

Arcade

1.0

Generated by Doxygen 1.8.15

1 How to implement your own game / library	1
2 Namespace Index	3
2.1 Namespace List	3
3 Hierarchical Index	5
3.1 Class Hierarchy	5
4 Class Index	7
4.1 Class List	7
5 Namespace Documentation	9
5.1 Arcade Namespace Reference	9
5.1.1 Detailed Description	9
5.2 Arcade::Display Namespace Reference	9
5.2.1 Detailed Description	10
5.3 Arcade::Exceptions Namespace Reference	10
5.3.1 Detailed Description	10
5.4 Arcade::Games Namespace Reference	10
5.4.1 Detailed Description	10
6 Class Documentation	11
6.1 Arcade::Display::ADisplayModule Class Reference	11
6.1.1 Detailed Description	11
6.1.2 Member Enumeration Documentation	12
6.1.2.1 SystemKeys	12
6.1.3 Constructor & Destructor Documentation	12
6.1.3.1 ADisplayModule()	12
6.1.4 Member Function Documentation	12
6.1.4.1 getLibName()	12
6.2 Arcade::Games::AGameModule Class Reference	13
6.2.1 Detailed Description	14
6.2.2 Constructor & Destructor Documentation	14
6.2.2.1 AGameModule()	14
6.2.3 Member Function Documentation	14
6.2.3.1 addToBestScores()	14
6.2.3.2 drawGameOver()	15
6.2.3.3 getBestScores()	15
6.2.3.4 getLibName()	15
6.2.3.5 getScore()	15
6.2.3.6 loadFromFile() [1/2]	15
6.2.3.7 loadFromFile() [2/2]	16
6.2.3.8 render()	16
6.2.3.9 saveToFile() [1/2]	16

6.2.3.10 saveToFile() [2/2]	17
6.2.3.11 setPlayerName()	17
6.3 Arcade::Exceptions::ArcadeException Class Reference	17
6.3.1 Detailed Description	18
6.3.2 Constructor & Destructor Documentation	18
6.3.2.1 ArcadeException()	18
6.3.3 Member Function Documentation	18
6.3.3.1 getComponent()	18
6.3.3.2 what()	19
6.4 Arcade::Exceptions::BadFileException Class Reference	19
6.4.1 Detailed Description	19
6.4.2 Constructor & Destructor Documentation	19
6.4.2.1 BadFileException()	19
6.5 Arcade::Exceptions::BadInstanciationException Class Reference	20
6.5.1 Detailed Description	20
6.5.2 Constructor & Destructor Documentation	20
6.5.2.1 BadInstanciationException()	20
6.6 Arcade::Games::Centipede Class Reference	21
6.6.1 Detailed Description	21
6.6.2 Member Function Documentation	21
6.6.2.1 update()	21
6.7 Arcade::Core Class Reference	22
6.7.1 Detailed Description	22
6.7.2 Constructor & Destructor Documentation	22
6.7.2.1 Core()	22
6.8 Arcade::DLInfos Struct Reference	22
6.8.1 Detailed Description	23
6.9 Arcade::DLLoader< T > Class Template Reference	23
6.9.1 Detailed Description	23
6.9.2 Member Function Documentation	24
6.9.2.1 getInstance()	24
6.9.2.2 getLibraries()	24
6.9.2.3 loadLibrary()	24
6.10 Arcade::Display::IDisplayModule Class Reference	25
6.10.1 Detailed Description	26
6.10.2 Member Enumeration Documentation	26
6.10.2.1 Colors	27
6.10.2.2 Keys	27
6.10.3 Member Function Documentation	28
6.10.3.1 getDelta()	28
6.10.3.2 getKeyCode()	28
6.10.3.3 getLibName()	29

6.10.3.4 isKeyPressed()	29
6.10.3.5 isKeyPressedOnce()	29
6.10.3.6 isOpen()	30
6.10.3.7 putCircle()	30
6.10.3.8 putFillCircle()	30
6.10.3.9 putFillRect()	31
6.10.3.10 putLine()	31
6.10.3.11 putPixel()	31
6.10.3.12 putRect()	32
6.10.3.13 putText()	32
6.10.3.14 setColor()	33
6.10.3.15 shouldBeRestarted()	33
6.10.3.16 shouldExit()	33
6.10.3.17 shouldGoToMenu()	34
6.10.3.18 switchToNextGame()	34
6.10.3.19 switchToNextLib()	34
6.10.3.20 switchToPreviousGame()	35
6.10.3.21 switchToPreviousLib()	35
6.11 Arcade::Games::IGameModule Class Reference	35
6.11.1 Detailed Description	36
6.11.2 Member Function Documentation	36
6.11.2.1 getBestScores()	36
6.11.2.2 getLibName()	37
6.11.2.3 getScore()	37
6.11.2.4 loadFromFile() [1/2]	37
6.11.2.5 loadFromFile() [2/2]	38
6.11.2.6 render()	38
6.11.2.7 saveToFile() [1/2]	38
6.11.2.8 saveToFile() [2/2]	39
6.11.2.9 setPlayerName()	39
6.11.2.10 update()	39
6.12 Arcade::Exceptions::InvalidLibraryException Class Reference	39
6.12.1 Detailed Description	40
6.12.2 Constructor & Destructor Documentation	40
6.12.2.1 InvalidLibraryException()	40
6.13 Arcade::Logger Class Reference	40
6.13.1 Detailed Description	41
6.13.2 Member Enumeration Documentation	41
6.13.2.1 LogLevel	41
6.13.3 Member Function Documentation	41
6.13.3.1 log()	41
6.13.3.2 setLogLevel()	42

6.14 Arcade::Display::Ncurses Class Reference	42
6.14.1 Detailed Description	44
6.14.2 Member Function Documentation	44
6.14.2.1 getDelta()	44
6.14.2.2 getKeyCode()	44
6.14.2.3 isKeyPressed()	44
6.14.2.4 isKeyPressedOnce()	45
6.14.2.5 isOpen()	45
6.14.2.6 putCircle()	45
6.14.2.7 putFillCircle()	46
6.14.2.8 putFillRect()	46
6.14.2.9 putLine()	47
6.14.2.10 putPixel()	47
6.14.2.11 putRect()	47
6.14.2.12 putText()	48
6.14.2.13 setColor()	48
6.14.2.14 shouldBeRestarted()	48
6.14.2.15 shouldExit()	49
6.14.2.16 shouldGoToMenu()	49
6.14.2.17 switchToNextGame()	49
6.14.2.18 switchToNextLib()	50
6.14.2.19 switchToPreviousGame()	50
6.14.2.20 switchToPreviousLib()	50
6.15 Arcade::Games::Nibbler Class Reference	51
6.15.1 Detailed Description	51
6.15.2 Member Function Documentation	51
6.15.2.1 render()	51
6.15.2.2 update()	52
6.16 Arcade::Games::Pacman Class Reference	52
6.16.1 Detailed Description	52
6.16.2 Member Function Documentation	53
6.16.2.1 render()	53
6.16.2.2 update()	53
6.17 Arcade::Games::Qix Class Reference	53
6.17.1 Detailed Description	54
6.17.2 Member Function Documentation	54
6.17.2.1 update()	54
6.18 Arcade::Display::SDL Class Reference	54
6.18.1 Detailed Description	56
6.18.2 Member Function Documentation	56
6.18.2.1 getDelta()	56
6.18.2.2 getKeyCode()	56

6.18.2.3 isKeyPressed()	56
6.18.2.4 isKeyPressedOnce()	57
6.18.2.5 isOpen()	57
6.18.2.6 putCircle()	57
6.18.2.7 putFillCircle()	58
6.18.2.8 putFillRect()	58
6.18.2.9 putLine()	59
6.18.2.10 putPixel()	59
6.18.2.11 putRect()	59
6.18.2.12 putText()	60
6.18.2.13 setColor()	60
6.18.2.14 shouldBeRestarted()	60
6.18.2.15 shouldExit()	61
6.18.2.16 shouldGoToMenu()	61
6.18.2.17 switchToNextGame()	61
6.18.2.18 switchToNextLib()	62
6.18.2.19 switchToPreviousGame()	62
6.18.2.20 switchToPreviousLib()	62
6.19 Arcade::Display::SFML Class Reference	63
6.19.1 Detailed Description	64
6.19.2 Member Function Documentation	64
6.19.2.1 getDelta()	64
6.19.2.2 getKeyCode()	65
6.19.2.3 isKeyPressed()	65
6.19.2.4 isKeyPressedOnce()	65
6.19.2.5 isOpen()	66
6.19.2.6 putCircle()	66
6.19.2.7 putFillCircle()	66
6.19.2.8 putFillRect()	67
6.19.2.9 putLine()	67
6.19.2.10 putPixel()	67
6.19.2.11 putRect()	68
6.19.2.12 putText()	68
6.19.2.13 setColor()	69
6.19.2.14 shouldBeRestarted()	69
6.19.2.15 shouldExit()	69
6.19.2.16 shouldGoToMenu()	70
6.19.2.17 switchToNextGame()	70
6.19.2.18 switchToNextLib()	70
6.19.2.19 switchToPreviousGame()	71
6.19.2.20 switchToPreviousLib()	71
6.20 Arcade::Games::Solarfox Class Reference	71

6.20.1 Detailed Description	72
6.20.2 Member Function Documentation	72
6.20.2.1 update()	72
Index	73

Chapter 1

How to implement your own game / library

Game

- First, you wanna create a c++ class implementing the `IGameModule` interface.
- Next you wanna compile it into a dynamic library (.so file).
- Put that library file in the `games/` folder located at the root of the arcade repository.
 - Note that it must follow the following naming convention : `lib_arcade_$gamename.so`.

Every class contained by the libraries located in the `games/` folder are going to be instantiated using the symbol `createLib` that your library **must** contain.

It should be as following:

```
extern "C" std::unique_ptr<Arcade::Games::IGameModule> createLib(void)
{
    return std::make_unique<MyGameModule>();
}
```

Library

- First, you wanna create a c++ class implementing the `IDisplayModule` interface.
- Next you wanna compile it into a dynamic library (.so file).
- Put that library file in the `lib/` folder located at the root of the arcade repository.
 - Note that it must follow the following naming convention : `lib_arcade_$libraryname.so`.

Every class contained by the libraries located in the `lib/` folder are going to be instantiated using the symbol `createLib` that your library **must** contain.

It should be as following:

```
extern "C" std::unique_ptr<Arcade::Display::IDisplayModule> createLib(void)
{
    return std::make_unique<MyDisplayModule>();
}
```


Chapter 2

Namespace Index

2.1 Namespace List

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

Arcade	9
Arcade::Display	9
Arcade::Exceptions	10
Arcade::Games	10

Chapter 3

Hierarchical Index

3.1 Class Hierarchy

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

Arcade::Core	22
Arcade::DLInfos	22
Arcade::DLLoader< T >	23
exception	
Arcade::Exceptions::ArcadeException	17
Arcade::Exceptions::BadFileException	19
Arcade::Exceptions::BadInstanciationException	20
Arcade::Exceptions::InvalidLibraryException	39
Arcade::Display::IDisplayModule	25
Arcade::Display::ADisplayModule	11
Arcade::Display::Ncurses	42
Arcade::Display::SDL	54
Arcade::Display::SFML	63
Arcade::Games::IGameModule	35
Arcade::Games::AGameModule	13
Arcade::Games::Centipede	21
Arcade::Games::Nibbler	51
Arcade::Games::Pacman	52
Arcade::Games::Qix	53
Arcade::Games::Solarfox	71
Arcade::Logger	40

Chapter 4

Class Index

4.1 Class List

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

Arcade::Display::ADisplayModule	11
Abstract class adding utilities and an enum to the IDisplayModule interface	
Arcade::Games::AGameModule	13
Abstract class implementing key methods of the IGameModule interface	
Arcade::Exceptions::ArcadeException	17
Base exception class for this projects' exceptions	
Arcade::Exceptions::BadFileException	19
Thrown when looking up to an external file that is inexistant	
Arcade::Exceptions::BadInstanciationException	20
Thrown when library objects failed to be instanciated	
Arcade::Games::Centipede	21
Centipede game	
Arcade::Core	22
Core class that handles all the interactions between the library modules and the game modules	
Arcade::DLInfos	22
Contains information about a given library	
Arcade::DLLoader< T >	23
Used for dynamically loading libraries	
Arcade::Display::IDisplayModule	25
Interface for the display modules used to display things	
Arcade::Games::IGameModule	35
Interface for the game modules used to handle games	
Arcade::Exceptions::InvalidLibraryException	39
Thrown when trying to use an invalid library file	
Arcade::Logger	40
Used to display error messages depending on the set log level	
Arcade::Display::Ncurses	42
Libncurses library	
Arcade::Games::Nibbler	51
Nibbler game	
Arcade::Games::Pacman	52
Pacman game	
Arcade::Games::Qix	53
Qix game	
Arcade::Display::SDL	54
SDL library	

Arcade::Display::SFML	
SFML library	63
Arcade::Games::Solarfox	
Solarfox game	71

Chapter 5

Namespace Documentation

5.1 Arcade Namespace Reference

Namespaces

- [Display](#)
- [Exceptions](#)
- [Games](#)

Classes

- class [Core](#)
Core class that handles all the interactions between the library modules and the game modules.
- struct [DLInfos](#)
Contains information about a given library.
- class [DLLoader](#)
Used for dynamically loading libraries.
- class [Logger](#)
Used to display error messages depending on the set log level.

