

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	12
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	13
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
4.8 arc::games::Food Class Reference	18
4.8.1 Member Function Documentation	18
4.8.1.1 getXpos()	19
4.8.1.2 getYpos()	19
4.8.1.3 setPos()	19
4.9 arc::utils::HighscoreHandler Class Reference	19
4.9.1 Member Function Documentation	20
4.9.1.1 addHighscore()	20
4.9.1.2 getHighscores()	20
4.9.1.3 setHighscores()	20
4.10 arc::display::IDisplayModule Class Reference	21
4.10.1 Detailed Description	21
4.10.2 Member Function Documentation	21
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	22
4.11.1 Detailed Description	23
4.11.2 Member Function Documentation	23
4.11.2.1 getObjects() [1/2]	23
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]	24
4.11.2.6 useEvent() [1/2]	24
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()	26
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	27

4.13.1 Constructor & Destructor Documentation	28
4.13.1.1 MenuItem()	28
4.13.2 Member Function Documentation	28
4.13.2.1 isSelected()	29
4.13.2.2 setSelected()	29
4.14 arc::games::MenuPropieties Struct Reference	29
4.15 arc::games::centipede::Mushroom Class Reference	29
4.15.1 Member Function Documentation	30
4.15.1.1 getlife()	30
4.15.1.2 isDead()	30
4.15.1.3 setlife()	30
4.16 arc::display::NcursesDisplay Class Reference	31
4.16.1 Member Function Documentation	31
4.16.1.1 drawInterface()	31
4.16.1.2 drawObjects()	31
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	34
4.18.2 Constructor & Destructor Documentation	34
4.18.2.1 Object()	34
4.18.3 Member Function Documentation	35
4.18.3.1 getPosition()	35
4.18.3.2 getType()	35
4.18.3.3 getValue()	35
4.18.3.4 setPosition()	35
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()	37
4.20 arc::display::Sdl2Display Class Reference	38
4.20.1 Member Function Documentation	38
4.20.1.1 drawInterface()	38
4.20.1.2 drawObjects()	39
4.20.1.3 getEvent()	39
4.20.1.4 placeObjectOnBoard()	39
4.21 arc::display::Sdl2Error Class Reference	39

4.21.1 Detailed Description	40
4.21.2 Constructor & Destructor Documentation	40
4.21.2.1 Sdl2Error()	40
4.21.3 Member Function Documentation	40
4.21.3.1 what()	40
4.22 arc::display::SfmlDisplay Class Reference	41
4.22.1 Member Function Documentation	41
4.22.1.1 drawInterface()	41
4.22.1.2 drawObjects()	42
4.22.1.3 getEvent()	42
4.22.1.4 placeObjectOnBoard()	42
4.23 arc::display::SfmlError Class Reference	43
4.23.1 Detailed Description	43
4.23.2 Constructor & Destructor Documentation	43
4.23.2.1 SfmlError()	43
4.23.3 Member Function Documentation	44
4.23.3.1 what()	44
4.24 arc::games::centipede::Shoot Class Reference	44
4.24.1 Constructor & Destructor Documentation	45
4.24.1.1 Shoot()	45
4.24.2 Member Function Documentation	45
4.24.2.1 checkHit()	45
4.24.2.2 getHit()	45
4.24.2.3 isHit()	46
4.25 arc::games::centipede::Snake Class Reference	46
4.25.1 Constructor & Destructor Documentation	46
4.25.1.1 Snake() [1/2]	46
4.25.1.2 Snake() [2/2]	47
4.25.2 Member Function Documentation	47
4.25.2.1 getCells()	47
4.26 arc::games::Snake Class Reference	48
4.26.1 Member Function Documentation	48
4.26.1.1 getBody()	48
4.26.1.2 getXpos()	48
4.26.1.3 getYpos()	49
4.26.1.4 hasPosition()	49
4.26.1.5 hasPrevPosition()	49
4.27 arc::games::centipede::SnakeCell Class Reference	50
4.27.1 Constructor & Destructor Documentation	51
4.27.1.1 SnakeCell()	51
4.27.2 Member Function Documentation	51
4.27.2.1 getCellType()	51

4.27.2.2	getDirection()	51
4.27.2.3	hasDownMushroom()	51
4.27.2.4	hasLeftMushroom()	52
4.27.2.5	hasRightMushroom()	52
4.27.2.6	hit()	52
4.27.2.7	pickADir()	53
4.27.2.8	pickASideDir()	53
4.27.2.9	setCellType()	53
4.27.2.10	setDirection()	54
4.28	arc::games::SnakeCell Class Reference	54
4.28.1	Member Function Documentation	54
4.28.1.1	getPrevXpos()	55
4.28.1.2	getPrevYpos()	55
4.28.1.3	getXpos()	55
4.28.1.4	getYpos()	55
4.29	arc::Sprite Class Reference	56
4.29.1	Detailed Description	56
4.29.2	Constructor & Destructor Documentation	56
4.29.2.1	Sprite()	56
4.29.3	Member Function Documentation	57
4.29.3.1	getHeight()	57
4.29.3.2	getScale()	57
4.29.3.3	getWidth()	57
4.29.3.4	setHeight()	57
4.29.3.5	setScale()	58
4.29.3.6	setWidth()	58
4.30	arc::Text Class Reference	58
4.30.1	Detailed Description	59
4.30.2	Constructor & Destructor Documentation	59
4.30.2.1	Text()	59
4.30.3	Member Function Documentation	59
4.30.3.1	getColor()	60
4.30.3.2	getSize()	60
4.30.3.3	setColor()	60
4.30.3.4	setSize()	60
4.31	arc::Vector Struct Reference	61
4.31.1	Detailed Description	61
5	File Documentation	63
5.1	Centipede.hpp	63
5.2	CentipedeGame.hpp	63
5.3	Mushroom.hpp	64

5.4 Player.hpp	64
5.5 Snake.hpp	65
5.6 Snake.hpp	66
5.7 AGame.hpp	66
5.8 Menu.hpp	67
5.9 MenuGame.hpp	67
5.10 MenuItem.hpp	68
5.11 Direction.hpp	68
5.12 Food.hpp	68
5.13 Nibbler.hpp	69
5.14 NibblerGame.hpp	69
5.15 SnakeCell.hpp	70
5.16 Ncurses.hpp	70
5.17 NcursesDisplay.hpp	70
5.18 Sdl2.hpp	71
5.19 Sdl2Display.hpp	71
5.20 SfmL.hpp	72
5.21 SfmLDisplay.hpp	72
5.22 Color.hpp	72
5.23 Core.hpp	73
5.24 Error.hpp	74
5.25 Error.hpp	74
5.26 Error.hpp	74
5.27 Events.hpp	75
5.28 IGameModule.hpp	75
5.29 IGameModule.hpp	76
5.30 IDisplayModule.hpp	76
5.31 Object.hpp	76
5.32 DLLoader.hpp	77
5.33 FileParser.hpp	78
5.34 HighscoreHandler.hpp	78
5.35 Vector.hpp	79
Index	81

