

Arcade

Generated by Doxygen 1.9.3

1 Hierarchical Index	1
1.1 Class Hierarchy	1
2 Class Index	3
2.1 Class List	3
3 File Index	5
3.1 File List	5
4 Class Documentation	7
4.1 arc::games::AGame Class Reference	7
4.1.1 Member Function Documentation	8
4.1.1.1 getObjects()	8
4.1.1.2 getScore()	8
4.1.1.3 isRunning()	8
4.2 arc::games::Centipede Class Reference	9
4.2.1 Member Function Documentation	9
4.2.1.1 getObjects()	9
4.2.1.2 splitSnake()	9
4.2.1.3 update()	10
4.2.1.4 useEvent()	10
4.3 arc::Color Struct Reference	10
4.3.1 Detailed Description	11
4.3.2 Constructor & Destructor Documentation	11
4.3.2.1 Color() [1/2]	11
4.3.2.2 Color() [2/2]	11
4.4 arc::Core Class Reference	12
4.4.1 Detailed Description	13
4.4.2 Member Function Documentation	13
4.4.2.1 getDisplay()	13
4.4.2.2 getDisplayName()	13
4.4.2.3 getGameName()	13
4.5 arc::DLLoader< T > Class Template Reference	14
4.5.1 Detailed Description	14
4.5.2 Constructor & Destructor Documentation	14
4.5.2.1 DLLoader()	14
4.5.3 Member Function Documentation	15
4.5.3.1 getInstance()	15
4.5.3.2 load()	15
4.5.3.3 operator->()	15
4.6 arc::Error Class Reference	16
4.6.1 Detailed Description	16
4.6.2 Constructor & Destructor Documentation	16

4.6.2.1 Error()	16
4.6.3 Member Function Documentation	17
4.6.3.1 what()	17
4.7 arc::utils::FileParser Class Reference	17
4.7.1 Detailed Description	17
4.7.2 Member Function Documentation	17
4.7.2.1 getAllLibraries()	17
4.7.2.2 getLibraryName()	18
4.7.2.3 isDisplayLibrary()	18
4.8 arc::games::Food Class Reference	19
4.8.1 Member Function Documentation	19
4.8.1.1 getClock()	19
4.8.1.2 getXpos()	19
4.8.1.3 getYpos()	19
4.9 arc::utils::HighscoreHandler Class Reference	20
4.9.1 Member Function Documentation	20
4.9.1.1 addHighscore()	20
4.9.1.2 getHighscores()	20
4.9.1.3 setHighscores()	21
4.10 arc::display::IDisplayModule Class Reference	21
4.10.1 Detailed Description	21
4.10.2 Member Function Documentation	22
4.10.2.1 drawInterface()	22
4.10.2.2 drawObjects()	22
4.10.2.3 getEvent()	22
4.11 arc::games::IGameModule Class Reference	23
4.11.1 Detailed Description	23
4.11.2 Member Function Documentation	23
4.11.2.1 getObjects() [1/2]	24
4.11.2.2 getObjects() [2/2]	24
4.11.2.3 isRunning()	24
4.11.2.4 update() [1/2]	24
4.11.2.5 update() [2/2]	25
4.11.2.6 useEvent() [1/2]	25
4.11.2.7 useEvent() [2/2]	25
4.12 arc::games::MenuGame Class Reference	25
4.12.1 Member Function Documentation	26
4.12.1.1 getObjects()	26
4.12.1.2 getProps()	27
4.12.1.3 isStarting()	27
4.12.1.4 update()	27
4.12.1.5 useEvent()	27

4.13 arc::games::menu::MenuItem Class Reference	28
4.13.1 Constructor & Destructor Documentation	28
4.13.1.1 MenuItem()	28
4.13.2 Member Function Documentation	29
4.13.2.1 isSelected()	29
4.13.2.2 setSelected()	29
4.14 arc::games::MenuProprieties Struct Reference	29
4.15 arc::games::centipede::Mushroom Class Reference	30
4.15.1 Member Function Documentation	30
4.15.1.1 getlife()	30
4.15.1.2 isDead()	31
4.15.1.3 setlife()	31
4.16 arc::display::NcursesDisplay Class Reference	31
4.16.1 Member Function Documentation	31
4.16.1.1 drawInterface()	32
4.16.1.2 drawObjects()	32
4.16.1.3 getEvent()	32
4.17 arc::games::NibblerGame Class Reference	32
4.17.1 Member Function Documentation	33
4.17.1.1 getObjects()	33
4.17.1.2 update()	33
4.17.1.3 useEvent()	33
4.18 arc::Object Class Reference	34
4.18.1 Detailed Description	35
4.18.2 Constructor & Destructor Documentation	35
4.18.2.1 Object()	35
4.18.3 Member Function Documentation	35
4.18.3.1 getPosition()	35
4.18.3.2 getType()	35
4.18.3.3 getValue()	36
4.18.3.4 setPosition()	36
4.18.3.5 setValue()	36
4.19 arc::games::centipede::Player Class Reference	36
4.19.1 Member Function Documentation	37
4.19.1.1 deleteShoot()	37
4.19.1.2 getShoots()	38
4.20 arc::display::Sdl2Display Class Reference	38
4.20.1 Member Function Documentation	39
4.20.1.1 drawInterface()	39
4.20.1.2 drawObjects()	39
4.20.1.3 getEvent()	39
4.20.1.4 placeObjectOnBoard()	40

4.21 arc::display::SfmlDisplay Class Reference	41
4.21.1 Member Function Documentation	41
4.21.1.1 drawInterface()	41
4.21.1.2 drawObjects()	42
4.21.1.3 getEvent()	42
4.21.1.4 placeObjectOnBoard()	42
4.22 arc::games::centipede::Shoot Class Reference	43
4.22.1 Constructor & Destructor Documentation	43
4.22.1.1 Shoot()	43
4.22.2 Member Function Documentation	44
4.22.2.1 checkHit()	44
4.22.2.2 getHit()	44
4.22.2.3 isHit()	44
4.23 arc::games::centipede::Snake Class Reference	45
4.23.1 Constructor & Destructor Documentation	45
4.23.1.1 Snake() [1/2]	45
4.23.1.2 Snake() [2/2]	46
4.23.2 Member Function Documentation	46
4.23.2.1 getCells()	46
4.24 arc::games::Snake Class Reference	46
4.24.1 Member Function Documentation	47
4.24.1.1 getBody()	47
4.24.1.2 getXpos()	47
4.24.1.3 getYpos()	47
4.24.1.4 hasPosition()	47
4.24.1.5 hasPrevPosition()	48
4.25 arc::games::centipede::SnakeCell Class Reference	48
4.25.1 Constructor & Destructor Documentation	49
4.25.1.1 SnakeCell()	49
4.25.2 Member Function Documentation	50
4.25.2.1 getCellType()	50
4.25.2.2 getDirection()	50
4.25.2.3 hasDownMushroom()	50
4.25.2.4 hasLeftMushroom()	51
4.25.2.5 hasRightMushroom()	51
4.25.2.6 hit()	51
4.25.2.7 pickADir()	52
4.25.2.8 pickASideDir()	52
4.25.2.9 setCellType()	52
4.25.2.10 setDirection()	53
4.26 arc::games::SnakeCell Class Reference	53
4.26.1 Member Function Documentation	53

4.26.1.1 getPrevXpos()	54
4.26.1.2 getPrevYpos()	54
4.26.1.3 getXpos()	54
4.26.1.4 getYpos()	54
4.27 arc::Sprite Class Reference	55
4.27.1 Detailed Description	55
4.27.2 Constructor & Destructor Documentation	55
4.27.2.1 Sprite()	55
4.27.3 Member Function Documentation	56
4.27.3.1 getHeight()	56
4.27.3.2 getScale()	56
4.27.3.3 getWidth()	56
4.27.3.4 setHeight()	56
4.27.3.5 setScale()	57
4.27.3.6 setWidth()	57
4.28 arc::Text Class Reference	57
4.28.1 Detailed Description	58
4.28.2 Constructor & Destructor Documentation	58
4.28.2.1 Text()	58
4.28.3 Member Function Documentation	58
4.28.3.1 getColor()	59
4.28.3.2 getSize()	59
4.28.3.3 setColor()	59
4.28.3.4 setSize()	59
4.29 arc::Vector Struct Reference	60
4.29.1 Detailed Description	60
5 File Documentation	61
5.1 Centipede.hpp	61
5.2 CentipedeGame.hpp	61
5.3 Mushroom.hpp	62
5.4 Player.hpp	62
5.5 Snake.hpp	63
5.6 Snake.hpp	64
5.7 AGame.hpp	64
5.8 Menu.hpp	65
5.9 MenuGame.hpp	65
5.10 MenuItem.hpp	66
5.11 Direction.hpp	66
5.12 Food.hpp	66
5.13 Nibbler.hpp	67
5.14 NibblerGame.hpp	67

5.15 SnakeCell.hpp	68
5.16 Ncurses.hpp	68
5.17 NcursesDisplay.hpp	68
5.18 Sdl2.hpp	69
5.19 Sdl2Display.hpp	69
5.20 SfmI.hpp	70
5.21 SfmIDisplay.hpp	70
5.22 Color.hpp	70
5.23 Core.hpp	71
5.24 Error.hpp	72
5.25 Events.hpp	72
5.26 IGameModule.hpp	73
5.27 IGameModule.hpp	73
5.28 IDisplayModule.hpp	73
5.29 Object.hpp	74
5.30 DLLoader.hpp	75
5.31 FileParser.hpp	76
5.32 HighscoreHandler.hpp	76
5.33 Vector.hpp	77
Index	79

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

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

arc::Color	10
arc::Core	12
arc::DLLoader< T >	14
arc::DLLoader< arc::display::IDisplayModule >	14
arc::DLLoader< arc::games::IGameModule >	14
std::exception	
arc::Error	16
arc::utils::FileParser	17
arc::games::Food	19
arc::utils::HighscoreHandler	20
arc::display::IDisplayModule	21
arc::display::NcursesDisplay	31
arc::display::Sdl2Display	38
arc::display::SfmlDisplay	41
arc::games::IGameModule	23
arc::games::AGame	7
arc::games::Centipede	9
arc::games::MenuGame	25
arc::games::NibblerGame	32
arc::games::MenuProprieties	29
arc::Object	34
arc::Sprite	55
arc::games::SnakeCell	53
arc::games::centipede::Mushroom	30
arc::games::centipede::Player	36
arc::games::centipede::Shoot	43
arc::games::centipede::SnakeCell	48
arc::Text	57
arc::games::menu::MenuItem	28
arc::games::centipede::Snake	45
arc::games::Snake	46
arc::Vector	60

Chapter 2

Class Index

2.1 Class List

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

arc::games::AGame	7
arc::games::Centipede	9
arc::Color	
Represents a color	10
arc::Core	
Arcade core, links both game and display libraries	12
arc::DLLoader< T >	
Loads shared libraries of games or displays	14
arc::Error	
General error class	16
arc::utils::FileParser	
Handles file manipulation	17
arc::games::Food	19
arc::utils::HighscoreHandler	20
arc::display::IDisplayModule	
Display module interface	21
arc::games::IGameModule	
Game module interface	23
arc::games::MenuGame	25
arc::games::menu::MenuItem	28
arc::games::MenuProprieties	29
arc::games::centipede::Mushroom	30
arc::display::NcursesDisplay	31
arc::games::NibblerGame	32
arc::Object	
Represents a drawable object	34
arc::games::centipede::Player	36
arc::display::Sdl2Display	38
arc::display::SfmlDisplay	41
arc::games::centipede::Shoot	43
arc::games::centipede::Snake	45
arc::games::Snake	46
arc::games::centipede::SnakeCell	48
arc::games::SnakeCell	53
arc::Sprite	
Represents a sprite object	55

arc::Text	
Represents a text object	57
arc::Vector	
Int vector	60

Chapter 3

File Index

3.1 File List

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

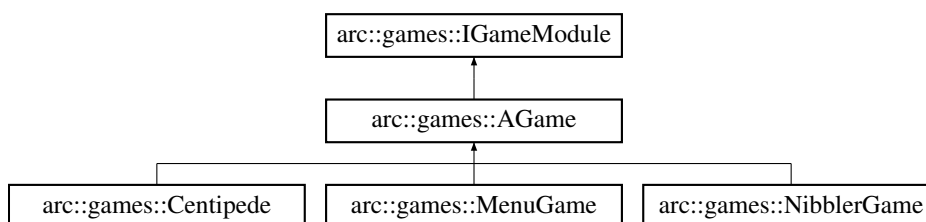
lib/games/centipede/includes/Centipede.hpp	61
lib/games/centipede/includes/CentipedeGame.hpp	61
lib/games/centipede/includes/Mushroom.hpp	62
lib/games/centipede/includes/Player.hpp	62
lib/games/centipede/includes/Snake.hpp	63
lib/games/includes/AGame.hpp	64
lib/games/menu/includes/Menu.hpp	65
lib/games/menu/includes/MenuGame.hpp	65
lib/games/menu/includes/MenuItem.hpp	66
lib/games/nibbler/includes/Direction.hpp	66
lib/games/nibbler/includes/Food.hpp	66
lib/games/nibbler/includes/Nibbler.hpp	67
lib/games/nibbler/includes/NibblerGame.hpp	67
lib/games/nibbler/includes/Snake.hpp	64
lib/games/nibbler/includes/SnakeCell.hpp	68
lib/graphics/ncurses/includes/Ncurses.hpp	68
lib/graphics/ncurses/includes/NcursesDisplay.hpp	68
lib/graphics/sdl2/includes/Sdl2.hpp	69
lib/graphics/sdl2/includes/Sdl2Display.hpp	69
lib/graphics/sfml/includes/Sfml.hpp	70
lib/graphics/sfml/includes/SfmlDisplay.hpp	70
src/includes/Color.hpp	70
src/includes/Core.hpp	71
src/includes/Error.hpp	72
src/includes/Events.hpp	72
src/includes/IGameModule.hpp	73
src/includes/Object.hpp	74
src/includes/Vector.hpp	77
src/includes/Interfaces/IDisplayModule.hpp	73
src/includes/Interfaces/IGameModule.hpp	73
src/includes/Utils/DLLoader.hpp	75
src/includes/Utils/FileParser.hpp	76
src/includes/Utils/HighscoreHandler.hpp	76

Chapter 4

Class Documentation

4.1 arc::games::AGame Class Reference

Inheritance diagram for arc::games::AGame:



Public Member Functions

- **AGame** (int score=0)
Construct a new [AGame](#) object.
- **~AGame** ()
Destroy the [AGame](#) object.
- int **getScore** () const
Get the score of the current game.
- bool **isRunning** () const override
Tells if game is still running or not.
- virtual const std::vector< std::shared_ptr< [arc::Object](#) > > **getObjects** () const override
Get the Objects object.