5.1.1 Detailed Description

Default namespace for the project.

5.2 Arcade::Display Namespace Reference

Classes

- class [ADisplayModule](#)
Abstract class adding utilities and an enum to the [IDisplayModule](#) interface.
- class [IDisplayModule](#)
Interface for the display modules used to display things.
- class [Ncurses](#)
Libncurses library.
- class [SDL](#)
SDL library.
- class [SFML](#)
SFML library.

5.2.1 Detailed Description

Contains elements related to the display libraries of the [Arcade](#) project.

5.3 Arcade::Exceptions Namespace Reference

Classes

- class [ArcadeException](#)
Base exception class for this projects' exceptions.
- class [BadFileException](#)
Thrown when looking up to an external file that is inexistant.
- class [BadInstanciacionException](#)
Thrown when library objects failed to be instanciated.
- class [InvalidLibraryException](#)
Thrown when trying to use an invalid library file.

5.3.1 Detailed Description

Contains a loadout of exceptions that are used in the [Arcade](#) project.

5.4 Arcade::Games Namespace Reference

Classes

- class [AGameModule](#)
Abstract class implementing key methods of the [IGameModule](#) interface.
- class [Centipede](#)
[Centipede](#) game.
- class [IGameModule](#)
Interface for the game modules used to handle games.
- class [Nibbler](#)
[Nibbler](#) game.
- class [Pacman](#)
[Pacman](#) game.
- class [Qix](#)
[Qix](#) game.
- class [Solarfox](#)
[Solarfox](#) game.

5.4.1 Detailed Description

Contains elements related to the game libraries of the [Arcade](#) project.

Chapter 6

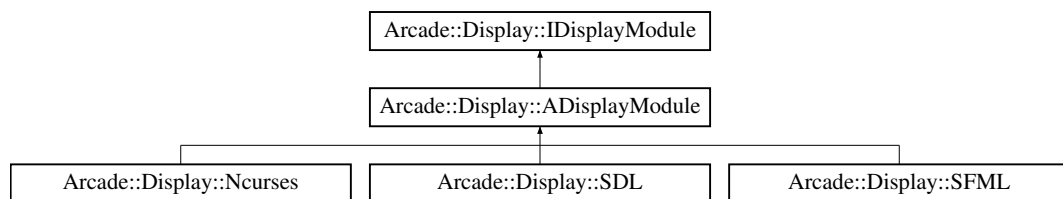
Class Documentation

6.1 Arcade::Display::ADisplayModule Class Reference

Abstract class adding utilities and an enum to the [IDisplayModule](#) interface.

```
#include <ADisplayModule.hpp>
```

Inheritance diagram for Arcade::Display::ADisplayModule:



Public Member Functions

- [ADisplayModule](#) (const std::string &libName)
Construct a new [ADisplayModule](#) object.
- const std::string &[getLibName](#) () const final
Gets the library name.

Protected Types

- enum [SystemKeys](#) {
 [ESCAPE](#) = Keys::KEYS_END, [M](#), [R](#), [F1](#),
 [F2](#), [F3](#), [F4](#), [SYSKEYS_END](#) }
Additional keys that are used by the library.

Additional Inherited Members

6.1.1 Detailed Description

Abstract class adding utilities and an enum to the [IDisplayModule](#) interface.

6.1.2 Member Enumeration Documentation

6.1.2.1 SystemKeys

```
enum Arcade::Display::ADisplayModule::SystemKeys [protected]
```

Additional keys that are used by the library.

Enumerator

ESCAPE	Escape key.
M	M key.
R	R key.
F1	F1 key.
F2	F2 key.
F3	F3 key.
F4	F4 key.
SYSKEYS_END	Key count.

6.1.3 Constructor & Destructor Documentation

6.1.3.1 ADisplayModule()

```
Arcade::Display::ADisplayModule::ADisplayModule (  
    const std::string & libName )
```

Construct a new [ADisplayModule](#) object.

Parameters

<i>libName</i>	The library's name
----------------	--------------------

6.1.4 Member Function Documentation

6.1.4.1 getLibName()

```
const std::string& Arcade::Display::ADisplayModule::getLibName ( ) const [final], [virtual]
```

Gets the library name.

Returns

The library's name

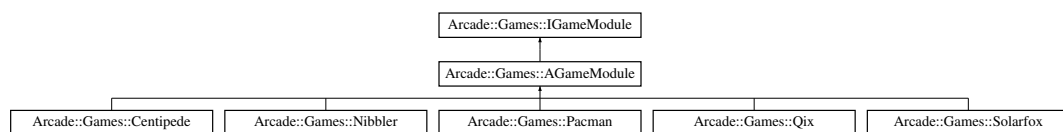
Implements [Arcade::Display::IDisplayModule](#).

6.2 Arcade::Games::AGameModule Class Reference

Abstract class implementing key methods of the [IGameModule](#) interface.

```
#include <AGameModule.hpp>
```

Inheritance diagram for Arcade::Games::AGameModule:

**Public Member Functions**

- [AGameModule](#) (std::string const &libName)
Construct a new [AGameModule](#) object.
- bool [loadFromFile](#) (const std::string &filepath) final
Loads highscores from a file.
- bool [loadFromFile](#) () final
Loads highscores from the default save file.
- bool [saveToFile](#) (const std::string &filepath) const final
Saves highscores to a file.
- bool [saveToFile](#) () const final
Saves highscores from the default save file.
- void [setPlayerName](#) (const std::string &name) final
Sets the player name.
- std::pair< std::string, int > [getScore](#) () const final
Gets the current score.
- std::vector< std::pair< std::string, int > > [getBestScores](#) () const final
Gets the best 16 scores.
- void [render](#) ([Arcade::Display::IDisplayModule](#) &lib) const override
Default game implementation (out of order)
- const std::string & [getLibName](#) () const final
Gets the library name.

Protected Member Functions

- void [addToBestScores](#) (int nb)
Adds a score to the scoreboard.
- void [drawGameOver](#) ([Arcade::Display::IDisplayModule](#) &displayModule) const
Display game over screen.

Protected Attributes

- `int _currentScore`
The current score of the active game session.
- `bool _isDead`
True if the player is dead.

6.2.1 Detailed Description

Abstract class implementing key methods of the [IGameModule](#) interface.

6.2.2 Constructor & Destructor Documentation

6.2.2.1 AGameModule()

```
Arcade::Games::AGameModule::AGameModule (
    std::string const & libName )
```

Construct a new [AGameModule](#) object.

Parameters

<i>libName</i>	The library's name
----------------	--------------------

6.2.3 Member Function Documentation

6.2.3.1 addToBestScores()

```
void Arcade::Games::AGameModule::addToBestScores (
    int nb ) [protected]
```

Adds a score to the scoreboard.

Parameters

<i>nb</i>	The score value
-----------	-----------------

6.2.3.2 drawGameOver()

```
void Arcade::Games::AGameModule::drawGameOver (
    Arcade::Display::IDisplayModule & displayModule ) const [protected]
```

Display game over screen.

Parameters

<i>displayModule</i>	The display module
----------------------	--------------------

6.2.3.3 getBestScores()

```
std::vector<std::pair<std::string, int> > Arcade::Games::AGameModule::getBestScores ( ) const
[final], [virtual]
```

Gets the best 16 scores.

Returns

std::vector<std::pair<std::string, int>> Vector of [name, score] value pairs

Implements [Arcade::Games::IGameModule](#).

6.2.3.4 getLibName()

```
const std::string& Arcade::Games::AGameModule::getLibName ( ) const [final], [virtual]
```

Gets the library name.

Returns

The library's name

Implements [Arcade::Games::IGameModule](#).

6.2.3.5 getScore()

```
std::pair<std::string, int> Arcade::Games::AGameModule::getScore ( ) const [final], [virtual]
```

Gets the current score.

Returns

std::pair<std::string, int> [Name, score] value pairs

Implements [Arcade::Games::IGameModule](#).

6.2.3.6 loadFromFile() [1/2]

```
bool Arcade::Games::AGameModule::loadFromFile (
    const std::string & filepath ) [final], [virtual]
```

Loads highscores from a file.

Parameters

<i>filepath</i>	The file path
-----------------	---------------

Returns

true Highscores were loaded
false An error occurred

Implements [Arcade::Games::IGameModule](#).

6.2.3.7 loadFromFile() [2/2]

```
bool Arcade::Games::AGameModule::loadFromFile ( ) [final], [virtual]
```

Loads highscores from the default save file.

Returns

true Highscores were loaded
false An error occurred

Implements [Arcade::Games::IGameModule](#).

6.2.3.8 render()

```
void Arcade::Games::AGameModule::render (
    Arcade::Display::IDisplayModule & lib ) const [override], [virtual]
```

Default game implementation (out of order)

Parameters

<i>lib</i>	The display module that will be used to put things on a canvas.
------------	---

Implements [Arcade::Games::IGameModule](#).

Reimplemented in [Arcade::Games::Nibbler](#), and [Arcade::Games::Pacman](#).

6.2.3.9 saveToFile() [1/2]

```
bool Arcade::Games::AGameModule::saveToFile (
    const std::string & filepath ) const [final], [virtual]
```

Saves highscores to a file.

Parameters

<i>filepath</i>	The file path
-----------------	---------------

Returns

true Highscores were saved
false An error occurred

Implements [Arcade::Games::IGameModule](#).

6.2.3.10 `saveToFile()` [2/2]

```
bool Arcade::Games::AGameModule::saveToFile ( ) const [final], [virtual]
```

Saves highscores from the default save file.

Returns

true Highscores were saved
false An error occurred

Implements [Arcade::Games::IGameModule](#).

6.2.3.11 `setPlayerName()`

```
void Arcade::Games::AGameModule::setPlayerName (
    const std::string & name ) [final], [virtual]
```

Sets the player name.

Parameters

<i>name</i>	The player name
-------------	-----------------

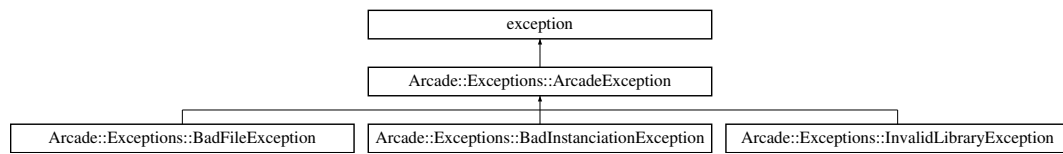
Implements [Arcade::Games::IGameModule](#).

6.3 Arcade::Exceptions::ArcadeException Class Reference

Base exception class for this projects' exceptions.

```
#include <ArcadeException.hpp>
```

Inheritance diagram for `Arcade::Exceptions::ArcadeException`:



Public Member Functions

- [ArcadeException](#) (`std::string const &message`, `std::string const &component`)
Construct a new [Arcade](#) Exception object.
- `const char * what` (`void`) `const noexcept` override
Gets the error message which describe why the exception occurred.
- `std::string const & getComponent` (`void`) `const noexcept`
Gets the name of component where the exception occurred.

6.3.1 Detailed Description

Base exception class for this projects' exceptions.

6.3.2 Constructor & Destructor Documentation

6.3.2.1 `ArcadeException()`

```

Arcade::Exceptions::ArcadeException::ArcadeException (
    std::string const & message,
    std::string const & component )
  
```

Construct a new [Arcade](#) Exception object.

Parameters

<i>message</i>	Message explaining the problem.
<i>component</i>	Additional information on where the problem occurred.

6.3.3 Member Function Documentation

6.3.3.1 `getComponent()`

```

std::string const& Arcade::Exceptions::ArcadeException::getComponent (
    void ) const [noexcept]
  
```

Gets the name of component where the exception occurred.

Returns

The component name

6.3.3.2 what()

```
const char* Arcade::Exceptions::ArcadeException::what (
    void ) const [override], [noexcept]
```

Gets the error message which describe why the exception occurred.

Returns

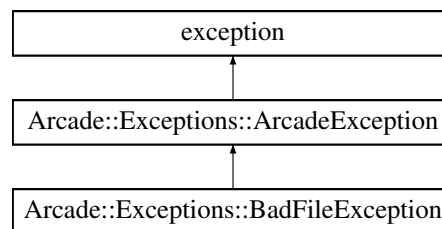
The error message

6.4 Arcade::Exceptions::BadFileException Class Reference

Thrown when looking up to an external file that is inexistant.

```
#include <BadFileException.hpp>
```

Inheritance diagram for Arcade::Exceptions::BadFileException:



Public Member Functions

- [BadFileException](#) (std::string const &message, std::string const &component)
Construct a new Bad File Exception object.