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 >	13
arc::DLLoader< arc::display::IDisplayModule >	13
arc::DLLoader< arc::games::IGameModule >	13
std::exception	
arc::Error	16
arc::display::Sdl2Error	39
arc::display::SfmlError	43
arc::utils::FileParser	17
arc::games::Food	18
arc::utils::HighscoreHandler	19
arc::display::IDisplayModule	21
arc::display::NcursesDisplay	31
arc::display::Sdl2Display	38
arc::display::SfmlDisplay	41
arc::games::IGameModule	22
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	56
arc::games::SnakeCell	54
arc::games::centipede::Mushroom	29
arc::games::centipede::Player	36
arc::games::centipede::Shoot	44
arc::games::centipede::SnakeCell	50
arc::Text	58
arc::games::menu::MenuItem	27
arc::games::centipede::Snake	46
arc::games::Snake	48
arc::Vector	61

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	13
arc::Error	
General error class	16
arc::utils::FileParser	
Handles file manipulation	17
arc::games::Food	18
arc::utils::HighscoreHandler	19
arc::display::IDisplayModule	
Display module interface	21
arc::games::IGameModule	
Game module interface	22
arc::games::MenuGame	25
arc::games::menu::MenuItem	27
arc::games::MenuProprieties	29
arc::games::centipede::Mushroom	29
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::Sdl2Error	
Error class of Sdl2 library	39
arc::display::SfmlDisplay	41
arc::display::SfmlError	
Error class of Sfml library	43
arc::games::centipede::Shoot	44
arc::games::centipede::Snake	46

arc::games::Snake	48
arc::games::centipede::SnakeCell	50
arc::games::SnakeCell	54
arc::Sprite	
Represents a sprite object	56
arc::Text	
Represents a text object	58
arc::Vector	
Int vector	61

Chapter 3

File Index

3.1 File List

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

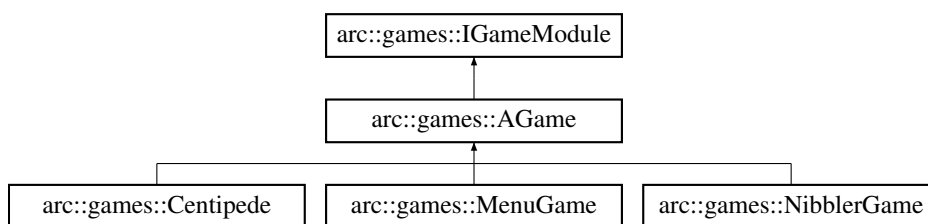
lib/games/centipede/includes/Centipede.hpp	63
lib/games/centipede/includes/CentipedeGame.hpp	63
lib/games/centipede/includes/Mushroom.hpp	64
lib/games/centipede/includes/Player.hpp	64
lib/games/centipede/includes/Snake.hpp	65
lib/games/includes/AGame.hpp	66
lib/games/menu/includes/Menu.hpp	67
lib/games/menu/includes/MenuGame.hpp	67
lib/games/menu/includes/MenuItem.hpp	68
lib/games/nibbler/includes/Direction.hpp	68
lib/games/nibbler/includes/Food.hpp	68
lib/games/nibbler/includes/Nibbler.hpp	69
lib/games/nibbler/includes/NibblerGame.hpp	69
lib/games/nibbler/includes/Snake.hpp	66
lib/games/nibbler/includes/SnakeCell.hpp	70
lib/graphics/ncurses/includes/Ncurses.hpp	70
lib/graphics/ncurses/includes/NcursesDisplay.hpp	70
lib/graphics/sdl2/includes/Error.hpp	74
lib/graphics/sdl2/includes/Sdl2.hpp	71
lib/graphics/sdl2/includes/Sdl2Display.hpp	71
lib/graphics/sfml/includes/Error.hpp	74
lib/graphics/sfml/includes/Sfml.hpp	72
lib/graphics/sfml/includes/SfmlDisplay.hpp	72
src/includes/Color.hpp	72
src/includes/Core.hpp	73
src/includes/Error.hpp	74
src/includes/Events.hpp	75
src/includes/IGameModule.hpp	75
src/includes/Object.hpp	76
src/includes/Vector.hpp	79
src/includes/Interfaces/IDisplayModule.hpp	76
src/includes/Interfaces/IGameModule.hpp	76
src/includes/Utils/DLLoader.hpp	77
src/includes/Utils/FileParser.hpp	78
src/includes/Utils/HighscoreHandler.hpp	78

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`

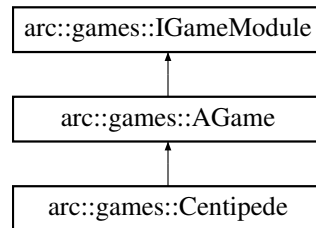
Implements [arc::games::IGameModule](#).

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.

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 DDLoader()

```
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:

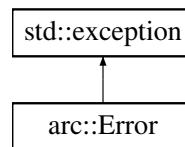
- src/includes/Utils/DLLoader.hpp

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.

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`

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

- int [getXpos](#) () const
Get the Pos X object.
- int [getYpos](#) () const
Get the Pos Y object.
- void [setPos](#) (int x, int y)
Set the Pos X and Y object.

4.8.1 Member Function Documentation

4.8.1.1 getXpos()

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

Get the Pos X object.

Returns

int

4.8.1.2 getYpos()

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

Get the Pos Y object.

Returns

int

4.8.1.3 setPos()