Protected Attributes

- int **m_score**
Current score.
- bool **m_isRunning**
Game state : running or not.
- std::vector< std::shared_ptr< [arc::Object](#) > > **m_objects**
All entities of the game.

4.1.1 Member Function Documentation

4.1.1.1 `getObjects()`

```
const std::vector< std::shared_ptr< arc::Object > > arc::games::AGame::getObjects ( ) const  
[override], [virtual]
```

Get the Objects object.

Returns

`const std::vector<std::shared_ptr<arc::Object>>`

Implements [arc::games::IGameModule](#).

Reimplemented in [arc::games::MenuGame](#), [arc::games::NibblerGame](#), and [arc::games::Centipede](#).

4.1.1.2 `getScore()`

```
int arc::games::AGame::getScore ( ) const
```

Get the score of the current game.

Returns

`int`

4.1.1.3 `isRunning()`

```
bool arc::games::AGame::isRunning ( ) const [override], [virtual]
```

Tells if game is still running or not.

Returns

`true or false`

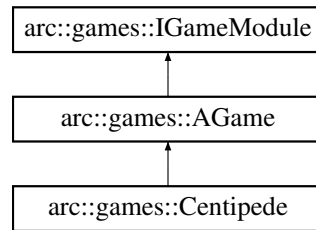
Implements [arc::games::IGameModule](#).

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

- `lib/games/includes/AGame.hpp`
- `lib/games/AGame.cpp`

4.2 arc::games::Centipede Class Reference

Inheritance diagram for arc::games::Centipede:



Public Member Functions

- **Centipede** ()
Construct a new [Centipede](#) object.
- **~Centipede** ()
Destroy the [Centipede](#) object.
- void **useEvent** (arc::Events event) override
Handle the given event.
- void **update** () override
Update the game entities.
- const std::vector< std::shared_ptr< [arc::Object](#) > > **getObjects** () const override
Get the Objects object.
- void **splitSnake** (std::shared_ptr< [arc::games::centipede::Snake](#) > snake, std::shared_ptr< [arc::games::centipede::SnakeCell](#) > cell)
Split the snake if it is hit by a shot.

Additional Inherited Members

4.2.1 Member Function Documentation

4.2.1.1 getObjects()

```
const std::vector< std::shared_ptr< arc::Object > > arc::games::Centipede::getObjects ( )
const [override], [virtual]
```

Get the Objects object.

Returns

```
const std::vector<std::shared_ptr<arc::Object>>
```

Reimplemented from [arc::games::AGame](#).

4.2.1.2 splitSnake()

```
void arc::games::Centipede::splitSnake (
    std::shared_ptr< arc::games::centipede::Snake > snake,
    std::shared_ptr< arc::games::centipede::SnakeCell > cell )
```

Split the snake if it is hit by a shot.

Parameters

<i>snake</i>	Snake to split
<i>cell</i>	Cell to split at

4.2.1.3 update()

```
void arc::games::Centipede::update ( ) [override], [virtual]
```

Update the game entities.

Implements [arc::games::IGameModule](#).

4.2.1.4 useEvent()

```
void arc::games::Centipede::useEvent (
    arc::Events event ) [override], [virtual]
```

Handle the given event.

Parameters

<i>event</i>	
--------------	--

Implements [arc::games::IGameModule](#).

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

- lib/games/centipede/includes/CentipedeGame.hpp
- lib/games/centipede/CentipedeGame.cpp

4.3 arc::Color Struct Reference

Represents a color.

```
#include <Color.hpp>
```

Public Types

- enum [ColorType](#) {
**RED , GREEN , BLUE , YELLOW ,
MAGENTA , CYAN , WHITE , BLACK }**
Default types.

Public Member Functions

- [Color](#) (uint8_t r, uint8_t g, uint8_t b, uint8_t a, [ColorType](#) color)
Construct a new [Color](#) object.
- [Color](#) ([ColorType](#) type)
Construct a new [Color](#) object.

Public Attributes

- uint8_t **r**
- uint8_t **g**
- uint8_t **b**
- uint8_t **a**
- [ColorType](#) **color**

4.3.1 Detailed Description

Represents a color.

4.3.2 Constructor & Destructor Documentation

4.3.2.1 [Color\(\)](#) [1/2]

```
arc::Color::Color (
    uint8_t r,
    uint8_t g,
    uint8_t b,
    uint8_t a,
    ColorType color )
```

Construct a new [Color](#) object.

Parameters

<i>r</i>	red value (0-255)
<i>g</i>	green value (0-255)
<i>b</i>	blue value (0-255)
<i>a</i>	opacity value (0-255)
<i>color</i>	

4.3.2.2 [Color\(\)](#) [2/2]

```
arc::Color::Color (
    ColorType type )
```

Construct a new [Color](#) object.

Parameters

<i>type</i>	Color type
-------------	----------------------------

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

- `src/includes/Color.hpp`
- `src/Color.cpp`

4.4 `arc::Core` Class Reference

Arcade core, links both game and display libraries.

```
#include <Core.hpp>
```

Public Member Functions

- **Core** (const std::string &lib)
Construct a new [Core](#) object.
- **~Core** ()
Destroy the [Core](#) object.
- std::unique_ptr< [arc::display::IDisplayModule](#) > [getDisplay](#) () const
Get the loaded display module.
- void **run** ()
starts the arcade machine
- const std::string & [getGameName](#) () const
Get the name of the loaded game.
- const std::string & [getDisplayName](#) () const
Get the name of the loaded display.
- bool **useEvent** (arc::Events event)
Handle the event.
- void **update** ()
Update the core.
- void **nextGame** ()
Switch to the next game.
- void **previousGame** ()
Switch to the previous game.
- void **nextDisplay** ()
Switch to the next display.
- void **previousDisplay** ()
Switch to the previous display.
- void **backToMenu** ()
Get back to menu.
- void **restartGame** ()
Restart the current game.

4.4.1 Detailed Description

Arcade core, links both game and display libraries.

4.4.2 Member Function Documentation

4.4.2.1 getDisplay()

```
std::unique_ptr< arc::display::IDisplayModule > arc::Core::getDisplay ( ) const
```

Get the loaded display module.

Returns

`std::unique_ptr<arc::display::IDisplayModule>`

4.4.2.2 getDisplayName()

```
const std::string & arc::Core::getDisplayName ( ) const
```

Get the name of the loaded display.

Returns

`const std::string&`

4.4.2.3 getGameName()

```
const std::string & arc::Core::getGameName ( ) const
```

Get the name of the loaded game.

Returns

`const std::string&`

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

- `src/includes/Core.hpp`
- `src/Core.cpp`

4.5 arc::DLLoader< T > Class Template Reference

Loads shared libraries of games or displays.

```
#include <DLLoader.hpp>
```

Public Member Functions

- **DLLoader** ()=default
Construct a new [DLLoader](#).
- **DLLoader** (const std::string &path)
- **DLLoader** ([DLLoader](#) &other)=delete
Unique pointer.
- **~DLLoader** ()
unload the library
- void **load** (const std::string &path)
Frees previous lib and loads a new one.
- void **free** ()
Free the currently loaded lib.
- T * **getInstance** () const
Get the loaded instance.
- T * **operator->** () const
Get the loaded instance.
- **DLLoader** & **operator=** ([DLLoader](#) &other)=delete
Unique pointer.

4.5.1 Detailed Description

```
template<class T>
class arc::DLLoader< T >
```

Loads shared libraries of games or displays.

Template Parameters

<i>T</i>	IGameModule or IDisplayModule
----------	-------------------------------

4.5.2 Constructor & Destructor Documentation

4.5.2.1 DLLoader()

```
template<class T >
arc::DLLoader< T >::DLLoader (
    const std::string & path ) [inline]
```

Parameters

<i>path</i>	path to the library to be loaded
-------------	----------------------------------

4.5.3 Member Function Documentation

4.5.3.1 getInstance()

```
template<class T >
T * arc::DLLoader< T >::getInstance ( ) const [inline]
```

Get the loaded instance.

Returns

Pointer to the loaded instance

4.5.3.2 load()

```
template<class T >
void arc::DLLoader< T >::load (
    const std::string & path ) [inline]
```

Frees previous lib and loads a new one.

Parameters

<i>path</i>	path to the new lib
-------------	---------------------

4.5.3.3 operator->()

```
template<class T >
T * arc::DLLoader< T >::operator-> ( ) const [inline]
```

Get the loaded instance.

Returns

Pointer to the loaded instance

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

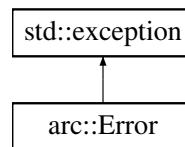
- src/includes/Utils/DLLoader.hpp

4.6 arc::Error Class Reference

General error class.

```
#include <Error.hpp>
```

Inheritance diagram for arc::Error:



Public Member Functions

- [Error](#) (const std::string &message)
Construct a new [Error](#) object.
- [~Error](#) ()
Destroy the [Error](#) object.
- const char * [what](#) () const noexcept final
Gets the error message.

Protected Attributes

- std::string **e_message**

4.6.1 Detailed Description

General error class.

4.6.2 Constructor & Destructor Documentation

4.6.2.1 Error()

```
arc::Error::Error (
    const std::string & message )
```

Construct a new [Error](#) object.

Parameters

<i>message</i>	error message
----------------	---------------

4.6.3 Member Function Documentation

4.6.3.1 what()

```
const char * arc::Error::what ( ) const [final], [noexcept]
```

Gets the error message.

Returns

const char* error message

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

- src/includes/Error.hpp
- src/Error.cpp

4.7 arc::utils::FileParser Class Reference

Handles file manipulation.

```
#include <FileParser.hpp>
```

Static Public Member Functions

- static std::string [getLibraryName](#) (const std::string &pathToLib)
Get the name of a .so arcade library.
- static std::vector< std::string > [getLibrariesNames](#) (const std::vector< std::string > libs)
Get the names of all libraries in a list.
- static std::array< std::vector< std::string >, 2 > [getAllLibraries](#) (const std::string &path="./lib/")
Get the all the available libraries.
- static bool [isDisplayLibrary](#) (const std::string libName)
Checks if given library is a display or not.

4.7.1 Detailed Description

Handles file manipulation.

4.7.2 Member Function Documentation

4.7.2.1 getAllLibraries()

```
std::array< std::vector< std::string >, 2 > arc::utils::FileParser::getAllLibraries (
    const std::string & path = "./lib/" ) [static]
```

Get the all the available libraries.

Parameters

<i>path</i>	path to the lib directory
-------------	---------------------------

Returns

`std::vector<std::string>`

4.7.2.2 getLibraryName()

```
std::string arc::utils::FileParser::getLibraryName (  
    const std::string & pathToLib )    [static]
```

Get the name of a .so arcade library.

Parameters

<i>pathToLib</i>	full path to the target library
------------------	---------------------------------

Returns

`std::string`

4.7.2.3 isDisplayLibrary()

```
bool arc::utils::FileParser::isDisplayLibrary (  
    const std::string libName )    [static]
```

Checks if given library is a display or not.

Parameters

<i>libName</i>	name of the library
----------------	---------------------

Returns

true if it is a display

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

- `src/includes/Utils/FileParser.hpp`
- `src/Utils/FileParser.cpp`

4.8 arc::games::Food Class Reference

Public Member Functions

- **Food** (std::vector< std::string > map, std::shared_ptr< [Snake](#) > snake)
- int [getXpos](#) () const
Get the Pos X object.
- int [getYpos](#) () const
Get the Pos Y object.
- clock_t [getClock](#) ()
Get the Clock object.

4.8.1 Member Function Documentation

4.8.1.1 [getClock\(\)](#)

```
clock_t arc::games::Food::getClock ( )
```

Get the Clock object.

Returns

clock_t

4.8.1.2 [getXpos\(\)](#)

```
int arc::games::Food::getXpos ( ) const
```

Get the Pos X object.

Returns

int

4.8.1.3 [getYpos\(\)](#)

```
int arc::games::Food::getYpos ( ) const
```

Get the Pos Y object.

Returns

int

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

- lib/games/nibbler/includes/Food.hpp
- lib/games/nibbler/Food.cpp

4.9 arc::utils::HighscoreHandler Class Reference

Public Member Functions

- **HighscoreHandler ()**
Constructor.
- **~HighscoreHandler ()=default**
Destructor.
- `std::vector< std::pair< std::string, int > > getHighscores () const`
Getter for the highscores.
- `void setHighscores (std::vector< std::pair< std::string, int > > highscores)`
Setter for the highscores.
- `void addHighscore (const std::string &name, int score)`
Add a highscore to the highscores.
- `void saveHighscores ()`
Save the highscores to a file.
- `std::vector< std::shared_ptr< arc::Object > > toObjects ()`
Convert the highscores to a list of Objects.

4.9.1 Member Function Documentation

4.9.1.1 addHighscore()

```
void arc::utils::HighscoreHandler::addHighscore (
    const std::string & name,
    int score )
```

Add a highscore to the highscores.

