to_display ) [virtual]
```

Display the text of the game. it will display the text in the vector of tuple see `_text`.

## Parameters

<i>text_to_display</i>	A vector of tuple containing the string, the position of the text
------------------------	---

Implements [IGraphic](#).

**4.13.2.7 displayConfig()**

```
void Sfml::displayConfig (
    size_t _menuChoice,
    std::vector< std::tuple< std::string, std::vector< size_t >, size_t >> _options
) [virtual]
```

Display the config of the game. it will display the config in the vector of tuple received. Highlighting the choice of the user.

## Parameters

<i>_menuChoice</i>	The choice of the user.
<i>_options</i>	A vector of tuple containing the name of the option, a vector of the value of the option and the index of the option.

Implements [IGraphic](#).

**4.13.2.8 get\_index\_tuple()**

```
int Sfml::get_index_tuple (
    char name ) [virtual]
```

Get the index of the tuple in the vector of tuple see [\\_assets](#) for more details.

**Parameters**

<i>name</i>	The character of the asset
-------------	----------------------------

**Returns**

The index of the tuple in the vector of tuple see [\\_assets](#). if the character is not found, it will return -1

Implements [IGraphic](#).

**4.13.2.9 get\_key()**

```
int Sfml::get_key ( ) [virtual]
```

Get the key pressed by the user.

**Returns**

The key pressed by the user

Implements [IGraphic](#).

**4.13.2.10 getConfigData()**

```
std::pair< int, int > Sfml::getConfigData ( )
```

Get the config data : resolution.

**Returns**

std::pair<int, int> : resolution

**4.13.2.11 init()**

```
void Sfml::init ( ) [virtual]
```

Initialize the window and the music.

#### 4.13.2.12 isConfirmSoundPlaying()

```
int Sfml::isConfirmSoundPlaying ( )
```

Check if the confirm sound is playing.

##### Returns

- 2 if the confirm sound is finished.
- 1 if the confirm sound is playing.
- 0 if the confirm sound is not playing.

#### 4.13.2.13 load\_assets()

```
void Sfml::load_assets (
    std::vector< std::tuple< char, std::string, int >> asset_toload ) [virtual]
```

Load the assets of the game. it will store the assets in the vector of tuple see [\\_assets](#) for more details.

##### Parameters

<i>asset_toload</i>	A vector of tuple containing the character, the path to the asset and the index for the spritesheet
---------------------	---

Implements [IGraphic](#).

#### 4.13.2.14 load\_background\_menu()

```
void Sfml::load_background_menu (
    std::string path )
```

Load the background of the menu. it will store the background in the vector of tuple see [\\_menu](#).

#### 4.13.2.15 load\_config()

```
void Sfml::load_config (
    std::vector< std::tuple< std::string, std::vector< size_t >, size_t >> _options
) [virtual]
```

Load the config of the game. it will store the config in the vector of tuple received.

## Parameters

<i>_options</i>	A vector of tuple containing the name of the option, a vector of the value of the option and the index of the option.
-----------------	---

Implements [IGraphic](#).

**4.13.2.16 load\_menu()**

```
void Sfml::load_menu (
    std::vector< std::string > & menu_toload ) [virtual]
```

Load the menu of the game. it will store the menu in the vector of tuple see [\\_menu](#).

## Parameters

<i>menu_toload</i>	A vector of string containing the path to the menu.
--------------------	---

Implements [IGraphic](#).

**4.13.2.17 load\_text()**

```
void Sfml::load_text (
    std::vector< std::tuple< std::string, Position >> text_toload ) [virtual]
```

Load the text of the game. it will store the text in the vector of tuple see [\\_text](#).

## Parameters

<i>text_toload</i>	A vector of tuple containing the string, the position of the text
--------------------	---

Implements [IGraphic](#).

**4.13.2.18 reverseMusic()**

```
void Sfml::reverseMusic (
    std::string name )
```

Stop the music and play the music in parameter.

## Parameters

<i>name</i>	The name of the music to play.
-------------	--------------------------------

### 4.13.3 Member Data Documentation

#### 4.13.3.1 `_active`

```
bool Sfml::_active [protected]
```

Check if the game is active.

#### 4.13.3.2 `_assets`

```
std::vector<std::tuple<char, sf::RectangleShape *, sf::Texture> > Sfml::_assets [protected]
```

A vector of tuple containing the character of the asset, the rectangle shape of the asset and the texture of the asset.

#### 4.13.3.3 `_configData`

```
std::vector<std::pair<std::string, std::string> > Sfml::_configData [protected]
```

A vector of pair containing the name of the option and the value of the option.

#### 4.13.3.4 `_event`

```
sf::Event Sfml::_event [protected]
```

The event of the window.

#### 4.13.3.5 `_menu`

```
std::vector<std::tuple<std::string, sf::RectangleShape *, sf::Texture> > Sfml::_menu [protected]
```

A vector of tuple containing the string of the menu, the rectangle shape of the menu and the texture of the menu.

#### 4.13.3.6 `_music`

```
std::vector<std::tuple<std::string, std::string, sf::Music *> > Sfml::_music [protected]
```

A vector of tuple containing the name of the music, the path of the music and the music.

#### 4.13.3.7 `_optionsText`

```
std::vector<std::tuple<std::string, std::string, sf::Text *> > Sfml::_optionsText [protected]
```

A vector of tuple containing the name of the option, the value of the option and the text.

#### 4.13.3.8 `_sound_menu`

```
std::vector<std::tuple<std::string, std::string, sf::Sound *, sf::SoundBuffer *> > Sfml::_↵  
sound_menu [protected]
```

A vector of tuple containing the name of the sound, the path of the sound, the sound and the sound buffer.

#### 4.13.3.9 `_string`

```
std::vector<std::tuple<std::string, Position, sf::Text *> > Sfml::_string [protected]
```

A vector of tuple containing the string of the text, the position of the text and the text.

#### 4.13.3.10 `_window`

```
sf::RenderWindow Sfml::_window [protected]
```

The window to render in SFML.

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

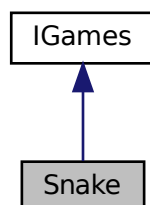
- [src\\_graphic/src\\_sfml/basic.hpp](#)
- [src\\_graphic/src\\_sfml/basic.cpp](#)



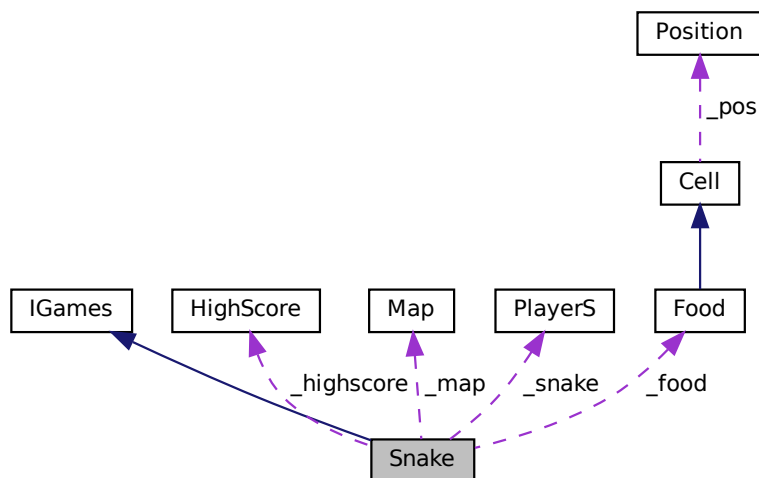
## 4.14 Snake Class Reference

```
#include <Snake_game.hpp>
```

Inheritance diagram for Snake:



Collaboration diagram for Snake:



### Public Member Functions

- `Snake` ()
- `Snake` (int x, int y)
- `~Snake` ()
- virtual void `init` ()  
*Initialize the game.*
- virtual void `initTexts` ()
- virtual bool `update` (int key)

*Update the game based on the key pressed.*

- virtual std::vector< std::tuple< char, std::string, int > > [loadAssets](#) ()
- virtual std::vector< std::tuple< char, [Position](#) > > [getElements](#) ()
- virtual std::tuple< std::vector< [Position](#) >, std::vector< [Position](#) > > [parse\\_map](#) (std::string map\_name)
- virtual double [getScore](#) ()

*Get the score of the game.*

- virtual size\_t [getSpeed](#) ()

*Get the speed of the game.*

- virtual void [handleCollision](#) ()

*Handle the collision of the game.*

- virtual void [handleInput](#) (int key)

*Handle the input of the game.*

- virtual bool [collisionMap](#) ([Position](#) pos)

*Check if the position is a wall.*

- virtual void [collisionFood](#) ()

*Check if the position is a food.*

- virtual std::vector< std::tuple< std::string, [Position](#) > > [loadTexts](#) ()

- virtual void [updateTexts](#) ()

*Update the texts of the game.*

- virtual std::vector< std::string > [writeScore](#) ()

*Write the score of the game.*

- virtual std::vector< std::string > [readScore](#) ()

*Read the score of the game.*

- virtual void [readConfig](#) ()

- std::string [getLogin](#) ()

## Protected Attributes

- bool [HighScoreIsSet](#) = false
- size\_t [\\_skinMultiplier](#) = 0
- size\_t [\\_speed](#) = 150
- size\_t [\\_score](#) = 0
- size\_t [\\_highScore](#) = 0
- size\_t [\\_level](#) = 1
- [Food](#) \* [\\_food](#)
- [Map](#) \* [\\_map](#)
- [PlayerS](#) \* [\\_snake](#)
- [HighScore](#) \* [\\_highscore](#)
- std::vector< std::tuple< std::string, [Position](#) > > [\\_string](#)
- std::vector< std::pair< std::string, std::string > > [\\_configData](#)

## 4.14.1 Constructor & Destructor Documentation

### 4.14.1.1 Snake() [1/2]

```
Snake::Snake ( )
```

#### 4.14.1.2 Snake() [2/2]

```
Snake::Snake (
    int x,
    int y )
```

#### 4.14.1.3 ~Snake()

```
Snake::~~Snake ( )
```

### 4.14.2 Member Function Documentation

#### 4.14.2.1 collisionFood()

```
void Snake::collisionFood ( ) [virtual]
```

Check if the position is a food.

##### Parameters

<i>pos</i>	The position to check
------------	-----------------------

##### Returns

true if the position is a food, false if not

Implements [IGames](#).

#### 4.14.2.2 collisionMap()