```
void arc::games::Food::setPos (
    int x,
    int y )
```

Set the Pos X and Y object.

Parameters

x	
---	--

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

highscores	
------------	--

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

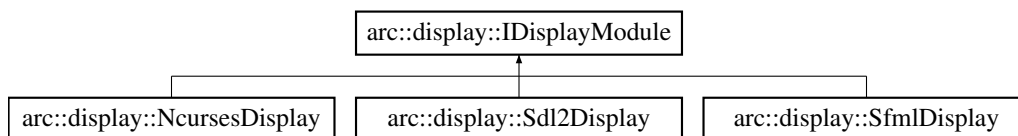
- src/includes/Utils/HighscoreHandler.hpp
- src/Utils/HighscoreHandler.cpp

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:

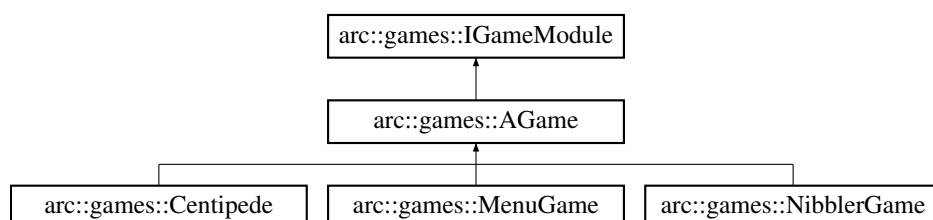
- src/includes/Interfaces/IDisplayModule.hpp

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::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](#).

4.11.2.3 `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>	
--------------	--

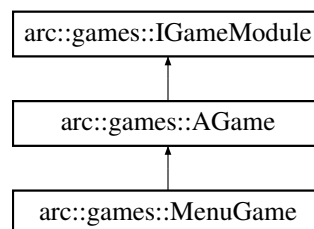
Implemented in [arc::games::MenuGame](#), [arc::games::NibblerGame](#), and [arc::games::Centipede](#).

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>	
--------------	--

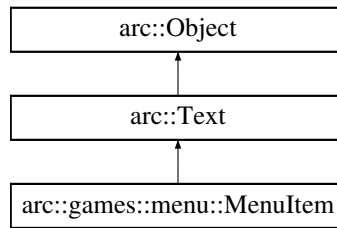
Implements [arc::games::IGameModule](#).

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

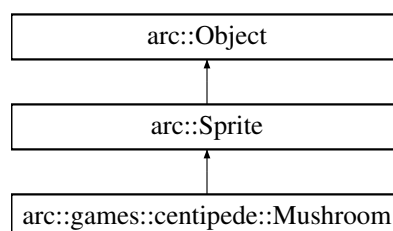
- std::string **username**
- std::string **gamelib**
- std::string **graphicslib**

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

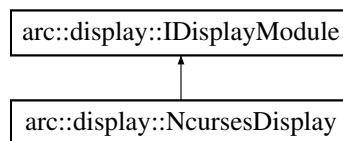
<i>life</i>	
-------------	--

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

<code>objs</code>	
-------------------	--

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

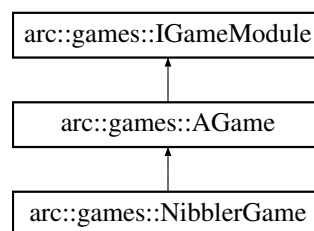
Implements [arc::display::IDisplayModule](#).

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 getObject()

```
const std::vector< std::shared_ptr< arc::Object > > arc::games::NibblerGame::getObject ( )  
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

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

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

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

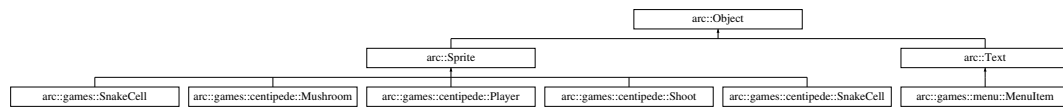
- lib/games/nibbler/includes/NibblerGame.hpp
- lib/games/nibbler/NibblerGame.cpp

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

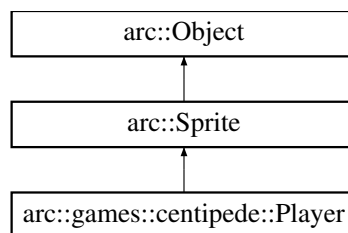
<i>value</i>	Value of the object
--------------	---------------------

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** (std::vector< std::shared_ptr< [arc::games::centipede::Mushroom](#) > > mushrooms, std::vector< std::shared_ptr< [arc::games::centipede::Snake](#) > > snakes)
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](#) > &deleted)

4.19.1 Member Function Documentation

4.19.1.1 deleteShoot()

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

Parameters

deleted	
-------------------------	--

4.19.1.2 getShoots()

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

Get the Shoots object.

Returns

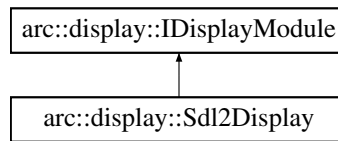
std::vector<std::shared_ptr<arc::games::centipede::Shoot>>

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:

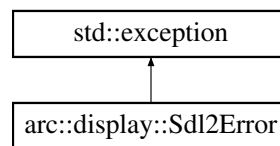
- lib/graphics/sdl2/includes/Sdl2Display.hpp
- lib/graphics/sdl2/Sdl2Display.cpp

4.21 arc::display::Sdl2Error Class Reference

[Error](#) class of Sdl2 library.

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

Inheritance diagram for `arc::display::Sdl2Error`:



Public Member Functions

- [Sdl2Error](#) (const std::string &message)
Create a new [Sdl2Error](#).
- `~Sdl2Error ()`
Destroy the Sdl 2 [Error](#) object.
- const char * [what](#) () const noexcept final
Get the error message.

4.21.1 Detailed Description

[Error](#) class of Sdl2 library.

4.21.2 Constructor & Destructor Documentation

4.21.2.1 Sdl2Error()

```
arc::display::Sdl2Error::Sdl2Error (
    const std::string & message )
```

Create a new [Sdl2Error](#).

Parameters

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

4.21.3 Member Function Documentation

4.21.3.1 what()

```
const char * arc::display::Sdl2Error::what ( ) const [final], [noexcept]
```

Get the error message.

Returns

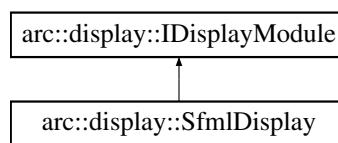
const char*

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

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

4.22 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.

4.22.1 Member Function Documentation

4.22.1.1 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](#).

4.22.1.2 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](#).

4.22.1.3 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](#).

4.22.1.4 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:

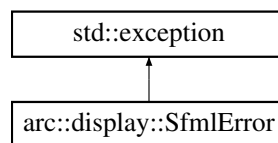
- lib/graphics/sfml/includes/SfmlDisplay.hpp
- lib/graphics/sfml/SfmlDisplay.cpp

4.23 arc::display::SfmlError Class Reference

[Error](#) class of Sfml library.