Parameters

<i>name</i>	Name of the player
<i>score</i>	Score of the player

4.9.1.2 getHighscores()

```
std::vector< std::pair< std::string, int > > arc::utils::HighscoreHandler::getHighscores ( )
const
```

Getter for the highscores.

Returns

`std::vector<std::pair<std::string, int>>`

4.9.1.3 setHighscores()

```
void arc::utils::HighscoreHandler::setHighscores (
    std::vector< std::pair< std::string, int > > highscores )
```

Setter for the highscores.

Parameters

<i>highscores</i>	
-------------------	--

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

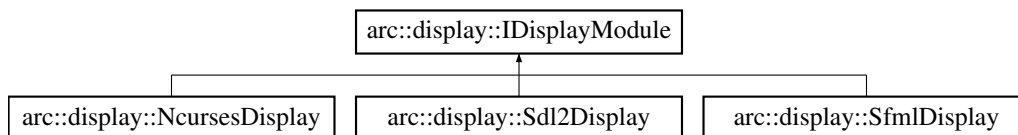
- src/includes/Utils/HighscoreHandler.hpp
- src/Utils/HighscoreHandler.cpp

4.10 arc::display::IDisplayModule Class Reference

Display module interface.

```
#include <IDisplayModule.hpp>
```

Inheritance diagram for arc::display::IDisplayModule:



Public Member Functions

- virtual **~IDisplayModule** ()=default
Destroy the IDisplayModule object.
- virtual void **drawObjects** (std::vector< std::shared_ptr< arc::Object > > objs)=0
draw all the objects generated by the game
- virtual void **drawInterface** (std::vector< std::shared_ptr< arc::Object > > objs)=0
Draw the interface of the game.
- virtual arc::Events **getEvent** () const =0
get any event

4.10.1 Detailed Description

Display module interface.

4.10.2 Member Function Documentation

4.10.2.1 drawInterface()

```
virtual void arc::display::IDisplayModule::drawInterface (
    std::vector< std::shared_ptr< arc::Object > > objs ) [pure virtual]
```

Draw the interface of the game.

Implemented in [arc::display::NcursesDisplay](#), [arc::display::Sdl2Display](#), and [arc::display::SfmlDisplay](#).

4.10.2.2 drawObjects()

```
virtual void arc::display::IDisplayModule::drawObjects (
    std::vector< std::shared_ptr< arc::Object > > objs ) [pure virtual]
```

draw all the objects generated by the game

Parameters

<i>objs</i>	
-------------	--

Implemented in [arc::display::NcursesDisplay](#), [arc::display::Sdl2Display](#), and [arc::display::SfmlDisplay](#).

4.10.2.3 getEvent()

```
virtual arc::Events arc::display::IDisplayModule::getEvent ( ) const [pure virtual]
```

get any event

Returns

const arc::Events

Implemented in [arc::display::NcursesDisplay](#), [arc::display::Sdl2Display](#), and [arc::display::SfmlDisplay](#).

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

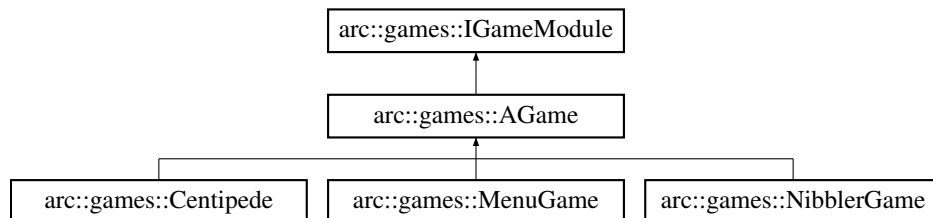
- [src/includes/Interfaces/IDisplayModule.hpp](#)

4.11 arc::games::IGameModule Class Reference

Game module interface.

```
#include <IGameModule.hpp>
```

Inheritance diagram for arc::games::IGameModule:



Public Member Functions

- **~IGameModule ()**=default
Destroy the IDisplayModule object.
- virtual void **useEvent** (arc::Events event)=0
apply the current event
- virtual void **update** ()=0
update the game
- virtual const std::vector< std::shared_ptr< **Object** > > **getObjects** () const =0
Get the objects to draw.
- virtual **~IGameModule ()**=default
Destroy the IDisplayModule object.
- virtual void **useEvent** (arc::Events event)=0
Apply the current event.
- virtual const std::vector< std::shared_ptr< **arc::Object** > > **getObjects** () const =0
Get the objects to draw.
- virtual bool **isRunning** () const =0
Tell if game is running or not.
- virtual void **update** ()=0
Update game's entities.

4.11.1 Detailed Description

Game module interface.

4.11.2 Member Function Documentation

4.11.2.1 `getObjects()` [1/2]

```
virtual const std::vector< std::shared_ptr< Object > > arc::games::IGameModule::getObjects (
) const [pure virtual]
```

Get the objects to draw.

Returns

`const std::vector<std::shared_ptr<IObject>>`

Implemented in [arc::games::MenuGame](#), [arc::games::NibblerGame](#), [arc::games::Centipede](#), and [arc::games::AGame](#).

4.11.2.2 `getObjects()` [2/2]

```
virtual const std::vector< std::shared_ptr< arc::Object > > arc::games::IGameModule::get←
Objects ( ) const [pure virtual]
```

Get the objects to draw.

Returns

`const std::vector<std::shared_ptr<IObject>>`

Implemented in [arc::games::MenuGame](#), [arc::games::NibblerGame](#), [arc::games::Centipede](#), and [arc::games::AGame](#).

4.11.2.3 `isRunning()`

```
virtual bool arc::games::IGameModule::isRunning ( ) const [pure virtual]
```

Tell if game is running or not.

Returns

true or false

Implemented in [arc::games::AGame](#).

4.11.2.4 `update()` [1/2]

```
virtual void arc::games::IGameModule::update ( ) [pure virtual]
```

update the game

Implemented in [arc::games::MenuGame](#), [arc::games::NibblerGame](#), and [arc::games::Centipede](#).

4.11.2.5 update() [2/2]

```
virtual void arc::games::IGameModule::update ( ) [pure virtual]
```

Update game's entities.

Implemented in [arc::games::MenuGame](#), [arc::games::NibblerGame](#), and [arc::games::Centipede](#).

4.11.2.6 useEvent() [1/2]

```
virtual void arc::games::IGameModule::useEvent (
    arc::Events event ) [pure virtual]
```

apply the current event

Parameters

<i>event</i>	
--------------	--

Implemented in [arc::games::MenuGame](#), [arc::games::NibblerGame](#), and [arc::games::Centipede](#).

4.11.2.7 useEvent() [2/2]

```
virtual void arc::games::IGameModule::useEvent (
    arc::Events event ) [pure virtual]
```

Apply the current event.

Parameters

<i>event</i>	
--------------	--

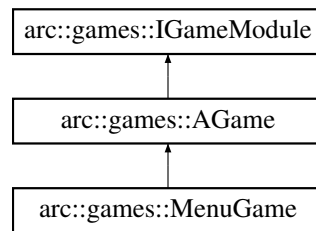
Implemented in [arc::games::MenuGame](#), [arc::games::NibblerGame](#), and [arc::games::Centipede](#).

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

- src/includes/IGameModule.hpp
- src/includes/Interfaces/IGameModule.hpp

4.12 arc::games::MenuGame Class Reference

Inheritance diagram for arc::games::MenuGame:



Public Member Functions

- **MenuGame ()**
Construct a new Menu Game object.
- **~MenuGame ()**
Destroy the Menu Game object.
- void **useEvent** (arc::Events event) final
Apply the current event.
- void **update** () final
Updates game's entities.
- const std::vector< std::shared_ptr< [arc::Object](#) > > **getObjects** () const final
Get the game objects.
- const [MenuProprieties](#) **getProps** () const
Get the properties of the game to start.
- bool **isStarting** () const
Checks if game is starting or not.
- bool **isSelectingGame** () const
Checks if user is selecting game or not.
- void **selectPreviousGame** ()
Selects previous game.
- void **selectNextGame** ()
Selects next game.
- void **selectPreviousDisplay** ()
Selects previous display.
- void **selectNextDisplay** ()
Selects next display.

Additional Inherited Members

4.12.1 Member Function Documentation

4.12.1.1 getObjects()

```
const std::vector< std::shared_ptr< arc::Object > > arc::games::MenuGame::getObjects ( )
const [final], [virtual]
```

Get the game objects.

Returns

Game objects

Reimplemented from [arc::games::AGame](#).

4.12.1.2 getProps()

```
const MenuProprieties arc::games::MenuGame::getProps ( ) const [inline]
```

Get the properties of the game to start.

Returns

const [MenuProprieties](#)

4.12.1.3 isStarting()

```
bool arc::games::MenuGame::isStarting ( ) const
```

Checks if game is starting or not.

Returns

true or false

4.12.1.4 update()

```
void arc::games::MenuGame::update ( ) [final], [virtual]
```

Updates game's entities.

Implements [arc::games::IGameModule](#).

4.12.1.5 useEvent()

```
void arc::games::MenuGame::useEvent (
    arc::Events event ) [final], [virtual]
```

Apply the current event.

Parameters

<i>event</i>	
--------------	--

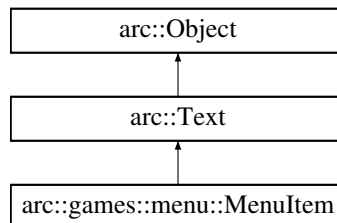
Implements [arc::games::IGameModule](#).

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

- lib/games/menu/includes/MenuGame.hpp
- lib/games/menu/MenuGame.cpp

4.13 arc::games::menu::MenuItem Class Reference

Inheritance diagram for arc::games::menu::MenuItem:



Public Member Functions

- **MenuItem** (const std::string value, **Vector** pos, int size, **Color** color)
Construct a new Menu Item object.
- **~MenuItem** ()=default
Destroy the Menu Item object.
- bool **isSelected** () const
Getter for the selected property.
- void **setSelected** (bool selected)
Setter for the selected property.

Additional Inherited Members

4.13.1 Constructor & Destructor Documentation

4.13.1.1 MenuItem()

```

arc::games::menu::MenuItem::MenuItem (
    const std::string value,
    Vector pos,
    int size,
    Color color )
  
```

Construct a new Menu Item object.

Parameters

<i>value</i>	
<i>pos</i>	
<i>size</i>	
<i>color</i>	

4.13.2 Member Function Documentation

4.13.2.1 isSelected()

```
bool arc::games::menu::MenuItem::isSelected ( ) const
```

Getter for the selected property.

Returns

true or false

4.13.2.2 setSelected()

```
void arc::games::menu::MenuItem::setSelected (
    bool selected )
```

Setter for the selected property.

Parameters

<i>selected</i>	
-----------------	--

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

- lib/games/menu/includes/MenuItem.hpp
- lib/games/menu/MenuItem.cpp

4.14 arc::games::MenuProprieties Struct Reference

Public Attributes

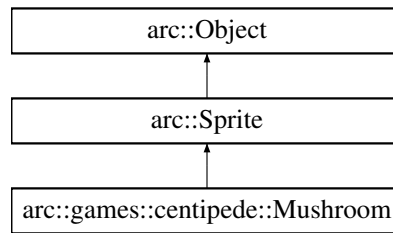
- std::string **username**
- std::string **gamelib**
- std::string **graphicslib**

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

- lib/games/menu/includes/MenuGame.hpp

4.15 arc::games::centipede::Mushroom Class Reference

Inheritance diagram for arc::games::centipede::Mushroom:



Public Member Functions

- **Mushroom** (int x, int y)
Construct a new [Mushroom](#) object.
- **~Mushroom** ()
Destroy the [Mushroom](#) object.
- void **update** ()
Update the state of the object.
- void **setlife** (int life)
- int **getlife** ()
- void **checkDead** ()
check if the object is dead
- bool **isDead** () const

Additional Inherited Members

4.15.1 Member Function Documentation

4.15.1.1 getlife()

```
int arc::games::centipede::Mushroom::getlife ( )
```

Returns

int

4.15.1.2 isDead()

```
bool arc::games::centipede::Mushroom::isDead ( ) const
```

Returns

true
false

4.15.1.3 setlife()

```
void arc::games::centipede::Mushroom::setlife (
    int life )
```

Parameters

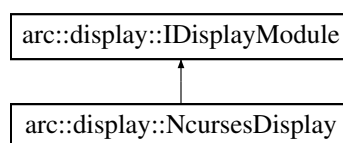
<i>life</i>	
-------------	--

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

- lib/games/centipede/includes/Mushroom.hpp
- lib/games/centipede/Mushroom.cpp

4.16 arc::display::NcursesDisplay Class Reference

Inheritance diagram for arc::display::NcursesDisplay:



Public Member Functions

- void [drawObjects](#) (std::vector< std::shared_ptr< [arc::Object](#) > > objs) override
draw all the objects generated by the game
- arc::Events [getEvent](#) () const override
get any event
- void [drawInterface](#) (std::vector< std::shared_ptr< [arc::Object](#) > > objs)
Draw the interface of the game.

4.16.1 Member Function Documentation

4.16.1.1 drawInterface()

```
void arc::display::NcursesDisplay::drawInterface (
    std::vector< std::shared_ptr< arc::Object > > objs ) [virtual]
```

Draw the interface of the game.

Implements [arc::display::IDisplayModule](#).

4.16.1.2 drawObjects()

```
void arc::display::NcursesDisplay::drawObjects (
    std::vector< std::shared_ptr< arc::Object > > objs ) [override], [virtual]
```

draw all the objects generated by the game

Parameters

<i>objs</i>	
-------------	--

Implements [arc::display::IDisplayModule](#).