```
bool Snake::collisionMap (
    Position pos ) [virtual]
```

Check if the position is a wall.

##### Parameters

<i>pos</i>	The position to check
------------	-----------------------

**Returns**

true if the position is a wall, false if not

Implements [IGames](#).

**4.14.2.3 getElements()**

```
std::vector< std::tuple< char, Position > > Snake::getElements ( ) [virtual]
```

Implements [IGames](#).

**4.14.2.4 getLogin()**

```
std::string Snake::getLogin ( )
```

**4.14.2.5 getScore()**

```
double Snake::getScore ( ) [virtual]
```

Get the score of the game.

**Returns**

A double containing the score

Implements [IGames](#).

**4.14.2.6 getSpeed()**

```
size_t Snake::getSpeed ( ) [virtual]
```

Get the speed of the game.

**Returns**

A size\_t containing the speed

Implements [IGames](#).

#### 4.14.2.7 handleCollision()

```
void Snake::handleCollision ( ) [virtual]
```

Handle the collision of the game.

Implements [IGames](#).

#### 4.14.2.8 handleInput()

```
void Snake::handleInput (
    int key ) [virtual]
```

Handle the input of the game.

#### Parameters

<i>key</i>	The key pressed
------------	-----------------

Implements [IGames](#).

#### 4.14.2.9 init()

```
void Snake::init ( ) [virtual]
```

Initialize the game.

This function is used to initialize the game

Implements [IGames](#).

#### 4.14.2.10 initTexts()

```
void Snake::initTexts ( ) [virtual]
```

#### 4.14.2.11 loadAssets()

```
std::vector< std::tuple< char, std::string, int > > Snake::loadAssets ( ) [virtual]
```

Implements [IGames](#).

#### 4.14.2.12 loadTexts()

```
std::vector< std::tuple< std::string, Position > > Snake::loadTexts ( ) [virtual]
```

Implements [IGames](#).

#### 4.14.2.13 parse\_map()

```
std::tuple< std::vector< Position >, std::vector< Position > > Snake::parse_map (
    std::string map_name ) [virtual]
```

Implements [IGames](#).

#### 4.14.2.14 readConfig()

```
void Snake::readConfig ( ) [virtual]
```

Implements [IGames](#).

#### 4.14.2.15 readScore()

```
std::vector< std::string > Snake::readScore ( ) [virtual]
```

Read the score of the game.

##### Returns

A vector of string containing the score

Implements [IGames](#).

#### 4.14.2.16 update()

```
bool Snake::update (
    int key ) [virtual]
```

Update the game based on the key pressed.

##### Parameters

key	The key pressed for handling the input later
-----	--

##### Returns

true if the game is still running, false if the game is over

Implements [IGames](#).

#### 4.14.2.17 updateTexts()

```
void Snake::updateTexts ( ) [virtual]
```

Update the texts of the game.

Implements [IGames](#).

#### 4.14.2.18 writeScore()

```
std::vector< std::string > Snake::writeScore ( ) [virtual]
```

Write the score of the game.

##### Returns

A vector of string containing the score

Implements [IGames](#).

### 4.14.3 Member Data Documentation

#### 4.14.3.1 \_configData

```
std::vector<std::pair<std::string, std::string> > Snake::_configData [protected]
```

#### 4.14.3.2 \_food

```
Food* Snake::_food [protected]
```

#### 4.14.3.3 \_highScore

```
size_t Snake::_highScore = 0 [protected]
```

#### 4.14.3.4 \_highscore

```
HighScore* Snake::_highscore [protected]
```

#### 4.14.3.5 \_level

```
size_t Snake::_level = 1 [protected]
```



#### 4.14.3.6 `_map`

```
Map* Snake::_map [protected]
```

#### 4.14.3.7 `_score`

```
size_t Snake::_score = 0 [protected]
```

#### 4.14.3.8 `_skinMultiplier`

```
size_t Snake::_skinMultiplier = 0 [protected]
```

#### 4.14.3.9 `_snake`

```
PlayerS* Snake::_snake [protected]
```

#### 4.14.3.10 `_speed`

```
size_t Snake::_speed = 150 [protected]
```

#### 4.14.3.11 `_string`

```
std::vector<std::tuple<std::string, Position> > Snake::_string [protected]
```

#### 4.14.3.12 `HighScoreIsSet`

```
bool Snake::HighScoreIsSet = false [protected]
```

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

- `src_game/Snake/Snake_game.hpp`
- `src_game/Snake/Snake_game.cpp`



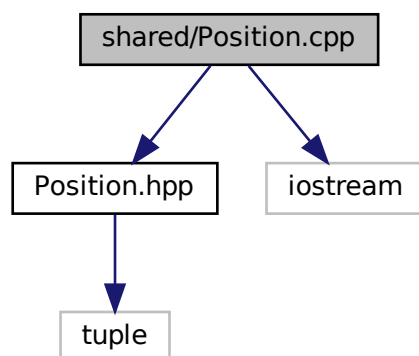
## Chapter 5

# File Documentation

### 5.1 shared/Position.cpp File Reference

```
#include "Position.hpp"  
#include <iostream>
```

Include dependency graph for Position.cpp:



### 5.2 shared/Position.hpp File Reference

[Position](#) class declaration used to store the position of an object.



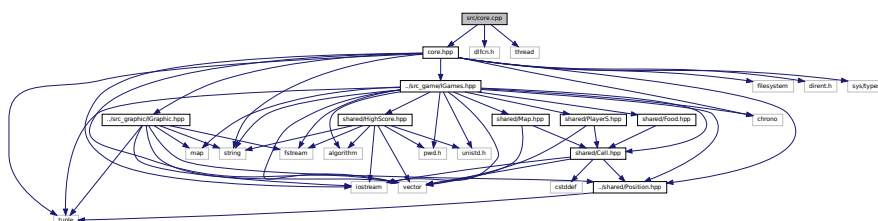
## Enumerator

UP	
DOWN	
LEFT	
RIGHT	
NONE	

### 5.3 src/core.cpp File Reference

```
#include "core.hpp"
#include <dlfcn.h>
#include <thread>
```

Include dependency graph for core.cpp:



## Functions

- `int main (int argc, char *argv[ ])`

### 5.3.1 Function Documentation

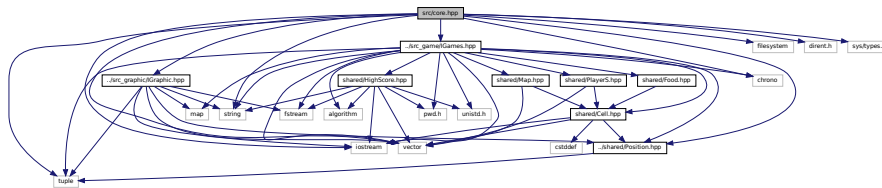
### 5.3.1.1 main()

```
int main (
    int argc,
    char * argv[] )
```

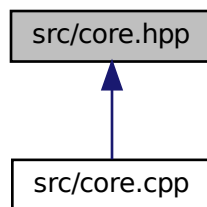
## 5.4 src/core.hpp File Reference

```
#include <iostream>
#include <string>
#include <vector>
#include <tuple>
#include <filesystem>
#include <dirent.h>
#include <sys/types.h>
#include "../src_graphic/IGraphic.hpp"
#include "../src_game/IGames.hpp"
#include "../shared/Position.hpp"
#include <chrono>
```

Include dependency graph for core.hpp:



This graph shows which files directly or indirectly include this file:



### Classes

- class [core](#)  
*Main Core class for the application.*

### Typedefs

- using [LibGraphicFuncPtr](#) = [IGraphic](#) \*(\*)()
- using [LibGameFuncPtr](#) = [IGames](#) \*(\*)()
- using [GetIdFuncPtr](#) = std::string(\*)()
- using [LibGraphicDeletePtr](#) = void(\*)([IGraphic](#) \*)
- using [LibGameDeletePtr](#) = void(\*)([IGames](#) \*)
- typedef enum [keys\\_e](#) [keys\\_t](#)  
*Enum for key values.*
- typedef enum [state\\_e](#) [state\\_t](#)  
*Enum for application state values.*

## Enumerations

- enum `keys_e` {  
    `KEY_DOWN` = 258 , `KEY_UP` = 259 , `KEY_LEFT` = 260 , `KEY_RIGHT` = 261 ,  
    `KEY_ENTER` = 10 , `KEY_BACKSPACE` = 8 , `KEY_ESCAPE` = 36 }  
    *Enum for key values.*
- enum `state_e` {  
    `UNDEFINED` = 0 , `MENU` = 1 , `GAME` = 2 , `HIGH_SCORE` = 3 ,  
    `CONFIG` = 4 , `END` = 5 }  
    *Enum for application state values.*

### 5.4.1 Typedef Documentation

#### 5.4.1.1 GetIdFuncPtr

```
using GetIdFuncPtr = std::string (*)()
```

#### 5.4.1.2 keys\_t

```
typedef enum keys_e keys_t
```

Enum for key values.

#### 5.4.1.3 LibGameDeletePtr

```
using LibGameDeletePtr = void (*) (IGames*)
```

#### 5.4.1.4 LibGameFuncPtr

```
using LibGameFuncPtr = IGames* (*)()
```

#### 5.4.1.5 LibGraphicDeletePtr

```
using LibGraphicDeletePtr = void (*) (IGraphic*)
```

#### 5.4.1.6 LibGraphicFuncPtr

```
using LibGraphicFuncPtr = IGraphic* (*)()
```

#### 5.4.1.7 state\_t

```
typedef enum state_e state_t
```

Enum for application state values.

### 5.4.2 Enumeration Type Documentation

#### 5.4.2.1 keys\_e

```
enum keys_e
```

Enum for key values.

Enumerator

KEY_DOWN	
KEY_UP	
KEY_LEFT	
KEY_RIGHT	
KEY_ENTER	
KEY_BACKSPACE	
KEY_ESCAPE	

#### 5.4.2.2 state\_e