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

Inheritance diagram for arc::display::SfmlError:



Public Member Functions

- [SfmlError](#) (const std::string &message)
Create a new [SfmlError](#).
- [~SfmlError](#) ()
Destroy the Sdl 2 [Error](#) object.
- const char * [what](#) () const noexcept final
Get the error message.

4.23.1 Detailed Description

[Error](#) class of Sfml library.

4.23.2 Constructor & Destructor Documentation

4.23.2.1 SfmlError()

```
arc::display::SfmlError::SfmlError (
    const std::string & message )
```

Create a new [SfmlError](#).

Parameters

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

4.23.3 Member Function Documentation

4.23.3.1 what()

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

Get the error message.

Returns

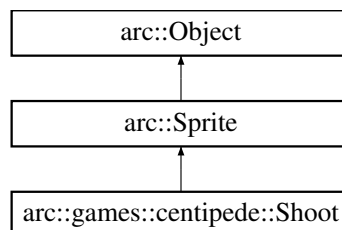
const char*

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

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

4.24 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::shared_ptr< [arc::games::centipede::Snake](#) > snakes)
- bool [isHit](#) () const
Tells if shoot has hit something or not.

Additional Inherited Members

4.24.1 Constructor & Destructor Documentation

4.24.1.1 Shoot()

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

Construct a new [Shoot](#) object.

Parameters

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

4.24.2 Member Function Documentation

4.24.2.1 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

4.24.2.2 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>	

4.24.2.3 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

4.25 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.

4.25.1 Constructor & Destructor Documentation

4.25.1.1 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

4.25.1.2 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
--------------	--------------------

4.25.2 Member Function Documentation

4.25.2.1 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

4.26 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)

4.26.1 Member Function Documentation

4.26.1.1 getBody()

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

Get the Body object.

Returns

std::vector<SnakeCell>

4.26.1.2 getXpos()

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

Get the x pos object.

Returns

int

4.26.1.3 getYpos()

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

Get the y pos object.

Returns

int

4.26.1.4 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

4.26.1.5 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

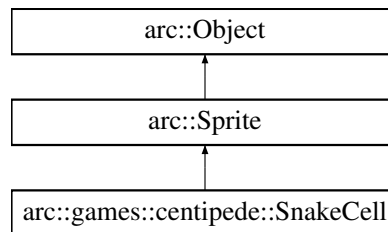
true
false

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

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

4.27 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.

4.27.1 Constructor & Destructor Documentation

4.27.1.1 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

4.27.2 Member Function Documentation

4.27.2.1 getCellType()

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

Getter for the type of the Cell.

Returns

Type

4.27.2.2 getDirection()

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

Getter for the direction of the object.

Returns

Direction

4.27.2.3 hasDownMushroom()

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

Parameters

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

Returns

true
false

4.27.2.4 hasLeftMushroom()

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

Parameters

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

Returns

true
false

4.27.2.5 hasRightMushroom()

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

Parameters

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

Returns

true
false

4.27.2.6 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>	
------------------	--

4.27.2.7 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
------------------	--------------------

4.27.2.8 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>	
------------------	--

4.27.2.9 setCellType()

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

Setter for type of the cell.

Parameters

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

4.27.2.10 setDirection()

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

Setter for the direction of the Cell.

Parameters

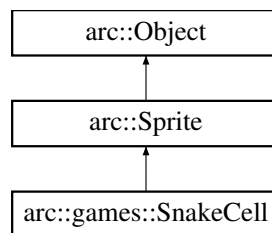
<i>dir</i>	
------------	--

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

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

4.28 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

4.28.1 Member Function Documentation

4.28.1.1 getPrevXpos()

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

Get the previous x position.

Returns

int

4.28.1.2 getPrevYpos()

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

Get the previous y position.

Returns

int

4.28.1.3 getXpos()

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

Get the x position.

Returns

int

4.28.1.4 getYpos()

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

Get the y position.

Returns

int

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

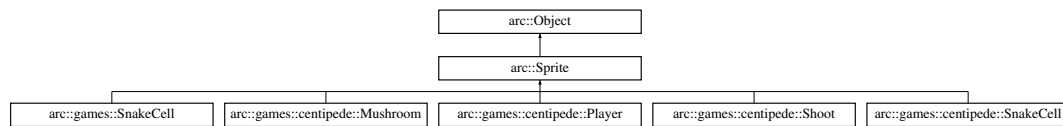
- lib/games/nibbler/includes/SnakeCell.hpp
- lib/games/nibbler/SnakeCell.cpp

4.29 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

4.29.1 Detailed Description

Represents a sprite object.

4.29.2 Constructor & Destructor Documentation

4.29.2.1 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

4.29.3 Member Function Documentation

4.29.3.1 getHeight()

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

Getter for the height of the sprite.

Returns

Height of the sprite

4.29.3.2 getScale()

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

Getter for the scale of the sprite.

Returns

Scale of the sprite

4.29.3.3 getWidth()

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

Getter for the width of the sprite.

Returns

Width of the sprite

4.29.3.4 setHeight()

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

Setter for the height of the sprite.

Parameters

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

4.29.3.5 setScale()

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

Setter for the scale of the sprite.

Parameters

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

4.29.3.6 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:

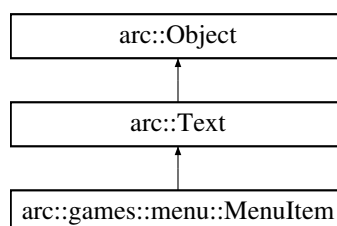
- src/includes/Object.hpp
- src/Object.cpp

4.30 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

4.30.1 Detailed Description

Represents a text object.

4.30.2 Constructor & Destructor Documentation

4.30.2.1 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

4.30.3 Member Function Documentation

4.30.3.1 getColor()

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

Getter for the color of the text.

Returns

Color of the text

4.30.3.2 getSize()

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

Getter for the size of the text.

Returns

Size of the text

4.30.3.3 setColor()

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

Setter for the color of the text.

Parameters

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

4.30.3.4 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

4.31 arc::Vector Struct Reference

Int vector.

