Arcade

Generated by Doxygen 1.9.3

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

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

Chapter 2

Class Index

2.1 Class List

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

Chapter 3

File Index

3.1 File List

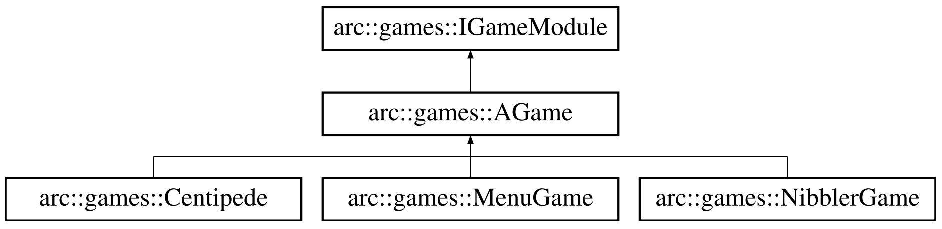
Here is a list of all documented files with brief descriptions:

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

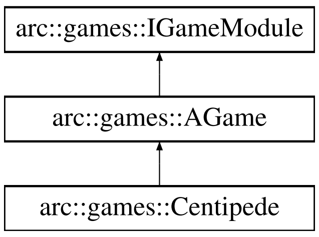
Implements arc::games::IGameModule.

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

snake	Snake to split
cell	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

event	

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

r	red value (0-255)
g	green value) (0-255)
b	blue value (0-255)
a	opacity value (0-255)
color	

4.3.2.2 Color() [2/2]

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

Construct a new Color object.

Parameters

type	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

T	IGameModule or IDisplayModule

4.5.2 Constructor & Destructor Documentation

4.5.2.1 DLLoader()

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

Parameters

path	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

path	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:

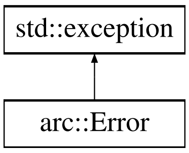
- src/includes/Utils/DLLoader.hpp

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

message	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

path	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

pathToLib	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

name	Name of the player
score	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:

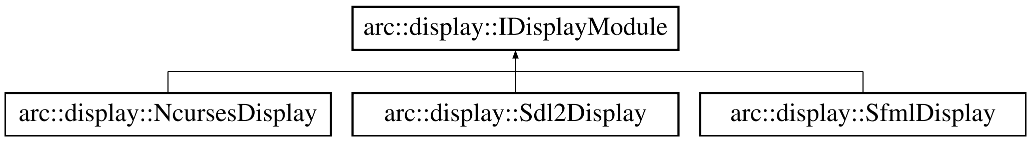
- src/includes/Utils/HighscoreHandler.hpp
- src/Utils/HighscoreHandler.cpp

4.10 arc::display::IDisplayModule Class Reference

Display module interface.

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

Inheritance diagram for arc::display::IDisplayModule:


arc::display::IDisplayModule
arc::display::NcursesDisplay
arc::display::Sdl2Display
arc::display::SfmlDisplay

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

objs	

Implemented in arc::display::NcursesDisplay, arc::display::Sdl2Display, and arc::display::SfmlDisplay.

4.10.2.3 getEvent()

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

get any event

Returns

const arc::Events

Implemented in arc::display::NcursesDisplay, arc::display::Sdl2Display, and arc::display::SfmlDisplay.

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

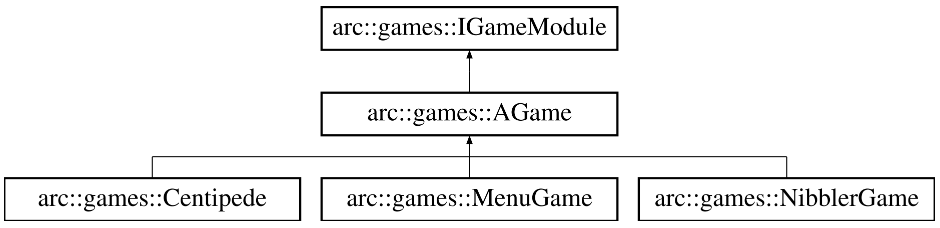
- src/includes/Interfaces/IDisplayModule.hpp

4.11 arc::games::IGameModule Class Reference

Game module interface.

`#include <IGameModule.hpp>`

Inheritance diagram for arc::games::IGameModule:



Public Member Functions

- ∼**IGameModule** ()=default

  Destroy the IDisplayModule object.
- virtual void useEvent (arc::Events event)=0

  apply the current event
- virtual void update ()=0

  update the game
- virtual const std::vector< std::shared_ptr< Object > > getObjects () const =0

  Get the objects to draw.
- virtual ∼**IGameModule** ()=default

  Destroy the IDisplayModule object.
- virtual void useEvent (arc::Events event)=0

  Apply the current event.
- virtual const std::vector< std::shared_ptr< arc::Object > > getObjects () const =0

  Get the objects to draw.
- virtual bool isRunning () const =0

  Tell if game is running or not.
- virtual void update ()=0

  Update game's entities.