```
enum state_e
```

Enum for application state values.

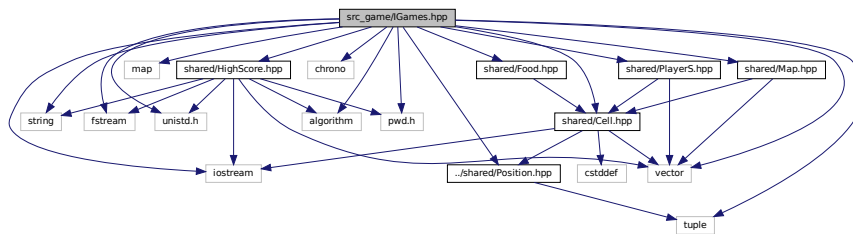
Enumerator

UNDEFINED	
MENU	
GAME	
HIGH_SCORE	
CONFIG	
END	

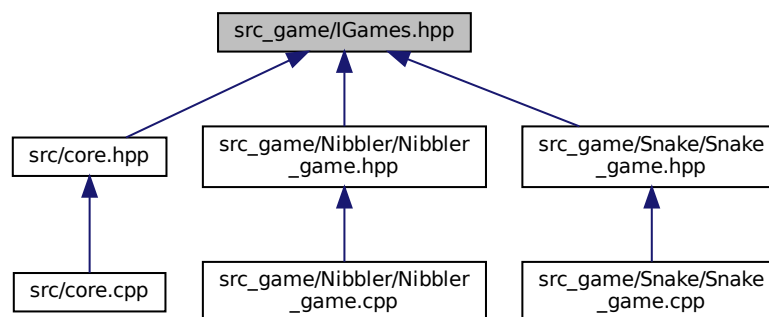


## 5.5 src\_game/IGames.hpp File Reference

```
#include <iostream>
#include <string>
#include <vector>
#include <tuple>
#include <map>
#include <fstream>
#include <chrono>
#include <unistd.h>
#include <algorithm>
#include <pwd.h>
#include "../shared/Position.hpp"
#include "shared/Cell.hpp"
#include "shared/Map.hpp"
#include "shared/Food.hpp"
#include "shared/PlayerS.hpp"
#include "shared/HighScore.hpp"
Include dependency graph for IGames.hpp:
```



This graph shows which files directly or indirectly include this file:



## Classes

- class [IGames](#)

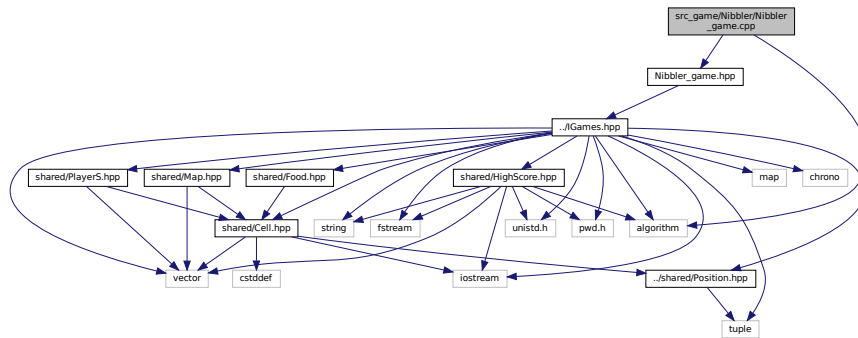
*Interface for handling games.*

## 5.6 src\_game/Nibbler/Nibbler\_game.cpp File Reference

```
#include "Nibbler_game.hpp"
```

```
#include <algorithm>
```

Include dependency graph for Nibbler\_game.cpp:



### Functions

- `IGames * create ()`
- `std::string getName ()`
- `void destroy (IGames *game)`
- `IGames * createGameInstance ()`

### 5.6.1 Function Documentation

#### 5.6.1.1 create()

```
IGames* create ( )
```

#### 5.6.1.2 createGameInstance()

```
IGames* createGameInstance ( )
```

#### 5.6.1.3 destroy()

```
void destroy (
    IGames * game )
```

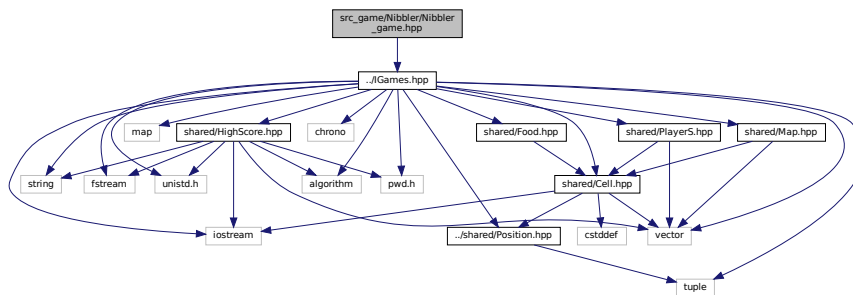
## 5.6.1.4 getName()

```
std::string getName ( )
```

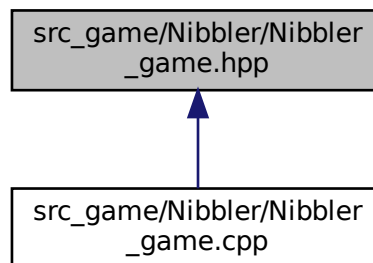
## 5.7 src\_game/Nibbler/Nibbler\_game.hpp File Reference

```
#include "../IGames.hpp"
```

Include dependency graph for Nibbler\_game.hpp:



This graph shows which files directly or indirectly include this file:



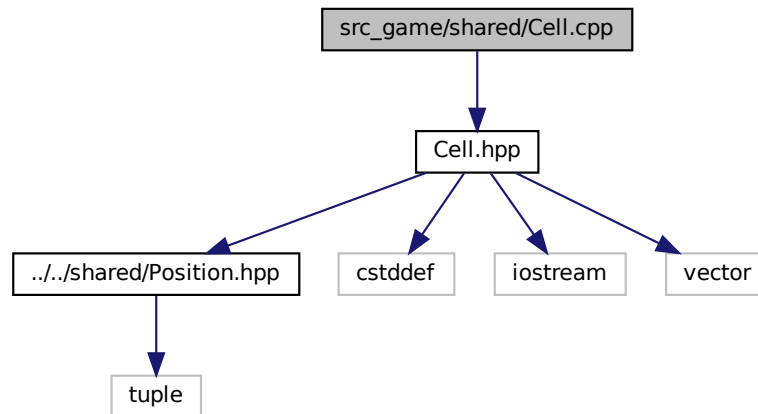
## Classes

- class [Nibbler](#)  
*[Nibbler](#) class used to manage the game [Nibbler](#).*

## 5.8 src\_game/shared/Cell.cpp File Reference

```
#include "Cell.hpp"
```

Include dependency graph for Cell.cpp:



## 5.9 src\_game/shared/Cell.hpp File Reference

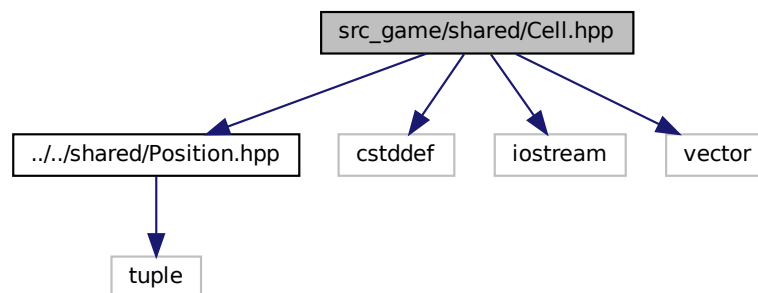
```
#include "../shared/Position.hpp"
```

```
#include <cstdint>
```

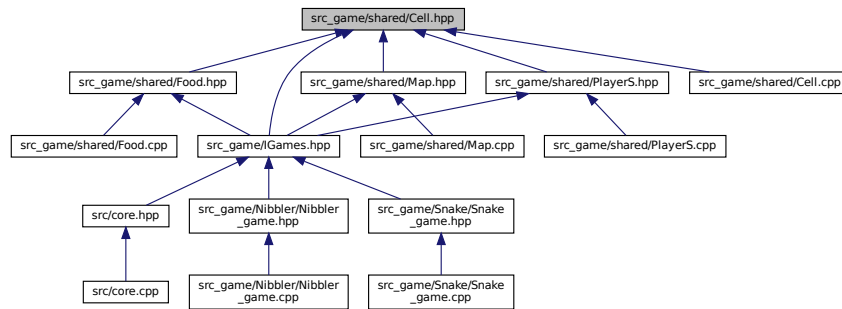
```
#include <iostream>
```

```
#include <vector>
```

Include dependency graph for Cell.hpp:



This graph shows which files directly or indirectly include this file:



## Classes

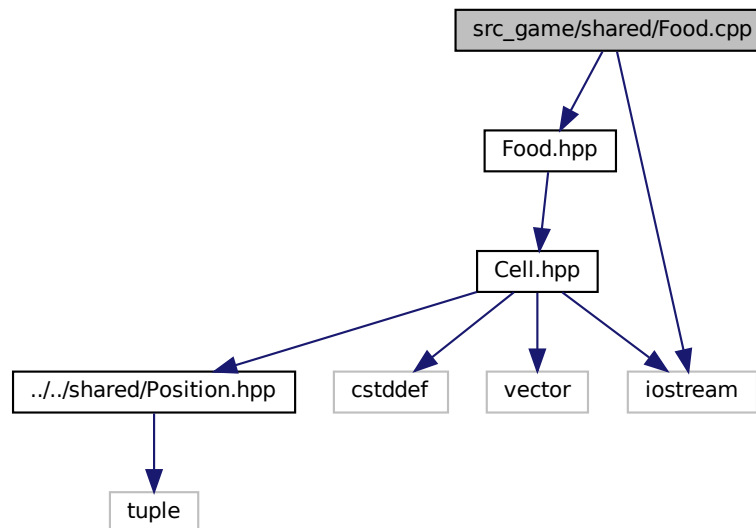
- class [Cell](#)

## 5.10 src\_game/shared/Food.cpp File Reference