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

Public Attributes

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

4.31.1 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;
36         std::clock_t shootMoveClock;
37     };
38 }
```

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:
34         bool m_isHit;
35     };
36
37     class Player : public arc::Sprite {
38     public:
39         Player();
40         ~Player();
41
42         enum Direction {LEFT, UP, RIGHT, DOWN, STAY};
43     };
44 }

```



```

93         void move(Direction dir);
94
99         void createShoot();
104        void update(std::vector<std::shared_ptr<arc::games::centipede::Mushroom>> mushrooms,
std::vector<std::shared_ptr<arc::games::centipede::Snake>> snakes);
105
111        std::vector<std::shared_ptr<arc::games::centipede::Shoot>> getShoots();
112
118        void deleteShoot(std::shared_ptr<arc::games::centipede::Shoot> &deleted);
119
120    private:
125        Direction dir;
126
131        std::vector<std::shared_ptr<arc::games::centipede::Shoot>> shoots;
132    };
133 }

```

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:
19         enum Type { HEAD, BODY };
24         enum Direction { DOWN, LEFT, RIGHT };
32         SnakeCell(int x, int y, Type type, Direction dir = DOWN);
33
38         ~SnakeCell();
39
44         void move();
45
50         void update();
51
57         void hit(std::vector<std::shared_ptr<arc::games::centipede::Mushroom>> mushrooms);
58
64         void pickADir(std::vector<std::shared_ptr<arc::games::centipede::Mushroom>> mushrooms);
65
71         void pickASideDir(std::vector<std::shared_ptr<arc::games::centipede::Mushroom>> mushrooms);
72
80         bool hasRightMushroom(std::vector<std::shared_ptr<arc::games::centipede::Mushroom>>
mushrooms);
81
89         bool hasLeftMushroom(std::vector<std::shared_ptr<arc::games::centipede::Mushroom>> mushrooms);
90
98         bool hasDownMushroom(std::vector<std::shared_ptr<arc::games::centipede::Mushroom>> mushrooms);
99
105         Direction getDirection() const;
106
112         Type getCellType() const;
113
119         void setCellType(Type type);
120
126         void setDirection(Direction dir);
127
128
129     private:
130         int x;
131         int y;
132         int frame;
133         Type type;
134         Direction dir;
135     };
136
137     std::string &operator<<(std::string& s, arc::games::centipede::SnakeCell::Direction& d);
138
139     class Snake
140     {
141     public:
149         Snake(int size, int x, int y);
150
156         Snake(std::vector<std::shared_ptr<arc::games::centipede::SnakeCell>> cells);
157
162         ~Snake();

```

```

163
169         std::vector<std::shared_ptr<arc::games::centipede::SnakeCell> getCells() const;
170
175         void setCells(std::vector<std::shared_ptr<arc::games::centipede::SnakeCell> cells);
176
181         void update();
182
187         void checkHit(std::vector<std::shared_ptr<arc::games::centipede::Mushroom> mushrooms);
188
189     private:
194         std::vector<std::shared_ptr<arc::games::centipede::SnakeCell> cells;
195     };
196 }

```

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
24         void moveSnake();
25
30         void eat();
31
37         int getXpos();
38
44         int getYpos();
45
50         void changeFacing(direction::Facing facing);
51
56         void updateOldFacing();
57
63         std::vector<SnakeCell> getBody();
64
69         const std::vector<std::shared_ptr<arc::Object> getObjects() const;
70
79         bool hasPosition(int x, int y);
80
89         bool hasPrevPosition(int x, int y);
90
91     private :
92         int s_Xpos;
93         int s_Ypos;
94         direction::Facing s_facing;
95         direction::Facing s_OldFacing;
96         std::vector<SnakeCell> body;
97 };
98 }

```

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:
13         AGame(int score = 0);
14
19         ~AGame();
20
26         int getScore() const;
27

```

```

33         bool isRunning() const override;
34
40         virtual const std::vector<std::shared_ptr<arc::Object>> getObjects() const override;
41
42     protected:
43         int m_score;
44
45         bool m_isRunning;
46
47         std::vector<std::shared_ptr<arc::Object>> m_objects;
48
49 }; /* class AGame */
50
51 } /* 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         bool isStarting() const;
35
36         bool isSelectingGame() const;
37
38         void selectPreviousGame();
39
40         void selectNextGame();
41
42         void selectPreviousDisplay();
43

```

```

97         void selectNextDisplay();
98
99     private:
104         MenuProprieties m_props;
105
110         bool m_isStarting;
111
116         bool m_isSelectingGame;
117
122         std::vector<std::shared_ptr<arc::games::menu::MenuItem>> m_games;
123
128         std::vector<std::shared_ptr<arc::games::menu::MenuItem>> m_displays;
129
134         std::vector<std::shared_ptr<arc::games::menu::MenuItem>> m_ui;
135
140         void useCharacterEvent(arc::Events event);
141
142     }; /* class MenuGame */
143
144 } /* 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:
17         MenuItem(const std::string value, Vector pos, int size, Color color);
18
23         ~MenuItem() = default;
24
30         bool isSelected() const;
31
37         void setSelected(bool selected);
38
39     private:
40         bool m_selected;
41     }; /* MenuItem */
42
43
44 } /* 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 namespace arc::games {
2     class Food {
3     public:
4         Food();
5         ~Food();
6
12         int getXpos() const;
13
19         int getYpos() const;
20
26         void setPos(int x, int y);

```

```

27
28     private:
29         int pos_x;
30         int pos_y;
31     };
32 }

```

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         Snake snake;
36         int n_highScore;
37         int n_lives;
38         int n_timeLeft;
39         std::vector<std::string> n_map;
40         clock_t n_clock;
41         //todo store this in a pointer
42         Food food;
43         int n_speed;
44
45         void spawnFood();
46     };
47 } // 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" {
8      NcursesDisplay *createInstance();
9  } /* extern "C" */
10
11 }; /* 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;
15
16         arc::Events getEvent() const override;
17
18         void drawInterface(std::vector<std::shared_ptr<arc::Object>> objs);
19
20     private : void getTexture(const std::string fileName, int y, int x);
21               void printMiddle(int y, int x, const std::string text, arc::Color color);
22               arc::Color getSpriteColor(std::string line);
23
24               void drawBorder();
25
26 }
```

```

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;
43
44         arc::Events interpretKeyboardEvent(const SDL_KeyboardEvent& event) const;
45
46     }; /* class Sdl2Display */
47
48 }; /* 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> obj> override;
20
21     void drawInterface(std::vector<std::shared_ptr<arc::Object> obj> override;
22
23     arc::Events getEvent() const override;
24
25     void placeObjectOnBoard(std::shared_ptr<arc::Object> obj);
26
27 private:
28     std::shared_ptr<sf::Texture> getTexture(const std::string& name);
29
30     void drawSprite(std::shared_ptr<arc::Object> obj);
31
32     void drawText(std::shared_ptr<arc::Object> obj);
33
34     std::shared_ptr<sf::RenderWindow> m_window;
35
36     std::map<std::string, std::shared_ptr<sf::Texture> m_textures;
37
38     std::unique_ptr<sf::Font> m_font;
39
40     arc::Events interpretKeyboardEvent(const sf::Event::KeyEvent& event) const;
41
42 }; /* class SfmlDisplay */
43
44 }; /* namespace arc::display */