4.11.1 Detailed Description

Game module interface.

4.11.2 Member Function Documentation

4.11.2.1 getObjects() [1/2]

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

Get the objects to draw.

Returns

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

Implemented in arc::games::MenuGame, arc::games::NibblerGame, arc::games::Centipede, and arc::games::AGame.

4.11.2.2 getObjects() [2/2]

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

Get the objects to draw.

Returns

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

Implemented in arc::games::MenuGame, arc::games::NibblerGame, arc::games::Centipede, and arc::games::AGame.

4.11.2.3 isRunning()

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

Tell if game is running or not.

Returns

true or false

Implemented in arc::games::AGame.

4.11.2.4 update() [1/2]

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

update the game

Implemented in arc::games::MenuGame, arc::games::NibblerGame, and arc::games::Centipede.

4.11.2.5 update() [2/2]

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

Update game's entities.

Implemented in arc::games::MenuGame, arc::games::NibblerGame, and arc::games::Centipede.

4.11.2.6 useEvent() [1/2]

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

apply the current event

Parameters

event	

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

event	

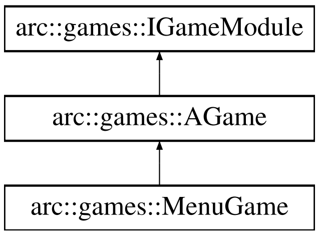
Implemented in arc::games::MenuGame, arc::games::NibblerGame, and arc::games::Centipede.

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

event	

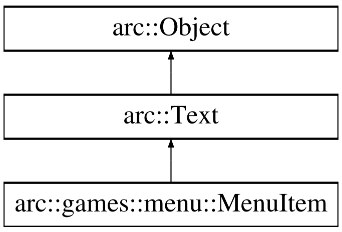
Implements arc::games::IGameModule.

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

value	
pos	
size	
color	

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

selected	

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

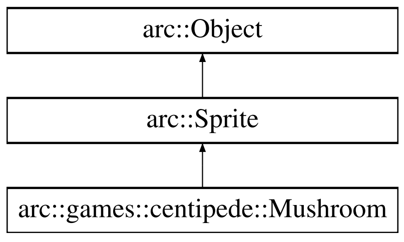
- std::string **username**
- std::string **gamelib**
- std::string **graphicslib**

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

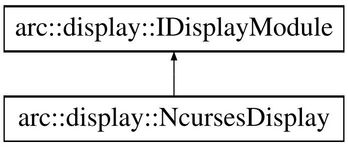
life	

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

objs	

Implements arc::display::IDisplayModule.

4.16.1.3 getEvent()

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

get any event

Returns

const arc::Events

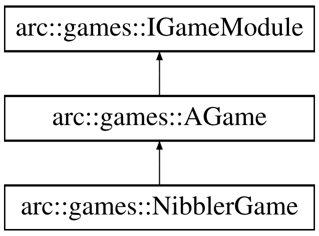
Implements arc::display::IDisplayModule.

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

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

4.17 arc::games::NibblerGame Class Reference

Inheritance diagram for arc::games::NibblerGame:



Public Member Functions

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

Additional Inherited Members

4.17.1 Member Function Documentation

4.17.1.1 getObjects()

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

Get the game objects.

Returns

Game objects

Reimplemented from arc::games::AGame.

4.17.1.2 update()

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

Updates game's entities.

Implements arc::games::IGameModule.

4.17.1.3 useEvent()

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

Apply the current event.

Parameters

event	

Implements arc::games::IGameModule.

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

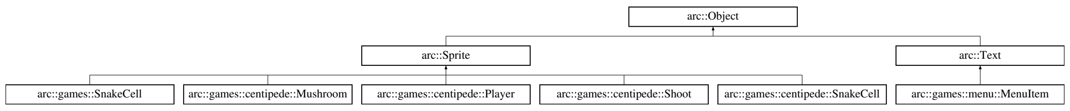
- lib/games/nibbler/includes/NibblerGame.hpp
- lib/games/nibbler/NibblerGame.cpp

4.18 arc::Object Class Reference

Represents a drawable object.

`#include <Object.hpp>`

Inheritance diagram for arc::Object:


arc::Object
arc::Sprite
arc::Text
arc::games::SnakeCell
arc::games::centipede::Mushroom
arc::games::centipede::Player
arc::games::centipede::Shoot
arc::games::centipede::SnakeCell
arc::games::menu::MenuItem

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

t	Type of the object
value	Value of the object
pos	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

pos	

4.18.3.5 setValue()

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

Setter for the value of the object.