```
#include "Food.hpp"
```

```
#include <iostream>
```

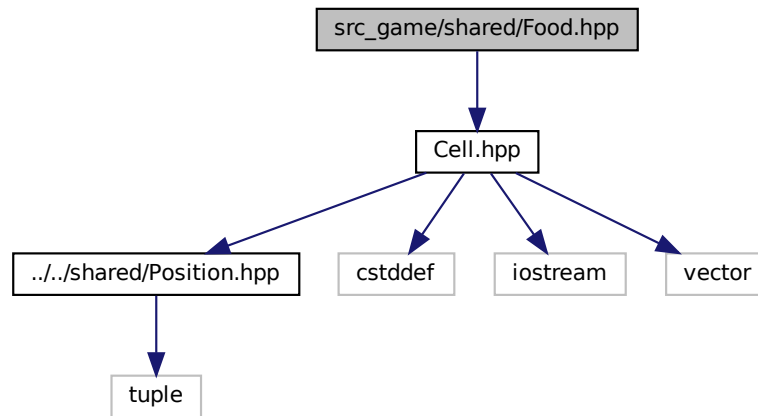
Include dependency graph for Food.cpp:



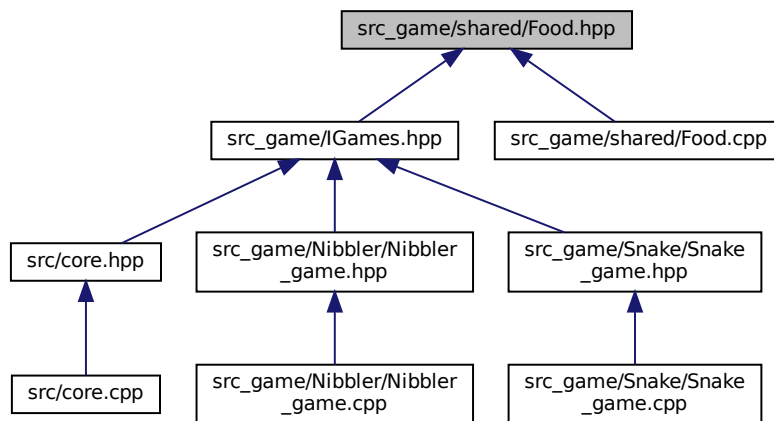
## 5.11 src\_game/shared/Food.hpp File Reference

```
#include "Cell.hpp"
```

Include dependency graph for Food.hpp:



This graph shows which files directly or indirectly include this file:



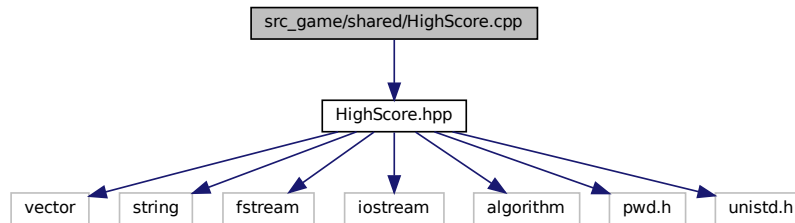
## Classes

- class [Food](#)

## 5.12 src\_game/shared/HighScore.cpp File Reference

```
#include "HighScore.hpp"
```

Include dependency graph for HighScore.cpp:



### Functions

- int `extractScore` (const std::string &line)

#### 5.12.1 Function Documentation

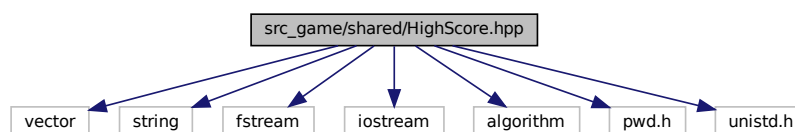
##### 5.12.1.1 `extractScore()`

```
int extractScore (
    const std::string & line )
```

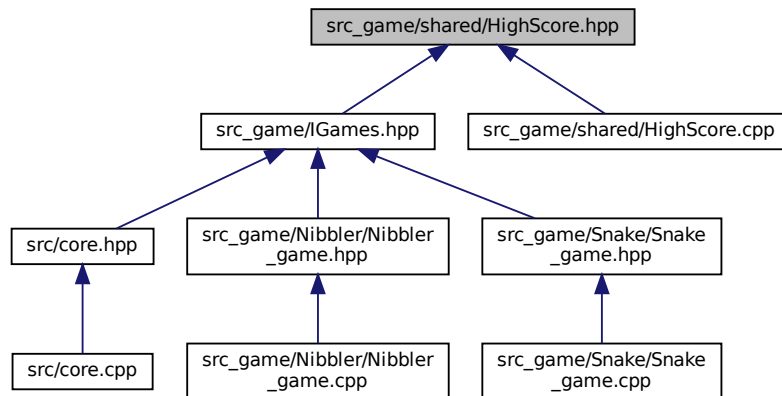
## 5.13 src\_game/shared/HighScore.hpp File Reference

```
#include <vector>
#include <string>
#include <fstream>
#include <iostream>
#include <algorithm>
#include <pwd.h>
#include <unistd.h>
```

Include dependency graph for HighScore.hpp:



This graph shows which files directly or indirectly include this file:



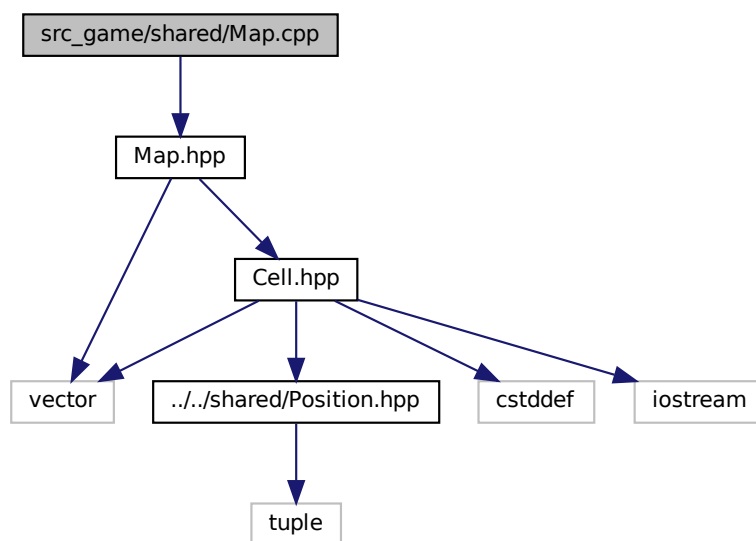
## Classes

- class [HighScore](#)

## 5.14 src\_game/shared/Map.cpp File Reference

```
#include "Map.hpp"
```

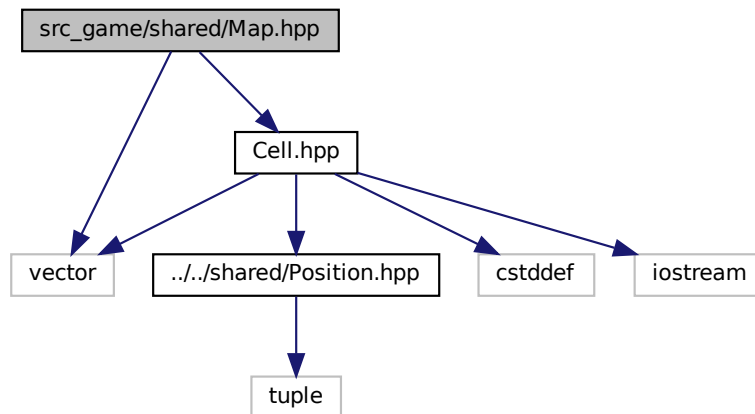
Include dependency graph for Map.cpp:



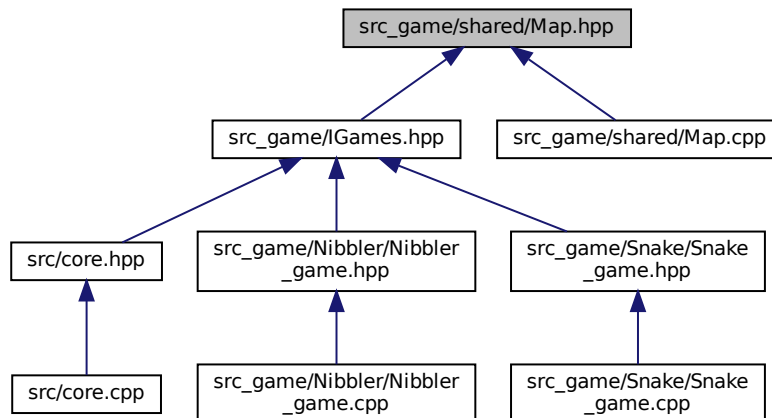


## 5.15 src\_game/shared/Map.hpp File Reference

```
#include <vector>
#include "Cell.hpp"
Include dependency graph for Map.hpp:
```



This graph shows which files directly or indirectly include this file:



### Classes

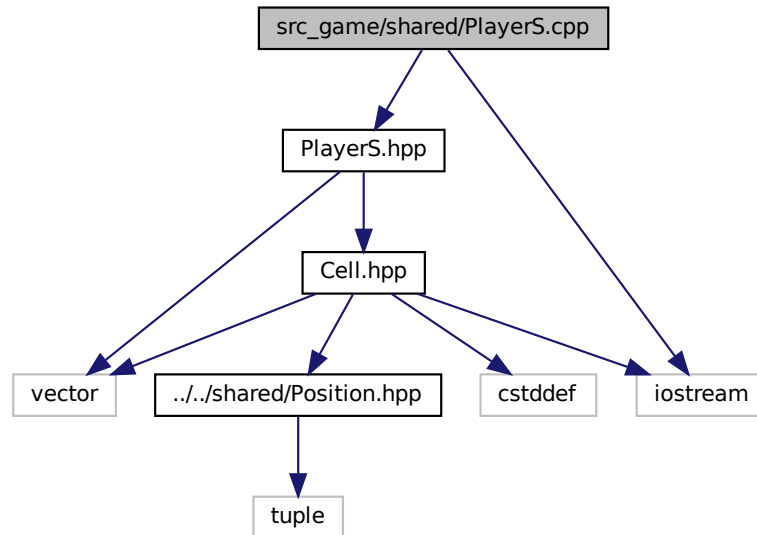
- class [Map](#)

## 5.16 src\_game/shared/PlayerS.cpp File Reference