```

5.22 Color.hpp

```

1 #include <cstdint>
2 #include <iostream>
3
4 #pragma once
5
6 namespace arc {
7
8     struct Color {
9         enum ColorType
10         {
11             RED,
12             GREEN,
13             BLUE,
14             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
35     Color(uint8_t r, uint8_t g, uint8_t b, uint8_t a, ColorType color);
36
37     Color(ColorType type);
38 }; /* struct Color */
39
40 } /* namespace arc */
41
42 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
13     class Core {
14     public:
15
16         Core(const std::string &lib);
17
18         ~Core();
19
20         std::unique_ptr<arc::display::IDisplayModule> getDisplay() const;
21
22         void run();
23
24         const std::string &getGameName() const;
25
26         const std::string &getDisplayName() const;
27
28         bool useEvent(arc::Events event);
29
30         void update();
31
32         void nextGame();
33
34         void previousGame();
35
36         void nextDisplay();
37
38         void previousDisplay();
39
40     private:
41
42         arc::DLLoader<arc::games::IGameModule> c_game;
43
44         arc::DLLoader<arc::display::IDisplayModule> c_display;
45
46         std::string currentDisplay;
47
48         std::string currentGame;
49
50         std::vector<std::string> c_games;
51
52         std::vector<std::string> c_displays;
53
54         std::string c_username;
55
56         std::vector<std::shared_ptr<arc::Object>> c_interface;
57
58         std::unique_ptr<arc::utils::HighscoreHandler> c_highscore;
59
60         int c_score;

```

```

157
158     }; /* class Core */
159
160 } /* namespace arc */

```

5.24 Error.hpp

```

1 #include <exception>
2 #include <string>
3
4 #pragma once
5
6 namespace arc::display {
7
12     class Sdl2Error : public std::exception {
13     public:
19         Sdl2Error(const std::string& message);
20
25         ~Sdl2Error();
26
32         const char* what() const noexcept final;
33
34     private:
39         const std::string& e_message;
40
41     }; /* class Sdl2Error */
42
43 } /* namespace arc::display */

```

5.25 Error.hpp

```

1 #include <exception>
2 #include <string>
3
4 #pragma once
5
6 namespace arc::display {
7
12 class SfmlError : public std::exception {
13 public:
19     SfmlError(const std::string& message);
20
25     ~SfmlError();
26
32     const char* what() const noexcept final;
33
34 private:
39     const std::string& e_message;
40
41 }; /* class SfmlError */
42
43 } /* namespace arc::display */

```

5.26 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 */

```

5.27 Events.hpp

```

1  #pragma once
2
3  namespace arc {
4
5      enum Events {
6          KeyUp,
7          KeyDown,
8          KeyRight,
9          KeyLeft,
10         KeyA,
11         KeyB,
12         KeyC,
13         KeyD,
14         KeyE,
15         KeyF,
16         KeyG,
17         KeyH,
18         KeyI,
19         KeyJ,
20         KeyK,
21         KeyL,
22         KeyM,
23         KeyN,
24         KeyO,
25         KeyP,
26         KeyQ,
27         KeyR,
28         KeyS,
29         KeyT,
30         KeyU,
31         KeyV,
32         KeyW,
33         KeyX,
34         KeyY,
35         KeyZ,
36         KeyEsc,
37         KeySpace,
38         KeyEnter,
39         KeyDel,
40         Key0,
41         Key1,
42         Key2,
43         Key3,
44         Key4,
45         Key5,
46         Key6,
47         Key7,
48         Key8,
49         Key9,
50         Exit,
51         None
52     }; /* enum Events */
53
54 } /* namespace arc*/

```

5.28 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 }

```

5.29 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 */

```

5.30 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         virtual ~IDisplayModule() = default;
14
15         virtual void drawObjects(std::vector<std::shared_ptr<arc::Object>> objs) = 0;
16
17         virtual void drawInterface(std::vector<std::shared_ptr<arc::Object>> objs) = 0;
18
19         virtual arc::Events getEvent() const = 0;
20     }; /* class IDisplayModule */
21
22 } /* namespace arc::display */

```

5.31 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     };
26
27 }

```

```

59         Vector getPosition() const;
60
61         void setValue(const std::string &value);
62
63         void setPosition(arc::Vector pos);
64
65     private:
66         Type m_type;
67         std::string m_value;
68         Vector m_position;
69     };
70
71     class Text : public Object {
72     public:
73         Text(const std::string content, Vector pos, int size, Color color);
74
75         ~Text() = default;
76
77         Color getColor() const;
78
79         int getSize() const;
80
81         void setColor(Color color);
82
83         void setSize(int size);
84
85     private:
86         Color m_color;
87         int m_size;
88     }; /* class Text */
89
90     class Sprite : public Object {
91     public:
92         Sprite(const std::string name, arc::Vector pos, int height = 0, int width = 0, arc::Vector
scale = arc::Vector(100, 100));
93
94         ~Sprite() = default;
95
96         int getHeight() const;
97
98         int getWidth() const;
99
100        Vector getScale() const;
101
102        int setHeight(int height);
103
104        int setWidth(int width);
105
106        void setScale(Vector scale);
107
108    private:
109        int m_height;
110        int m_width;
111        arc::Vector m_scale;
112    }; /* class Sprite */
113
114 } /* namespace arc */

```

5.32 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)

```

```

37         {
38             this->load(path);
39         }
40
41         DLoader(DLoader& other) = delete;
42
43         ~DLoader()
44         {
45             this->free();
46         }
47
48         void load(const std::string &path)
49         {
50             this->free();
51             this->l_lib = dlopen(path.c_str(), RTLD_NOW | RTLD_LOCAL);
52             if (!l_lib)
53                 throw new arc::Error("Could not open lib: " + path + ", " + dlerror());
54             void* func = dlsym(this->l_lib, "createInstance");
55             if (func == NULL)
56                 throw new arc::Error("Wrong lib format: " + path + ", " + dlerror());
57             l_instance = reinterpret_cast<T* (*)>(func)();
58         }
59
60         void free()
61         {
62             if (this->l_instance)
63                 delete l_instance;
64             if (this->l_lib)
65                 dlclose(this->l_lib);
66             l_instance = nullptr;
67             l_lib = nullptr;
68         }
69
70         T *getInstance() const
71         {
72             return l_instance;
73         }
74
75         T* operator->() const
76         {
77             return l_instance;
78         }
79
80         DLoader& operator=(DLoader& other) = delete;
81
82     private:
83         void *l_lib;
84
85         T* l_instance;
86
87 }; /* class DLoader */
88
89 } /* namespace arc */

