# Arcade

Generated by Doxygen 1.9.1

# Chapter 1

# Namespace Index

## 1.1 Namespace List

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

# Chapter 2

# Hierarchical Index

## 2.1 Class Hierarchy

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

# Chapter 3

# Class Index

## 3.1 Class List

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

# Chapter 4

# Namespace Documentation

## 4.1 Arcade Namespace Reference

Namespace containing all the Arcade classes.

### Namespaces

- Centipede

  *The Centipede game library namespace.*
- Core

  *Namespace for the core of the arcade. It contains classes that are used to load, manage, and communicate between libraries.*
- Graphics

  *The namespace containing all the graphical libraries.*
- Nibbler

  *The Nibbler game library namespace.*

### Classes

- class IEntity

  *Interface of an entity. An entity is the building block of a game; That is to say, anything displayed by a graphical library is an entity (except for scores and controls). For instance, if you want to display walls, create a WallEntity and sets it every position where you want a wall to be. If you want particles, create a ParticleEntity with a small size and a specific texture and add to it as many positions as you want.*
- class IGameData

  *Interface for the game data. This interface contains all the method required to represent a game.*
- class IGame

  *The IGame class is the interface that all games must implement. For a game library to be compatible with the arcade, it must contains the following symbols:*
- class IDisplay

  *Interface for the display.*
- class TestInterface
- class XDisplay

  *A wrapper around the X11 display.*

### Typedefs

- typedef std::map< std::string, std::string > **ControlMap**

### Enumerations

- enum Key {
  Unknown = -1 , A = 0 , B , C ,
  D , E , F , G ,
  H , I , J , K ,
  L , M , N , O ,
  P , Q , R , S ,
  T , U , V , W ,
  X , Y , Z , Num0 ,
  Num1 , Num2 , Num3 , Num4 ,
  Num5 , Num6 , Num7 , Num8 ,
  Num9 , Escape , LControl , LShift ,
  LAlt , LSystem , RControl , RShift ,
  RAlt , RSystem , Menu , LBracket ,
  RBracket , Semicolon , Comma , Period ,
  Apostrophe , Slash , Backslash , Grave ,
  Equal , Hyphen , Space , Enter ,
  Backspace , Tab , PageUp , PageDown ,
  End , Home , Insert , Delete ,
  Add , Subtract , Multiply , Divide ,
  Left , Right , Up , Down ,
  Numpad0 , Numpad1 , Numpad2 , Numpad3 ,
  Numpad4 , Numpad5 , Numpad6 , Numpad7 ,
  Numpad8 , Numpad9 , F1 , F2 ,
  F3 , F4 , F5 , F6 ,
  F7 , F8 , F9 , F10 ,
  F11 , F12 , F13 , F14 ,
  F15 , Pause , KeyCount }

  *Enum of all the possible keys that can be pressed.*

## 4.1.1 Detailed Description

Namespace containing all the Arcade classes.

## 4.1.2 Enumeration Type Documentation

### 4.1.2.1 Key

`enum Arcade::Key`

Enum of all the possible keys that can be pressed.

**Enumerator**

| | |
|---|---|
| Unknown | Unhandled key. |
| A | The A key. |
| B | The B key. |
| C | The C key. |
| D | The D key. |
| E | The E key. |
| F | The F key. |
| G | The G key. |
| H | The H key. |
| I | The I key. |
| J | The J key. |
| K | The K key. |
| L | The L key. |
| M | The M key. |
| N | The N key. |
| O | The O key. |
| P | The P key. |
| Q | The Q key. |
| R | The R key. |
| S | The S key. |
| T | The T key. |
| U | The U key. |
| V | The V key. |
| W | The W key. |
| X | The X key. |
| Y | The Y key. |
| Z | The Z key. |
| Num0 | The 0 key. |
| Num1 | The 1 key. |
| Num2 | The 2 key. |
| Num3 | The 3 key. |
| Num4 | The 4 key. |
| Num5 | The 5 key. |
| Num6 | The 6 key. |
| Num7 | The 7 key. |
| Num8 | The 8 key. |
| Num9 | The 9 key. |
| Escape | The Escape key. |
| LControl | The left Control key. |
| LShift | The left Shift key. |
| LAlt | The left Alt key. |
| LSystem | The left OS specific key: window (Windows and Linux), apple (macOS), ... |
| RControl | The right Control key. |
| RShift | The right Shift key. |
| RAlt | The right Alt key. |
| RSystem | The right OS specific key: window (Windows and Linux), apple (macOS), ... |
| Menu | The Menu key. |

**Enumerator**

| | |
|---|---|
| LBracket | The [ key. |
| RBracket | The ] key. |
| Semicolon | The ; key. |
| Comma | The , key. |
| Period | The . key. |
| Apostrophe | The ' key. |
| Slash | The / key. |
| Backslash | The \ key. |
| Grave | The ` key. |
| Equal | The = key. |
| Hyphen | The - key (hyphen) |
| Space | The Space key. |
| Enter | The Enter/Return keys. |
| Backspace | The Backspace key. |
| Tab | The Tabulation key. |
| PageUp | The Page up key. |
| PageDown | The Page down key. |
| End | The End key. |
| Home | The Home key. |
| Insert | The Insert key. |
| Delete | The Delete key. |
| Add | The + key. |
| Subtract | The - key (minus, usually from numpad) |
| Multiply | The $*$ key. |
| Divide | The / key. |
| Left | Left arrow. |
| Right | Right arrow. |
| Up | Up arrow. |
| Down | Down arrow. |
| Numpad0 | The numpad 0 key. |
| Numpad1 | The numpad 1 key. |
| Numpad2 | The numpad 2 key. |
| Numpad3 | The numpad 3 key. |
| Numpad4 | The numpad 4 key. |
| Numpad5 | The numpad 5 key. |
| Numpad6 | The numpad 6 key. |
| Numpad7 | The numpad 7 key. |
| Numpad8 | The numpad 8 key. |
| Numpad9 | The numpad 9 key. |
| F1 | The F1 key. |
| F2 | The F2 key. |
| F3 | The F3 key. |
| F4 | The F4 key. |
| F5 | The F5 key. |
| F6 | The F6 key. |
| F7 | The F7 key. |
| F8 | The F8 key. |