```
#include "PlayerS.hpp"
```

```
#include <iostream>
```

Include dependency graph for PlayerS.cpp:

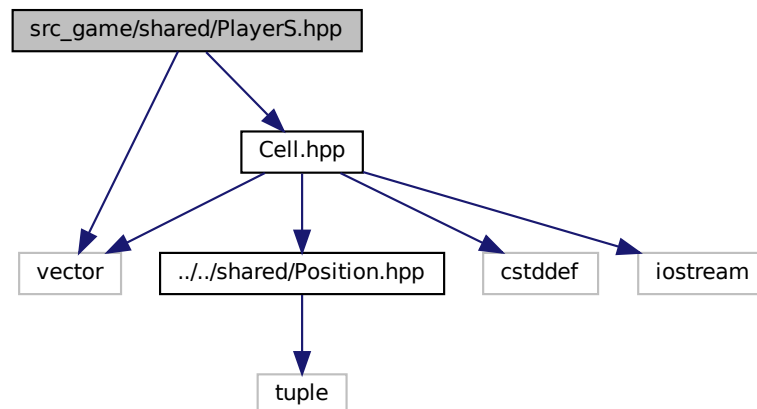


## 5.17 src\_game/shared/PlayerS.hpp File Reference

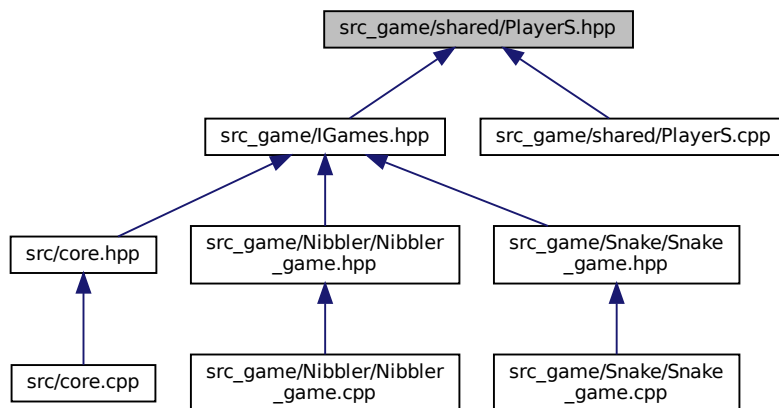
```
#include <vector>
```

```
#include "Cell.hpp"
```

Include dependency graph for PlayerS.hpp:



This graph shows which files directly or indirectly include this file:



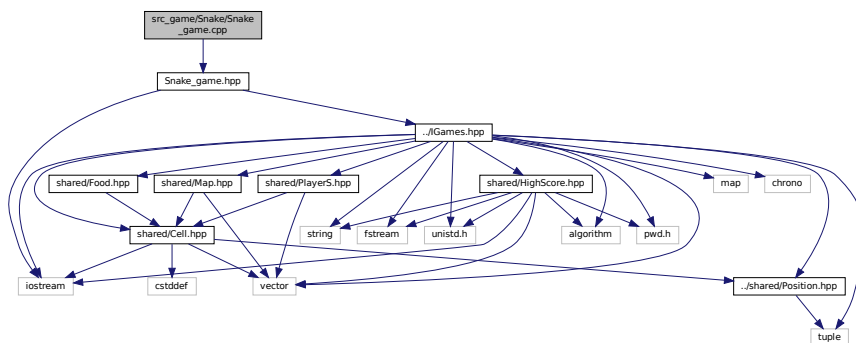
## Classes

- class [PlayerS](#)

## 5.18 src\_game/Snake/Snake\_game.cpp File Reference

```
#include "Snake_game.hpp"
```

Include dependency graph for Snake\_game.cpp:



## Functions

- [IGames](#) \* [create](#) ()
- [IGames](#) \* [createGameInstance](#) ()
- std::string [getName](#) ()
- void [destroy](#) ([IGames](#) \*game)

## 5.18.1 Function Documentation

### 5.18.1.1 create()

```
IGames* create ( )
```

### 5.18.1.2 createGameInstance()

```
IGames* createGameInstance ( )
```

### 5.18.1.3 destroy()

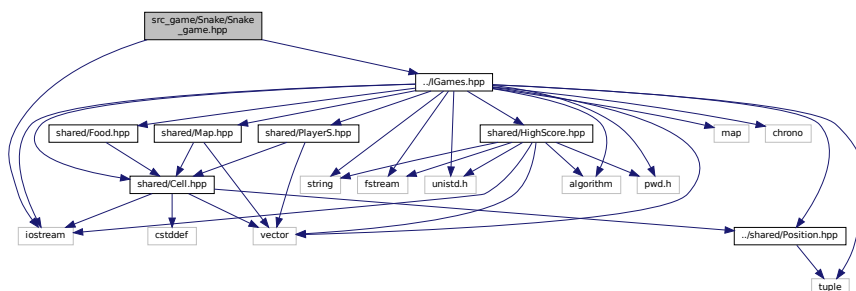
```
void destroy (
    IGames * game )
```

### 5.18.1.4 getName()

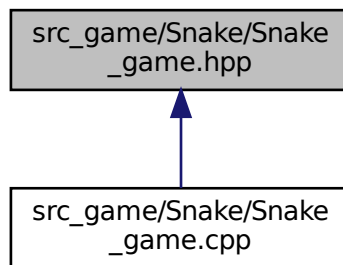
```
std::string getName ( )
```

## 5.19 src\_game/Snake/Snake\_game.hpp File Reference

```
#include "../IGames.hpp"
#include <iostream>
Include dependency graph for Snake_game.hpp:
```



This graph shows which files directly or indirectly include this file:



## Classes

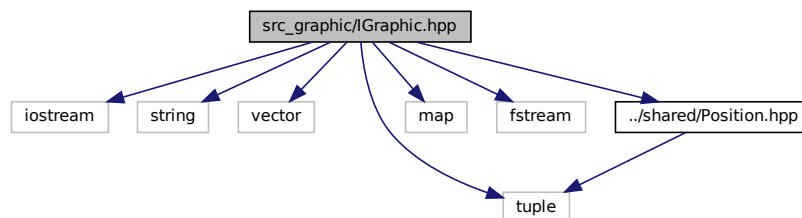
- class [Snake](#)

## 5.20 src\_graphic/IGraphic.hpp File Reference

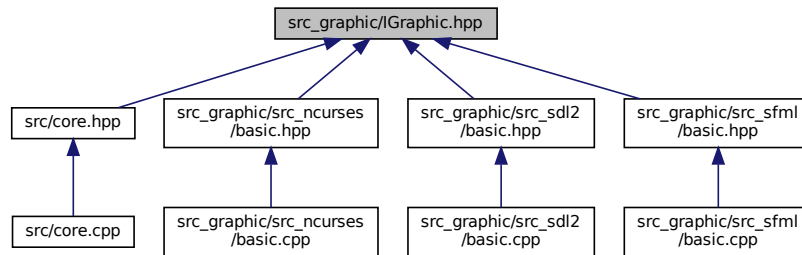
Header file for the [IGraphic](#) interface.

```
#include <iostream>
#include <string>
#include <vector>
#include <tuple>
#include <map>
#include <fstream>
#include "../shared/Position.hpp"
```

Include dependency graph for IGraphic.hpp:



This graph shows which files directly or indirectly include this file:



## Classes

- class [IGraphic](#)  
*Interface for handling game assets.*

### 5.20.1 Detailed Description

Header file for the [IGraphic](#) interface.

#### Author

Titien Carellas

#### Version

1.0

#### Date

2023-04-03

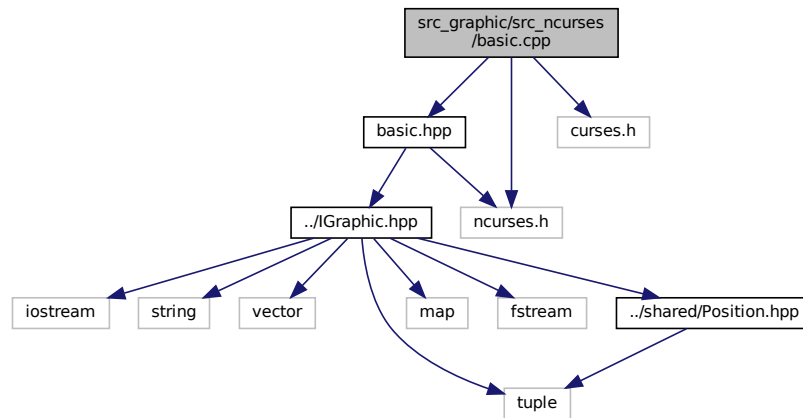
This file defines the [IGraphic](#) interface that provides the necessary functions for loading, displaying, and handling game assets.

## 5.21 src\_graphic/src\_ncurses/basic.cpp File Reference

```
#include "basic.hpp"
#include <curses.h>
```

```
#include <ncurses.h>
```

Include dependency graph for basic.cpp:



## Macros

- `#define RECT_SIZE Ncurses::get_screen_x() / 1.5 / 20 / 1.5`
- `#define OFFSET_X (Ncurses::get_screen_x() - (20 * RECT_SIZE)) / 2`
- `#define OFFSET_Y Ncurses::get_screen_y() / 4`

## Functions

- `IGraphic * createGraphicInstance ()`
- `std::string getName ()`
- `void destroy (IGraphic *graphic)`

### 5.21.1 Macro Definition Documentation

#### 5.21.1.1 OFFSET\_X

```
#define OFFSET_X (Ncurses::get_screen_x() - (20 * RECT_SIZE)) / 2
```

#### 5.21.1.2 OFFSET\_Y

```
#define OFFSET_Y Ncurses::get_screen_y() / 4
```

### 5.21.1.3 RECT\_SIZE

```
#define RECT_SIZE Ncurses::get_screen_x() / 1.5 / 20 / 1.5
```

## 5.21.2 Function Documentation

### 5.21.2.1 createGraphicInstance()

```
IGraphic* createGraphicInstance ( )
```

### 5.21.2.2 destroy()

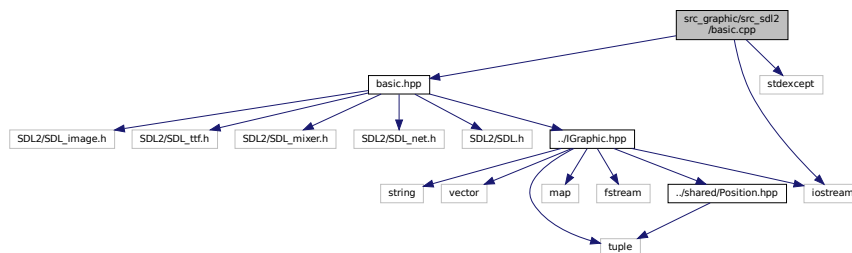
```
void destroy (
    IGraphic * graphic )
```

### 5.21.2.3 getName()

```
std::string getName ( )
```

## 5.22 src\_graphic/src\_sdl2/basic.cpp File Reference

```
#include "basic.hpp"
#include <iostream>
#include <stdexcept>
Include dependency graph for basic.cpp:
```





## Functions

- `IGraphic * create ()`
- `std::string getName ()`
- `void sdl2_load_assets (std::vector< std::tuple< char, std::string, int >> asset_toload)`

### 5.22.1 Function Documentation

#### 5.22.1.1 create()