```

5.33 FileParser.hpp

```

1 #include <array>
2 #include <string>
3 #include <vector>
4
5 #pragma once
6
7 namespace arc::utils {
8
9     class FileParser {
10     public:
11
12         static std::string getLibraryName(const std::string &pathToLib);
13
14         static std::vector<std::string> getLibrariesNames(const std::vector<std::string> libs);
15
16         static std::array<std::vector<std::string>, 2> getAllLibraries(const std::string& path =
17             "./lib/");
18     }; /* class FileParser */
19
20 } /* namespace arc::utils */

```

5.34 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
16         ~HighscoreHandler() = default;
17
18         std::vector<std::pair<std::string, int>> getHighscores() const;
19
20         void setHighscores(std::vector<std::pair<std::string, int>> highscores);
21
22         void addHighscore(const std::string& name, int score);
23
24         void saveHighscores();
25
26         std::vector<std::shared_ptr<arc::Object>> toObjects();
27
28     private:
29         std::vector<std::pair<std::string, int>> m_highscores;
30         std::string m_filePath;
31
32     }; /* class HighscoreHandler */
33
34 } /* namespace arc::utils */

```

5.35 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, [21](#)
 - drawObjects, [22](#)
 - getEvent, [22](#)
- arc::display::NcursesDisplay, [31](#)
 - drawInterface, [31](#)
 - drawObjects, [31](#)
 - getEvent, [32](#)
- arc::display::Sdl2Display, [38](#)
 - drawInterface, [38](#)
 - drawObjects, [38](#)
 - getEvent, [39](#)
 - placeObjectOnBoard, [39](#)
- arc::display::Sdl2Error, [39](#)
 - Sdl2Error, [40](#)
 - what, [40](#)
- arc::display::SfmlDisplay, [41](#)
 - drawInterface, [41](#)
 - drawObjects, [42](#)
 - getEvent, [42](#)
 - placeObjectOnBoard, [42](#)
- arc::display::SfmlError, [43](#)
 - SfmlError, [43](#)
 - what, [44](#)
- arc::DLLoader< T >, [13](#)
 - 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, [29](#)
 - getlife, [30](#)
 - isDead, [30](#)
 - setlife, [30](#)
- arc::games::centipede::Player, [36](#)
 - deleteShoot, [37](#)
 - getShoots, [37](#)
- arc::games::centipede::Shoot, [44](#)
 - checkHit, [45](#)
 - getHit, [45](#)
 - isHit, [46](#)
 - Shoot, [45](#)
- arc::games::centipede::Snake, [46](#)
 - getCells, [47](#)
 - Snake, [46, 47](#)
- arc::games::centipede::SnakeCell, [50](#)
 - getCellType, [51](#)
 - getDirection, [51](#)
 - hasDownMushroom, [51](#)
 - hasLeftMushroom, [52](#)
 - hasRightMushroom, [52](#)
 - hit, [52](#)
 - pickADir, [53](#)
 - pickASideDir, [53](#)
 - setCellType, [53](#)
 - setDirection, [53](#)
 - SnakeCell, [51](#)
- arc::games::Food, [18](#)
 - getXpos, [18](#)
 - getYpos, [19](#)
 - setPos, [19](#)
- arc::games::IGameModule, [22](#)
 - getObjects, [23](#)
 - isRunning, [24](#)
 - update, [24](#)
 - useEvent, [24, 25](#)
- arc::games::menu::MenuItem, [27](#)
 - isSelected, [28](#)
 - MenuItem, [28](#)
 - setSelected, [29](#)
- arc::games::MenuGame, [25](#)
 - getObjects, [26](#)
 - getProps, [26](#)
 - isStarting, [26](#)
 - update, [27](#)
 - useEvent, [27](#)
- arc::games::MenuProprieties, [29](#)
- arc::games::NibblerGame, [32](#)
 - getObjects, [33](#)

- update, [33](#)
 - useEvent, [33](#)
- arc::games::Snake, [48](#)
 - getBody, [48](#)
 - getXpos, [48](#)
 - getYpos, [48](#)
 - hasPosition, [49](#)
 - hasPrevPosition, [49](#)
- arc::games::SnakeCell, [54](#)
 - getPrevXpos, [54](#)
 - getPrevYpos, [55](#)
 - getXpos, [55](#)
 - getYpos, [55](#)
- arc::Object, [34](#)
 - getPosition, [35](#)
 - getType, [35](#)
 - getValue, [35](#)
 - Object, [34](#)
 - setPosition, [35](#)
 - setValue, [36](#)
- arc::Sprite, [56](#)
 - getHeight, [57](#)
 - getScale, [57](#)
 - getWidth, [57](#)
 - setHeight, [57](#)
 - setScale, [58](#)
 - setWidth, [58](#)
 - Sprite, [56](#)
- arc::Text, [58](#)
 - getColor, [59](#)
 - getSize, [60](#)
 - setColor, [60](#)
 - setSize, [60](#)
 - Text, [59](#)
- arc::utils::FileParser, [17](#)
 - getAllLibraries, [17](#)
 - getLibraryName, [18](#)
- arc::utils::HighscoreHandler, [19](#)
 - addHighscore, [20](#)
 - getHighscores, [20](#)
 - setHighscores, [20](#)
- arc::Vector, [61](#)
- checkHit
 - arc::games::centipede::Shoot, [45](#)
- Color
 - arc::Color, [11](#)
- deleteShoot
 - arc::games::centipede::Player, [37](#)
- DLLoader
 - arc::DLLoader< T >, [14](#)
- drawInterface
 - arc::display::IDisplayModule, [21](#)
 - arc::display::NcursesDisplay, [31](#)
 - arc::display::Sdl2Display, [38](#)
 - arc::display::SfmlDisplay, [41](#)
- drawObjects
 - arc::display::IDisplayModule, [22](#)
 - arc::display::NcursesDisplay, [31](#)
 - arc::display::Sdl2Display, [38](#)
 - arc::display::SfmlDisplay, [42](#)
- Error
 - arc::Error, [16](#)
- getAllLibraries