Parameters

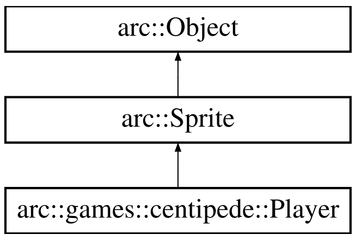
value	Value of the object

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

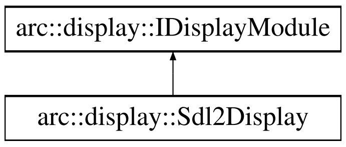
std::vector<std::shared_ptr<arc::games::centipede::Shoot>>

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 Sld2Display 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

objs	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

objs	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 occured (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

obj	object to be placed

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

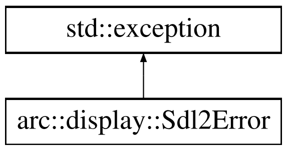
- lib/graphics/sdl2/includes/Sdl2Display.hpp
- lib/graphics/sdl2/Sdl2Display.cpp

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

message	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

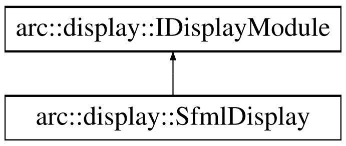
const char∗

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 Sld2Display 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

objs	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

objs	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 occured (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

obj	object to be placed

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

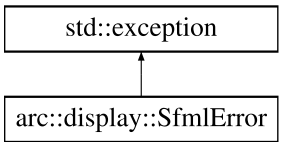
- lib/graphics/sfml/includes/SfmlDisplay.hpp
- lib/graphics/sfml/SfmlDisplay.cpp

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

message	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

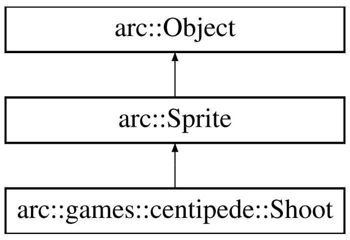
const char∗

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↩ ::vector< 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

x	
y	

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

mushrooms	
snakes	

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

mushrooms	
snakes	

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

size	size of the snake
x	position of the snake on the x axis
y	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

cells	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

x	X position
y	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

x	X position
y	Y position

Returns

true

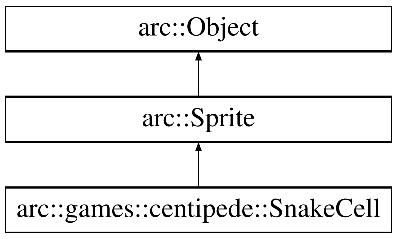
false

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

x	position of the Cell on the x axis
y	position of the Cell on the y axiss
type	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

mushrooms	

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

mushrooms	

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

mushrooms	

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

mushrooms	

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

mushrooms	= 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 oposite direction

Parameters

mushrooms	

4.27.2.9 setCellType()

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

Setter for type of the cell.

Parameters

type	

4.27.2.10 setDirection()

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

Setter for the direction of the Cell.

Parameters

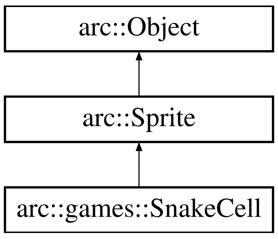
dir	

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:

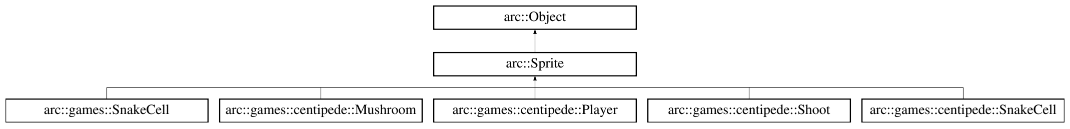
- lib/games/nibbler/includes/SnakeCell.hpp
- lib/games/nibbler/SnakeCell.cpp

4.29 arc::Sprite Class Reference

Represents a sprite object.

`#include <Object.hpp>`

Inheritance diagram for arc::Sprite:


arc::Object
arc::Sprite
arc::games::SnakeCell
arc::games::centipede::Mushroom
arc::games::centipede::Player
arc::games::centipede::Shoot
arc::games::centipede::SnakeCell

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

name	Name of the sprite
pos	Position of the sprite
height	Height of the sprite
width	Width of the sprite
scale	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

height	Height of the sprite

4.29.3.5 setScale()

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

Setter for the scale of the sprite.

Parameters

scale	Scale of the sprite

4.29.3.6 setWidth()

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

Setter for the width of the sprite.

Parameters

width	Width of the sprite

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

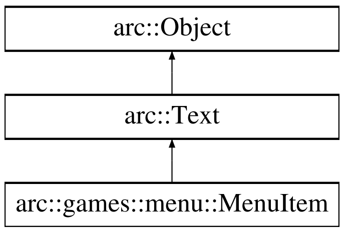
- src/includes/Object.hpp
- src/Object.cpp

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

content	Content of the text
pos	Position of the text
size	Size of the text
color	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

color	Color of the text

4.30.3.4 setSize()

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

Setter for the size of the text.