4.16.1.3 getEvent()

```
arc::Events arc::display::NcursesDisplay::getEvent ( ) const [override], [virtual]
```

get any event

Returns

const arc::Events

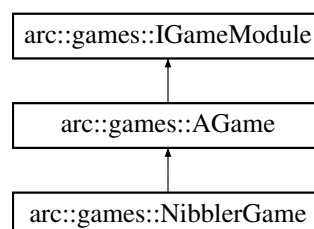
Implements [arc::display::IDisplayModule](#).

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

- lib/graphics/ncurses/includes/NcursesDisplay.hpp
- lib/graphics/ncurses/NcursesDisplay.cpp

4.17 arc::games::NibblerGame Class Reference

Inheritance diagram for arc::games::NibblerGame:



Public Member Functions

- **NibblerGame** ()
Construct a new Nibbler Game object.
- **~NibblerGame** ()
Destroy the Nibbler Game object.
- void **useEvent** (arc::Events event) final
Apply the current event.
- void **update** () final
Updates game's entities.
- const std::vector< std::shared_ptr< [arc::Object](#) > > **getObjects** () const final
Get the game objects.

Additional Inherited Members

4.17.1 Member Function Documentation

4.17.1.1 getObjects()

```
const std::vector< std::shared_ptr< arc::Object > > arc::games::NibblerGame::getObjects ( )  
const [final], [virtual]
```

Get the game objects.

Returns

Game objects

Reimplemented from [arc::games::AGame](#).

4.17.1.2 update()

```
void arc::games::NibblerGame::update ( ) [final], [virtual]
```

Updates game's entities.

Implements [arc::games::IGameModule](#).

4.17.1.3 useEvent()

```
void arc::games::NibblerGame::useEvent (  
    arc::Events event ) [final], [virtual]
```

Apply the current event.

Parameters

<code>event</code>	
--------------------	--

Implements [arc::games::IGameModule](#).

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

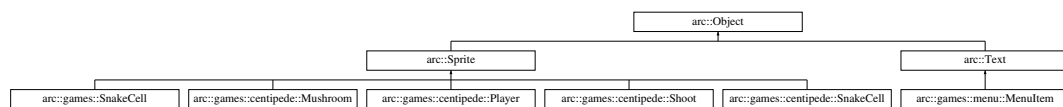
- lib/games/nibbler/includes/NibblerGame.hpp
- lib/games/nibbler/NibblerGame.cpp

4.18 arc::Object Class Reference

Represents a drawable object.

```
#include <Object.hpp>
```

Inheritance diagram for arc::Object:



Public Types

- enum class [Type](#) { **TEXT** , **SPRITE** }
Enumeration of the different types of objects.

Public Member Functions

- [Object](#) ([Type](#) t, const std::string value, [Vector](#) pos)
Constructor.
- [~Object](#) ()=default
Destructor.
- [Type](#) [getType](#) () const
Getter for the type of the object.
- const std::string & [getValue](#) () const
Getter for the value of the object.
- [Vector](#) [getPosition](#) () const
Getter for the position of the object.
- void [setValue](#) (const std::string &value)
Setter for the value of the object.
- void [setPosition](#) ([arc::Vector](#) pos)
Setter for the position of the object.

4.18.1 Detailed Description

Represents a drawable object.

4.18.2 Constructor & Destructor Documentation

4.18.2.1 Object()

```
arc::Object::Object (
    Type t,
    const std::string value,
    Vector pos )
```

Constructor.

Parameters

<i>t</i>	Type of the object
<i>value</i>	Value of the object
<i>pos</i>	Position of the object

4.18.3 Member Function Documentation

4.18.3.1 getPosition()

```
arc::Vector arc::Object::getPosition ( ) const
```

Getter for the position of the object.

Returns

Position of the object

4.18.3.2 getType()

```
arc::Object::Type arc::Object::getType ( ) const
```

Getter for the type of the object.

Returns

Type of the object

4.18.3.3 `getValue()`

```
const std::string & arc::Object::getValue ( ) const
```

Getter for the value of the object.

Returns

Value of the object

4.18.3.4 `setPosition()`

```
void arc::Object::setPosition (
    arc::Vector pos )
```

Setter for the position of the object.

Parameters

<i>pos</i>	
------------	--

4.18.3.5 `setValue()`

```
void arc::Object::setValue (
    const std::string & value )
```

Setter for the value of the object.

Parameters

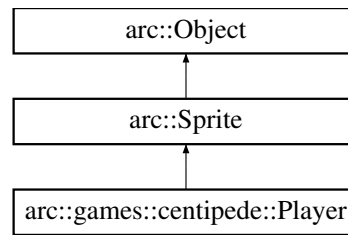
<i>value</i>	Value of the object
--------------	---------------------

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

- `src/includes/Object.hpp`
- `src/Object.cpp`

4.19 `arc::games::centipede::Player` Class Reference

Inheritance diagram for `arc::games::centipede::Player`:



Public Types

- enum [Direction](#) {
LEFT, UP, RIGHT, DOWN,
STAY }

Direction of the [Player](#).

Public Member Functions

- **Player** ()
Construct a new [Player](#) object.
- **~Player** ()
Destroy the [Player](#) object.
- void **move** ([Direction](#) dir)
Move the [Player](#).
- void **createShoot** ()
Create a [Shoot](#) object.
- void **update** ()
Update the [Player](#).
- std::vector< std::shared_ptr< [arc::games::centipede::Shoot](#) > > **getShoots** ()
Get the Shoots object.
- void **deleteShoot** (std::shared_ptr< [arc::games::centipede::Shoot](#) > &shoot)
- bool **lose** (std::vector< std::shared_ptr< [arc::games::centipede::Mushroom](#) > > mushrooms, std::vector< std::shared_ptr< [arc::games::centipede::Snake](#) > > snakes)
check if you lose

4.19.1 Member Function Documentation

4.19.1.1 deleteShoot()

```
void arc::games::centipede::Player::deleteShoot (
    std::shared_ptr< arc::games::centipede::Shoot > & shoot )
```

Parameters

<i>shoot</i>	
--------------	--

4.19.1.2 getShoots()

```
std::vector< std::shared_ptr< arc::games::centipede::Shoot > > arc::games::centipede::↵
Player::getShoots ( )
```

Get the Shoots object.

Returns

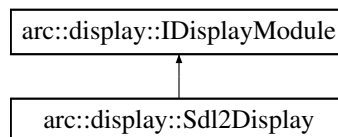
std::vector<std::shared_ptr<arc::games::centipede::Shoot>>

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

- lib/games/centipede/includes/Player.hpp
- lib/games/centipede/Player.cpp

4.20 arc::display::Sdl2Display Class Reference

Inheritance diagram for arc::display::Sdl2Display:



Public Member Functions

- **Sdl2Display ()**
Create a new Sdl2Display object.
- **Sdl2Display (Sdl2Display &other)=delete**
Unique pointer.
- **~Sdl2Display ()**
Destroy a Sdl2Display object.
- void **drawObjects** (std::vector< std::shared_ptr< arc::Object > > objs) override
Draw all the objects generated by the game.
- void **drawInterface** (std::vector< std::shared_ptr< arc::Object > > objs) override
Draw the interface of the game.
- arc::Events **getEvent** () const override
Get any event.
- void **placeObjectOnBoard** (std::shared_ptr< arc::Object > obj)
Place an object on the board.
- **Sdl2Display & operator= (Sdl2Display &other)=delete**
Unique pointer.

4.20.1 Member Function Documentation

4.20.1.1 drawInterface()

```
void arc::display::Sdl2Display::drawInterface (
    std::vector< std::shared_ptr< arc::Object > > objs ) [override], [virtual]
```

Draw the interface of the game.

Parameters

<i>objs</i>	objects to be drawn
-------------	---------------------

Implements [arc::display::IDisplayModule](#).

4.20.1.2 drawObjects()

```
void arc::display::Sdl2Display::drawObjects (
    std::vector< std::shared_ptr< arc::Object > > objs ) [override], [virtual]
```

Draw all the objects generated by the game.

Parameters

<i>objs</i>	objects to be drawn
-------------	---------------------

Implements [arc::display::IDisplayModule](#).

4.20.1.3 getEvent()

```
arc::Events arc::display::Sdl2Display::getEvent ( ) const [override], [virtual]
```

Get any event.

Returns

const arc::Events - event that occurred (or arc::Events::NONE)

Implements [arc::display::IDisplayModule](#).

4.20.1.4 placeObjectOnBoard()

```
void arc::display::Sdl2Display::placeObjectOnBoard (
    std::shared_ptr< arc::Object > obj )
```

Place an object on the board.

Parameters

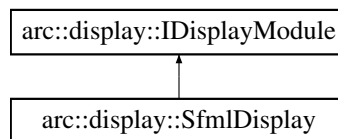
<i>obj</i>	object to be placed
------------	---------------------

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

- lib/graphics/sdl2/includes/Sdl2Display.hpp
- lib/graphics/sdl2/Sdl2Display.cpp

4.21 arc::display::SfmlDisplay Class Reference

Inheritance diagram for arc::display::SfmlDisplay:



Public Member Functions

- **SfmlDisplay ()**
Create a new [SfmlDisplay](#) object.
- **~SfmlDisplay ()**
Destroy a [SfmlDisplay](#) object.
- void **drawObjects** (std::vector< std::shared_ptr< [arc::Object](#) > > objs) override
Draw all the objects generated by the game.
- void **drawInterface** (std::vector< std::shared_ptr< [arc::Object](#) > > objs) override
Draw the interface of the game.
- arc::Events **getEvent** () const override
Get any event.
- void **placeObjectOnBoard** (std::shared_ptr< [arc::Object](#) > obj)
Place an object on the board.

4.21.1 Member Function Documentation

4.21.1.1 drawInterface()

```
void arc::display::SfmlDisplay::drawInterface (
    std::vector< std::shared_ptr< arc::Object > > objs )  [override], [virtual]
```

Draw the interface of the game.

Parameters

<i>objs</i>	objects to be drawn
-------------	---------------------

Implements [arc::display::IDisplayModule](#).

4.21.1.2 drawObjects()

```
void arc::display::SfmlDisplay::drawObjects (
    std::vector< std::shared_ptr< arc::Object > > objs ) [override], [virtual]
```

Draw all the objects generated by the game.

Parameters

<i>objs</i>	objects to be drawn
-------------	---------------------

Implements [arc::display::IDisplayModule](#).

4.21.1.3 getEvent()

```
arc::Events arc::display::SfmlDisplay::getEvent ( ) const [override], [virtual]
```

Get any event.

Returns

const arc::Events - event that occurred (or arc::Events::NONE)

Implements [arc::display::IDisplayModule](#).

4.21.1.4 placeObjectOnBoard()

```
void arc::display::SfmlDisplay::placeObjectOnBoard (
    std::shared_ptr< arc::Object > obj )
```

Place an object on the board.

Parameters

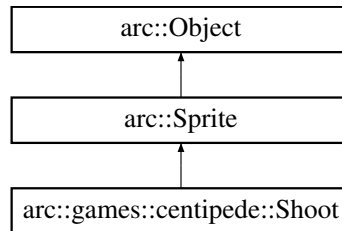
<i>obj</i>	object to be placed
------------	---------------------

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

- lib/graphics/sfml/includes/SfmlDisplay.hpp
- lib/graphics/sfml/SfmlDisplay.cpp

4.22 arc::games::centipede::Shoot Class Reference

Inheritance diagram for arc::games::centipede::Shoot:



Public Member Functions

- [Shoot](#) (int x, int y)
Construct a new [Shoot](#) object.
- [~Shoot](#) ()
Destroy the [Shoot](#) object.
- void [Update](#) ()
Move the shoot.
- std::shared_ptr< [arc::games::centipede::SnakeCell](#) > [getHit](#) (std::shared_ptr< [arc::games::centipede::Snake](#) > snake)
check if cell after is not mushroom or snakes
- void [checkHit](#) (std::vector< std::shared_ptr< [arc::games::centipede::Mushroom](#) > > mushrooms, std::vector< std::shared_ptr< [arc::games::centipede::Snake](#) > > snakes)
- bool [isHit](#) () const
Tells if shoot has hit something or not.

Additional Inherited Members

4.22.1 Constructor & Destructor Documentation

4.22.1.1 Shoot()

```
arc::games::centipede::Shoot::Shoot (
    int x,
    int y )
```

Construct a new [Shoot](#) object.

Parameters

<i>x</i>	
<i>y</i>	

4.22.2 Member Function Documentation

4.22.2.1 checkHit()

```
void arc::games::centipede::Shoot::checkHit (
    std::vector< std::shared_ptr< arc::games::centipede::Mushroom > > mushrooms,
    std::vector< std::shared_ptr< arc::games::centipede::Snake > > snakes )
```

Parameters

<i>mushrooms</i>	
<i>snakes</i>	

Returns

true
false

4.22.2.2 getHit()

```
std::shared_ptr< arc::games::centipede::SnakeCell > arc::games::centipede::Shoot::getHit (
    std::shared_ptr< arc::games::centipede::Snake > snake )
```

check if cell after is not mushroom or snakes

Parameters

<i>mushrooms</i>	
<i>snakes</i>	

4.22.2.3 isHit()

```
bool arc::games::centipede::Shoot::isHit ( ) const
```

Tells if shoot has hit something or not.

Returns

true
false

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

- lib/games/centipede/includes/Player.hpp
- lib/games/centipede/Player.cpp

4.23 arc::games::centipede::Snake Class Reference

Public Member Functions

- [Snake](#) (int size, int x, int y)
Construct a new [Snake](#) object.
- [Snake](#) (std::vector< std::shared_ptr< [arc::games::centipede::SnakeCell](#) > > cells)
Construct a new [Snake](#) object.
- [~Snake](#) ()
Destroy the [Snake](#) object.
- std::vector< std::shared_ptr< [arc::games::centipede::SnakeCell](#) > > [getCells](#) () const
Getter for the cells of the snake.
- void [setCells](#) (std::vector< std::shared_ptr< [arc::games::centipede::SnakeCell](#) > > cells)
Setter for the cells of the snake.
- void [update](#) ()
Update the state of the [Snake](#).
- void [checkHit](#) (std::vector< std::shared_ptr< [arc::games::centipede::Mushroom](#) > > mushrooms)
If sprite hit something.