6.4.1 Detailed Description

Thrown when looking up to an external file that is inexistant.

6.4.2 Constructor & Destructor Documentation

6.4.2.1 BadFileException()

```
Arcade::Exceptions::BadFileException::BadFileException (
    std::string const & message,
    std::string const & component )
```

Construct a new Bad File Exception object.

Parameters

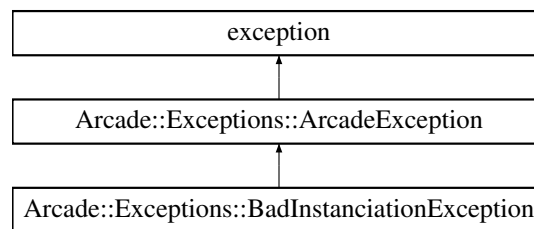
<i>message</i>	Message explaining the problem.
<i>component</i>	Additional information on where the problem occurred.

6.5 Arcade::Exceptions::BadInstanciationException Class Reference

Thrown when library objects failed to be instanciated.

```
#include <BadInstanciationException.hpp>
```

Inheritance diagram for Arcade::Exceptions::BadInstanciationException:



Public Member Functions

- [BadInstanciationException](#) (std::string const &message, std::string const &component)
Construct a new Bad InstanciationException object.

6.5.1 Detailed Description

Thrown when library objects failed to be instanciated.

6.5.2 Constructor & Destructor Documentation

6.5.2.1 BadInstanciationException()

```

Arcade::Exceptions::BadInstanciationException::BadInstanciationException (
    std::string const & message,
    std::string const & component )

```

Construct a new Bad InstanciationException object.

Parameters

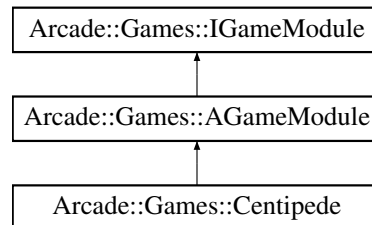
<i>message</i>	Message explaining the problem.
<i>component</i>	Additional information on where the problem occurred.

6.6 Arcade::Games::Centipede Class Reference

[Centipede](#) game.

```
#include <Centipede.hpp>
```

Inheritance diagram for Arcade::Games::Centipede:



Public Member Functions

- void [reset](#) () final
Resets and restarts the game.
- void [update](#) (const [Arcade::Display::IDisplayModule](#) &lib) final
Updates the game.

Additional Inherited Members

6.6.1 Detailed Description

[Centipede](#) game.

6.6.2 Member Function Documentation

6.6.2.1 update()

```
void Arcade::Games::Centipede::update (
    const Arcade::Display::IDisplayModule & lib ) [final], [virtual]
```

Updates the game.

Parameters

<i>lib</i>	The display module that will be used to get events that occurred
------------	--

Implements [Arcade::Games::IGameModule](#).

6.7 Arcade::Core Class Reference

[Core](#) class that handles all the interactions between the library modules and the game modules.

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

Public Member Functions

- [Core](#) (const std::string &startLibraryPath)
Construct a new [Core](#) object.
- void [play](#) ()
Starts the arcade program.

6.7.1 Detailed Description

[Core](#) class that handles all the interactions between the library modules and the game modules.

6.7.2 Constructor & Destructor Documentation

6.7.2.1 Core()

```
Arcade::Core::Core (
    const std::string & startLibraryPath ) [explicit]
```

Construct a new [Core](#) object.

Parameters

<i>startLibraryPath</i>	The library path that will be first used when the program will be started.
-------------------------	--

6.8 Arcade::DLInfos Struct Reference

Contains information about a given library.

```
#include <DLInfos.h>
```

Public Attributes

- std::string [path](#)
Path of the library.
- std::string [name](#)

- Name of the library*
([Arcade::Games::IGameModule::getLibName](#) or [Arcade::Display::IDisplayModule::getLibName](#))
- `std::vector< std::pair< std::string, int > > scores`
For [Arcade::Games::IGameModule](#), vector containing the best scores (obtained by [Arcade::Games::IGameModule::getBestScores](#)).
*For [Arcade::Display::IDisplayModule](#), **empty vector**.*

6.8.1 Detailed Description

Contains information about a given library.

6.9 Arcade::DLLoader< T > Class Template Reference

Used for dynamically loading libraries.

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

Public Member Functions

- **DLLoader** ([DLLoader](#) const &)=delete
- void **operator=** ([DLLoader](#) const &)=delete
- `std::vector< DLInfos > getLibraries` (const std::string &dirPath) const
Gets the available libraries in a given folder.
- `std::unique_ptr< T > loadLibrary` (const std::string &path) const
Loads the given library.

Static Public Member Functions

- static [DLLoader](#) const & [getInstance](#) (void)
Gets an instance of this object.

6.9.1 Detailed Description

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

Used for dynamically loading libraries.

Template Parameters

<i>T</i>	Library class type. Either Arcade::Games::IGameModule or Arcade::Display::IDisplayModule .
----------	--

6.9.2 Member Function Documentation

6.9.2.1 getInstance()

```
template<class T >
static DLLoader const& Arcade::DLLoader< T >::getInstance (
    void ) [static]
```

Gets an instance of this object.

Returns

The current [DLLoader](#) instance.

6.9.2.2 getLibraries()

```
template<class T >
std::vector<DLInfos> Arcade::DLLoader< T >::getLibraries (
    const std::string & dirPath ) const
```

Gets the available libraries in a given folder.

Parameters

<i>dirPath</i>	Folder to lookup
----------------	------------------

Returns

`std::vector<DLInfos>` Vector containing informations about the library.

6.9.2.3 loadLibrary()

```
template<class T >
std::unique_ptr<T> Arcade::DLLoader< T >::loadLibrary (
    const std::string & path ) const
```

Loads the given library.

Parameters

<i>path</i>	File containing the library to extract.
-------------	---

Returns

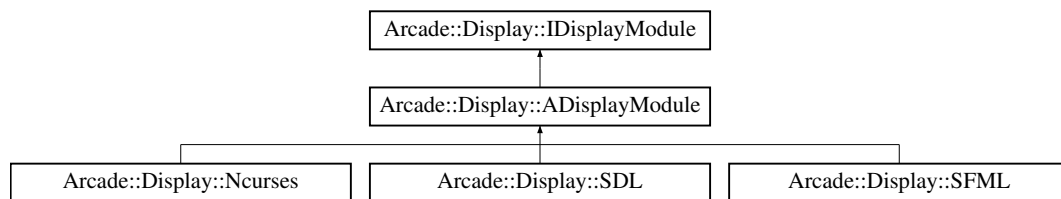
`std::unique_ptr<T>` Unique pointer to the T instance.

6.10 Arcade::Display::IDisplayModule Class Reference

Interface for the display modules used to display things.

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

Inheritance diagram for Arcade::Display::IDisplayModule:



Public Types

- enum `Colors` {
`DEFAULT`, `BLACK`, `RED`, `GREEN`,
`YELLOW`, `BLUE`, `MAGENTA`, `CYAN`,
`LIGHT_GRAY`, `DARK_GRAY`, `LIGHT_RED`, `LIGHT_GREEN`,
`LIGHT_YELLOW`, `LIGHT_BLUE`, `LIGHT_MAGENTA`, `LIGHT_CYAN`,
`WHITE`, `COLORS_END` }

Available colors.

- enum `Keys` {
`LEFT`, `RIGHT`, `UP`, `DOWN`,
`Z`, `Q`, `S`, `D`,
`A`, `E`, `W`, `X`,
`SPACE`, `J`, `K`, `U`,
`I`, `ENTER`, `KEYS_END` }

Available keys.

Public Member Functions

- virtual void `reset` ()=0
Resets the library.
- virtual void `open` ()=0
Opens / initializes the window.
- virtual void `close` ()=0
Close / destroy the window.
- virtual bool `isOpen` () const =0
Check window status.
- virtual bool `switchToNextLib` () const =0
Checks whether you need to change the current display library.
- virtual bool `switchToPreviousLib` () const =0
Checks whether you need to change the current display library.

- virtual bool `switchToNextGame` () const =0
Checks whether you need to change the current game library.
- virtual bool `switchToPreviousGame` () const =0
Checks whether you need to change the current game library.
- virtual bool `shouldBeRestarted` () const =0
Checks whether you need to restart the current game.
- virtual bool `shouldGoToMenu` () const =0
Checks whether you need to go back to the menu.
- virtual bool `shouldExit` () const =0
Checks whether you need to exit the program.
- virtual bool `isKeyPressed` (IDisplayModule::Keys key) const =0
Checks whether the current key is being pressed.
- virtual bool `isKeyPressedOnce` (IDisplayModule::Keys key) const =0
Checks whether the current key was pressed during the last frame.
- virtual float `getDelta` () const =0
Gets the number of frames since last update.
- virtual void `clear` () const =0
*Clears the canvas. **Call this after the IDisplayModule::update method.***
- virtual void `update` ()=0
*Runs an update over the events that occurred. **Call this before the IDisplayModule::clear method.***
- virtual void `render` () const =0
Renders the canvas.
- virtual char `getKeyCode` () const =0
Gets the last pressed character from the keyboard.
- virtual void `setColor` (IDisplayModule::Colors color)=0
Defines the color of the elements that will be drawn.
- virtual void `putPixel` (float x, float y) const =0
Displays a pixel.
- virtual void `putLine` (float x1, float y1, float x2, float y2) const =0
Displays a line.
- virtual void `putRect` (float x, float y, float w, float h) const =0
Displays a rectangle.
- virtual void `putFillRect` (float x, float y, float w, float h) const =0
Displays a filled rectangle.
- virtual void `putCircle` (float x, float y, float rad) const =0
Displays a circle.
- virtual void `putFillCircle` (float x, float y, float rad) const =0
Displays a filled circle.
- virtual void `putText` (const std::string &text, unsigned int size, float x, float y) const =0
Displays text.
- virtual const std::string & `getLibName` () const =0
Gets the library name.

6.10.1 Detailed Description

Interface for the display modules used to display things.

6.10.2 Member Enumeration Documentation

6.10.2.1 Colors

```
enum Arcade::Display::IDisplayModule::Colors
```

Available colors.

Enumerator

DEFAULT	The color the window clears to.
BLACK	Black color.
RED	Red color.
GREEN	Green color.
YELLOW	Yellow color.
BLUE	Blue color.
MAGENTA	Magenta color.
CYAN	Cyan color.
LIGHT_GRAY	Light gray color.
DARK_GRAY	Dark gray color.
LIGHT_RED	Light red color.
LIGHT_GREEN	Light green color.
LIGHT_YELLOW	Light yellow color.
LIGHT_BLUE	Light blue color.
LIGHT_MAGENTA	Light magenta color.
LIGHT_CYAN	Light cyan color.
WHITE	White color.
COLORS_END	Color count.

6.10.2.2 Keys

```
enum Arcade::Display::IDisplayModule::Keys
```

Available keys.

Enumerator

LEFT	Left key.
RIGHT	Right key.
UP	Up key.
DOWN	Down key.
Z	Z key.
Q	Q key.
S	S key.
D	D key.
A	A key.
E	E key.
W	W key.
X	X key.

Enumerator

SPACE	Space key.
J	J key.
K	K key.
U	U key.
I	I key.
ENTER	Return key.
KEYS_END	Key count.

6.10.3 Member Function Documentation**6.10.3.1 getDelta()**

```
virtual float Arcade::Display::IDisplayModule::getDelta ( ) const [pure virtual]
```

Gets the number of frames since last update.

Returns

float Frame count

Implemented in [Arcade::Display::SDL](#), [Arcade::Display::Ncurses](#), and [Arcade::Display::SFML](#).

6.10.3.2 getKeyCode()

```
virtual char Arcade::Display::IDisplayModule::getKeyCode ( ) const [pure virtual]
```

Gets the last pressed character from the keyboard.

Returns

\0 if nothing was pressed, \b if backspace was pressed, \n if return was pressed, otherwise, a character.

Implemented in [Arcade::Display::SDL](#), [Arcade::Display::Ncurses](#), and [Arcade::Display::SFML](#).

6.10.3.3 getLibName()

```
virtual const std::string& Arcade::Display::IDisplayModule::getLibName ( ) const [pure virtual]
```

Gets the library name.

Returns

The library's name

Implemented in [Arcade::Display::ADisplayModule](#).

6.10.3.4 isKeyPressed()

```
virtual bool Arcade::Display::IDisplayModule::isKeyPressed (
    IDisplayModule::Keys key ) const [pure virtual]
```

Checks whether the current key is being pressed.

Parameters

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

Returns

true Key is pressed
false Key is not pressed

Implemented in [Arcade::Display::SDL](#), [Arcade::Display::Ncurses](#), and [Arcade::Display::SFML](#).