Parameters

size	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

```
#include <CentipedeGame.hpp>

#include <exception>
#include <string>

#pragma once

namespace arc::games
{

    extern "C"
    {
        Centipede *createInstance();
    }; /* extern "C" */

}; /* namespace arc::display */
```

5.2 CentipedeGame.hpp

```
#pragma once

#include <AGame.hpp>
#include <Mushroom.hpp>
#include <Player.hpp>
#include <Snake.hpp>

#include <ctime>
#include <iostream>
#include <memory>
#include <vector>

namespace arc::games {

    class Centipede : public arc::games::AGame {
        public:
            Centipede();

            ~Centipede();

            void useEvent(arc::Events event) override;

            void update() override;

            const std::vector<std::shared_ptr<arc::Object» getObjects() const override;

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

        private :
            std::vector<std::shared_ptr<arc::games::centipede::Snake» snakes;
            std::shared_ptr<arc::games::centipede::Player> player;
            std::vector<std::shared_ptr<arc::games::centipede::Mushroom» mushrooms;
            std::clock_t clock;
            std::clock_t shootClock;
            std::clock_t shootMoveClock;
    };
};
```

5.3 Mushroom.hpp

```cpp
/*
** EPITECH PROJECT, 2022
** B-OOP-400-LYN-4-1-arcade-marvin.flamand
** File description:
** Mushroom
*/

#pragma once

#include <Object.hpp>

namespace arc::games::centipede {

    class Mushroom : public arc::Sprite
    {
    public:
        Mushroom(int x, int y);
        ~Mushroom();

        void update();

        void setlife(int life);

        int getlife();

        void checkDead();

        bool isDead() const;

    private:
        int life;
        bool m_isDead;
    };
}
```

5.4 Player.hpp

```cpp
/*
** EPITECH PROJECT, 2022
** B-OOP-400-LYN-4-1-arcade-marvin.flamand
** File description:
** Player
*/

#pragma once

#include <Object.hpp>
#include <Snake.hpp>
#include <Mushroom.hpp>
#include <iostream>

namespace arc::games::centipede {

    class Shoot : public arc::Sprite
    {
        public:
            Shoot(int x, int y);

            ~Shoot();

            void Update();

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

            void checkHit(std::vector<std::shared_ptr<arc::games::centipede::Mushroom» mushrooms,
    std::vector<std::shared_ptr<arc::games::centipede::Snake» snakes);

            bool isHit() const;

        private:
            bool m_isHit;
    };
    class Player : public arc::Sprite {
        public:
            Player();

            ~Player();

            enum Direction {LEFT, UP, RIGHT, DOWN, STAY};

            void move(Direction dir);

            void createShoot();
             void update(std::vector<std::shared_ptr<arc::games::centipede::Mushroom» mushrooms,
       std::vector<std::shared_ptr<arc::games::centipede::Snake» snakes);

             std::vector<std::shared_ptr<arc::games::centipede::Shoot» getShoots();

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

         private:
             Direction dir;

             std::vector<std::shared_ptr<arc::games::centipede::Shoot» shoots;
     };
}
```

5.5 Snake.hpp

```
#pragma once

#include <Object.hpp>

#include <ctime>
#include <iostream>
#include <vector>
#include <memory>
#include <Mushroom.hpp>
namespace arc::games::centipede  {

    class SnakeCell : public arc::Sprite
    {
        public:
            enum Type { HEAD, BODY };
            enum Direction { DOWN, LEFT, RIGHT };
            SnakeCell(int x, int y, Type type, Direction dir = DOWN);

            ~SnakeCell();

            void move();

            void update();

            void hit(std::vector<std::shared_ptr<arc::games::centipede::Mushroom» mushrooms);

            void pickADir(std::vector<std::shared_ptr<arc::games::centipede::Mushroom» mushrooms);

            void pickASideDir(std::vector<std::shared_ptr<arc::games::centipede::Mushroom» mushrooms);

            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;

             Type getCellType() const;

             void setCellType(Type type);

             void setDirection(Direction dir);


         private:
             int x;
             int y;
             int frame;
             Type type;
             Direction dir;
     };

     std::string &operator«(std::string& s, arc::games::centipede::SnakeCell::Direction& d);

     class Snake
     {
     public:
         Snake(int size, int x, int y);

         Snake(std::vector<std::shared_ptr<arc::games::centipede::SnakeCell» cells);

         ~Snake();
         std::vector<std::shared_ptr<arc::games::centipede::SnakeCell» getCells() const;

         void setCells(std::vector<std::shared_ptr<arc::games::centipede::SnakeCell» cells);

         void update();

         void checkHit(std::vector<std::shared_ptr<arc::games::centipede::Mushroom» mushrooms);

     private:
         std::vector<std::shared_ptr<arc::games::centipede::SnakeCell» cells;
     };
 }
```