4.23.1 Constructor & Destructor Documentation

4.23.1.1 Snake() [1/2]

```
arc::games::centipede::Snake::Snake (
    int size,
    int x,
    int y )
```

Construct a new [Snake](#) object.

Parameters

<i>size</i>	size of the snake
<i>x</i>	position of the snake on the x axis
<i>y</i>	position of the snake on the y axis

4.23.1.2 Snake() [2/2]

```
arc::games::centipede::Snake::Snake (
    std::vector< std::shared_ptr< arc::games::centipede::SnakeCell > > cells )
```

Construct a new [Snake](#) object.

Parameters

<i>cells</i>	cells of the snake
--------------	--------------------

4.23.2 Member Function Documentation

4.23.2.1 getCells()

```
std::vector< std::shared_ptr< arc::games::centipede::SnakeCell > > arc::games::centipede::Snake::getCells ( ) const
```

Getter for the cells of the snake.

Returns

```
std::vector<std::shared_ptr<arc::games::centipede::SnakeCell>>
```

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

- lib/games/centipede/includes/Snake.hpp
- lib/games/centipede/Snake.cpp

4.24 arc::games::Snake Class Reference

Public Member Functions

- **Snake** (int x, int y)
- void **moveSnake** ()
move the snake by one cell, in 's_facing' direction
- void **eat** ()
add a body cell
- int [getXpos](#) ()
Get the x pos object.
- int [getYpos](#) ()
Get the y pos object.

- void **changeFacing** (direction::Facing facing)
change the facing direction of the snake
- void **updateOldFacing** ()
set the OldFacing to Facing
- std::vector< [SnakeCell](#) > **getBody** ()
Get the Body object.
- const std::vector< std::shared_ptr< [arc::Object](#) > > **getObjects** () const
get a vector of object of the whole snake
- bool **hasPosition** (int x, int y)
check if the snake has a cell at position (x, y)
- bool **hasPrevPosition** (int x, int y)
check if the snake has a cell at previous position (x, y)

4.24.1 Member Function Documentation

4.24.1.1 **getBody()**

```
std::vector< arc::games::SnakeCell > arc::games::Snake::getBody ( )
```

Get the Body object.

Returns

```
std::vector<SnakeCell>
```

4.24.1.2 **getXpos()**

```
int arc::games::Snake::getXpos ( )
```

Get the x pos object.

Returns

```
int
```

4.24.1.3 **getYpos()**

```
int arc::games::Snake::getYpos ( )
```

Get the y pos object.

Returns

```
int
```

4.24.1.4 **hasPosition()**

```
bool arc::games::Snake::hasPosition (
    int x,
    int y )
```

check if the snake has a cell at position (x, y)

Parameters

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

Returns

true
false

4.24.1.5 hasPrevPosition()

```
bool arc::games::Snake::hasPrevPosition (
    int x,
    int y )
```

check if the snake has a cell at previous position (x, y)

Parameters

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

Returns

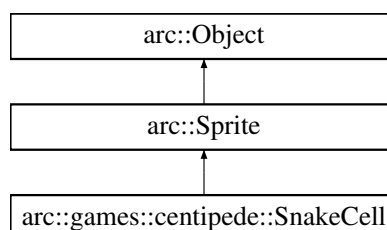
true
false

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

- lib/games/nibbler/includes/Snake.hpp
- lib/games/nibbler/Snake.cpp

4.25 arc::games::centipede::SnakeCell Class Reference

Inheritance diagram for arc::games::centipede::SnakeCell:



Public Types

- enum [Type](#) { **HEAD** , **BODY** }
Type of the Cell.
- enum [Direction](#) { **DOWN** , **LEFT** , **RIGHT** }
direction of the Cell

Public Member Functions

- [SnakeCell](#) (int x, int y, [Type](#) type, [Direction](#) dir=DOWN)
Construct a new [Snake](#) Cell object.
- [~SnakeCell](#) ()
Destroy the [Snake](#) Cell object.
- void **move** ()
Move the snake cell.
- void **update** ()
Update the state of the Cell.
- void **hit** (std::vector< std::shared_ptr< [arc::games::centipede::Mushroom](#) > > mushrooms)
check if cell after is not mushroom
- void **pickADir** (std::vector< std::shared_ptr< [arc::games::centipede::Mushroom](#) > > mushrooms)
check if dir = down and if we go left or right
- void **pickASideDir** (std::vector< std::shared_ptr< [arc::games::centipede::Mushroom](#) > > mushrooms)
check if direction = left or right if we can go down or go in oposite direction
- bool **hasRightMushroom** (std::vector< std::shared_ptr< [arc::games::centipede::Mushroom](#) > > mushrooms)
- bool **hasLeftMushroom** (std::vector< std::shared_ptr< [arc::games::centipede::Mushroom](#) > > mushrooms)
- bool **hasDownMushroom** (std::vector< std::shared_ptr< [arc::games::centipede::Mushroom](#) > > mushrooms)
- [Direction](#) **getDirection** () const
Getter for the direction of the object.
- [Type](#) **getCellType** () const
Getter for the type of the Cell.
- void **setCellType** ([Type](#) type)
Setter for type of the cell.
- void **setDirection** ([Direction](#) dir)
Setter for the direction of the Cell.

4.25.1 Constructor & Destructor Documentation

4.25.1.1 SnakeCell()

```
arc::games::centipede::SnakeCell::SnakeCell (
    int x,
    int y,
    Type type,
    Direction dir = DOWN )
```

Construct a new [Snake](#) Cell object.

Parameters

<i>x</i>	position of the Cell on the x axis
<i>y</i>	position of the Cell on the y axiss
<i>type</i>	type of the Cell

4.25.2 Member Function Documentation

4.25.2.1 getCellType()

```
arc::games::centipede::SnakeCell::Type arc::games::centipede::SnakeCell::getCellType ( ) const
```

Getter for the type of the Cell.

Returns

Type

4.25.2.2 getDirection()

```
arc::games::centipede::SnakeCell::Direction arc::games::centipede::SnakeCell::getDirection ( )  
const
```

Getter for the direction of the object.

Returns

Direction

4.25.2.3 hasDownMushroom()

```
bool arc::games::centipede::SnakeCell::hasDownMushroom (  
    std::vector< std::shared_ptr< arc::games::centipede::Mushroom > > mushrooms )
```

Parameters

<i>mushrooms</i>	
------------------	--

Returns

true
false

4.25.2.4 hasLeftMushroom()

```
bool arc::games::centipede::SnakeCell::hasLeftMushroom (
    std::vector< std::shared_ptr< arc::games::centipede::Mushroom > > mushrooms )
```

Parameters

<i>mushrooms</i>	
------------------	--

Returns

true
false

4.25.2.5 hasRightMushroom()

```
bool arc::games::centipede::SnakeCell::hasRightMushroom (
    std::vector< std::shared_ptr< arc::games::centipede::Mushroom > > mushrooms )
```

Parameters

<i>mushrooms</i>	
------------------	--

Returns

true
false

4.25.2.6 hit()

```
void arc::games::centipede::SnakeCell::hit (
    std::vector< std::shared_ptr< arc::games::centipede::Mushroom > > mushrooms )
```

check if cell after is not mushroom

Parameters

<i>mushrooms</i>	
------------------	--

4.25.2.7 pickADir()

```
void arc::games::centipede::SnakeCell::pickADir (
    std::vector< std::shared_ptr< arc::games::centipede::Mushroom > > mushrooms )
```

check if dir = down and if we go left or right

Parameters

<i>mushrooms</i>	= list of mushroom
------------------	--------------------

4.25.2.8 pickASideDir()

```
void arc::games::centipede::SnakeCell::pickASideDir (
    std::vector< std::shared_ptr< arc::games::centipede::Mushroom > > mushrooms )
```

check if direction = left or right if we can go down or go in opposite direction

Parameters

<i>mushrooms</i>	
------------------	--

4.25.2.9 setCellType()

```
void arc::games::centipede::SnakeCell::setCellType (
    Type type )
```

Setter for type of the cell.

Parameters

<i>type</i>	
-------------	--

4.25.2.10 setDirection()

```
void arc::games::centipede::SnakeCell::setDirection (
    Direction dir )
```

Setter for the direction of the Cell.

Parameters

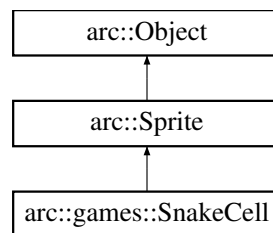
<i>dir</i>	
------------	--

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

- lib/games/centipede/includes/Snake.hpp
- lib/games/centipede/Snake.cpp

4.26 arc::games::SnakeCell Class Reference

Inheritance diagram for arc::games::SnakeCell:



Public Member Functions

- **SnakeCell** (int x, int y)
- void **setPrevPos** ()
Set the position of the previous.
- int **getXpos** ()
Get the x position.
- int **getYpos** ()
Get the y position.
- int **getPrevXpos** ()
Get the previous x position.
- int **getPrevYpos** ()
Get the previous y position.
- void **updateAxis** ()
Update the Axis variable.

Additional Inherited Members

4.26.1 Member Function Documentation

4.26.1.1 `getPrevXpos()`

```
int arc::games::SnakeCell::getPrevXpos ( )
```

Get the previous x position.

Returns

int

4.26.1.2 `getPrevYpos()`

```
int arc::games::SnakeCell::getPrevYpos ( )
```

Get the previous y position.

Returns

int

4.26.1.3 `getXpos()`

```
int arc::games::SnakeCell::getXpos ( )
```

Get the x position.

Returns

int

4.26.1.4 `getYpos()`

```
int arc::games::SnakeCell::getYpos ( )
```

Get the y position.

Returns

int

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

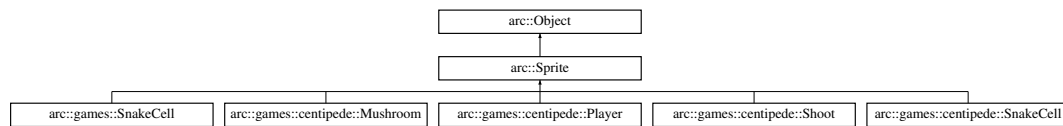
- lib/games/nibbler/includes/SnakeCell.hpp
- lib/games/nibbler/SnakeCell.cpp

4.27 arc::Sprite Class Reference

Represents a sprite object.

```
#include <Object.hpp>
```

Inheritance diagram for arc::Sprite:



Public Member Functions

- [Sprite](#) (const std::string name, [arc::Vector](#) pos, int height=0, int width=0, [arc::Vector](#) scale=[arc::Vector](#)(100, 100))
Constructor.
- [~Sprite](#) ()=default
Destructor.
- int [getHeight](#) () const
Getter for the height of the sprite.
- int [getWidth](#) () const
Getter for the width of the sprite.
- [Vector](#) [getScale](#) () const
Getter for the scale of the sprite.
- int [setHeight](#) (int height)
Setter for the height of the sprite.
- int [setWidth](#) (int width)
Setter for the width of the sprite.
- void [setScale](#) ([Vector](#) scale)
Setter for the scale of the sprite.

Additional Inherited Members

4.27.1 Detailed Description

Represents a sprite object.

4.27.2 Constructor & Destructor Documentation

4.27.2.1 Sprite()

```
arc::Sprite::Sprite (
    const std::string name,
    arc::Vector pos,
    int height = 0,
    int width = 0,
    arc::Vector scale = arc::Vector(100, 100) )
```

Constructor.

Parameters

<i>name</i>	Name of the sprite
<i>pos</i>	Position of the sprite
<i>height</i>	Height of the sprite
<i>width</i>	Width of the sprite
<i>scale</i>	Scale of the sprite

4.27.3 Member Function Documentation

4.27.3.1 getHeight()

```
int arc::Sprite::getHeight ( ) const
```

Getter for the height of the sprite.

Returns

Height of the sprite

4.27.3.2 getScale()

```
arc::Vector arc::Sprite::getScale ( ) const
```

Getter for the scale of the sprite.

Returns

Scale of the sprite

4.27.3.3 getWidth()

```
int arc::Sprite::getWidth ( ) const
```

Getter for the width of the sprite.

Returns

Width of the sprite

4.27.3.4 setHeight()

```
int arc::Sprite::setHeight (
    int height )
```

Setter for the height of the sprite.

Parameters

<i>height</i>	Height of the sprite
---------------	----------------------

4.27.3.5 setScale()

```
void arc::Sprite::setScale (
    Vector scale )
```

Setter for the scale of the sprite.

Parameters

<i>scale</i>	Scale of the sprite
--------------	---------------------

4.27.3.6 setWidth()

```
int arc::Sprite::setWidth (
    int width )
```

Setter for the width of the sprite.

Parameters

<i>width</i>	Width of the sprite
--------------	---------------------

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

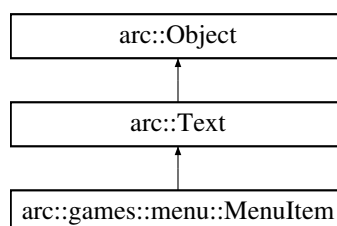
- src/includes/Object.hpp
- src/Object.cpp

4.28 arc::Text Class Reference

Represents a text object.

```
#include <Object.hpp>
```

Inheritance diagram for arc::Text:



Public Member Functions

- [Text](#) (const std::string content, [Vector](#) pos, int size, [Color](#) color)
Constructor.
- [~Text](#) ()=default
Destructor.
- [Color](#) [getColor](#) () const
Getter for the color of the text.
- int [getSize](#) () const
Getter for the size of the text.
- void [setColor](#) ([Color](#) color)
Setter for the color of the text.
- void [setSize](#) (int size)
Setter for the size of the text.

Additional Inherited Members

4.28.1 Detailed Description

Represents a text object.