6.10.3.5 isKeyPressedOnce()

```
virtual bool Arcade::Display::IDisplayModule::isKeyPressedOnce (
    IDisplayModule::Keys key ) const [pure virtual]
```

Checks whether the current key was pressed during the last frame.

Parameters

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

Returns

true Key is pressed
false Key is not pressed

Implemented in [Arcade::Display::SDL](#), [Arcade::Display::Ncurses](#), and [Arcade::Display::SFML](#).

6.10.3.6 isOpen()

```
virtual bool Arcade::Display::IDisplayModule::isOpen ( ) const [pure virtual]
```

Check window status.

Returns

true Window is open
false Window is closed

Implemented in [Arcade::Display::SDL](#), [Arcade::Display::Ncurses](#), and [Arcade::Display::SFML](#).

6.10.3.7 putCircle()

```
virtual void Arcade::Display::IDisplayModule::putCircle (
    float x,
    float y,
    float rad ) const [pure virtual]
```

Displays a cirle.

Parameters

<i>x</i>	X coordinates
<i>y</i>	Y coordinates
<i>rad</i>	Radius of the circle

Implemented in [Arcade::Display::SDL](#), [Arcade::Display::Ncurses](#), and [Arcade::Display::SFML](#).

6.10.3.8 putFillCircle()

```
virtual void Arcade::Display::IDisplayModule::putFillCircle (
    float x,
    float y,
    float rad ) const [pure virtual]
```

Displays a filled cirle.

Parameters

<i>x</i>	X coordinates
<i>y</i>	Y coordinates
<i>rad</i>	Radius of the circle

Implemented in [Arcade::Display::SDL](#), [Arcade::Display::Ncurses](#), and [Arcade::Display::SFML](#).

6.10.3.9 putFillRect()

```
virtual void Arcade::Display::IDisplayModule::putFillRect (
    float x,
    float y,
    float w,
    float h ) const [pure virtual]
```

Displays a filled rectangle.

Parameters

<i>x</i>	X coordinates
<i>y</i>	Y coordinates
<i>w</i>	Width of the rectangle
<i>h</i>	Height of the rectangle

Implemented in [Arcade::Display::SDL](#), [Arcade::Display::Ncurses](#), and [Arcade::Display::SFML](#).

6.10.3.10 putLine()

```
virtual void Arcade::Display::IDisplayModule::putLine (
    float x1,
    float y1,
    float x2,
    float y2 ) const [pure virtual]
```

Displays a line.

Parameters

<i>x1</i>	X coordinates for the first point
<i>y1</i>	Y coordinates for the first point
<i>x2</i>	X coordinates for the second point
<i>y2</i>	Y coordinates for the second point

Implemented in [Arcade::Display::SDL](#), [Arcade::Display::Ncurses](#), and [Arcade::Display::SFML](#).

6.10.3.11 putPixel()

```
virtual void Arcade::Display::IDisplayModule::putPixel (
    float x,
    float y ) const [pure virtual]
```

Displays a pixel.

Parameters

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

Implemented in [Arcade::Display::SDL](#), [Arcade::Display::Ncurses](#), and [Arcade::Display::SFML](#).

6.10.3.12 putRect()

```
virtual void Arcade::Display::IDisplayModule::putRect (
    float x,
    float y,
    float w,
    float h ) const [pure virtual]
```

Displays a rectangle.

Parameters

<i>x</i>	X coordinates
<i>y</i>	Y coordinates
<i>w</i>	Width of the rectangle
<i>h</i>	Height of the rectangle

Implemented in [Arcade::Display::SDL](#), [Arcade::Display::Ncurses](#), and [Arcade::Display::SFML](#).

6.10.3.13 putText()

```
virtual void Arcade::Display::IDisplayModule::putText (
    const std::string & text,
    unsigned int size,
    float x,
    float y ) const [pure virtual]
```

Displays text.

Parameters

<i>text</i>	The text content
<i>size</i>	The text size
<i>x</i>	X coordinates
<i>y</i>	Y coordinates

Implemented in [Arcade::Display::SDL](#), [Arcade::Display::Ncurses](#), and [Arcade::Display::SFML](#).

6.10.3.14 setColor()

```
virtual void Arcade::Display::IDisplayModule::setColor (
    IDisplayModule::Colors color ) [pure virtual]
```

Defines the color of the elements that will be drawn.

Parameters

<i>color</i>	The color
--------------	-----------

Implemented in [Arcade::Display::SDL](#), [Arcade::Display::Ncurses](#), and [Arcade::Display::SFML](#).

6.10.3.15 shouldBeRestarted()

```
virtual bool Arcade::Display::IDisplayModule::shouldBeRestarted ( ) const [pure virtual]
```

Checks whether you need to restart the current game.

Returns

true Restart the game
false Do nothing

Implemented in [Arcade::Display::SDL](#), [Arcade::Display::Ncurses](#), and [Arcade::Display::SFML](#).

6.10.3.16 shouldExit()

```
virtual bool Arcade::Display::IDisplayModule::shouldExit ( ) const [pure virtual]
```

Checks whether you need to exit the program.

Returns

true Exit the program
false Do nothing

Implemented in [Arcade::Display::SDL](#), [Arcade::Display::Ncurses](#), and [Arcade::Display::SFML](#).

6.10.3.17 shouldGoToMenu()

```
virtual bool Arcade::Display::IDisplayModule::shouldGoToMenu ( ) const [pure virtual]
```

Checks whether you need to go back to the menu.

Returns

true Go back to menu
false Do nothing

Implemented in [Arcade::Display::SDL](#), [Arcade::Display::Ncurses](#), and [Arcade::Display::SFML](#).

6.10.3.18 switchToNextGame()

```
virtual bool Arcade::Display::IDisplayModule::switchToNextGame ( ) const [pure virtual]
```

Checks whether you need to change the current game library.

Returns

true Switch to next available library
false Do nothing

Implemented in [Arcade::Display::SDL](#), [Arcade::Display::Ncurses](#), and [Arcade::Display::SFML](#).

6.10.3.19 switchToNextLib()

```
virtual bool Arcade::Display::IDisplayModule::switchToNextLib ( ) const [pure virtual]
```

Checks whether you need to change the current display library.

Returns

true Switch to next available library
false Do nothing

Implemented in [Arcade::Display::SDL](#), [Arcade::Display::Ncurses](#), and [Arcade::Display::SFML](#).

6.10.3.20 switchToPreviousGame()

```
virtual bool Arcade::Display::IDisplayModule::switchToPreviousGame ( ) const [pure virtual]
```

Checks whether you need to change the current game library.

Returns

true Switch to previous available library
false Do nothing

Implemented in [Arcade::Display::SDL](#), [Arcade::Display::Ncurses](#), and [Arcade::Display::SFML](#).

6.10.3.21 switchToPreviousLib()

```
virtual bool Arcade::Display::IDisplayModule::switchToPreviousLib ( ) const [pure virtual]
```

Checks whether you need to change the current display library.

Returns

true Switch to previous available library
false Do nothing

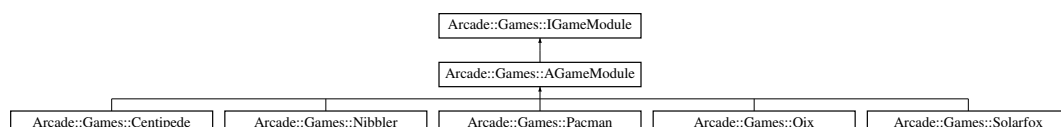
Implemented in [Arcade::Display::SDL](#), [Arcade::Display::Ncurses](#), and [Arcade::Display::SFML](#).

6.11 Arcade::Games::IGameModule Class Reference

Interface for the game modules used to handle games.

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

Inheritance diagram for Arcade::Games::IGameModule:



Public Member Functions

- virtual void [reset](#) ()=0
Resets and restarts the game.
- virtual bool [loadFromFile](#) (const std::string &filepath)=0
Loads highscores from a file.
- virtual bool [loadFromFile](#) ()=0
Loads highscores from the default save file.
- virtual bool [saveToFile](#) (const std::string &filepath) const =0
Saves highscores to a file.
- virtual bool [saveToFile](#) () const =0
Saves highscores from the default save file.
- virtual void [setPlayerName](#) (const std::string &name)=0
Sets the player name.
- virtual std::pair< std::string, int > [getScore](#) () const =0
Gets the current score.
- virtual std::vector< std::pair< std::string, int > > [getBestScores](#) () const =0
Gets the best 16 scores.
- virtual void [update](#) (const [Arcade::Display::IDisplayModule](#) &lib)=0
Updates the game.
- virtual void [render](#) ([Arcade::Display::IDisplayModule](#) &lib) const =0
Renders the game on the display module.
THIS MUST ONLY DRAW THINGS ON THE CANVAS, so don't call the [Arcade::Display::IDisplayModule::update](#) or [Arcade::Display::IDisplayModule::render](#) methods.
- virtual const std::string & [getLibName](#) () const =0
Gets the library name.

6.11.1 Detailed Description

Interface for the game modules used to handle games.

6.11.2 Member Function Documentation

6.11.2.1 [getBestScores\(\)](#)

```
virtual std::vector<std::pair<std::string, int> > Arcade::Games::IGameModule::getBestScores (
) const [pure virtual]
```

Gets the best 16 scores.

Returns

std::vector<std::pair<std::string, int>> Vector of [name, score] value pairs

Implemented in [Arcade::Games::AGameModule](#).

6.11.2.2 getLibName()

```
virtual const std::string& Arcade::Games::IGameModule::getLibName ( ) const [pure virtual]
```

Gets the library name.

Returns

The library's name

Implemented in [Arcade::Games::AGameModule](#).

6.11.2.3 getScore()

```
virtual std::pair<std::string, int> Arcade::Games::IGameModule::getScore ( ) const [pure virtual]
```

Gets the current score.

Returns

std::pair<std::string, int> [Name, score] value pairs

Implemented in [Arcade::Games::AGameModule](#).

6.11.2.4 loadFromFile() [1/2]

```
virtual bool Arcade::Games::IGameModule::loadFromFile (
    const std::string & filepath ) [pure virtual]
```

Loads highscores from a file.

Parameters

<i>filepath</i>	The file path
-----------------	---------------

Returns

true Highscores were loaded
false An error occurred

Implemented in [Arcade::Games::AGameModule](#).

6.11.2.5 loadFromFile() [2/2]

```
virtual bool Arcade::Games::IGameModule::loadFromFile ( ) [pure virtual]
```

Loads highscores from the default save file.

Returns

true Highscores were loaded
false An error occurred

Implemented in [Arcade::Games::AGameModule](#).

6.11.2.6 render()

```
virtual void Arcade::Games::IGameModule::render (
    Arcade::Display::IDisplayModule & lib ) const [pure virtual]
```

Renders the game on the display module.

THIS MUST ONLY DRAW THINGS ON THE CANVAS, so don't call the [Arcade::Display::IDisplayModule::update](#) or [Arcade::Display::IDisplayModule::render](#) methods.

Parameters

<i>lib</i>	The display module that will be used to put things on a canvas.
------------	---

Implemented in [Arcade::Games::AGameModule](#), [Arcade::Games::Nibbler](#), and [Arcade::Games::Pacman](#).

6.11.2.7 saveToFile() [1/2]

```
virtual bool Arcade::Games::IGameModule::saveToFile (
    const std::string & filepath ) const [pure virtual]
```

Saves highscores to a file.

Parameters

<i>filepath</i>	The file path
-----------------	---------------

Returns

true Highscores were saved
false An error occurred

Implemented in [Arcade::Games::AGameModule](#).

6.11.2.8 saveToFile() [2/2]

```
virtual bool Arcade::Games::IGameModule::saveToFile ( ) const [pure virtual]
```

Saves highscores from the default save file.

Returns

true Highscores were saved
false An error occurred

Implemented in [Arcade::Games::AGameModule](#).

6.11.2.9 setPlayerName()

```
virtual void Arcade::Games::IGameModule::setPlayerName (
    const std::string & name ) [pure virtual]
```

Sets the player name.

Parameters

<i>name</i>	The player name
-------------	-----------------

Implemented in [Arcade::Games::AGameModule](#).

6.11.2.10 update()

```
virtual void Arcade::Games::IGameModule::update (
    const Arcade::Display::IDisplayModule & lib ) [pure virtual]
```

Updates the game.

Parameters

<i>lib</i>	The display module that will be used to get events that occurred
------------	--

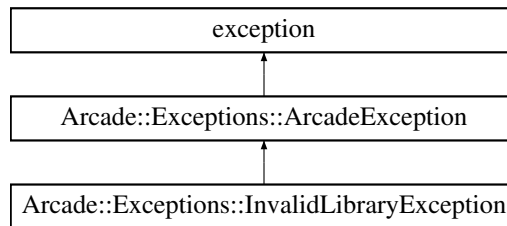
Implemented in [Arcade::Games::Nibbler](#), [Arcade::Games::Pacman](#), [Arcade::Games::Centipede](#), [Arcade::Games::Qix](#), and [Arcade::Games::Solarfox](#).