```
IGraphic* create ( )
```

#### 5.22.1.2 getName()

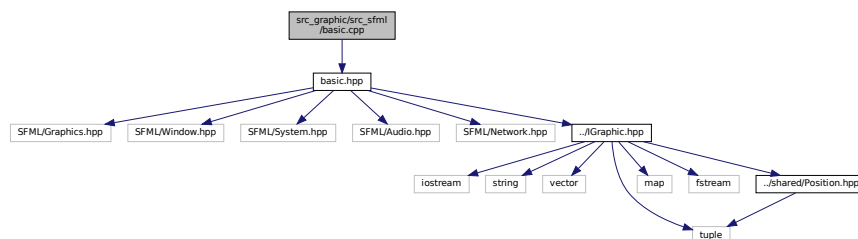
```
std::string getName ( )
```

#### 5.22.1.3 sdl2\_load\_assets()

```
void sdl2_load_assets (
    std::vector< std::tuple< char, std::string, int >> asset_toload )
```

## 5.23 src\_graphic/src\_sfml/basic.cpp File Reference

```
#include "basic.hpp"
Include dependency graph for basic.cpp:
```



## Macros

- `#define RECT_SIZE _window.getSize().x / 1.5 / 20 / 1.5`
- `#define OFFSET_X (_window.getSize().x - (20 * RECT_SIZE)) / 2`
- `#define OFFSET_Y _window.getSize().y / 4`
- `#define CD_SIZE _window.getSize().x / 7.5`

## Functions

- `IGraphic * createGraphicInstance ()`
- `std::string getName ()`
- `void destroy (IGraphic *graphic)`

## 5.23.1 Macro Definition Documentation

### 5.23.1.1 CD\_SIZE

```
#define CD_SIZE _window.getSize().x / 7.5
```

### 5.23.1.2 OFFSET\_X

```
#define OFFSET_X (_window.getSize().x - (20 * RECT_SIZE) ) / 2
```

### 5.23.1.3 OFFSET\_Y

```
#define OFFSET_Y _window.getSize().y / 4
```

### 5.23.1.4 RECT\_SIZE

```
#define RECT_SIZE _window.getSize().x / 1.5 / 20 / 1.5
```

## 5.23.2 Function Documentation

### 5.23.2.1 createGraphicInstance()

```
IGraphic* createGraphicInstance ( )
```

## 5.23.2.2 destroy()

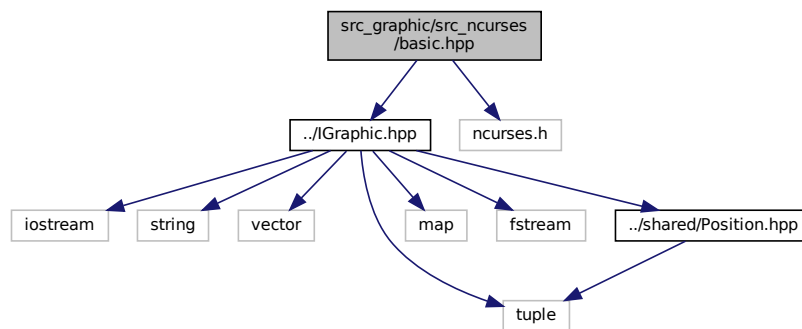
```
void destroy (
    IGraphic * graphic )
```

## 5.23.2.3 getName()

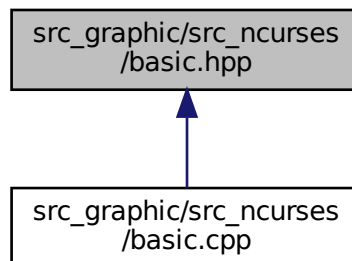
```
std::string getName ( )
```

## 5.24 src\_graphic/src\_ncurses/basic.hpp File Reference

```
#include "../IGraphic.hpp"
#include <ncurses.h>
Include dependency graph for basic.hpp:
```



This graph shows which files directly or indirectly include this file:



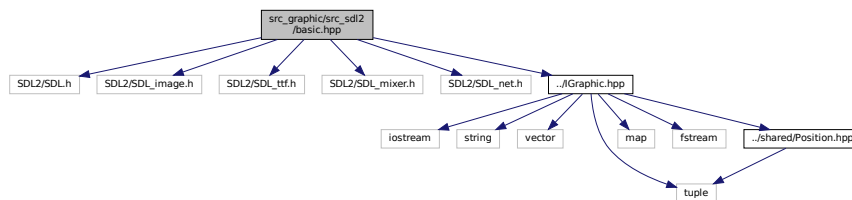
## Classes

- class [Ncurses](#)

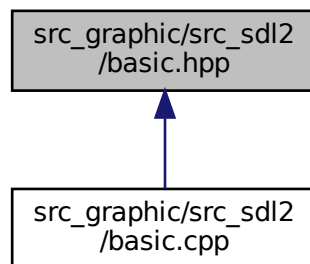
## 5.25 src\_graphic/src\_sdl2/basic.hpp File Reference

```
#include <SDL2/SDL.h>
#include <SDL2/SDL_image.h>
#include <SDL2/SDL_ttf.h>
#include <SDL2/SDL_mixer.h>
#include <SDL2/SDL_net.h>
#include "../IGraphic.hpp"
```

Include dependency graph for basic.hpp:



This graph shows which files directly or indirectly include this file:

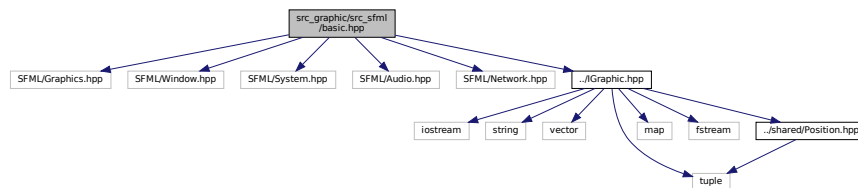


## Classes

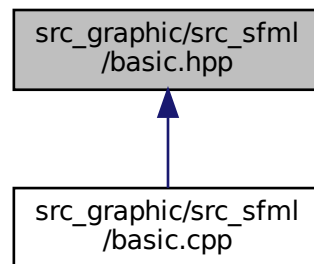
- class [Sdl2](#)

## 5.26 src\_graphic/src\_sfml/basic.hpp File Reference