4.28.2 Constructor & Destructor Documentation

4.28.2.1 Text()

```
arc::Text::Text (  
    const std::string content,  
    Vector pos,  
    int size,  
    Color color )
```

Constructor.

Parameters

<i>content</i>	Content of the text
<i>pos</i>	Position of the text
<i>size</i>	Size of the text
<i>color</i>	Color of the text

4.28.3 Member Function Documentation

4.28.3.1 getColor()

```
arc::Color arc::Text::getColor ( ) const
```

Getter for the color of the text.

Returns

Color of the text

4.28.3.2 getSize()

```
int arc::Text::getSize ( ) const
```

Getter for the size of the text.

Returns

Size of the text

4.28.3.3 setColor()

```
void arc::Text::setColor (
    Color color )
```

Setter for the color of the text.

Parameters

<i>color</i>	Color of the text
--------------	-------------------

4.28.3.4 setSize()

```
void arc::Text::setSize (
    int size )
```

Setter for the size of the text.

Parameters

<i>size</i>	Size of the text
-------------	------------------

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

- src/includes/Object.hpp
- src/Object.cpp

4.29 arc::Vector Struct Reference

Int vector.

```
#include <Vector.hpp>
```

Public Attributes

- int **x**
- int **y**

4.29.1 Detailed Description

Int vector.

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

- src/includes/Vector.hpp

Chapter 5

File Documentation

5.1 Centipede.hpp

```
1 #include <CentipedeGame.hpp>
2
3 #include <exception>
4 #include <string>
5
6 #pragma once
7
8 namespace arc::games
9 {
10
11     extern "C"
12     {
13         Centipede *createInstance();
14     }; /* extern "C" */
15
16 }; /* namespace arc::display */
```

5.2 CentipedeGame.hpp

```
1 #pragma once
2
3 #include <AGame.hpp>
4 #include <Mushroom.hpp>
5 #include <Player.hpp>
6 #include <Snake.hpp>
7
8 #include <ctime>
9 #include <iostream>
10 #include <memory>
11 #include <vector>
12
13 namespace arc::games {
14
15     class Centipede : public arc::games::AGame {
16     public:
17         Centipede();
18
19         ~Centipede();
20
21         void useEvent(arc::Events event) override;
22
23         void update() override;
24
25         const std::vector<std::shared_ptr<arc::Object>> getObjects() const override;
26
27         void splitSnake(std::shared_ptr<arc::games::centipede::Snake> snake,
28             std::shared_ptr<arc::games::centipede::SnakeCell> cell);
29
30     private :
31         std::vector<std::shared_ptr<arc::games::centipede::Snake>> snakes;
32         std::shared_ptr<arc::games::centipede::Player> player;
33         std::vector<std::shared_ptr<arc::games::centipede::Mushroom>> mushrooms;
34         std::clock_t clock;
35         std::clock_t shootClock;
```

```

63         std::clock_t shootMoveClock;
64         std::clock_t snakePopClock;
65
66     };
67 };

```

5.3 Mushroom.hpp

```

1  /*
2  ** EPITECH PROJECT, 2022
3  ** B-OOP-400-LYN-4-1-arcade-marvin.flamand
4  ** File description:
5  ** Mushroom
6  */
7
8  #pragma once
9
10 #include <Object.hpp>
11
12 namespace arc::games::centipede {
13
14     class Mushroom : public arc::Sprite
15     {
16     public:
17         Mushroom(int x, int y);
18         ~Mushroom();
19
20         void update();
21
22         void setlife(int life);
23
24         int getlife();
25
26         void checkDead();
27
28         bool isDead() const;
29
30     private:
31         int life;
32         bool m_isDead;
33     };
34 }

```

5.4 Player.hpp

```

1  /*
2  ** EPITECH PROJECT, 2022
3  ** B-OOP-400-LYN-4-1-arcade-marvin.flamand
4  ** File description:
5  ** Player
6  */
7
8  #pragma once
9
10 #include <Object.hpp>
11 #include <Snake.hpp>
12 #include <Mushroom.hpp>
13 #include <iostream>
14
15 namespace arc::games::centipede {
16
17     class Shoot : public arc::Sprite
18     {
19     public:
20         Shoot(int x, int y);
21         ~Shoot();
22
23         void Update();
24
25         std::shared_ptr<arc::games::centipede::SnakeCell>
26         getHit(std::shared_ptr<arc::games::centipede::Snake> snake);
27
28         void checkHit(std::vector<std::shared_ptr<arc::games::centipede::Mushroom> mushrooms,
29             std::vector<std::shared_ptr<arc::games::centipede::Snake> snakes);
30
31         bool isHit() const;
32
33     private:

```

```

67         bool m_isHit;
68     };
69     class Player : public arc::Sprite {
70     public:
71         Player();
72
73         ~Player();
74
75         enum Direction {LEFT, UP, RIGHT, DOWN, STAY};
76
77         void move(Direction dir);
78
79         void createShoot();
80         void update();
81
82         std::vector<std::shared_ptr<arc::games::centipede::Shoot> > getShoots();
83
84         void deleteShoot(std::shared_ptr<arc::games::centipede::Shoot> &shoot);
85
86         bool lose(std::vector<std::shared_ptr<arc::games::centipede::Mushroom> > mushrooms,
87                 std::vector<std::shared_ptr<arc::games::centipede::Snake> > snakes);
88     private:
89         Direction dir;
90
91         std::vector<std::shared_ptr<arc::games::centipede::Shoot> > shoots;
92     };
93 }

```

5.5 Snake.hpp

```

1  #pragma once
2
3  #include <Object.hpp>
4
5  #include <ctime>
6  #include <iostream>
7  #include <vector>
8  #include <memory>
9  #include <Mushroom.hpp>
10 namespace arc::games::centipede {
11
12     class SnakeCell : public arc::Sprite
13     {
14     public:
15         enum Type { HEAD, BODY };
16         enum Direction { DOWN, LEFT, RIGHT };
17         SnakeCell(int x, int y, Type type, Direction dir = DOWN);
18
19         ~SnakeCell();
20
21         void move();
22
23         void update();
24
25         void hit(std::vector<std::shared_ptr<arc::games::centipede::Mushroom> > mushrooms);
26
27         void pickADir(std::vector<std::shared_ptr<arc::games::centipede::Mushroom> > mushrooms);
28
29         void pickASideDir(std::vector<std::shared_ptr<arc::games::centipede::Mushroom> > mushrooms);
30
31         bool hasRightMushroom(std::vector<std::shared_ptr<arc::games::centipede::Mushroom> >
32                                mushrooms);
33
34         bool hasLeftMushroom(std::vector<std::shared_ptr<arc::games::centipede::Mushroom> > mushrooms);
35
36         bool hasDownMushroom(std::vector<std::shared_ptr<arc::games::centipede::Mushroom> > mushrooms);
37
38         Direction getDirection() const;
39
40         Type getCellType() const;
41
42         void setCellType(Type type);
43
44         void setDirection(Direction dir);
45     private:
46         int x;
47         int y;
48         int frame;
49         Type type;
50         Direction dir;
51
52     };
53 }

```

```

135     };
136
137     std::string &operator«(std::string& s, arc::games::centipede::SnakeCell::Direction& d);
138
139     class Snake
140     {
141     public:
142         Snake(int size, int x, int y);
143
144         Snake(std::vector<std::shared_ptr<arc::games::centipede::SnakeCell> cells);
145
146         ~Snake();
147
148         std::vector<std::shared_ptr<arc::games::centipede::SnakeCell> getCells() const;
149
150         void setCells(std::vector<std::shared_ptr<arc::games::centipede::SnakeCell> cells);
151
152         void update();
153
154         void checkHit(std::vector<std::shared_ptr<arc::games::centipede::Mushroom> mushrooms);
155
156     private:
157         std::vector<std::shared_ptr<arc::games::centipede::SnakeCell> cells;
158     };
159 }

```

5.6 Snake.hpp

```

1  /*
2  ** EPITECH PROJECT, 2022
3  ** Arcade
4  ** File description:
5  ** snake
6  */
7
8  #pragma once
9  #include "SnakeCell.hpp"
10 #include "Direction.hpp"
11 #include <vector>
12
13 namespace arc::games {
14 class Snake {
15     public:
16         Snake(int x, int y);
17         ~Snake();
18
19         void moveSnake();
20
21         void eat();
22
23         int getXpos();
24         int getYpos();
25
26         void changeFacing(direction::Facing facing);
27         void updateOldFacing();
28
29         std::vector<SnakeCell> getBody();
30
31         const std::vector<std::shared_ptr<arc::Object> getObjects() const;
32
33         bool hasPosition(int x, int y);
34         bool hasPrevPosition(int x, int y);
35
36     private :
37         int s_Xpos;
38         int s_Ypos;
39         direction::Facing s_facing;
40         direction::Facing s_OldFacing;
41         std::vector<SnakeCell> body;
42 };
43 }

```

5.7 AGame.hpp

```

1 #include <Interfaces/IGameModule.hpp>
2

```



```

3 #pragma once
4
5 namespace arc::games {
6
7 class AGame : public arc::games::IGameModule {
8     public:
9         AGame(int score = 0);
10
11         ~AGame();
12
13         int getScore() const;
14
15         bool isRunning() const override;
16
17         virtual const std::vector<std::shared_ptr<arc::Object>> getObjects() const override;
18
19     protected:
20         int m_score;
21
22         bool m_isRunning;
23
24         std::vector<std::shared_ptr<arc::Object>> m_objects;
25
26 }; /* class AGame */
27
28 } /* namespace arc::games */

```

5.8 Menu.hpp

```

1 #include <MenuGame.hpp>
2
3 #include <exception>
4 #include <string>
5
6 #pragma once
7
8 namespace arc::games {
9
10     extern "C" {
11         MenuGame *createInstance();
12     }; /* extern "C" */
13
14 }; /* namespace arc::display */

```

5.9 MenuGame.hpp

```

1 #include <memory>
2 #include <string>
3
4
5 #include <AGame.hpp>
6 #include <MenuItem.hpp>
7
8 #pragma once
9
10 namespace arc::games {
11
12     struct MenuProprieties {
13         std::string username;
14         std::string gamelib;
15         std::string graphicslib;
16     };
17
18     class MenuGame : public AGame {
19     public:
20         MenuGame();
21
22         ~MenuGame();
23
24         void useEvent(arc::Events event) final;
25
26         void update() final;
27
28         const std::vector<std::shared_ptr<arc::Object>> getObjects() const final;
29
30         const MenuProprieties getProps() const {
31             return m_props;
32         }
33     };
34
35 }

```

```

60         }
61
62         bool isStarting() const;
63
64         bool isSelectingGame() const;
65
66         void selectPreviousGame();
67
68         void selectNextGame();
69
70         void selectPreviousDisplay();
71
72         void selectNextDisplay();
73
74     private:
75         MenuProprieties m_props;
76
77         bool m_isStarting;
78
79         bool m_isSelectingGame;
80
81         std::vector<std::shared_ptr<arc::games::menu::MenuItem>> m_games;
82
83         std::vector<std::shared_ptr<arc::games::menu::MenuItem>> m_displays;
84
85         std::vector<std::shared_ptr<arc::games::menu::MenuItem>> m_ui;
86
87         void useCharacterEvent(arc::Events event);
88
89     }; /* class MenuGame */
90 } /* namespace arc::games */

```

5.10 MenuItem.hpp

```

1 #pragma once
2
3 #include <Object.hpp>
4
5 namespace arc::games::menu {
6
7     class MenuItem : public arc::Text {
8     public:
9         MenuItem(const std::string value, Vector pos, int size, Color color);
10
11         ~MenuItem() = default;
12
13         bool isSelected() const;
14
15         void setSelected(bool selected);
16
17     private:
18         bool m_selected;
19     }; /* MenuItem */
20 } /* namespace arc::games::menu */

```

5.11 Direction.hpp

```

1 /*
2  ** EPITECH PROJECT, 2022
3  ** B-OOP-400-LYN-4-1-arcade-marvin.flamand
4  ** File description:
5  ** Direction
6  */
7
8 #pragma once
9
10 namespace direction {
11     enum Facing { UP, RIGHT, DOWN, LEFT };
12     enum axis { HORIZONTAL, VERTICAL };
13 }

```

5.12 Food.hpp

```

1 #pragma once

```

```

2 #include <time.h>
3 #include <string>
4 #include "Snake.hpp"
5 namespace arc::games {
6     class Food {
7     public:
8         Food(std::vector<std::string> map, std::shared_ptr<Snake> snake);
9         ~Food();
10
11         int getXpos() const;
12
13         int getYpos() const;
14
15         clock_t getClock();
16
17     private : int pos_x;
18               int pos_y;
19               clock_t f_clock;
20     };
21 }

```

5.13 Nibbler.hpp

```

1 #include <NibblerGame.hpp>
2
3 #include <exception>
4 #include <string>
5
6 #pragma once
7
8 namespace arc::games {
9
10 extern "C" {
11 NibblerGame *createInstance();
12 }; /* extern "C" */
13
14 }; /* namespace arc::display */

```

5.14 NibblerGame.hpp

```

1 /*
2 ** EPITECH PROJECT, 2022
3 ** Arcade
4 ** File description:
5 ** Nibbler
6 */
7
8 #pragma once
9 #include "Direction.hpp"
10 #include "Snake.hpp"
11 #include <AGame.hpp>
12 #include <fcntl.h>
13 #include <fstream>
14 #include <iostream>
15 #include <ncurses.h>
16 #include <stdlib.h>
17 #include <string>
18 #include <time.h>
19 #include <Food.hpp>
20
21 namespace arc::games {
22 class NibblerGame : public AGame {
23 public:
24     NibblerGame();
25
26     ~NibblerGame();
27
28     void useEvent(arc::Events event) final;
29
30     void update() final;
31
32     const std::vector<std::shared_ptr<arc::Object>> getObjects() const final;
33
34 private:
35     std::shared_ptr<Snake> snake;
36     int n_highScore;
37     int n_lives;
38     int n_timeLeft;