6.12 Arcade::Exceptions::InvalidLibraryException Class Reference

Thrown when trying to use an invalid library file.

```
#include <InvalidLibraryException.hpp>
```

Inheritance diagram for Arcade::Exceptions::InvalidLibraryException:



Public Member Functions

- [InvalidLibraryException](#) (std::string const &message, std::string const &component)
Construct a new Invalid Library Exception object.

6.12.1 Detailed Description

Thrown when trying to use an invalid library file.

6.12.2 Constructor & Destructor Documentation

6.12.2.1 InvalidLibraryException()

```
Arcade::Exceptions::InvalidLibraryException::InvalidLibraryException (
    std::string const & message,
    std::string const & component )
```

Construct a new Invalid Library Exception object.

Parameters

<i>message</i>	Message explaining the problem.
<i>component</i>	Additional information on where the problem occurred.

6.13 Arcade::Logger Class Reference

Used to display error messages depending on the set log level.

```
#include <Logger.hpp>
```


Public Types

- enum [LogLevel](#) { [ERROR](#), [DEBUG](#) }
- Available log levels.*

Public Member Functions

- Logger** (const [Logger](#) &)=delete
- void **operator=** (const [Logger](#) &)=delete

Static Public Member Functions

- template<typename ... Args>
static void [log](#) ([LogLevel](#) level, Args &&...args)
Logs the given message to the output.
- static void [setLogLevel](#) ([LogLevel](#) level)
Sets the current log level.

6.13.1 Detailed Description

Used to display error messages depending on the set log level.

6.13.2 Member Enumeration Documentation

6.13.2.1 LogLevel

```
enum Arcade::Logger::LogLevel
```

Available log levels.

Enumerator

ERROR	Displays unexpected errors.
DEBUG	Used for debugging.

6.13.3 Member Function Documentation

6.13.3.1 log()

```
template<typename ... Args>  
static void Arcade::Logger::log (
```

```
LogLevel level,
Args &&... args ) [inline], [static]
```

Logs the given message to the output.

Template Parameters

<i>Args</i>	Type list for each argument
-------------	-----------------------------

Parameters

<i>level</i>	The log level of the message
<i>args</i>	Every argument are going to be assembled using <code>std::cerr << ... << args</code>

6.13.3.2 setLogLevel()

```
static void Arcade::Logger::setLogLevel (
LogLevel level ) [static]
```

Sets the current log level.

Parameters

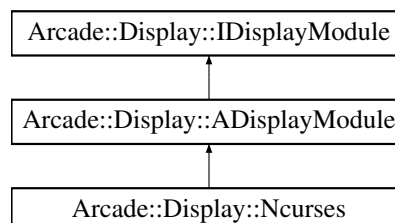
<i>level</i>	Log level value.
--------------	------------------

6.14 Arcade::Display::Ncurses Class Reference

Libncurses library.

```
#include <Ncurses.hpp>
```

Inheritance diagram for Arcade::Display::Ncurses:



Public Member Functions

- void `reset` () final

- Resets the library.*

 - void `open` () final
- Opens / initializes the window.*

 - void `close` () final
- Close / destroy the window.*

 - bool `isOpen` () const final
- Check window status.*

 - bool `switchToNextLib` () const final
- Checks whether you need to change the current display library.*

 - bool `switchToPreviousLib` () const final
- Checks whether you need to change the current display library.*

 - bool `switchToNextGame` () const final
- Checks whether you need to change the current game library.*

 - bool `switchToPreviousGame` () const final
- Checks whether you need to change the current game library.*

 - bool `shouldBeRestarted` () const final
- Checks whether you need to restart the current game.*

 - bool `shouldGoToMenu` () const final
- Checks whether you need to go back to the menu.*

 - bool `shouldExit` () const final
- Checks whether you need to exit the program.*

 - bool `isKeyPressed` (IDisplayModule::Keys key) const final
- Checks whether the current key is being pressed.*

 - bool `isKeyPressedOnce` (IDisplayModule::Keys key) const final
- Checks whether the current key was pressed during the last frame.*

 - float `getDelta` () const final
- Gets the number of frames since last update.*

 - void `clear` () const final
- Clears the canvas. **Call this after the IDisplayModule::update method.***

 - void `update` () final
- Runs an update over the events that occurred. **Call this before the IDisplayModule::clear method.***

 - void `render` () const final
- Renders the canvas.*

 - char `getKeyCode` () const final
- Gets the last pressed character from the keyboard.*

 - void `setColor` (IDisplayModule::Colors color) final
- Defines the color of the elements that will be drawn.*

 - void `putPixel` (float x, float y) const final
- Displays a pixel.*

 - void `putLine` (float x1, float y1, float x2, float y2) const final
- Displays a line.*

 - void `putRect` (float x, float y, float w, float h) const final
- Displays a rectangle.*

 - void `putFillRect` (float x, float y, float w, float h) const final
- Displays a filled rectangle.*

 - void `putCircle` (float x, float y, float rad) const final
- Displays a circle.*

 - void `putFillCircle` (float x, float y, float rad) const final
- Displays a filled circle.*

 - void `putText` (const std::string &text, unsigned int size, float x, float y) const final
- Displays text.*

Additional Inherited Members

6.14.1 Detailed Description

Libncurses library.

6.14.2 Member Function Documentation

6.14.2.1 `getDelta()`

```
float Arcade::Display::Ncurses::getDelta ( ) const [final], [virtual]
```

Gets the number of frames since last update.

Returns

float Frame count

Implements [Arcade::Display::IDisplayModule](#).

6.14.2.2 `getKeyCode()`

```
char Arcade::Display::Ncurses::getKeyCode ( ) const [final], [virtual]
```

Gets the last pressed character from the keyboard.

Returns

\0 if nothing was pressed, \b if backspace was pressed, \n if return was pressed, otherwise, a character.

Implements [Arcade::Display::IDisplayModule](#).

6.14.2.3 `isKeyPressed()`

```
bool Arcade::Display::Ncurses::isKeyPressed (
    IDisplayModule::Keys key ) const [final], [virtual]
```

Checks whether the current key is being pressed.

Parameters

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

Returns

true Key is pressed
false Key is not pressed

Implements [Arcade::Display::IDisplayModule](#).

6.14.2.4 isKeyPressedOnce()

```
bool Arcade::Display::Ncurses::isKeyPressedOnce (
    IDisplayModule::Keys key ) const [final], [virtual]
```

Checks whether the current key was pressed during the last frame.

Parameters

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

Returns

true Key is pressed
false Key is not pressed

Implements [Arcade::Display::IDisplayModule](#).

6.14.2.5 isOpen()

```
bool Arcade::Display::Ncurses::isOpen ( ) const [final], [virtual]
```

Check window status.

Returns

true Window is open
false Window is closed

Implements [Arcade::Display::IDisplayModule](#).

6.14.2.6 putCircle()

```
void Arcade::Display::Ncurses::putCircle (
    float x,
    float y,
    float rad ) const [final], [virtual]
```

Displays a circle.

Parameters

<i>x</i>	X coordinates
<i>y</i>	Y coordinates
<i>rad</i>	Radius of the circle

Implements [Arcade::Display::IDisplayModule](#).

6.14.2.7 putFillCircle()

```
void Arcade::Display::Ncurses::putFillCircle (
    float x,
    float y,
    float rad ) const [final], [virtual]
```

Displays a filled circle.

Parameters

<i>x</i>	X coordinates
<i>y</i>	Y coordinates
<i>rad</i>	Radius of the circle

Implements [Arcade::Display::IDisplayModule](#).

6.14.2.8 putFillRect()

```
void Arcade::Display::Ncurses::putFillRect (
    float x,
    float y,
    float w,
    float h ) const [final], [virtual]
```

Displays a filled rectangle.

Parameters

<i>x</i>	X coordinates
<i>y</i>	Y coordinates
<i>w</i>	Width of the rectangle
<i>h</i>	Height of the rectangle

Implements [Arcade::Display::IDisplayModule](#).

6.14.2.9 putLine()

```
void Arcade::Display::Ncurses::putLine (
    float x1,
    float y1,
    float x2,
    float y2 ) const [final], [virtual]
```

Displays a line.

Parameters

<i>x1</i>	X coordinates for the first point
<i>y1</i>	Y coordinates for the first point
<i>x2</i>	X coordinates for the second point
<i>y2</i>	Y coordinates for the second point

Implements [Arcade::Display::IDisplayModule](#).

6.14.2.10 putPixel()

```
void Arcade::Display::Ncurses::putPixel (
    float x,
    float y ) const [final], [virtual]
```

Displays a pixel.

Parameters

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

Implements [Arcade::Display::IDisplayModule](#).

6.14.2.11 putRect()

```
void Arcade::Display::Ncurses::putRect (
    float x,
    float y,
    float w,
    float h ) const [final], [virtual]
```

Displays a rectangle.

Parameters

<i>x</i>	X coordinates
----------	---------------

Parameters

<i>y</i>	Y coordinates
<i>w</i>	Width of the rectangle
<i>h</i>	Height of the rectangle

Implements [Arcade::Display::IDisplayModule](#).

6.14.2.12 putText()

```
void Arcade::Display::Ncurses::putText (
    const std::string & text,
    unsigned int size,
    float x,
    float y ) const [final], [virtual]
```

Displays text.

Parameters

<i>text</i>	The text content
<i>size</i>	The text size
<i>x</i>	X coordinates
<i>y</i>	Y coordinates

Implements [Arcade::Display::IDisplayModule](#).

6.14.2.13 setColor()

```
void Arcade::Display::Ncurses::setColor (
    IDisplayModule::Colors color ) [final], [virtual]
```

Defines the color of the elements that will be drawn.

Parameters

<i>color</i>	The color
--------------	-----------

Implements [Arcade::Display::IDisplayModule](#).

6.14.2.14 shouldBeRestarted()

```
bool Arcade::Display::Ncurses::shouldBeRestarted ( ) const [final], [virtual]
```

Checks whether you need to restart the current game.

Returns

true Restart the game
false Do nothing

Implements [Arcade::Display::IDisplayModule](#).

6.14.2.15 shouldExit()

```
bool Arcade::Display::Ncurses::shouldExit ( ) const [final], [virtual]
```

Checks whether you need to exit the program.

Returns

true Exit the program
false Do nothing

Implements [Arcade::Display::IDisplayModule](#).

6.14.2.16 shouldGoToMenu()

```
bool Arcade::Display::Ncurses::shouldGoToMenu ( ) const [final], [virtual]
```

Checks whether you need to go back to the menu.

Returns

true Go back to menu
false Do nothing

Implements [Arcade::Display::IDisplayModule](#).

6.14.2.17 switchToNextGame()

```
bool Arcade::Display::Ncurses::switchToNextGame ( ) const [final], [virtual]
```

Checks whether you need to change the current game library.

Returns

true Switch to next available library
false Do nothing

Implements [Arcade::Display::IDisplayModule](#).

6.14.2.18 switchToNextLib()

```
bool Arcade::Display::Ncurses::switchToNextLib ( ) const [final], [virtual]
```

Checks whether you need to change the current display library.

Returns

true Switch to next available library
false Do nothing

Implements [Arcade::Display::IDisplayModule](#).

6.14.2.19 switchToPreviousGame()

```
bool Arcade::Display::Ncurses::switchToPreviousGame ( ) const [final], [virtual]
```

Checks whether you need to change the current game library.

Returns

true Switch to previous available library
false Do nothing

Implements [Arcade::Display::IDisplayModule](#).

6.14.2.20 switchToPreviousLib()

```
bool Arcade::Display::Ncurses::switchToPreviousLib ( ) const [final], [virtual]
```

Checks whether you need to change the current display library.

Returns

true Switch to previous available library
false Do nothing

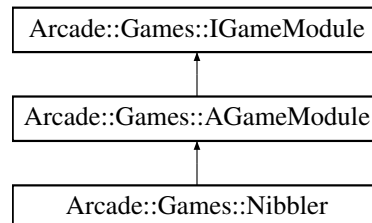
Implements [Arcade::Display::IDisplayModule](#).

6.15 Arcade::Games::Nibbler Class Reference

[Nibbler](#) game.

```
#include <Nibbler.hpp>
```

Inheritance diagram for Arcade::Games::Nibbler:



Public Member Functions

- void [reset](#) () final
Resets and restarts the game.
- void [update](#) (const [Arcade::Display::IDisplayModule](#) &displayModule) final
Updates the game.
- void [render](#) ([Arcade::Display::IDisplayModule](#) &displayModule) const final
Renders the [Nibbler](#) game on the display module.

Additional Inherited Members

6.15.1 Detailed Description

[Nibbler](#) game.

6.15.2 Member Function Documentation

6.15.2.1 render()

```
void Arcade::Games::Nibbler::render (
    Arcade::Display::IDisplayModule & displayModule ) const [final], [virtual]
```

Renders the [Nibbler](#) game on the display module.

Parameters

<i>displayModule</i>	The display module that will be used to put things on a canvas.
----------------------	---

Reimplemented from [Arcade::Games::AGameModule](#).