 - arc::utils::FileParser, [17](#)
- getBody
 - arc::games::Snake, [48](#)
- getCells
 - arc::games::centipede::Snake, [47](#)
- getCellType
 - arc::games::centipede::SnakeCell, [51](#)
- getColor
 - arc::Text, [59](#)
- getDirection
 - arc::games::centipede::SnakeCell, [51](#)
- 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, [57](#)
- getHighscores
 - arc::utils::HighscoreHandler, [20](#)
- getHit
 - arc::games::centipede::Shoot, [45](#)
- 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](#)
 - arc::games::MenuGame, [26](#)
 - arc::games::NibblerGame, [33](#)
- getPosition
 - arc::Object, [35](#)
- getPrevXpos
 - arc::games::SnakeCell, [54](#)
- getPrevYpos
 - arc::games::SnakeCell, [55](#)
- getProps
 - arc::games::MenuGame, [26](#)
- getScale
 - arc::Sprite, [57](#)
- getScore

- arc::games::AGame, 8
- getShoots
 - arc::games::centipede::Player, 37
- getSize
 - arc::Text, 60
- getType
 - arc::Object, 35
- getValue
 - arc::Object, 35
- getWidth
 - arc::Sprite, 57
- getXpos
 - arc::games::Food, 18
 - arc::games::Snake, 48
 - arc::games::SnakeCell, 55
- getYpos
 - arc::games::Food, 19
 - arc::games::Snake, 48
 - arc::games::SnakeCell, 55
- hasDownMushroom
 - arc::games::centipede::SnakeCell, 51
- hasLeftMushroom
 - arc::games::centipede::SnakeCell, 52
- hasPosition
 - arc::games::Snake, 49
- hasPrevPosition
 - arc::games::Snake, 49
- hasRightMushroom
 - arc::games::centipede::SnakeCell, 52
- hit
 - arc::games::centipede::SnakeCell, 52
- isDead
 - arc::games::centipede::Mushroom, 30
- isHit
 - arc::games::centipede::Shoot, 46
- isRunning
 - arc::games::AGame, 8
 - arc::games::IGameModule, 24
- isSelected
 - arc::games::menu::MenuItem, 28
- isStarting
 - arc::games::MenuGame, 26
- lib/games/centipede/includes/Centipede.hpp, 63
- lib/games/centipede/includes/CentipedeGame.hpp, 63
- lib/games/centipede/includes/Mushroom.hpp, 64
- lib/games/centipede/includes/Player.hpp, 64
- lib/games/centipede/includes/Snake.hpp, 65
- lib/games/includes/AGame.hpp, 66
- lib/games/menu/includes/Menu.hpp, 67
- lib/games/menu/includes/MenuGame.hpp, 67
- lib/games/menu/includes/MenuItem.hpp, 68
- lib/games/nibbler/includes/Direction.hpp, 68
- lib/games/nibbler/includes/Food.hpp, 68
- lib/games/nibbler/includes/Nibbler.hpp, 69
- lib/games/nibbler/includes/NibblerGame.hpp, 69
- lib/games/nibbler/includes/Snake.hpp, 66
- lib/games/nibbler/includes/SnakeCell.hpp, 70
- lib/graphics/ncurses/includes/Ncurses.hpp, 70
- lib/graphics/ncurses/includes/NcursesDisplay.hpp, 70
- lib/graphics/sdl2/includes/Error.hpp, 74
- lib/graphics/sdl2/includes/Sdl2.hpp, 71
- lib/graphics/sdl2/includes/Sdl2Display.hpp, 71
- lib/graphics/sfml/includes/Error.hpp, 74
- lib/graphics/sfml/includes/Sfml.hpp, 72
- lib/graphics/sfml/includes/SfmlDisplay.hpp, 72
- load
 - arc::DLLoader< T >, 15
- MenuItem
 - arc::games::menu::MenuItem, 28
- Object
 - arc::Object, 34
- operator->
 - arc::DLLoader< T >, 15
- pickADir
 - arc::games::centipede::SnakeCell, 53
- pickASideDir
 - arc::games::centipede::SnakeCell, 53
- placeObjectOnBoard
 - arc::display::Sdl2Display, 39
 - arc::display::SfmlDisplay, 42
- Sdl2Error
 - arc::display::Sdl2Error, 40
- setCellType
 - arc::games::centipede::SnakeCell, 53
- setColor
 - arc::Text, 60
- setDirection
 - arc::games::centipede::SnakeCell, 53
- setHeight
 - arc::Sprite, 57
- setHighscores
 - arc::utils::HighscoreHandler, 20
- setlife
 - arc::games::centipede::Mushroom, 30
- setPos
 - arc::games::Food, 19
- setPosition
 - arc::Object, 35
- setScale
 - arc::Sprite, 58
- setSelected
 - arc::games::menu::MenuItem, 29
- setSize
 - arc::Text, 60
- setValue
 - arc::Object, 36
- setWidth
 - arc::Sprite, 58
- SfmlError
 - arc::display::SfmlError, 43
- Shoot

- arc::games::centipede::Shoot, [45](#)
- Snake
 - arc::games::centipede::Snake, [46](#), [47](#)
- SnakeCell
 - arc::games::centipede::SnakeCell, [51](#)
- splitSnake
 - arc::games::Centipede, [9](#)
- Sprite
 - arc::Sprite, [56](#)
- src/includes/Color.hpp, [72](#)
- src/includes/Core.hpp, [73](#)
- src/includes/Error.hpp, [74](#)
- src/includes/Events.hpp, [75](#)
- src/includes/IGameModule.hpp, [75](#)
- src/includes/Interfaces/IDisplayModule.hpp, [76](#)
- src/includes/Interfaces/IGameModule.hpp, [76](#)
- src/includes/Object.hpp, [76](#)
- src/includes/Utils/DLLoader.hpp, [77](#)
- src/includes/Utils/FileParser.hpp, [78](#)
- src/includes/Utils/HighscoreHandler.hpp, [78](#)
- src/includes/Vector.hpp, [79](#)
- Text
 - arc::Text, [59](#)
- 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, [24](#), [25](#)
 - arc::games::MenuGame, [27](#)
 - arc::games::NibblerGame, [33](#)
- what
 - arc::display::Sdl2Error, [40](#)
 - arc::display::SfmlError, [44](#)
 - arc::Error, [17](#)