5.6 Snake.hpp

```
/*
** EPITECH PROJECT, 2022
** Arcade
** File description:
** snake
*/

#pragma once
#include "SnakeCell.hpp"
#include "Direction.hpp"
#include <vector>

namespace arc::games {
class Snake {
    public:
        Snake(int x, int y);
        ~Snake();

        void moveSnake();

        void eat();

        int getXpos();

        int getYpos();

        void changeFacing(direction::Facing facing);

        void updateOldFacing();

        std::vector<SnakeCell> getBody();

        const std::vector<std::shared_ptr<arc::Object» getObjects() const;

        bool hasPosition(int x, int y);

        bool hasPrevPosition(int x, int y);

    private :
        int s_Xpos;
        int s_Ypos;
        direction::Facing s_facing;
        direction::Facing s_OldFacing;
        std::vector<SnakeCell> body;
};
}
```

5.7 AGame.hpp

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

#pragma once

namespace arc::games {

class AGame : public arc::games::IGameModule {
    public:
        AGame(int score = 0);

        ~AGame();

        int getScore() const;

        bool isRunning() const override;

        virtual const std::vector<std::shared_ptr<arc::Object» getObjects() const override;

    protected:
        int m_score;

        bool m_isRunning;


        std::vector<std::shared_ptr<arc::Object» m_objects;

}; /* class AGame */

} /* namespace arc::games */
```

5.8 Menu.hpp

```
#include <MenuGame.hpp>

#include <exception>
#include <string>

#pragma once

namespace arc::games {

    extern "C" {
        MenuGame *createInstance();
    }; /* extern "C" */

}; /* namespace arc::display */
```

5.9 MenuGame.hpp

```
#include <memory>
#include <string>


#include <AGame.hpp>
#include <MenuItem.hpp>

#pragma once


namespace arc::games {

    struct MenuProprieties {
        std::string username;
        std::string gamelib;
        std::string graphicslib;
    };

    class MenuGame : public AGame {
        public:
            MenuGame();

            ~MenuGame();

            void useEvent(arc::Events event) final;

            void update() final;

            const std::vector<std::shared_ptr<arc::Object» getObjects() const final;

            const MenuProprieties getProps() const {
                return m_props;
            }

            bool isStarting() const;

            bool isSelectingGame() const;

            void selectPreviousGame();

            void selectNextGame();

            void selectPreviousDisplay();

            void selectNextDisplay();

        private:
             MenuProprieties m_props;

             bool m_isStarting;

             bool m_isSelectingGame;

             std::vector<std::shared_ptr<arc::games::menu::MenuItem>> m_games;

             std::vector<std::shared_ptr<arc::games::menu::MenuItem>> m_displays;

             std::vector<std::shared_ptr<arc::games::menu::MenuItem>> m_ui;

             void useCharacterEvent(arc::Events event);

    }; /* class MenuGame */

} /* namespace arc::games */
```

5.10 MenuItem.hpp

```
#pragma once

#include <Object.hpp>

namespace arc::games::menu {

    class MenuItem : public arc::Text {
        public:
             MenuItem(const std::string value, Vector pos, int size, Color color);

             ~MenuItem() = default;

             bool isSelected() const;

             void setSelected(bool selected);

         private:
             bool m_selected;
    }; /* MenuItem */


} /* namespace arc::games::menu */
```

5.11 Direction.hpp

```
/*
** EPITECH PROJECT, 2022
** B-OOP-400-LYN-4-1-arcade-marvin.flamand
** File description:
** Direction
*/

#pragma once

namespace direction {
    enum Facing { UP, RIGHT, DOWN, LEFT };
    enum axis { HORIZONTAL, VERTICAL };
}
```

5.12 Food.hpp

```
namespace arc::games {
    class Food {
        public:
            Food();
            ~Food();

             int getXpos() const;

             int getYpos() const;

             void setPos(int x, int y);

        private:
            int pos_x;
            int pos_y;
    };
}
```

5.13 Nibbler.hpp

```
#include <NibblerGame.hpp>

#include <exception>
#include <string>

#pragma once

namespace arc::games {

extern "C" {
NibblerGame *createInstance();
}; /* extern "C" */

}; /* namespace arc::display */
```

5.14 NibblerGame.hpp

```
/*
** EPITECH PROJECT, 2022
** Arcade
** File description:
** Nibbler
*/

#pragma once
#include "Direction.hpp"
#include "Snake.hpp"
#include <AGame.hpp>
#include <fcntl.h>
#include <fstream>
#include <iostream>
#include <ncurses.h>
#include <stdlib.h>
#include <string>
#include <time.h>
#include <Food.hpp>

namespace arc::games {
class NibblerGame : public AGame {
public:
    NibblerGame();