6.15.2.2 update()

```
void Arcade::Games::Nibbler::update (
    const Arcade::Display::IDisplayModule & lib ) [final], [virtual]
```

Updates the game.

Parameters

<i>lib</i>	The display module that will be used to get events that occurred
------------	--

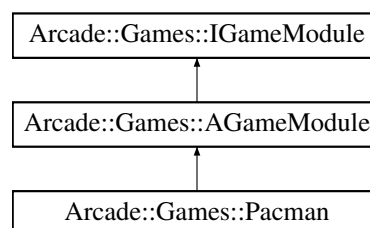
Implements [Arcade::Games::IGameModule](#).

6.16 Arcade::Games::Pacman Class Reference

[Pacman](#) game.

```
#include <Pacman.hpp>
```

Inheritance diagram for Arcade::Games::Pacman:



Public Member Functions

- void [reset](#) () final
Resets and restarts the game.
- void [update](#) (const [Arcade::Display::IDisplayModule](#) &displayModule) final
Updates the game.
- void [render](#) ([Arcade::Display::IDisplayModule](#) &displayModule) const final
Renders the [Pacman](#) game on the display module.

Additional Inherited Members

6.16.1 Detailed Description

[Pacman](#) game.

6.16.2 Member Function Documentation

6.16.2.1 render()

```
void Arcade::Games::Pacman::render (
    Arcade::Display::IDisplayModule & displayModule ) const [final], [virtual]
```

Renders the [Pacman](#) game on the display module.

Parameters

<i>displayModule</i>	The display module that will be used to put things on a canvas.
----------------------	---

Reimplemented from [Arcade::Games::AGameModule](#).

6.16.2.2 update()

```
void Arcade::Games::Pacman::update (
    const Arcade::Display::IDisplayModule & lib ) [final], [virtual]
```

Updates the game.

Parameters

<i>lib</i>	The display module that will be used to get events that occurred
------------	--

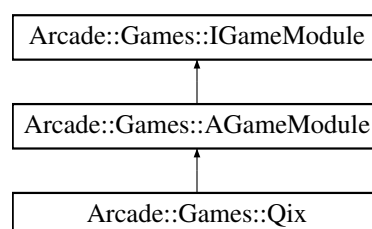
Implements [Arcade::Games::IGameModule](#).

6.17 Arcade::Games::Qix Class Reference

[Qix](#) game.

```
#include <Qix.hpp>
```

Inheritance diagram for `Arcade::Games::Qix`:



Public Member Functions

- void [reset](#) () final
Resets and restarts the game.
- void [update](#) (const [Arcade::Display::IDisplayModule](#) &lib) final
Updates the game.

Additional Inherited Members

6.17.1 Detailed Description

[Qix](#) game.

6.17.2 Member Function Documentation

6.17.2.1 update()

```
void Arcade::Games::Qix::update (
    const Arcade::Display::IDisplayModule & lib ) [final], [virtual]
```

Updates the game.

Parameters

<i>lib</i>	The display module that will be used to get events that occurred
------------	--

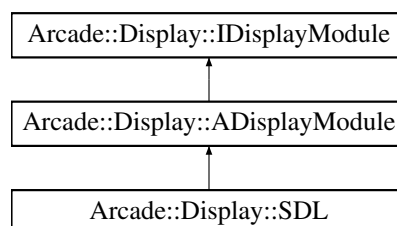
Implements [Arcade::Games::IGameModule](#).

6.18 Arcade::Display::SDL Class Reference

[SDL](#) library.

```
#include <SDL.hpp>
```

Inheritance diagram for [Arcade::Display::SDL](#):



Public Member Functions

- void [reset](#) () final
Resets the library.
- void [open](#) () final
Opens / initializes the window.
- void [close](#) () final
Close / destroy the window.
- bool [isOpen](#) () const final
Check window status.
- bool [switchToNextLib](#) () const final
Checks whether you need to change the current display library.
- bool [switchToPreviousLib](#) () const final
Checks whether you need to change the current display library.
- bool [switchToNextGame](#) () const final
Checks whether you need to change the current game library.
- bool [switchToPreviousGame](#) () const final
Checks whether you need to change the current game library.
- bool [shouldBeRestarted](#) () const final
Checks whether you need to restart the current game.
- bool [shouldGoToMenu](#) () const final
Checks whether you need to go back to the menu.
- bool [shouldExit](#) () const final
Checks whether you need to exit the program.
- bool [isKeyPressed](#) (IDisplayModule::Keys key) const final
Checks whether the current key is being pressed.
- bool [isKeyPressedOnce](#) (IDisplayModule::Keys key) const final
Checks whether the current key was pressed during the last frame.
- float [getDelta](#) () const final
Gets the number of frames since last update.
- void [clear](#) () const final
*Clears the canvas. **Call this after the IDisplayModule::update method.***
- void [update](#) () final
*Runs an update over the events that occurred. **Call this before the IDisplayModule::clear method.***
- void [render](#) () const final
Renders the canvas.
- char [getKeyCode](#) () const final
Gets the last pressed character from the keyboard.
- void [setColor](#) (IDisplayModule::Colors color) final
Defines the color of the elements that will be drawn.
- void [putPixel](#) (float x, float y) const final
Displays a pixel.
- void [putLine](#) (float x1, float y1, float x2, float y2) const final
Displays a line.
- void [putRect](#) (float x, float y, float w, float h) const final
Displays a rectangle.
- void [putFillRect](#) (float x, float y, float w, float h) const final
Displays a filled rectangle.
- void [putCircle](#) (float x, float y, float rad) const final
Displays a circle.
- void [putFillCircle](#) (float x, float y, float rad) const final
Displays a filled circle.
- void [putText](#) (const std::string &text, unsigned int size, float x, float y) const final
Displays text.

Additional Inherited Members

6.18.1 Detailed Description

[SDL](#) library.

6.18.2 Member Function Documentation

6.18.2.1 getDelta()

```
float Arcade::Display::SDL::getDelta ( ) const [final], [virtual]
```

Gets the number of frames since last update.

Returns

float Frame count

Implements [Arcade::Display::IDisplayModule](#).

6.18.2.2 getKeyCode()

```
char Arcade::Display::SDL::getKeyCode ( ) const [final], [virtual]
```

Gets the last pressed character from the keyboard.

Returns

\0 if nothing was pressed, \b if backspace was pressed, \n if return was pressed, otherwise, a character.

Implements [Arcade::Display::IDisplayModule](#).

6.18.2.3 isKeyPressed()

```
bool Arcade::Display::SDL::isKeyPressed (
    IDisplayModule::Keys key ) const [final], [virtual]
```

Checks whether the current key is being pressed.

Parameters

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

Returns

true Key is pressed
false Key is not pressed

Implements [Arcade::Display::IDisplayModule](#).

6.18.2.4 isKeyPressedOnce()

```
bool Arcade::Display::SDL::isKeyPressedOnce (
    IDisplayModule::Keys key ) const [final], [virtual]
```

Checks whether the current key was pressed during the last frame.

Parameters

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

Returns

true Key is pressed
false Key is not pressed

Implements [Arcade::Display::IDisplayModule](#).

6.18.2.5 isOpen()

```
bool Arcade::Display::SDL::isOpen ( ) const [final], [virtual]
```

Check window status.

Returns

true Window is open
false Window is closed

Implements [Arcade::Display::IDisplayModule](#).

6.18.2.6 putCircle()

```
void Arcade::Display::SDL::putCircle (
    float x,
    float y,
    float rad ) const [final], [virtual]
```

Displays a circle.

Parameters

<i>x</i>	X coordinates
<i>y</i>	Y coordinates
<i>rad</i>	Radius of the circle

Implements [Arcade::Display::IDisplayModule](#).

6.18.2.7 putFillCircle()

```
void Arcade::Display::SDL::putFillCircle (
    float x,
    float y,
    float rad ) const [final], [virtual]
```

Displays a filled circle.

Parameters

<i>x</i>	X coordinates
<i>y</i>	Y coordinates
<i>rad</i>	Radius of the circle

Implements [Arcade::Display::IDisplayModule](#).

6.18.2.8 putFillRect()

```
void Arcade::Display::SDL::putFillRect (
    float x,
    float y,
    float w,
    float h ) const [final], [virtual]
```

Displays a filled rectangle.

Parameters

<i>x</i>	X coordinates
<i>y</i>	Y coordinates
<i>w</i>	Width of the rectangle
<i>h</i>	Height of the rectangle

Implements [Arcade::Display::IDisplayModule](#).

6.18.2.9 putLine()

```
void Arcade::Display::SDL::putLine (
    float x1,
    float y1,
    float x2,
    float y2 ) const [final], [virtual]
```

Displays a line.

Parameters

<i>x1</i>	X coordinates for the first point
<i>y1</i>	Y coordinates for the first point
<i>x2</i>	X coordinates for the second point
<i>y2</i>	Y coordinates for the second point

Implements [Arcade::Display::IDisplayModule](#).

6.18.2.10 putPixel()

```
void Arcade::Display::SDL::putPixel (
    float x,
    float y ) const [final], [virtual]
```

Displays a pixel.

Parameters

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

Implements [Arcade::Display::IDisplayModule](#).

6.18.2.11 putRect()

```
void Arcade::Display::SDL::putRect (
    float x,
    float y,
    float w,
    float h ) const [final], [virtual]
```

Displays a rectangle.

Parameters

<i>x</i>	X coordinates
----------	---------------

Parameters

<i>y</i>	Y coordinates
<i>w</i>	Width of the rectangle
<i>h</i>	Height of the rectangle

Implements [Arcade::Display::IDisplayModule](#).

6.18.2.12 putText()

```
void Arcade::Display::SDL::putText (
    const std::string & text,
    unsigned int size,
    float x,
    float y ) const [final], [virtual]
```

Displays text.

Parameters

<i>text</i>	The text content
<i>size</i>	The text size
<i>x</i>	X coordinates
<i>y</i>	Y coordinates

Implements [Arcade::Display::IDisplayModule](#).

6.18.2.13 setColor()

```
void Arcade::Display::SDL::setColor (
    IDisplayModule::Colors color ) [final], [virtual]
```

Defines the color of the elements that will be drawn.

Parameters

<i>color</i>	The color
--------------	-----------

Implements [Arcade::Display::IDisplayModule](#).

6.18.2.14 shouldBeRestarted()

```
bool Arcade::Display::SDL::shouldBeRestarted ( ) const [final], [virtual]
```

Checks whether you need to restart the current game.

Returns

true Restart the game
false Do nothing

Implements [Arcade::Display::IDisplayModule](#).

6.18.2.15 shouldExit()

```
bool Arcade::Display::SDL::shouldExit ( ) const [final], [virtual]
```

Checks whether you need to exit the program.

Returns

true Exit the program
false Do nothing

Implements [Arcade::Display::IDisplayModule](#).

6.18.2.16 shouldGoToMenu()

```
bool Arcade::Display::SDL::shouldGoToMenu ( ) const [final], [virtual]
```

Checks whether you need to go back to the menu.

Returns

true Go back to menu
false Do nothing

Implements [Arcade::Display::IDisplayModule](#).

6.18.2.17 switchToNextGame()

```
bool Arcade::Display::SDL::switchToNextGame ( ) const [final], [virtual]
```

Checks whether you need to change the current game library.

Returns

true Switch to next available library
false Do nothing

Implements [Arcade::Display::IDisplayModule](#).

6.18.2.18 switchToNextLib()

```
bool Arcade::Display::SDL::switchToNextLib ( ) const [final], [virtual]
```

Checks whether you need to change the current display library.

Returns

true Switch to next available library
false Do nothing

Implements [Arcade::Display::IDisplayModule](#).

6.18.2.19 switchToPreviousGame()

```
bool Arcade::Display::SDL::switchToPreviousGame ( ) const [final], [virtual]
```

Checks whether you need to change the current game library.

Returns

true Switch to previous available library
false Do nothing

Implements [Arcade::Display::IDisplayModule](#).

6.18.2.20 switchToPreviousLib()

```
bool Arcade::Display::SDL::switchToPreviousLib ( ) const [final], [virtual]
```

Checks whether you need to change the current display library.

Returns

true Switch to previous available library
false Do nothing

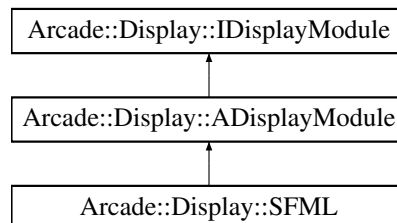
Implements [Arcade::Display::IDisplayModule](#).