```

```

62     std::vector<std::string> n_map;
63     clock_t n_clock;
64     std::vector<std::shared_ptr<arc::games::Food> n_foods;
65     int n_speed;
66 };
67 } // namespace arc::games

```

5.15 SnakeCell.hpp

```

1  /*
2  ** EPITECH PROJECT, 2022
3  ** Arcade
4  ** File description:
5  ** SnakeCell
6  */
7
8  #pragma once
9  #include "Direction.hpp"
10 #include <memory>
11 #include "Object.hpp"
12
13 namespace arc::games {
14 class SnakeCell : public Sprite{
15     public:
16         SnakeCell(int x, int y);
17         ~SnakeCell();
18
19         void setPrevPos();
20
21         int getXpos();
22
23         int getYpos();
24
25         int getPrevXpos();
26
27         int getPrevYpos();
28
29         void updateAxis();
30
31     private :
32         int sc_prevXpos;
33         int sc_prevYpos;
34 };
35 }

```

5.16 Ncurses.hpp

```

1  #include "NcursesDisplay.hpp"
2
3  #pragma once
4
5  namespace arc::display {
6
7  extern "C" {
13 NcursesDisplay *createInstance();
14 } /* extern "C" */
15
16 }; /* namespace arc::display */

```

5.17 NcursesDisplay.hpp

```

1  #pragma once
2
3  #include "Interfaces/IDisplayModule.hpp"
4  #include <fcntl.h>
5  #include <fstream>
6  #include <ncurses.h>
7
8  namespace arc::display {
9  class NcursesDisplay : public arc::display::IDisplayModule {
10     public:
11         NcursesDisplay();
12         ~NcursesDisplay();
13
14         void drawObjects(std::vector<std::shared_ptr<arc::Object> objs) override;

```

```

20
26         arc::Events getEvent() const override;
27
32         void drawInterface(std::vector<std::shared_ptr<arc::Object>> objs);
33
34     private : void getTexture(const std::string fileName, int y, int x);
35               void printMiddle(int y, int x, const std::string text, arc::Color color);
36               arc::Color getSpriteColor(std::string line);
37
42         void drawBorder();
43
48         void printInterface(int y, int x, const std::string text, arc::Color color);
49
54         void clearBoard();
55
56     };
57 }

```

5.18 Sdl2.hpp

```

1 #include <Sdl2Display.hpp>
2
3 #include <exception>
4 #include <string>
5
6 #pragma once
7
8 namespace arc::display {
9
10     extern "C" {
16         Sdl2Display *createInstance();
17     } /* extern "C" */
18
19 }; /* namespace arc::display */

```

5.19 Sdl2Display.hpp

```

1 #include <Interfaces/IDisplayModule.hpp>
2
3 #include <SDL2/SDL.h>
4 #include <SDL2/SDL_ttf.h>
5 #include <map>
6
7 #pragma once
8
9
10 namespace arc::display {
11
12     class Sdl2Display : public IDisplayModule {
13     public:
14         Sdl2Display();
15
16         Sdl2Display(Sdl2Display& other) = delete;
17
18         ~Sdl2Display();
19
20         void drawObjects(std::vector<std::shared_ptr<arc::Object>> objs) override;
21
22         void drawInterface(std::vector<std::shared_ptr<arc::Object>> objs) override;
23
24         arc::Events getEvent() const override;
25
26         void placeObjectOnBoard(std::shared_ptr<arc::Object> obj);
27
28         Sdl2Display& operator=(Sdl2Display& other) = delete;
29
30     private:
31
32         SDL_Texture *getTexture(const std::string& name);
33
34         void drawSprite(std::shared_ptr<arc::Object> obj);
35
36         void drawText(std::shared_ptr<arc::Object> obj);
37
38         SDL_Window *m_window;
39
40         SDL_Renderer *m_renderer;
41
42         std::map<std::string, SDL_Texture*> m_textures;

```

```

107
113         arc::Events interpretKeyboardEvent(const SDL_KeyboardEvent& event) const;
114
115     }; /* class Sdl2Display */
116
117 }; /* namespace arc::display */

```

5.20 Sfml.hpp

```

1 #include <SfmlDisplay.hpp>
2 #include <exception>
3 #include <string>
4
5 #pragma once
6
7 namespace arc::display {
8
9     extern "C" {
15         SfmlDisplay *createInstance();
16     } /* extern "C" */
17
18 }; /* namespace arc::display */

```

5.21 SfmlDisplay.hpp

```

1 #include <Interfaces/IDisplayModule.hpp>
2
3 #include <SFML/Audio.hpp>
4 #include <SFML/Graphics.hpp>
5 #include <SFML/Window.hpp>
6 #include <SFML/System.hpp>
7 #include <map>
8
9 #pragma once
10
11 namespace arc::display {
12
13     class SfmlDisplay : public IDisplayModule {
14     public:
15         SfmlDisplay();
16
17         ~SfmlDisplay();
18
19         void drawObjects(std::vector<std::shared_ptr<arc::Object> objs) override;
20
21         void drawInterface(std::vector<std::shared_ptr<arc::Object> objs) override;
22
23         arc::Events getEvent() const override;
24
25         void placeObjectOnBoard(std::shared_ptr<arc::Object> obj);
26     private:
27         std::shared_ptr<sf::Texture> getTexture(const std::string& name);
28
29         void drawSprite(std::shared_ptr<arc::Object> obj);
30
31         void drawText(std::shared_ptr<arc::Object> obj);
32
33         std::shared_ptr<sf::RenderWindow> m_window;
34
35         std::map<std::string, std::shared_ptr<sf::Texture> m_textures;
36
37         std::unique_ptr<sf::Font> m_font;
38
39         arc::Events interpretKeyboardEvent(const sf::Event::KeyEvent& event) const;
40     }; /* class SfmlDisplay */
41
42 }; /* namespace arc::display */

```

5.22 Color.hpp

```

1 #include <cstdint>
2 #include <iostream>
3

```

```

4 #pragma once
5
6 namespace arc {
7
12     struct Color {
17         enum ColorType
18         {
19             RED,
20             GREEN,
21             BLUE,
22             YELLOW,
23             MAGENTA,
24             CYAN,
25             WHITE,
26             BLACK,
27         };
28
29         uint8_t r;
30         uint8_t g;
31         uint8_t b;
32         uint8_t a;
33         ColorType color;
34
44         Color(uint8_t r, uint8_t g, uint8_t b, uint8_t a, ColorType color);
45
51         Color(ColorType type);
52     }; /* struct Color */
53
54 } /* namespace arc */
55
56 std::ostream& operator<<(std::ostream& os, arc::Color& c);

```

5.23 Core.hpp

```

1 #include <Interfaces/IGameModule.hpp>
2 #include <Interfaces/IDisplayModule.hpp>
3 #include <Utils/DLLoader.hpp>
4 #include <Utils/HighscoreHandler.hpp>
5
6 #include <memory>
7 #include <string>
8
9 #pragma once
10
11 namespace arc {
12
17     class Core {
18     public:
19
24         Core(const std::string &lib);
25
30         ~Core();
31
38         std::unique_ptr<arc::display::IDisplayModule> getDisplay() const;
39
44         void run();
45
51         const std::string &getGameName() const;
52
58         const std::string &getDisplayName() const;
59
64         bool useEvent(arc::Events event);
65
70         void update();
71
76         void nextGame();
77
82         void previousGame();
83
88         void nextDisplay();
89
94         void previousDisplay();
95
100        void backToMenu();
101
106        void restartGame();
107
108    private:
109
114        arc::DLLoader<arc::games::IGameModule> c_game;
115
120        arc::DLLoader<arc::display::IDisplayModule> c_display;
121

```

```

126         std::string currentDisplay;
127
132         std::string currentGame;
133
138         std::vector<std::string> c_games;
139
144         std::vector<std::string> c_displays;
145
150         std::string c_username;
151
156         std::vector<std::shared_ptr<arc::Object> c_interface;
157
162         std::unique_ptr<arc::utils::HighscoreHandler> c_highscore;
163
168         int c_score;
169
170     }; /* class Core */
171
172 } /* namespace arc */

```

5.24 Error.hpp

```

1 #include <exception>
2 #include <string>
3
4 #pragma once
5
6 namespace arc {
7
12     class Error : public std::exception {
13     public:
19         Error(const std::string &message);
20
25         ~Error();
26
32         const char *what() const noexcept final;
33
34     protected:
35         std::string e_message;
36     }; /* class arc::Error */
37
38 } /* namespace arc */

```

5.25 Events.hpp

```

1 #pragma once
2
3 namespace arc {
4
9     enum Events {
10         KeyUp,
11         KeyDown,
12         KeyRight,
13         KeyLeft,
14         KeyA,
15         KeyB,
16         KeyC,
17         KeyD,
18         KeyE,
19         KeyF,
20         KeyG,
21         KeyH,
22         KeyI,
23         KeyJ,
24         KeyK,
25         KeyL,
26         KeyM,
27         KeyN,
28         KeyO,
29         KeyP,
30         KeyQ,
31         KeyR,
32         KeyS,
33         KeyT,
34         KeyU,
35         KeyV,
36         KeyW,
37         KeyX,
38         KeyY,

```



```

39     KeyZ,
40     KeyEsc,
41     KeySpace,
42     KeyEnter,
43     KeyDel,
44     Key0,
45     Key1,
46     Key2,
47     Key3,
48     Key4,
49     Key5,
50     Key6,
51     Key7,
52     Key8,
53     Key9,
54     Exit,
55     None
56 }; /* enum Events */
57
58 } /* namespace arc*/

```

5.26 IGameModule.hpp

```

1 #include "Events.hpp"
2 #include "Object.hpp"
3
4 #include <memory>
5 #include <vector>
6
7 #pragma once
8
9 namespace arc::games {
10
11     class IGameModule {
12     public:
13         ~IGameModule() = default;
14
15         virtual void useEvent(arc::Events event) = 0;
16
17         virtual void update() = 0;
18
19         virtual const std::vector<std::shared_ptr<Object>> getObjects() const = 0;
20     }; /* class IGameModule */
21
22 }

```

5.27 IGameModule.hpp

```

1 #include <Events.hpp>
2 #include <Object.hpp>
3
4 #include <memory>
5 #include <vector>
6
7 #pragma once
8
9 namespace arc::games {
10
11     class IGameModule {
12     public:
13         virtual ~IGameModule() = default;
14
15         virtual void useEvent(arc::Events event) = 0;
16
17         virtual const std::vector<std::shared_ptr<arc::Object>> getObjects() const = 0;
18
19         virtual bool isRunning() const = 0;
20
21         virtual void update() = 0;
22     }; /* class IGameModule */
23
24 } /* namespace arc::games */

```

5.28 IDisplayModule.hpp

```

1 #include <Events.hpp>

```

```

2 #include <Object.hpp>
3
4 #include <memory>
5 #include <vector>
6
7 #pragma once
8
9 namespace arc::display {
10
11     class IDisplayModule {
12     public:
13
14         virtual ~IDisplayModule() = default;
15
16         virtual void drawObjects(std::vector<std::shared_ptr<arc::Object>> objs) = 0;
17
18         virtual void drawInterface(std::vector<std::shared_ptr<arc::Object>> objs) = 0;
19
20         virtual arc::Events getEvent() const = 0;
21     }; /* class IDisplayModule */
22
23 } /* namespace arc::display */

```

5.29 Object.hpp

```

1 #include <Color.hpp>
2 #include <Vector.hpp>
3
4 #include <string>
5
6 #pragma once
7
8 namespace arc {
9
10     class Object {
11     public:
12         enum class Type {
13             TEXT,
14             SPRITE
15         };
16
17         Object(Type t, const std::string value, Vector pos);
18
19         ~Object() = default;
20
21         Type getType() const;
22
23         const std::string &getValue() const;
24
25         Vector getPosition() const;
26
27         void setValue(const std::string &value);
28
29         void setPosition(arc::Vector pos);
30
31     private:
32         Type m_type;
33         std::string m_value;
34         Vector m_position;
35     };
36
37     class Text : public Object {
38     public:
39         Text(const std::string content, Vector pos, int size, Color color);
40
41         ~Text() = default;
42
43         Color getColor() const;
44
45         int getSize() const;
46
47         void setColor(Color color);
48
49         void setSize(int size);
50
51     private:
52         Color m_color;
53         int m_size;
54     }; /* class Text */
55
56     class Sprite : public Object {
57     public:

```

```

152         Sprite(const std::string name, arc::Vector pos, int height = 0, int width = 0, arc::Vector
scale = arc::Vector(100, 100));
153
154         ~Sprite() = default;
155
156         int getHeight() const;
157
158         int getWidth() const;
159
160         Vector getScale() const;
161
162         int setHeight(int height);
163
164         int setWidth(int width);
165
166         void setScale(Vector scale);
167
168     private:
169         int m_height;
170         int m_width;
171         arc::Vector m_scale;
172     }; /* class Sprite */
173
174 } /* namespace arc */