    ~NibblerGame();

    void useEvent(arc::Events event) final;

    void update() final;

    const std::vector<std::shared_ptr<arc::Object>> getObjects() const final;


private:
    Snake snake;
    int n_highScore;
    int n_lives;
    int n_timeLeft;
    std::vector<std::string> n_map;
    clock_t n_clock;
    //todo store this in a pointer
    Food food;
    int n_speed;

    void spawnFood();
};
} // namespace arc::games
```

5.15 SnakeCell.hpp

```
/*
** EPITECH PROJECT, 2022
** Arcade
** File description:
** SnakeCell
*/

#pragma once
#include "Direction.hpp"
#include <memory>
#include "Object.hpp"

namespace arc::games {
class SnakeCell : public Sprite{
    public:
        SnakeCell(int x, int y);
        ~SnakeCell();

        void setPrevPos();

        int getXpos();

        int getYpos();

        int getPrevXpos();

        int getPrevYpos();

        void updateAxis();

    private :
        int sc_prevXpos;
        int sc_prevYpos;
};
}
```

5.16 Ncurses.hpp

```
#include "NcursesDisplay.hpp"

#pragma once

namespace arc::display {

extern "C" {
NcursesDisplay *createInstance();
} /* extern "C" */

}; /* namespace arc::display */
```

5.17 NcursesDisplay.hpp

```
#pragma once

#include "Interfaces/IDisplayModule.hpp"
#include <fcntl.h>
#include <fstream>
#include <ncurses.h>

namespace arc::display {
class NcursesDisplay : public arc::display::IDisplayModule {
        public:
            NcursesDisplay();
            ~NcursesDisplay();

            void drawObjects(std::vector<std::shared_ptr<arc::Object>> objs) override;

            arc::Events getEvent() const override;

            void drawInterface(std::vector<std::shared_ptr<arc::Object>> objs);

        private : void getTexture(const std::string fileName, int y, int x);
            void printMiddle(int y, int x, const std::string text, arc::Color color);
            arc::Color getSpriteColor(std::string line);

            void drawBorder();

            void printInterface(int y, int x, const std::string text, arc::Color color);

            void clearBoard();

    };
}
```

5.18 Sdl2.hpp

```
#include <Sdl2Display.hpp>

#include <exception>
#include <string>

#pragma once

namespace arc::display {

    extern "C" {
        Sdl2Display *createInstance();
    } /* extern "C" */

}; /* namespace arc::display */
```

5.19 Sdl2Display.hpp

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

#include <SDL2/SDL.h>
#include <SDL2/SDL_ttf.h>
#include <map>

#pragma once


namespace arc::display {

    class Sdl2Display : public IDisplayModule {
        public:
            Sdl2Display();

            Sdl2Display(Sdl2Display& other) = delete;

            ~Sdl2Display();

            void drawObjects(std::vector<std::shared_ptr<arc::Object» objs) override;

            void drawInterface(std::vector<std::shared_ptr<arc::Object» objs) override;

            arc::Events getEvent() const override;

            void placeObjectOnBoard(std::shared_ptr<arc::Object> obj);

            Sdl2Display& operator=(Sdl2Display& other) = delete;

        private:

            SDL_Texture *getTexture(const std::string& name);

            void drawSprite(std::shared_ptr<arc::Object> obj);

            void drawText(std::shared_ptr<arc::Object> obj);

            SDL_Window *m_window;

            SDL_Renderer *m_renderer;

            std::map<std::string, SDL_Texture*> m_textures;

            arc::Events interpretKeyboardEvent(const SDL_KeyboardEvent& event) const;
    }; /* class Sdl2Display */

}; /* namespace arc::display */
```

5.20 Sfml.hpp

```
#include <SfmlDisplay.hpp>
#include <exception>
#include <string>

#pragma once

namespace arc::display {

    extern "C" {
         SfmlDisplay *createInstance();
     } /* extern "C" */

}; /* namespace arc::display */
```

5.21 SfmlDisplay.hpp

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

#include <SFML/Audio.hpp>
#include <SFML/Graphics.hpp>
#include <SFML/Window.hpp>
#include <SFML/System.hpp>
#include <map>

#pragma once

namespace arc::display {

class SfmlDisplay : public IDisplayModule {
public:
    SfmlDisplay();

    ~SfmlDisplay();

    void drawObjects(std::vector<std::shared_ptr<arc::Object» objs) override;

    void drawInterface(std::vector<std::shared_ptr<arc::Object» objs) override;

    arc::Events getEvent() const override;

    void placeObjectOnBoard(std::shared_ptr<arc::Object> obj);

private:
    std::shared_ptr<sf::Texture> getTexture(const std::string& name);

    void drawSprite(std::shared_ptr<arc::Object> obj);

    void drawText(std::shared_ptr<arc::Object> obj);

    std::shared_ptr<sf::RenderWindow> m_window;

    std::map<std::string, std::shared_ptr<sf::Texture» m_textures;

    std::unique_ptr<sf::Font> m_font;

    arc::Events interpretKeyboardEvent(const sf::Event::KeyEvent& event) const;