6.19 Arcade::Display::SFML Class Reference

SFML library.

```
#include <SFML.hpp>
```

Inheritance diagram for Arcade::Display::SFML:



Public Member Functions

- void **reset** () final
Resets the library.
- void **open** () final
Opens / initializes the window.
- void **close** () final
Close / destroy the window.
- bool **isOpen** () const final
Check window status.
- bool **switchToNextLib** () const final
Checks whether you need to change the current display library.
- bool **switchToPreviousLib** () const final
Checks whether you need to change the current display library.
- bool **switchToNextGame** () const final
Checks whether you need to change the current game library.
- bool **switchToPreviousGame** () const final
Checks whether you need to change the current game library.
- bool **shouldBeRestarted** () const final
Checks whether you need to restart the current game.
- bool **shouldGoToMenu** () const final
Checks whether you need to go back to the menu.
- bool **shouldExit** () const final
Checks whether you need to exit the program.
- bool **isKeyPressed** (IDisplayModule::Keys key) const final
Checks whether the current key is being pressed.
- bool **isKeyPressedOnce** (IDisplayModule::Keys key) const final
Checks whether the current key was pressed during the last frame.
- float **getDelta** () const final
Gets the number of frames since last update.
- void **clear** () const final
*Clears the canvas. **Call this after the IDisplayModule::update method.***
- void **update** () final
*Runs an update over the events that occurred. **Call this before the IDisplayModule::clear method.***

- void `render` () const final
Renders the canvas.
- char `getKeyCode` () const final
Gets the last pressed character from the keyboard.
- void `setColor` (IDisplayModule::Colors color) final
Defines the color of the elements that will be drawn.
- void `putPixel` (float x, float y) const final
Displays a pixel.
- void `putLine` (float x1, float y1, float x2, float y2) const final
Displays a line.
- void `putRect` (float x, float y, float w, float h) const final
Displays a rectangle.
- void `putFillRect` (float x, float y, float w, float h) const final
Displays a filled rectangle.
- void `putCircle` (float x, float y, float rad) const final
Displays a circle.
- void `putFillCircle` (float x, float y, float rad) const final
Displays a filled circle.
- void `putText` (const std::string &text, unsigned int size, float x, float y) const final
Displays text.

Additional Inherited Members

6.19.1 Detailed Description

SFML library.

6.19.2 Member Function Documentation

6.19.2.1 `getDelta()`

```
float Arcade::Display::SFML::getDelta ( ) const [final], [virtual]
```

Gets the number of frames since last update.

Returns

float Frame count

Implements `Arcade::Display::IDisplayModule`.

6.19.2.2 getKeyCode()

```
char Arcade::Display::SFML::getKeyCode ( ) const [final], [virtual]
```

Gets the last pressed character from the keyboard.

Returns

\0 if nothing was pressed, \b if backspace was pressed, \n if return was pressed, otherwise, a character.

Implements [Arcade::Display::IDisplayModule](#).

6.19.2.3 isKeyPressed()

```
bool Arcade::Display::SFML::isKeyPressed (
    IDisplayModule::Keys key ) const [final], [virtual]
```

Checks whether the current key is being pressed.

Parameters

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

Returns

true Key is pressed
false Key is not pressed

Implements [Arcade::Display::IDisplayModule](#).

6.19.2.4 isKeyPressedOnce()

```
bool Arcade::Display::SFML::isKeyPressedOnce (
    IDisplayModule::Keys key ) const [final], [virtual]
```

Checks whether the current key was pressed during the last frame.

Parameters

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

Returns

true Key is pressed
false Key is not pressed

Implements [Arcade::Display::IDisplayModule](#).

6.19.2.5 isOpen()

```
bool Arcade::Display::SFML::isOpen ( ) const [final], [virtual]
```

Check window status.

Returns

true Window is open
false Window is closed

Implements [Arcade::Display::IDisplayModule](#).

6.19.2.6 putCircle()

```
void Arcade::Display::SFML::putCircle (
    float x,
    float y,
    float rad ) const [final], [virtual]
```

Displays a cirle.

Parameters

<i>x</i>	X coordinates
<i>y</i>	Y coordinates
<i>rad</i>	Radius of the circle

Implements [Arcade::Display::IDisplayModule](#).

6.19.2.7 putFillCircle()

```
void Arcade::Display::SFML::putFillCircle (
    float x,
    float y,
    float rad ) const [final], [virtual]
```

Displays a filled cirle.

Parameters

<i>x</i>	X coordinates
<i>y</i>	Y coordinates
<i>rad</i>	Radius of the circle

Implements [Arcade::Display::IDisplayModule](#).

6.19.2.8 putFillRect()

```
void Arcade::Display::SFML::putFillRect (
    float x,
    float y,
    float w,
    float h ) const [final], [virtual]
```

Displays a filled rectangle.

Parameters

<i>x</i>	X coordinates
<i>y</i>	Y coordinates
<i>w</i>	Width of the rectangle
<i>h</i>	Height of the rectangle

Implements [Arcade::Display::IDisplayModule](#).

6.19.2.9 putLine()

```
void Arcade::Display::SFML::putLine (
    float x1,
    float y1,
    float x2,
    float y2 ) const [final], [virtual]
```

Displays a line.

Parameters

<i>x1</i>	X coordinates for the first point
<i>y1</i>	Y coordinates for the first point
<i>x2</i>	X coordinates for the second point
<i>y2</i>	Y coordinates for the second point

Implements [Arcade::Display::IDisplayModule](#).

6.19.2.10 putPixel()

```
void Arcade::Display::SFML::putPixel (
    float x,
    float y ) const [final], [virtual]
```

Displays a pixel.

Parameters

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

Implements [Arcade::Display::IDisplayModule](#).

6.19.2.11 putRect()

```
void Arcade::Display::SFML::putRect (
    float x,
    float y,
    float w,
    float h ) const [final], [virtual]
```

Displays a rectangle.

Parameters

<i>x</i>	X coordinates
<i>y</i>	Y coordinates
<i>w</i>	Width of the rectangle
<i>h</i>	Height of the rectangle

Implements [Arcade::Display::IDisplayModule](#).

6.19.2.12 putText()

```
void Arcade::Display::SFML::putText (
    const std::string & text,
    unsigned int size,
    float x,
    float y ) const [final], [virtual]
```

Displays text.

Parameters

<i>text</i>	The text content
<i>size</i>	The text size
<i>x</i>	X coordinates
<i>y</i>	Y coordinates

Implements [Arcade::Display::IDisplayModule](#).

6.19.2.13 setColor()

```
void Arcade::Display::SFML::setColor (
    IDisplayModule::Colors color ) [final], [virtual]
```

Defines the color of the elements that will be drawn.

Parameters

<i>color</i>	The color
--------------	-----------

Implements [Arcade::Display::IDisplayModule](#).

6.19.2.14 shouldBeRestarted()

```
bool Arcade::Display::SFML::shouldBeRestarted ( ) const [final], [virtual]
```

Checks whether you need to restart the current game.

Returns

true Restart the game
false Do nothing

Implements [Arcade::Display::IDisplayModule](#).

6.19.2.15 shouldExit()

```
bool Arcade::Display::SFML::shouldExit ( ) const [final], [virtual]
```

Checks whether you need to exit the program.

Returns

true Exit the program
false Do nothing

Implements [Arcade::Display::IDisplayModule](#).

6.19.2.16 shouldGoToMenu()

```
bool Arcade::Display::SFML::shouldGoToMenu ( ) const [final], [virtual]
```

Checks whether you need to go back to the menu.

Returns

true Go back to menu
false Do nothing

Implements [Arcade::Display::IDisplayModule](#).

6.19.2.17 switchToNextGame()

```
bool Arcade::Display::SFML::switchToNextGame ( ) const [final], [virtual]
```

Checks whether you need to change the current game library.

Returns

true Switch to next available library
false Do nothing

Implements [Arcade::Display::IDisplayModule](#).

6.19.2.18 switchToNextLib()

```
bool Arcade::Display::SFML::switchToNextLib ( ) const [final], [virtual]
```

Checks whether you need to change the current display library.

Returns

true Switch to next available library
false Do nothing

Implements [Arcade::Display::IDisplayModule](#).

6.19.2.19 switchToPreviousGame()

```
bool Arcade::Display::SFML::switchToPreviousGame ( ) const [final], [virtual]
```

Checks whether you need to change the current game library.

Returns

true Switch to previous available library
false Do nothing

Implements [Arcade::Display::IDisplayModule](#).

6.19.2.20 switchToPreviousLib()

```
bool Arcade::Display::SFML::switchToPreviousLib ( ) const [final], [virtual]
```

Checks whether you need to change the current display library.

Returns

true Switch to previous available library
false Do nothing

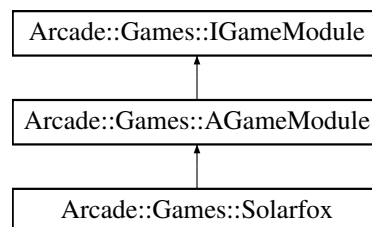
Implements [Arcade::Display::IDisplayModule](#).

6.20 Arcade::Games::Solarfox Class Reference

[Solarfox](#) game.

```
#include <Solarfox.hpp>
```

Inheritance diagram for Arcade::Games::Solarfox:



Public Member Functions

- void [reset](#) () final
Resets and restarts the game.
- void [update](#) (const [Arcade::Display::IDisplayModule](#) &lib) final
Updates the game.

Additional Inherited Members

6.20.1 Detailed Description

[Solarfox](#) game.

6.20.2 Member Function Documentation

6.20.2.1 `update()`

```
void Arcade::Games::Solarfox::update (
    const Arcade::Display::IDisplayModule & lib ) [final], [virtual]
```

Updates the game.

Parameters

<i>lib</i>	The display module that will be used to get events that occurred
------------	--

Implements [Arcade::Games::IGameModule](#).

Index

A

- Arcade::Display::IDisplayModule, [27](#)
- addToBestScores
 - Arcade::Games::AGameModule, [14](#)
- ADisplayModule
 - Arcade::Display::ADisplayModule, [12](#)
- AGameModule
 - Arcade::Games::AGameModule, [14](#)
- Arcade, [9](#)
- Arcade::Core, [22](#)
 - Core, [22](#)
- Arcade::Display, [9](#)
- Arcade::Display::ADisplayModule, [11](#)
 - ADisplayModule, [12](#)
 - ESCAPE, [12](#)
 - F1, [12](#)
 - F2, [12](#)
 - F3, [12](#)
 - F4, [12](#)
 - getLibName, [12](#)
 - M, [12](#)
 - R, [12](#)
 - SYSKEYS_END, [12](#)
 - SystemKeys, [12](#)
- Arcade::Display::IDisplayModule, [25](#)
 - A, [27](#)
 - BLACK, [27](#)
 - BLUE, [27](#)
 - Colors, [26](#)
 - COLORS_END, [27](#)
 - CYAN, [27](#)
 - D, [27](#)
 - DARK_GRAY, [27](#)
 - DEFAULT, [27](#)
 - DOWN, [27](#)
 - E, [27](#)
 - ENTER, [28](#)
 - getDelta, [28](#)
 - getKeyCode, [28](#)
 - getLibName, [28](#)
 - GREEN, [27](#)
 - I, [28](#)
 - isKeyPressed, [29](#)
 - isKeyPressedOnce, [29](#)
 - isOpen, [30](#)
 - J, [28](#)
 - K, [28](#)
 - Keys, [27](#)
 - KEYS_END, [28](#)
 - LEFT, [27](#)
 - LIGHT_BLUE, [27](#)
 - LIGHT_CYAN, [27](#)
 - LIGHT_GRAY, [27](#)
 - LIGHT_GREEN, [27](#)
 - LIGHT_MAGENTA, [27](#)
 - LIGHT_RED, [27](#)
 - LIGHT_YELLOW, [27](#)
 - MAGENTA, [27](#)
 - putCircle, [30](#)
 - putFillCircle, [30](#)
 - putFillRect, [31](#)
 - putLine, [31](#)
 - putPixel, [31](#)
 - putRect, [32](#)
 - putText, [32](#)
 - Q, [27](#)
 - RED, [27](#)
 - RIGHT, [27](#)
 - S, [27](#)
 - setColor, [33](#)
 - shouldBeRestarted, [33](#)
 - shouldExit, [33](#)
 - shouldGoToMenu, [33](#)
 - SPACE, [28](#)
 - switchToNextGame, [34](#)
 - switchToNextLib, [34](#)
 - switchToPreviousGame, [34](#)
 - switchToPreviousLib, [35](#)
 - U, [28](#)
 - UP, [27](#)
 - W, [27](#)
 - WHITE, [27](#)
 - X, [27](#)
 - YELLOW, [27](#)
 - Z, [27](#)
- Arcade::Display::Ncurses, [42](#)
 - getDelta, [44](#)
 - getKeyCode, [44](#)
 - isKeyPressed, [44](#)
 - isKeyPressedOnce, [45](#)
 - isOpen, [45](#)
 - putCircle, [45](#)
 - putFillCircle, [46](#)
 - putFillRect, [46](#)
 - putLine, [46](#)
 - putPixel, [47](#)
 - putRect, [47](#)
 - putText, [48](#)