```
#include <SFML/Graphics.hpp>
#include <SFML/Window.hpp>
#include <SFML/System.hpp>
#include <SFML/Audio.hpp>
#include <SFML/Network.hpp>
#include "../IGraphic.hpp"
Include dependency graph for basic.hpp:
```



This graph shows which files directly or indirectly include this file:



### Classes

- class [Sfml](#)



# Index

- [\\_active](#)
    - [Sdl2, 66](#)
    - [Sfml, 75](#)
  - [\\_assets](#)
    - [Sdl2, 66](#)
    - [Sfml, 75](#)
  - [\\_clock\\_end](#)
    - [core, 17](#)
    - [Nibbler, 52](#)
  - [\\_clock\\_end\\_fixed](#)
    - [core, 17](#)
  - [\\_clock\\_start](#)
    - [core, 17](#)
    - [Nibbler, 52](#)
  - [\\_clock\\_start\\_fixed](#)
    - [core, 18](#)
  - [\\_configData](#)
    - [Nibbler, 52](#)
    - [Sfml, 75](#)
    - [Snake, 84](#)
  - [\\_configMenuChoice](#)
    - [core, 18](#)
  - [\\_currentGameLibraryPath](#)
    - [core, 18](#)
  - [\\_currentGraphicLibrary](#)
    - [core, 18](#)
  - [\\_event](#)
    - [Sdl2, 66](#)
    - [Sfml, 75](#)
  - [\\_food](#)
    - [Nibbler, 53](#)
    - [Snake, 84](#)
  - [\\_game](#)
    - [core, 18](#)
  - [\\_gameLibraries](#)
    - [core, 18](#)
  - [\\_graphicLibraries](#)
    - [core, 19](#)
  - [\\_graphical](#)
    - [core, 19](#)
  - [\\_highScore](#)
    - [Nibbler, 53](#)
    - [Snake, 84](#)
  - [\\_highscore](#)
    - [Nibbler, 53](#)
    - [Snake, 84](#)
  - [\\_key](#)
    - [core, 19](#)
  - [\\_level](#)
    - [Nibbler, 53](#)
    - [Snake, 84](#)
  - [\\_map](#)
    - [Map, 36](#)
    - [Nibbler, 53](#)
    - [Snake, 84](#)
  - [\\_menu](#)
    - [Sdl2, 66](#)
    - [Sfml, 75](#)
  - [\\_menustate](#)
    - [core, 19](#)
  - [\\_music](#)
    - [Sfml, 75](#)
  - [\\_optionsConfig](#)
    - [core, 19](#)
  - [\\_optionsMenu](#)
    - [core, 19](#)
  - [\\_optionsText](#)
    - [Sfml, 76](#)
  - [\\_pos](#)
    - [Cell, 10](#)
  - [\\_renderer](#)
    - [Sdl2, 66](#)
  - [\\_rotation](#)
    - [Position, 61](#)
  - [\\_score](#)
    - [Nibbler, 53](#)
    - [Snake, 85](#)
  - [\\_skinMultiplier](#)
    - [Nibbler, 53](#)
    - [Snake, 85](#)
  - [\\_snake](#)
    - [Nibbler, 53](#)
    - [Snake, 85](#)
  - [\\_sound\\_menu](#)
    - [Sfml, 76](#)
  - [\\_speed](#)
    - [Nibbler, 54](#)
    - [Snake, 85](#)
  - [\\_state](#)
    - [core, 20](#)
  - [\\_string](#)
    - [Nibbler, 54](#)
    - [Sdl2, 67](#)
    - [Sfml, 76](#)
    - [Snake, 85](#)
  - [\\_window](#)
    - [Sdl2, 67](#)
    - [Sfml, 76](#)

- `_x`
  - Position, [61](#)
- `_y`
  - Position, [61](#)
- `~Cell`
  - Cell, [8](#)
- `~IGames`
  - IGames, [24](#)
- `~IGraphic`
  - IGraphic, [30](#)
- `~Map`
  - Map, [35](#)
- `~Ncurses`
  - Ncurses, [38](#)
- `~Nibbler`
  - Nibbler, [46](#)
- `~PlayerS`
  - PlayerS, [55](#)
- `~Position`
  - Position, [58](#)
- `~Sdl2`
  - Sdl2, [63](#)
- `~Sfml`
  - Sfml, [69](#)
- `~Snake`
  - Snake, [79](#)
- `~core`
  - core, [13](#)
- `addTime`
  - Nibbler, [47](#)
- `autoMove`
  - Nibbler, [47](#)
- `basic.cpp`
  - CD\_SIZE, [110](#)
  - create, [109](#)
  - createGraphicInstance, [108](#), [110](#)
  - destroy, [108](#), [110](#)
  - getName, [108](#), [109](#), [111](#)
  - OFFSET\_X, [107](#), [110](#)
  - OFFSET\_Y, [107](#), [110](#)
  - RECT\_SIZE, [107](#), [110](#)
  - sdl2\_load\_assets, [109](#)
- `CD_SIZE`
  - basic.cpp, [110](#)
- `Cell, 7`
  - \_pos, [10](#)
  - ~Cell, [8](#)
  - Cell, [8](#)
  - checkCollision, [8](#)
  - getPos, [9](#)
  - getRotation, [9](#)
  - getX, [9](#)
  - getY, [9](#)
  - setPos, [9](#)
  - setRotation, [9](#)
- `check_time`
  - core, [13](#)
- `checkCollision`
  - Cell, [8](#)
  - PlayerS, [55](#)
- `checkEatHimself`
  - PlayerS, [55](#)
- `collisionFood`
  - IGames, [25](#)
  - Nibbler, [47](#)
  - Snake, [79](#)
- `collisionMap`
  - IGames, [26](#)
  - Nibbler, [48](#)
  - Snake, [79](#)
- `CONFIG`
  - core.hpp, [92](#)
- `core, 10`
  - \_clock\_end, [17](#)
  - \_clock\_end\_fixed, [17](#)
  - \_clock\_start, [17](#)
  - \_clock\_start\_fixed, [18](#)
  - \_configMenuChoice, [18](#)
  - \_currentGameLibraryPath, [18](#)
  - \_currentGraphicLibrary, [18](#)
  - \_game, [18](#)
  - \_gameLibraries, [18](#)
  - \_graphicLibraries, [19](#)
  - \_graphical, [19](#)
  - \_key, [19](#)
  - \_menustate, [19](#)
  - \_optionsConfig, [19](#)
  - \_optionsMenu, [19](#)
  - \_state, [20](#)
  - ~core, [13](#)
  - check\_time, [13](#)
  - core, [13](#)
  - destroyGame, [20](#)
  - destroyGraphic, [20](#)
  - handleInputConfig, [14](#)
  - handleInputGame, [14](#)
  - handleInputMenu, [14](#)
  - initConfig, [14](#)
  - initGame, [14](#)
  - initMenu, [15](#)
  - loadGame, [15](#)
  - loadGraphical, [15](#)
  - loadLibrariesFromDirectory, [15](#)
  - loop, [16](#)
  - loopConfig, [16](#)
  - loopEnd, [16](#)
  - loopGame, [16](#)
  - loopHighScore, [16](#)
  - loopMenu, [16](#)
  - switchGraphics, [17](#)
  - writeConfig, [17](#)
- `core.cpp`
  - main, [89](#)
- `core.hpp`



- CONFIG, [92](#)
- END, [92](#)
- GAME, [92](#)
- GetIdFuncPtr, [91](#)
- HIGH\_SCORE, [92](#)
- KEY\_BACKSPACE, [92](#)
- KEY\_DOWN, [92](#)
- KEY\_ENTER, [92](#)
- KEY\_ESCAPE, [92](#)
- KEY\_LEFT, [92](#)
- KEY\_RIGHT, [92](#)
- KEY\_UP, [92](#)
- keys\_e, [92](#)
- keys\_t, [91](#)
- LibGameDeletePtr, [91](#)
- LibGameFuncPtr, [91](#)
- LibGraphicDeletePtr, [91](#)
- LibGraphicFuncPtr, [91](#)
- MENU, [92](#)
- state\_e, [92](#)
- state\_t, [92](#)
- UNDEFINED, [92](#)
- create
  - basic.cpp, [109](#)
  - Nibbler\_game.cpp, [94](#)
  - Snake\_game.cpp, [104](#)
- createGameInstance
  - Nibbler\_game.cpp, [94](#)
  - Snake\_game.cpp, [104](#)
- createGraphicInstance
  - basic.cpp, [108](#), [110](#)
- debug\_print
  - IGraphic, [30](#)
  - Ncurses, [38](#)
  - Sdl2, [63](#)
  - Sfml, [69](#)
- deleteAssets
  - IGraphic, [31](#)
  - Ncurses, [38](#)
  - Sfml, [69](#)
- destroy
  - basic.cpp, [108](#), [110](#)
  - Nibbler\_game.cpp, [94](#)
  - Snake\_game.cpp, [104](#)
- destroyGame
  - core, [20](#)
- destroyGraphic
  - core, [20](#)
- direction\_t
  - Position.hpp, [88](#)
- display
  - Ncurses, [38](#), [39](#)
- display\_asset
  - IGraphic, [31](#)
  - Ncurses, [39](#)
  - Sdl2, [63](#)
  - Sfml, [70](#)
- display\_high\_score
  - IGraphic, [31](#)
  - Ncurses, [39](#)
  - Sfml, [70](#)
- display\_menu
  - IGraphic, [31](#)
  - Ncurses, [40](#)
  - Sdl2, [64](#)
  - Sfml, [70](#)
- display\_separation\_x
  - Ncurses, [40](#)
- display\_separation\_y
  - Ncurses, [40](#)
- display\_text
  - IGraphic, [32](#)
  - Ncurses, [40](#)
  - Sdl2, [64](#)
  - Sfml, [71](#)
- displayConfig
  - IGraphic, [32](#)
  - Ncurses, [41](#)
  - Sfml, [71](#)
- DOWN
  - Position.hpp, [89](#)
- END
  - core.hpp, [92](#)
- extractScore
  - HighScore.cpp, [99](#)
- Food, [21](#)
  - Food, [21](#), [22](#)
  - respawn, [22](#)
- GAME
  - core.hpp, [92](#)
- get\_index\_tuple
  - IGraphic, [33](#)
  - Ncurses, [41](#)
  - Sdl2, [64](#)
  - Sfml, [71](#)
- get\_key
  - IGraphic, [33](#)
  - Ncurses, [41](#)
  - Sdl2, [64](#)
  - Sfml, [72](#)
- get\_screen\_x
  - Ncurses, [42](#)
- get\_screen\_y
  - Ncurses, [42](#)
- getBodyPos
  - PlayerS, [55](#)
- getConfigData
  - Sfml, [72](#)
- getElements
  - IGames, [26](#)
  - Nibbler, [48](#)
  - Snake, [80](#)
- getHeadPos
  - PlayerS, [55](#)

- GetIdFuncPtr
  - core.hpp, 91
- getIndexString
  - Nibbler, 48
- getLogin
  - HighScore, 22
  - Nibbler, 48
  - Snake, 80
- getMap
  - Map, 35
- getName
  - basic.cpp, 108, 109, 111
  - Nibbler\_game.cpp, 94
  - Snake\_game.cpp, 104
- getPos
  - Cell, 9
  - Position, 59
- getPosition
  - Map, 36
- getPositionsVector
  - Map, 36
- getPosPair
  - Position, 59
- getRemainingTime
  - Nibbler, 49
- getRotation
  - Cell, 9
  - PlayerS, 56
  - Position, 59
- getScore
  - IGames, 26
  - Nibbler, 49
  - Snake, 80
- getSize
  - PlayerS, 56
- getSnakePosVector
  - PlayerS, 56
- getSnakeTailPosVector
  - PlayerS, 56
- getSpeed
  - IGames, 26
  - Nibbler, 49
  - Snake, 80
- getTailPos
  - PlayerS, 56
- getTop10Reverse
  - HighScore, 23
- getX
  - Cell, 9
  - Position, 59
- getY
  - Cell, 9
  - Position, 60
- grow
  - PlayerS, 56
- handleCollision
  - IGames, 27
  - Nibbler, 49
  - Snake, 80
- handleInput
  - IGames, 27
  - Nibbler, 50
  - Snake, 81
- handleInputConfig
  - core, 14
- handleInputGame
  - core, 14
- handleInputMenu
  - core, 14
- HIGH\_SCORE
  - core.hpp, 92
- HighScore, 22
  - getLogin, 22
  - getTop10Reverse, 23
  - readScore, 23
  - writeScore, 23
- HighScore.cpp
  - extractScore, 99
- HighScoresIsSet
  - Nibbler, 54
  - Snake, 85
- IGames, 23
  - ~IGames, 24
  - collisionFood, 25
  - collisionMap, 26
  - getElements, 26
  - getScore, 26
  - getSpeed, 26
  - handleCollision, 27
  - handleInput, 27
  - init, 27