```

5.30 DLLoader.hpp

```

1 #include <Error.hpp>
2
3 #include <dlfcn.h>
4 #include <iostream>
5 #include <memory>
6 #include <string>
7
8 #include <MenuGame.hpp>
9
10 #pragma once
11
12 namespace arc {
13
14     template <class T>
15     class DLLoader {
16     public:
17
18         DLLoader() = default;
19
20         DLLoader(const std::string& path)
21             : l_lib(nullptr)
22             , l_instance(nullptr)
23         {
24             this->load(path);
25         }
26
27         DLLoader(DLLoader& other) = delete;
28
29         ~DLLoader()
30         {
31             this->free();
32         }
33
34         void load(const std::string &path)
35         {
36             this->free();
37             this->l_lib = dlopen(path.c_str(), RTLD_NOW | RTLD_LOCAL);
38             if (!l_lib)
39                 throw new arc::Error("Could not open lib: " + path + ", " + dlerror());
40             void* func = dlsym(this->l_lib, "createInstance");
41             if (func == NULL)
42                 throw new arc::Error("Wrong lib format: " + path + ", " + dlerror());
43             l_instance = reinterpret_cast<T* (*)()>(func)();
44             if (l_instance == NULL)
45                 throw new arc::Error("Could not create instance of lib: " + path + ", " + dlerror());
46         }
47
48         void free()
49         {
50             if (this->l_instance)
51                 delete l_instance;
52             if (this->l_lib)
53                 dlclose(this->l_lib);
54             l_instance = nullptr;
55             l_lib = nullptr;
56         }
57
58     };
59
60 }

```

```

88
94         T *getInstance() const
95         {
96             return l_instance;
97         }
98
104        T* operator->() const
105        {
106            return l_instance;
107        }
108
113        DLoader& operator=(DLoader& other) = delete;
114
115    private:
120        void *l_lib;
121
126        T* l_instance;
127
128    }; /* class DLOpener */
129
130 } /* namespace arc */

```

5.31 FileParser.hpp

```

1 #include <array>
2 #include <string>
3 #include <vector>
4
5 #pragma once
6
7 namespace arc::utils {
8
13     class FileParser {
14     public:
15
22         static std::string getLibraryName(const std::string &pathToLib);
23
28         static std::vector<std::string> getLibrariesNames(const std::vector<std::string> libs);
29
36         static std::array<std::vector<std::string>, 2> getAllLibraries(const std::string& path =
        "./lib/");
37
44         static bool isDisplayLibrary(const std::string libName);
45     }; /* class FileParser */
46
47 } /* namespace arc::utils */

```

5.32 HighscoreHandler.hpp

```

1 #include <Object.hpp>
2
3 #include <map>
4 #include <memory>
5 #include <string>
6 #include <vector>
7
8 #pragma once
9
10 namespace arc::utils {
11
12     class HighscoreHandler {
13     public:
14         HighscoreHandler();
15
19         ~HighscoreHandler() = default;
20
25         std::vector<std::pair<std::string, int>> getHighscores() const;
26
32         void setHighscores(std::vector<std::pair<std::string, int>> highscores);
33
39         void addHighscore(const std::string& name, int score);
40
47         void saveHighscores();
48
53         std::vector<std::shared_ptr<arc::Object>> toObjects();
54
59     private:
60         std::vector<std::pair<std::string, int>> m_highscores;
61         std::string m_filePath;
62

```

```
63
64     }; /* class HighscoreHandler */
65
66 } /* namespace arc::utils */
```

5.33 Vector.hpp

```
1 #pragma once
2
3 namespace arc {
4
5     struct Vector {
6         int x;
7         int y;
8     }; /* struct Vector */
9
10 } /* namespace arc */
```


Index

- addHighscore
 - arc::utils::HighscoreHandler, 20
- arc::Color, 10
 - Color, 11
- arc::Core, 12
 - getDisplay, 13
 - getDisplayName, 13
 - getGameName, 13
- arc::display::IDisplayModule, 21
 - drawInterface, 22
 - drawObjects, 22
 - getEvent, 22
- arc::display::NcursesDisplay, 31
 - drawInterface, 31
 - drawObjects, 32
 - getEvent, 32
- arc::display::Sdl2Display, 38
 - drawInterface, 39
 - drawObjects, 39
 - getEvent, 39
 - placeObjectOnBoard, 39
- arc::display::SfmlDisplay, 41
 - drawInterface, 41
 - drawObjects, 42
 - getEvent, 42
 - placeObjectOnBoard, 42
- arc::DLLoader< T >, 14
 - DLLoader, 14
 - getInstance, 15
 - load, 15
 - operator->, 15
- arc::Error, 16
 - Error, 16
 - what, 17
- arc::games::AGame, 7
 - getObjects, 8
 - getScore, 8
 - isRunning, 8
- arc::games::Centipede, 9
 - getObjects, 9
 - splitSnake, 9
 - update, 10
 - useEvent, 10
- arc::games::centipede::Mushroom, 30
 - getlife, 30
 - isDead, 30
 - setlife, 31
- arc::games::centipede::Player, 36
 - deleteShoot, 37
 - getShoots, 38
- arc::games::centipede::Shoot, 43
 - checkHit, 44
 - getHit, 44
 - isHit, 44
 - Shoot, 43
- arc::games::centipede::Snake, 45
 - getCells, 46
 - Snake, 45, 46
- arc::games::centipede::SnakeCell, 48
 - getCellType, 50
 - getDirection, 50
 - hasDownMushroom, 50
 - hasLeftMushroom, 51
 - hasRightMushroom, 51
 - hit, 51
 - pickADir, 52
 - pickASideDir, 52
 - setCellType, 52
 - setDirection, 52
 - SnakeCell, 49
- arc::games::Food, 19
 - getClock, 19
 - getXpos, 19
 - getYpos, 19
- arc::games::IGameModule, 23
 - getObjects, 23, 24
 - isRunning, 24
 - update, 24
 - useEvent, 25
- arc::games::menu::MenuItem, 28
 - isSelected, 29
 - MenuItem, 28
 - setSelected, 29
- arc::games::MenuGame, 25
 - getObjects, 26
 - getProps, 26
 - isStarting, 27
 - update, 27
 - useEvent, 27
- arc::games::MenuProprieties, 29
- arc::games::NibblerGame, 32
 - getObjects, 33
 - update, 33
 - useEvent, 33
- arc::games::Snake, 46
 - getBody, 47
 - getXpos, 47
 - getYpos, 47

- hasPosition, 47
 - hasPrevPosition, 48
- arc::games::SnakeCell, 53
 - getPrevXpos, 53
 - getPrevYpos, 54
 - getXpos, 54
 - getYpos, 54
- arc::Object, 34
 - getPosition, 35
 - getType, 35
 - getValue, 35
 - Object, 35
 - setPosition, 36
 - setValue, 36
- arc::Sprite, 55
 - getHeight, 56
 - getScale, 56
 - getWidth, 56
 - setHeight, 56
 - setScale, 57
 - setWidth, 57
 - Sprite, 55
- arc::Text, 57
 - getColor, 58
 - getSize, 59
 - setColor, 59
 - setSize, 59
 - Text, 58
- arc::utils::FileParser, 17
 - getAllLibraries, 17
 - getLibraryName, 18
 - isDisplayLibrary, 18
- arc::utils::HighscoreHandler, 20
 - addHighscore, 20
 - getHighscores, 20
 - setHighscores, 20
- arc::Vector, 60
- checkHit
 - arc::games::centipede::Shoot, 44
- Color
 - arc::Color, 11
- deleteShoot
 - arc::games::centipede::Player, 37
- DLLoader
 - arc::DLLoader< T >, 14
- drawInterface
 - arc::display::IDisplayModule, 22
 - arc::display::NcursesDisplay, 31
 - arc::display::Sdl2Display, 39
 - arc::display::SfmlDisplay, 41
- drawObjects
 - arc::display::IDisplayModule, 22
 - arc::display::NcursesDisplay, 32
 - arc::display::Sdl2Display, 39
 - arc::display::SfmlDisplay, 42
- Error
 - arc::Error, 16
- getAllLibraries
 - arc::utils::FileParser, 17
- getBody
 - arc::games::Snake, 47
- getCells
 - arc::games::centipede::Snake, 46
- getCellType
 - arc::games::centipede::SnakeCell, 50
- getClock
 - arc::games::Food, 19
- getColor
 - arc::Text, 58
- getDirection
 - arc::games::centipede::SnakeCell, 50
- getDisplay
 - arc::Core, 13
- getDisplayName
 - arc::Core, 13
- getEvent
 - arc::display::IDisplayModule, 22
 - arc::display::NcursesDisplay, 32
 - arc::display::Sdl2Display, 39
 - arc::display::SfmlDisplay, 42
- getGameName
 - arc::Core, 13
- getHeight
 - arc::Sprite, 56
- getHighscores
 - arc::utils::HighscoreHandler, 20
- getHit
 - arc::games::centipede::Shoot, 44
- getInstance
 - arc::DLLoader< T >, 15
- getLibraryName
 - arc::utils::FileParser, 18
- getlife
 - arc::games::centipede::Mushroom, 30
- getObjects
 - arc::games::AGame, 8
 - arc::games::Centipede, 9
 - arc::games::IGameModule, 23, 24
 - arc::games::MenuGame, 26
 - arc::games::NibblerGame, 33
- getPosition
 - arc::Object, 35
- getPrevXpos
 - arc::games::SnakeCell, 53
- getPrevYpos
 - arc::games::SnakeCell, 54
- getProps
 - arc::games::MenuGame, 26
- getScale
 - arc::Sprite, 56
- getScore
 - arc::games::AGame, 8
- getShoots
 - arc::games::centipede::Player, 38

- getSize
 - arc::Text, 59
- getType
 - arc::Object, 35
- getValue
 - arc::Object, 35
- getWidth
 - arc::Sprite, 56
- getXpos
 - arc::games::Food, 19
 - arc::games::Snake, 47
 - arc::games::SnakeCell, 54
- getYpos
 - arc::games::Food, 19
 - arc::games::Snake, 47
 - arc::games::SnakeCell, 54
- hasDownMushroom
 - arc::games::centipede::SnakeCell, 50
- hasLeftMushroom
 - arc::games::centipede::SnakeCell, 51
- hasPosition
 - arc::games::Snake, 47
- hasPrevPosition
 - arc::games::Snake, 48
- hasRightMushroom
 - arc::games::centipede::SnakeCell, 51
- hit
 - arc::games::centipede::SnakeCell, 51
- isDead
 - arc::games::centipede::Mushroom, 30
- isDisplayLibrary
 - arc::utils::FileParser, 18
- isHit
 - arc::games::centipede::Shoot, 44
- isRunning
 - arc::games::AGame, 8
 - arc::games::IGameModule, 24
- isSelected
 - arc::games::menu::MenuItem, 29
- isStarting
 - arc::games::MenuGame, 27
- lib/games/centipede/includes/Centipede.hpp, 61
- lib/games/centipede/includes/CentipedeGame.hpp, 61
- lib/games/centipede/includes/Mushroom.hpp, 62
- lib/games/centipede/includes/Player.hpp, 62
- lib/games/centipede/includes/Snake.hpp, 63
- lib/games/includes/AGame.hpp, 64
- lib/games/menu/includes/Menu.hpp, 65
- lib/games/menu/includes/MenuGame.hpp, 65
- lib/games/menu/includes/MenuItem.hpp, 66
- lib/games/nibbler/includes/Direction.hpp, 66
- lib/games/nibbler/includes/Food.hpp, 66
- lib/games/nibbler/includes/Nibbler.hpp, 67
- lib/games/nibbler/includes/NibblerGame.hpp, 67
- lib/games/nibbler/includes/Snake.hpp, 64
- lib/games/nibbler/includes/SnakeCell.hpp, 68
- lib/graphics/ncurses/includes/Ncurses.hpp, 68
- lib/graphics/ncurses/includes/NcursesDisplay.hpp, 68
- lib/graphics/sdl2/includes/Sdl2.hpp, 69
- lib/graphics/sdl2/includes/Sdl2Display.hpp, 69
- lib/graphics/sfml/includes/Sfml.hpp, 70
- lib/graphics/sfml/includes/SfmlDisplay.hpp, 70
- load
 - arc::DLLoader< T >, 15
- MenuItem
 - arc::games::menu::MenuItem, 28
- Object
 - arc::Object, 35
- operator->
 - arc::DLLoader< T >, 15
- pickADir
 - arc::games::centipede::SnakeCell, 52
- pickASideDir
 - arc::games::centipede::SnakeCell, 52
- placeObjectOnBoard
 - arc::display::Sdl2Display, 39
 - arc::display::SfmlDisplay, 42
- setCellType
 - arc::games::centipede::SnakeCell, 52
- setColor
 - arc::Text, 59
- setDirection
 - arc::games::centipede::SnakeCell, 52
- setHeight
 - arc::Sprite, 56
- setHighscores
 - arc::utils::HighscoreHandler, 20
- setlife
 - arc::games::centipede::Mushroom, 31
- setPosition
 - arc::Object, 36
- setScale
 - arc::Sprite, 57
- setSelected
 - arc::games::menu::MenuItem, 29
- setSize
 - arc::Text, 59
- setValue
 - arc::Object, 36
- setWidth
 - arc::Sprite, 57
- Shoot
 - arc::games::centipede::Shoot, 43
- Snake
 - arc::games::centipede::Snake, 45, 46
- SnakeCell
 - arc::games::centipede::SnakeCell, 49
- splitSnake
 - arc::games::Centipede, 9
- Sprite
 - arc::Sprite, 55

src/includes/Color.hpp, [70](#)
src/includes/Core.hpp, [71](#)
src/includes/Error.hpp, [72](#)
src/includes/Events.hpp, [72](#)
src/includes/IGameModule.hpp, [73](#)
src/includes/Interfaces/IDisplayModule.hpp, [73](#)
src/includes/Interfaces/IGameModule.hpp, [73](#)
src/includes/Object.hpp, [74](#)
src/includes/Utils/DLLoader.hpp, [75](#)
src/includes/Utils/FileParser.hpp, [76](#)
src/includes/Utils/HighscoreHandler.hpp, [76](#)
src/includes/Vector.hpp, [77](#)

Text

arc::Text, [58](#)

update

arc::games::Centipede, [10](#)
arc::games::IGameModule, [24](#)
arc::games::MenuGame, [27](#)
arc::games::NibblerGame, [33](#)

useEvent

arc::games::Centipede, [10](#)
arc::games::IGameModule, [25](#)
arc::games::MenuGame, [27](#)
arc::games::NibblerGame, [33](#)

what

arc::Error, [17](#)