- setColor, 48
- shouldBeRestarted, 48
- shouldExit, 49
- shouldGoToMenu, 49
- switchToNextGame, 49
- switchToNextLib, 49
- switchToPreviousGame, 50
- switchToPreviousLib, 50
- Arcade::Display::SDL, 54
 - getDelta, 56
 - getKeyCode, 56
 - isKeyPressed, 56
 - isKeyPressedOnce, 57
 - isOpen, 57
 - putCircle, 57
 - putFillCircle, 58
 - putFillRect, 58
 - putLine, 58
 - putPixel, 59
 - putRect, 59
 - putText, 60
 - setColor, 60
 - shouldBeRestarted, 60
 - shouldExit, 61
 - shouldGoToMenu, 61
 - switchToNextGame, 61
 - switchToNextLib, 61
 - switchToPreviousGame, 62
 - switchToPreviousLib, 62
- Arcade::Display::SFML, 63
 - getDelta, 64
 - getKeyCode, 64
 - isKeyPressed, 65
 - isKeyPressedOnce, 65
 - isOpen, 66
 - putCircle, 66
 - putFillCircle, 66
 - putFillRect, 67
 - putLine, 67
 - putPixel, 67
 - putRect, 68
 - putText, 68
 - setColor, 69
 - shouldBeRestarted, 69
 - shouldExit, 69
 - shouldGoToMenu, 69
 - switchToNextGame, 70
 - switchToNextLib, 70
 - switchToPreviousGame, 70
 - switchToPreviousLib, 71
- Arcade::DLInfos, 22
- Arcade::DLLoader< T >, 23
 - getInstance, 24
 - getLibraries, 24
 - loadLibrary, 24
- Arcade::Exceptions, 10
- Arcade::Exceptions::ArcadeException, 17
 - ArcadeException, 18
- getComponent, 18
- what, 19
- Arcade::Exceptions::BadFileException, 19
 - BadFileException, 19
- Arcade::Exceptions::BadInstanciacionException, 20
 - BadInstanciacionException, 20
- Arcade::Exceptions::InvalidLibraryException, 39
 - InvalidLibraryException, 40
- Arcade::Games, 10
- Arcade::Games::AGameModule, 13
 - addToBestScores, 14
 - AGameModule, 14
 - drawGameOver, 14
 - getBestScores, 15
 - getLibName, 15
 - getScore, 15
 - loadFromFile, 15, 16
 - render, 16
 - saveToFile, 16, 17
 - setPlayerName, 17
- Arcade::Games::Centipede, 21
 - update, 21
- Arcade::Games::IGameModule, 35
 - getBestScores, 36
 - getLibName, 36
 - getScore, 37
 - loadFromFile, 37
 - render, 38
 - saveToFile, 38
 - setPlayerName, 39
 - update, 39
- Arcade::Games::Nibbler, 51
 - render, 51
 - update, 52
- Arcade::Games::Pacman, 52
 - render, 53
 - update, 53
- Arcade::Games::Qix, 53
 - update, 54
- Arcade::Games::Solarfox, 71
 - update, 72
- Arcade::Logger, 40
 - DEBUG, 41
 - ERROR, 41
 - log, 41
 - LogLevel, 41
 - setLogLevel, 42
- ArcadeException
 - Arcade::Exceptions::ArcadeException, 18
- BadFileException
 - Arcade::Exceptions::BadFileException, 19
- BadInstanciacionException
 - Arcade::Exceptions::BadInstanciacionException, 20
- BLACK
 - Arcade::Display::IDisplayModule, 27
- BLUE
 - Arcade::Display::IDisplayModule, 27

- Colors
 - Arcade::Display::IDisplayModule, [26](#)
- COLORS_END
 - Arcade::Display::IDisplayModule, [27](#)
- Core
 - Arcade::Core, [22](#)
- CYAN
 - Arcade::Display::IDisplayModule, [27](#)
- D
 - Arcade::Display::IDisplayModule, [27](#)
- DARK_GRAY
 - Arcade::Display::IDisplayModule, [27](#)
- DEBUG
 - Arcade::Logger, [41](#)
- DEFAULT
 - Arcade::Display::IDisplayModule, [27](#)
- DOWN
 - Arcade::Display::IDisplayModule, [27](#)
- drawGameOver
 - Arcade::Games::AGameModule, [14](#)
- E
 - Arcade::Display::IDisplayModule, [27](#)
- ENTER
 - Arcade::Display::IDisplayModule, [28](#)
- ERROR
 - Arcade::Logger, [41](#)
- ESCAPE
 - Arcade::Display::ADisplayModule, [12](#)
- F1
 - Arcade::Display::ADisplayModule, [12](#)
- F2
 - Arcade::Display::ADisplayModule, [12](#)
- F3
 - Arcade::Display::ADisplayModule, [12](#)
- F4
 - Arcade::Display::ADisplayModule, [12](#)
- getBestScores
 - Arcade::Games::AGameModule, [15](#)
 - Arcade::Games::IGameModule, [36](#)
- getComponent
 - Arcade::Exceptions::ArcadeException, [18](#)
- getDelta
 - Arcade::Display::IDisplayModule, [28](#)
 - Arcade::Display::Ncurses, [44](#)
 - Arcade::Display::SDL, [56](#)
 - Arcade::Display::SFML, [64](#)
- getInstance
 - Arcade::DLLoader< T >, [24](#)
- getKeyCode
 - Arcade::Display::IDisplayModule, [28](#)
 - Arcade::Display::Ncurses, [44](#)
 - Arcade::Display::SDL, [56](#)
 - Arcade::Display::SFML, [64](#)
- getLibName
 - Arcade::Display::ADisplayModule, [12](#)
- Arcade::Display::IDisplayModule, [28](#)
- Arcade::Games::AGameModule, [15](#)
- Arcade::Games::IGameModule, [36](#)
- getLibraries
 - Arcade::DLLoader< T >, [24](#)
- getScore
 - Arcade::Games::AGameModule, [15](#)
 - Arcade::Games::IGameModule, [37](#)
- GREEN
 - Arcade::Display::IDisplayModule, [27](#)
- I
 - Arcade::Display::IDisplayModule, [28](#)
- InvalidLibraryException
 - Arcade::Exceptions::InvalidLibraryException, [40](#)
- isKeyPressed
 - Arcade::Display::IDisplayModule, [29](#)
 - Arcade::Display::Ncurses, [44](#)
 - Arcade::Display::SDL, [56](#)
 - Arcade::Display::SFML, [65](#)
- isKeyPressedOnce
 - Arcade::Display::IDisplayModule, [29](#)
 - Arcade::Display::Ncurses, [45](#)
 - Arcade::Display::SDL, [57](#)
 - Arcade::Display::SFML, [65](#)
- isOpen
 - Arcade::Display::IDisplayModule, [30](#)
 - Arcade::Display::Ncurses, [45](#)
 - Arcade::Display::SDL, [57](#)
 - Arcade::Display::SFML, [66](#)
- J
 - Arcade::Display::IDisplayModule, [28](#)
- K
 - Arcade::Display::IDisplayModule, [28](#)
- Keys
 - Arcade::Display::IDisplayModule, [27](#)
- KEYS_END
 - Arcade::Display::IDisplayModule, [28](#)
- LEFT
 - Arcade::Display::IDisplayModule, [27](#)
- LIGHT_BLUE
 - Arcade::Display::IDisplayModule, [27](#)
- LIGHT_CYAN
 - Arcade::Display::IDisplayModule, [27](#)
- LIGHT_GRAY
 - Arcade::Display::IDisplayModule, [27](#)
- LIGHT_GREEN
 - Arcade::Display::IDisplayModule, [27](#)
- LIGHT_MAGENTA
 - Arcade::Display::IDisplayModule, [27](#)
- LIGHT_RED
 - Arcade::Display::IDisplayModule, [27](#)
- LIGHT_YELLOW
 - Arcade::Display::IDisplayModule, [27](#)
- loadFromFile
 - Arcade::Games::AGameModule, [15](#), [16](#)

- Arcade::Games::IGameModule, 37
- loadLibrary
 - Arcade::DLLoader< T >, 24
- log
 - Arcade::Logger, 41
- LogLevel
 - Arcade::Logger, 41
- M
 - Arcade::Display::ADisplayModule, 12
- MAGENTA
 - Arcade::Display::IDisplayModule, 27
- putCircle
 - Arcade::Display::IDisplayModule, 30
 - Arcade::Display::Ncurses, 45
 - Arcade::Display::SDL, 57
 - Arcade::Display::SFML, 66
- putFillCircle
 - Arcade::Display::IDisplayModule, 30
 - Arcade::Display::Ncurses, 46
 - Arcade::Display::SDL, 58
 - Arcade::Display::SFML, 66
- putFillRect
 - Arcade::Display::IDisplayModule, 31
 - Arcade::Display::Ncurses, 46
 - Arcade::Display::SDL, 58
 - Arcade::Display::SFML, 67
- putLine
 - Arcade::Display::IDisplayModule, 31
 - Arcade::Display::Ncurses, 46
 - Arcade::Display::SDL, 58
 - Arcade::Display::SFML, 67
- putPixel
 - Arcade::Display::IDisplayModule, 31
 - Arcade::Display::Ncurses, 47
 - Arcade::Display::SDL, 59
 - Arcade::Display::SFML, 67
- putRect
 - Arcade::Display::IDisplayModule, 32
 - Arcade::Display::Ncurses, 47
 - Arcade::Display::SDL, 59
 - Arcade::Display::SFML, 68
- putText
 - Arcade::Display::IDisplayModule, 32
 - Arcade::Display::Ncurses, 48
 - Arcade::Display::SDL, 60
 - Arcade::Display::SFML, 68
- Q
 - Arcade::Display::IDisplayModule, 27
- R
 - Arcade::Display::ADisplayModule, 12
- RED
 - Arcade::Display::IDisplayModule, 27
- render
 - Arcade::Games::AGameModule, 16
 - Arcade::Games::IGameModule, 38
 - Arcade::Games::Nibbler, 51
 - Arcade::Games::Pacman, 53
- RIGHT
 - Arcade::Display::IDisplayModule, 27
- S
 - Arcade::Display::IDisplayModule, 27
- saveToFile
 - Arcade::Games::AGameModule, 16, 17
 - Arcade::Games::IGameModule, 38
- setColor
 - Arcade::Display::IDisplayModule, 33
 - Arcade::Display::Ncurses, 48
 - Arcade::Display::SDL, 60
 - Arcade::Display::SFML, 69
- setLogLevel
 - Arcade::Logger, 42
- setPlayerName
 - Arcade::Games::AGameModule, 17
 - Arcade::Games::IGameModule, 39
- shouldBeRestarted
 - Arcade::Display::IDisplayModule, 33
 - Arcade::Display::Ncurses, 48
 - Arcade::Display::SDL, 60
 - Arcade::Display::SFML, 69
- shouldExit
 - Arcade::Display::IDisplayModule, 33
 - Arcade::Display::Ncurses, 49
 - Arcade::Display::SDL, 61
 - Arcade::Display::SFML, 69
- shouldGoToMenu
 - Arcade::Display::IDisplayModule, 33
 - Arcade::Display::Ncurses, 49
 - Arcade::Display::SDL, 61
 - Arcade::Display::SFML, 69
- SPACE
 - Arcade::Display::IDisplayModule, 28
- switchToNextGame
 - Arcade::Display::IDisplayModule, 34
 - Arcade::Display::Ncurses, 49
 - Arcade::Display::SDL, 61
 - Arcade::Display::SFML, 70
- switchToNextLib
 - Arcade::Display::IDisplayModule, 34
 - Arcade::Display::Ncurses, 49
 - Arcade::Display::SDL, 61
 - Arcade::Display::SFML, 70
- switchToPreviousGame
 - Arcade::Display::IDisplayModule, 34
 - Arcade::Display::Ncurses, 50
 - Arcade::Display::SDL, 62
 - Arcade::Display::SFML, 70
- switchToPreviousLib
 - Arcade::Display::IDisplayModule, 35
 - Arcade::Display::Ncurses, 50
 - Arcade::Display::SDL, 62
 - Arcade::Display::SFML, 71
- SYSKEYS_END
 - Arcade::Display::ADisplayModule, 12

SystemKeys

- Arcade::Display::ADisplayModule, [12](#)

U

- Arcade::Display::IDisplayModule, [28](#)

UP

- Arcade::Display::IDisplayModule, [27](#)

update

- Arcade::Games::Centipede, [21](#)
- Arcade::Games::IGameModule, [39](#)
- Arcade::Games::Nibbler, [52](#)
- Arcade::Games::Pacman, [53](#)
- Arcade::Games::Qix, [54](#)
- Arcade::Games::Solarfox, [72](#)

W

- Arcade::Display::IDisplayModule, [27](#)

what

- Arcade::Exceptions::ArcadeException, [19](#)

WHITE

- Arcade::Display::IDisplayModule, [27](#)

X

- Arcade::Display::IDisplayModule, [27](#)

YELLOW

- Arcade::Display::IDisplayModule, [27](#)

Z

- Arcade::Display::IDisplayModule, [27](#)