  - loadAssets, 27
  - loadTexts, 28
  - parse\_map, 28
  - readConfig, 28
  - readScore, 28
  - update, 28
  - updateTexts, 29
  - writeScore, 29
- IGraphic, 29
  - ~IGraphic, 30
  - debug\_print, 30
  - deleteAssets, 31
  - display\_asset, 31
  - display\_high\_score, 31
  - display\_menu, 31
  - display\_text, 32
  - displayConfig, 32
  - get\_index\_tuple, 33
  - get\_key, 33
  - load\_assets, 33
  - load\_config, 34
  - load\_menu, 34
  - load\_text, 34
- init
  - IGames, 27

- Ncurses, [42](#)
- Nibbler, [50](#)
- Sdl2, [65](#)
- Sfml, [72](#)
- Snake, [82](#)
- initConfig
  - core, [14](#)
- initGame
  - core, [14](#)
- initMenu
  - core, [15](#)
- initTexts
  - Nibbler, [50](#)
  - Snake, [82](#)
- isConfirmSoundPlaying
  - Sfml, [72](#)
- KEY\_BACKSPACE
  - core.hpp, [92](#)
- KEY\_DOWN
  - core.hpp, [92](#)
- KEY\_ENTER
  - core.hpp, [92](#)
- KEY\_ESCAPE
  - core.hpp, [92](#)
- KEY\_LEFT
  - core.hpp, [92](#)
- KEY\_RIGHT
  - core.hpp, [92](#)
- KEY\_UP
  - core.hpp, [92](#)
- keys\_e
  - core.hpp, [92](#)
- keys\_t
  - core.hpp, [91](#)
- LEFT
  - Position.hpp, [89](#)
- LibGameDeletePtr
  - core.hpp, [91](#)
- LibGameFuncPtr
  - core.hpp, [91](#)
- LibGraphicDeletePtr
  - core.hpp, [91](#)
- LibGraphicFuncPtr
  - core.hpp, [91](#)
- load\_assets
  - IGraphic, [33](#)
  - Ncurses, [42](#)
  - Sdl2, [65](#)
  - Sfml, [73](#)
- load\_background\_menu
  - Sfml, [73](#)
- load\_config
  - IGraphic, [34](#)
  - Ncurses, [42](#)
  - Sfml, [73](#)
- load\_menu
  - IGraphic, [34](#)
- Ncurses, [43](#)
- Sdl2, [65](#)
- Sfml, [74](#)
- load\_text
  - IGraphic, [34](#)
  - Ncurses, [43](#)
  - Sdl2, [66](#)
  - Sfml, [74](#)
- loadAssets
  - IGames, [27](#)
  - Nibbler, [50](#)
  - Snake, [82](#)
- loadGame
  - core, [15](#)
- loadGraphical
  - core, [15](#)
- loadLibrariesFromDirectory
  - core, [15](#)
- loadTexts
  - IGames, [28](#)
  - Nibbler, [50](#)
  - Snake, [82](#)
- loop
  - core, [16](#)
- loopConfig
  - core, [16](#)
- loopEnd
  - core, [16](#)
- loopGame
  - core, [16](#)
- loopHighScore
  - core, [16](#)
- loopMenu
  - core, [16](#)
- main
  - core.cpp, [89](#)
- Map, [35](#)
  - \_map, [36](#)
  - ~Map, [35](#)
  - getMap, [35](#)
  - getPosition, [36](#)
  - getPositionsVector, [36](#)
  - Map, [35](#)
- MENU
  - core.hpp, [92](#)
- move
  - PlayerS, [56](#)
- Ncurses, [36](#)
  - ~Ncurses, [38](#)
  - debug\_print, [38](#)
  - deleteAssets, [38](#)
  - display, [38](#), [39](#)
  - display\_asset, [39](#)
  - display\_high\_score, [39](#)
  - display\_menu, [40](#)
  - display\_separation\_x, [40](#)
  - display\_separation\_y, [40](#)

- display\_text, 40
- displayConfig, 41
- get\_index\_tuple, 41
- get\_key, 41
- get\_screen\_x, 42
- get\_screen\_y, 42
- init, 42
- load\_assets, 42
- load\_config, 42
- load\_menu, 43
- load\_text, 43
- Ncurses, 38
- window\_clear, 43
- window\_refresh, 43
- Nibbler, 44
  - \_clock\_end, 52
  - \_clock\_start, 52
  - \_configData, 52
  - \_food, 53
  - \_highScore, 53
  - \_highscore, 53
  - \_level, 53
  - \_map, 53
  - \_score, 53
  - \_skinMultiplier, 53
  - \_snake, 53
  - \_speed, 54
  - \_string, 54
  - ~Nibbler, 46
  - addTime, 47
  - autoMove, 47
  - collisionFood, 47
  - collisionMap, 48
  - getElements, 48
  - getIndexString, 48
  - getLogin, 48
  - getRemainingTime, 49
  - getScore, 49
  - getSpeed, 49
  - handleCollision, 49
  - handleInput, 50
  - HighScoreIsSet, 54
  - init, 50
  - initTexts, 50
  - loadAssets, 50
  - loadTexts, 50
  - Nibbler, 46
  - parse\_map, 51
  - readConfig, 51
  - readScore, 51
  - TimeOut, 51
  - update, 51
  - updateTexts, 52
  - writeScore, 52
- Nibbler\_game.cpp
  - create, 94
  - createGameInstance, 94
  - destroy, 94
  - getName, 94
- NONE
  - Position.hpp, 89
- OFFSET\_X
  - basic.cpp, 107, 110
- OFFSET\_Y
  - basic.cpp, 107, 110
- parse\_map
  - IGames, 28
  - Nibbler, 51
  - Snake, 82
- PlayerS, 54
  - ~PlayerS, 55
  - checkCollision, 55
  - checkEatHimself, 55
  - getBodyPos, 55
  - getHeadPos, 55
  - getRotation, 56
  - getSize, 56
  - getSnakePosVector, 56
  - getSnakeTailPosVector, 56
  - getTailPos, 56
  - grow, 56
  - move, 56
  - PlayerS, 55
  - setRotation, 56
- Position, 57
  - \_rotation, 61
  - \_x, 61
  - \_y, 61
  - ~Position, 58
  - getPos, 59
  - getPosPair, 59
  - getRotation, 59
  - getX, 59
  - getY, 60
  - Position, 58
  - setRotation, 60
  - setX, 60
  - setY, 60
- Position.hpp
  - direction\_t, 88
  - DOWN, 89
  - LEFT, 89
  - NONE, 89
  - RIGHT, 89
  - UP, 89
- readConfig
  - IGames, 28
  - Nibbler, 51
  - Snake, 82
- readScore
  - HighScore, 23
  - IGames, 28
  - Nibbler, 51
  - Snake, 83

- RECT\_SIZE
  - basic.cpp, [107](#), [110](#)
- respawn
  - Food, [22](#)
- reverseMusic
  - Sfml, [74](#)
- RIGHT
  - Position.hpp, [89](#)
- Sdl2, [62](#)
  - \_active, [66](#)
  - \_assets, [66](#)
  - \_event, [66](#)
  - \_menu, [66](#)
  - \_renderer, [66](#)
  - \_string, [67](#)
  - \_window, [67](#)
  - ~Sdl2, [63](#)
  - debug\_print, [63](#)
  - display\_asset, [63](#)
  - display\_menu, [64](#)
  - display\_text, [64](#)
  - get\_index\_tuple, [64](#)
  - get\_key, [64](#)
  - init, [65](#)
  - load\_assets, [65](#)
  - load\_menu, [65](#)
  - load\_text, [66](#)
  - Sdl2, [63](#)
- sdl2\_load\_assets
  - basic.cpp, [109](#)
- setPos
  - Cell, [9](#)
- setRotation
  - Cell, [9](#)
  - PlayerS, [56](#)
  - Position, [60](#)
- setX
  - Position, [60](#)
- setY
  - Position, [60](#)
- Sfml, [67](#)
  - \_active, [75](#)
  - \_assets, [75](#)
  - \_configData, [75](#)
  - \_event, [75](#)
  - \_menu, [75](#)
  - \_music, [75](#)
  - \_optionsText, [76](#)
  - \_sound\_menu, [76](#)
  - \_string, [76](#)
  - \_window, [76](#)
  - ~Sfml, [69](#)
  - debug\_print, [69](#)
  - deleteAssets, [69](#)
  - display\_asset, [70](#)
  - display\_high\_score, [70](#)
  - display\_menu, [70](#)
  - display\_text, [71](#)
  - displayConfig, [71](#)
  - get\_index\_tuple, [71](#)
  - get\_key, [72](#)
  - getConfigData, [72](#)
  - init, [72](#)
  - isConfirmSoundPlaying, [72](#)
  - load\_assets, [73](#)
  - load\_background\_menu, [73](#)
  - load\_config, [73](#)
  - load\_menu, [74](#)
  - load\_text, [74](#)
  - reverseMusic, [74](#)
  - Sfml, [69](#)
- shared/Position.cpp, [87](#)
- shared/Position.hpp, [87](#)
- Snake, [77](#)
  - \_configData, [84](#)
  - \_food, [84](#)
  - \_highScore, [84](#)
  - \_highscore, [84](#)
  - \_level, [84](#)
  - \_map, [84](#)
  - \_score, [85](#)
  - \_skinMultiplier, [85](#)
  - \_snake, [85](#)
  - \_speed, [85](#)
  - \_string, [85](#)
  - ~Snake, [79](#)
  - collisionFood, [79](#)
  - collisionMap, [79](#)
  - getElements, [80](#)
  - getLogin, [80](#)
  - getScore, [80](#)
  - getSpeed, [80](#)
  - handleCollision, [80](#)
  - handleInput, [81](#)
  - HighScoresSet, [85](#)
  - init, [82](#)
  - initTexts, [82](#)
  - loadAssets, [82](#)
  - loadTexts, [82](#)
  - parse\_map, [82](#)
  - readConfig, [82](#)
  - readScore, [83](#)
  - Snake, [78](#)
  - update, [83](#)
  - updateTexts, [83](#)
  - writeScore, [83](#)
- Snake\_game.cpp
  - create, [104](#)
  - createGameInstance, [104](#)
  - destroy, [104](#)
  - getName, [104](#)
- src/core.cpp, [89](#)
- src/core.hpp, [90](#)
- src\_game/IGames.hpp, [93](#)
- src\_game/Nibbler/Nibbler\_game.cpp, [94](#)
- src\_game/Nibbler/Nibbler\_game.hpp, [95](#)

- src\_game/shared/Cell.cpp, [96](#)
- src\_game/shared/Cell.hpp, [96](#)
- src\_game/shared/Food.cpp, [97](#)
- src\_game/shared/Food.hpp, [98](#)
- src\_game/shared/HighScore.cpp, [99](#)
- src\_game/shared/HighScore.hpp, [99](#)
- src\_game/shared/Map.cpp, [100](#)
- src\_game/shared/Map.hpp, [101](#)
- src\_game/shared/PlayerS.cpp, [102](#)
- src\_game/shared/PlayerS.hpp, [102](#)
- src\_game/Snake/Snake\_game.cpp, [103](#)
- src\_game/Snake/Snake\_game.hpp, [104](#)
- src\_graphic/IGraphic.hpp, [105](#)
- src\_graphic/src\_ncurses/basic.cpp, [106](#)
- src\_graphic/src\_ncurses/basic.hpp, [111](#)
- src\_graphic/src\_sdl2/basic.cpp, [108](#)
- src\_graphic/src\_sdl2/basic.hpp, [112](#)
- src\_graphic/src\_sfml/basic.cpp, [109](#)
- src\_graphic/src\_sfml/basic.hpp, [113](#)
- state\_e
  - core.hpp, [92](#)
- state\_t
  - core.hpp, [92](#)
- switchGraphics
  - core, [17](#)
- TimeOut
  - Nibbler, [51](#)
- UNDEFINED
  - core.hpp, [92](#)
- UP
  - Position.hpp, [89](#)
- update
  - IGames, [28](#)
  - Nibbler, [51](#)
  - Snake, [83](#)
- updateTexts
  - IGames, [29](#)
  - Nibbler, [52](#)
  - Snake, [83](#)
- window\_clear
  - Ncurses, [43](#)
- window\_refresh
  - Ncurses, [43](#)
- writeConfig
  - core, [17](#)
- writeScore
  - HighScore, [23](#)
  - IGames, [29](#)
  - Nibbler, [52](#)
  - Snake, [83](#)