}; /* class SfmlDisplay */

}; /* namespace arc::display */
```

5.22 Color.hpp

```
#include <cstdint>
#include <iostream>

#pragma once

namespace arc {

    struct Color {
        enum ColorType
        {
            RED,
            GREEN,
            BLUE,
            YELLOW,
            MAGENTA,
            CYAN,
            WHITE,
            BLACK,
        };

        uint8_t r;
        uint8_t g;
        uint8_t b;
        uint8_t a;
        ColorType color;

        Color(uint8_t r, uint8_t g, uint8_t b, uint8_t a, ColorType color);

        Color(ColorType type);
    }; /* struct Color */

} /* namespace arc */

std::ostream& operator«(std::ostream& os, arc::Color& c);
```

5.23 Core.hpp

```
#include <Interfaces/IGameModule.hpp>
#include <Interfaces/IDisplayModule.hpp>
#include <Utils/DLLoader.hpp>
#include <Utils/HighscoreHandler.hpp>

#include <memory>
#include <string>

#pragma once

namespace arc {

    class Core {
        public:

            Core(const std::string &lib);

            ~Core();

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

            void run();

            const std::string &getGameName() const;

            const std::string &getDisplayName() const;

            bool useEvent(arc::Events event);

            void update();

            void nextGame();

            void previousGame();

            void nextDisplay();

            void previousDisplay();

        private:

            arc::DLLoader<arc::games::IGameModule> c_game;

            arc::DLLoader<arc::display::IDisplayModule> c_display;

            std::string currentDisplay;

            std::string currentGame;

            std::vector<std::string> c_games;

            std::vector<std::string> c_displays;

            std::string c_username;

            std::vector<std::shared_ptr<arc::Object» c_interface;

            std::unique_ptr<arc::utils::HighscoreHandler> c_highscore;

            int c_score;

    }; /* class Core */

} /* namespace arc */
```

5.24 Error.hpp

```
#include <exception>
#include <string>

#pragma once

namespace arc::display {

    class Sdl2Error : public std::exception {
    public:
        Sdl2Error(const std::string& message);

        ~Sdl2Error();

        const char* what() const noexcept final;

    private:
        const std::string& e_message;

    }; /* class Sdl2Error */

} /* namespace arc::display */
```

5.25 Error.hpp

```
#include <exception>
#include <string>

#pragma once

namespace arc::display {

class SfmlError : public std::exception {
public:
    SfmlError(const std::string& message);

    ~SfmlError();

    const char* what() const noexcept final;

private:
    const std::string& e_message;

}; /* class SfmlError */

} /* namespace arc::display */
```

5.26 Error.hpp

```
#include <exception>
#include <string>

#pragma once

namespace arc {

    class Error : public std::exception {
        public:
            Error(const std::string &message);

            ~Error();

            const char *what() const noexcept final;

        protected:
            std::string e_message;
    }; /* class arc::Error */

} /* namespace arc */
```

5.27 Events.hpp

```
#pragma once

namespace arc {

enum Events {
    KeyUp,
    KeyDown,
    KeyRight,
    KeyLeft,
    KeyA,
    KeyB,
    KeyC,
    KeyD,
    KeyE,
    KeyF,
    KeyG,
    KeyH,
    KeyI,
    KeyJ,
    KeyK,
    KeyL,
    KeyM,
    KeyN,
    KeyO,
    KeyP,
    KeyQ,
    KeyR,
    KeyS,
    KeyT,
    KeyU,
    KeyV,
    KeyW,
    KeyX,
    KeyY,
    KeyZ,
    KeyEsc,
    KeySpace,
    KeyEnter,
    KeyDel,
    Key0,
    Key1,
    Key2,
    Key3,
    Key4,
    Key5,
    Key6,
    Key7,
    Key8,
    Key9,
    Exit,
    None
}; /* enum Events */

} /* namespace arc*/
```

5.28 IGameModule.hpp

```
#include "Events.hpp"
#include "Object.hpp"

#include <memory>
#include <vector>

#pragma once

namespace arc::games {

    class IGameModule {
        public:
            ~IGameModule() = default;

            virtual void useEvent(arc::Events event) = 0;

            virtual void update() = 0;

            virtual const std::vector<std::shared_ptr<Object>> getObjects() const = 0;
    }; /* class IGameModule */

}
```

5.29 IGameModule.hpp

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

#include <memory>
#include <vector>

#pragma once

namespace arc::games {

    class IGameModule {
        public:
            virtual ~IGameModule() = default;

            virtual void useEvent(arc::Events event) = 0;

            virtual const std::vector<std::shared_ptr<arc::Object» getObjects() const = 0;

            virtual bool isRunning() const = 0;

            virtual void update() = 0;
    }; /* class IGameModule */

} /* namespace arc::games */
```

5.30 IDisplayModule.hpp

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

#include <memory>
#include <vector>

#pragma once

namespace arc::display {

    class IDisplayModule {
        public:

            virtual ~IDisplayModule() = default;

            virtual void drawObjects(std::vector<std::shared_ptr<arc::Object» objs) = 0;

            virtual void drawInterface(std::vector<std::shared_ptr<arc::Object» objs) = 0;

            virtual arc::Events getEvent() const = 0;
    }; /* class IDisplayModule */

} /* namespace arc::display */
```

5.31 Object.hpp

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

#include <string>

#pragma once

namespace arc {

    class Object {
        public:
            enum class Type {
                TEXT,
                SPRITE
            };

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

            ~Object() = default;

            Type getType() const;

            const std::string &getValue() const;

```

```cpp
            Vector getPosition() const;

            void setValue(const std::string &value);

            void setPosition(arc::Vector pos);

            private:
                Type m_type;
                std::string m_value;
                Vector m_position;
    };

    class Text : public Object {
        public:
            Text(const std::string content, Vector pos, int size, Color color);

            ~Text() = default;

            Color getColor() const;

            int getSize() const;

            void setColor(Color color);

            void setSize(int size);

        private:
            Color m_color;
            int m_size;

    }; /* class Text */

    class Sprite : public Object {
        public:
            Sprite(const std::string name, arc::Vector pos, int height = 0, int width = 0, arc::Vector scale = arc::Vector(100, 100));

            ~Sprite() = default;

            int getHeight() const;

            int getWidth() const;

            Vector getScale() const;

            int setHeight(int height);

            int setWidth(int width);

            void setScale(Vector scale);

        private:
            int m_height;
            int m_width;
            arc::Vector m_scale;
    }; /* class Sprite */

} /* namespace arc */
```

5.32 DLLoader.hpp

```cpp
#include <Error.hpp>

#include <dlfcn.h>
#include <iostream>
#include <memory>
#include <string>

#include <MenuGame.hpp>

#pragma once

namespace arc {

    template <class T>
    class DLLoader {
        public:

            DLLoader() = default;

            DLLoader(const std::string& path)
                : l_lib(nullptr)
                , l_instance(nullptr)
            {
                this->load(path);
            }

            DLLoader(DLLoader& other) = delete;

            ~DLLoader()
            {
                this->free();
            }

            void load(const std::string &path)
            {
                this->free();
                this->l_lib = dlopen(path.c_str(), RTLD_NOW | RTLD_LOCAL);
                if (!l_lib)
                    throw new arc::Error("Could not open lib: " + path + ", " + dlerror());
                void* func = dlsym(this->l_lib, "createInstance");
                if (func == NULL)
                    throw new arc::Error("Wrong lib format: " + path + ", " + dlerror());
                l_instance = reinterpret_cast<T* (*)()>(func)();
            }

            void free()
            {
                if (this->l_instance)
                    delete l_instance;
                if (this->l_lib)
                    dlclose(this->l_lib);
                l_instance = nullptr;
                l_lib = nullptr;
            }

            T *getInstance() const
            {
                return l_instance;
            }

            T* operator->() const
            {
                return l_instance;
            }

            DLLoader& operator=(DLLoader& other) = delete;

        private:
            void *l_lib;

            T* l_instance;

    }; /* class DLOpener */

} /* namespace arc */
```

5.33 FileParser.hpp

```
#include <array>
#include <string>
#include <vector>

#pragma once

namespace arc::utils {

    class FileParser {
        public:

            static std::string getLibraryName(const std::string &pathToLib);

            static std::vector<std::string> getLibrariesNames(const std::vector<std::string> libs);

            static std::array<std::vector<std::string>, 2> getAllLibraries(const std::string& path =
    "./lib/");
    }; /* class FileParser */

} /* namespace arc::utils */
```

5.34 HighscoreHandler.hpp

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

```

#include <map>
#include <memory>
#include <string>
#include <vector>

#pragma once

namespace arc::utils {

    class HighscoreHandler {
        public:
            HighscoreHandler();

            ~HighscoreHandler() = default;

            std::vector<std::pair<std::string, int» getHighscores() const;

            void setHighscores(std::vector<std::pair<std::string, int» highscores);

            void addHighscore(const std::string& name, int score);

            void saveHighscores();

            std::vector<std::shared_ptr<arc::Object» toObjects();

            private:
                std::vector<std::pair<std::string, int» m_highscores;
                std::string m_filePath;

    }; /* class HighscoreHandler */

} /* namespace arc::utils */
```

5.35 Vector.hpp

```
#pragma once

namespace arc {

    struct Vector {
        int x;
        int y;
    }; /* struct Vector */

} /* namespace arc */
```

Index