**Enumerator**

| | |
|---|---|
| F9 | The F9 key. |
| F10 | The F10 key. |
| F11 | The F11 key. |
| F12 | The F12 key. |
| F13 | The F13 key. |
| F14 | The F14 key. |
| F15 | The F15 key. |
| Pause | The Pause key. |
| KeyCount | Keep last – the total number of keyboard keys. |

## 4.2 Arcade::Centipede Namespace Reference

The Centipede game library namespace.

### Classes

- class Game

  *The Centipede game class.*
- class Entity

  *A Centipede Entity.*
- class GameData

  *The Centipede GameData.*
- class Snake

  *A Centipede snake.*

### 4.2.1 Detailed Description

The Centipede game library namespace.

## 4.3 Arcade::Core Namespace Reference

Namespace for the core of the arcade. It contains classes that are used to load, manage, and communicate between libraries.

### Classes

- class Core

  *Class that handles the communication between the graphical and game libraries.*
- class LibHandle

  *A wrapper around a dynamic library handle.*
- class LibLoader

  *The LibLoader for arcade-like libraries.*

### 4.3.1 Detailed Description

Namespace for the core of the arcade. It contains classes that are used to load, manage, and communicate between libraries.

## 4.4 Arcade::Graphics Namespace Reference

The namespace containing all the graphical libraries.

### Namespaces

- NCurses

  *A wrapper around the ncurses library.*
- Sdl

  *A wrapper around the SDL2 library.*
- SFML

  *The SFML graphical library.*

### 4.4.1 Detailed Description

The namespace containing all the graphical libraries.

## 4.5 Arcade::Graphics::NCurses Namespace Reference

A wrapper around the ncurses library.

### Classes

- class Menu

  *A ncurses menu.*
- class NCurses

  *The NCurses graphical library.*
- class Texture

  *A ncurses texture.*
- class Window

  *A ncurses window.*

### Enumerations

- enum Color {
  **BLACK** = COLOR_BLACK , **RED** = COLOR_RED , **GREEN** = COLOR_GREEN , **YELLOW** = COLOR_↩
  YELLOW ,
  **BLUE** = COLOR_BLUE , **MAGENTA** = COLOR_MAGENTA , **CYAN** = COLOR_CYAN , **WHITE** = COLOR↩
  _WHITE }

  *A ncurses color.*

### 4.5.1 Detailed Description

A wrapper around the ncurses library.

## 4.6 Arcade::Graphics::Sdl Namespace Reference

A wrapper around the SDL2 library.

### Classes

- class Font

  *A SDL2 font.*
- class RectangleShape

  *A rectangle shape, with a size, a position and a color.*
- class RenderWindow

  *A window that can be drawn on.*
- class Sdl

  *The SDL2 graphical library.*
- class Sprite

  *A sprite, with a texture, a size, a position and a texture rect.*
- class Text

  *A text, with a font, a color and a text.*
- struct TextureRect
- class Texture

  *A wrapper around a SDL2 texture.*

### 4.6.1 Detailed Description

A wrapper around the SDL2 library.

## 4.7 Arcade::Graphics::SFML Namespace Reference

The SFML graphical library.

### Classes

- class SFML

  *The SFML graphical library.*

### 4.7.1 Detailed Description

The SFML graphical library.

## 4.8 Arcade::Nibbler Namespace Reference

The Nibbler game library namespace.

### Classes

- class Entity

  *A Nibbler Entity.*
- class GameData

  *The Nibbler GameData.*
- class Game

  *The Nibbler game class.*

### 4.8.1 Detailed Description

The Nibbler game library namespace.

# Chapter 5

# Class Documentation

## 5.1 Arcade::Core::Core Class Reference

Class that handles the communication between the graphical and game libraries.

`#include <Core.hpp>`

### Classes

- class LibraryNotLoadedException

  *Exception thrown when a library could not be loaded.*
- class NoLibraryException

  *Exception thrown when no library is found.*

### Public Member Functions

- Core (int ac, char **av)

  *Constructor for the Core class. It will pre-load the available libraries. Throws:*
- int run (const std::string &libName)

  *Runs the arcade. This function will select (via a menu) the graphic/game lib to play with, and run the main loop.*

### 5.1.1 Detailed Description

Class that handles the communication between the graphical and game libraries.

### 5.1.2 Constructor & Destructor Documentation

#### 5.1.2.1 Core()

```
Arcade::Core::Core::Core (
            int ac,
            char ** av )
```

Constructor for the Core class. It will pre-load the available libraries. Throws:

- Arcade::Core::NoLibraryException if no game / graphics library is found.
- Arcade::Core::LibraryNotLoadedException If the given library (av[1]) could not be loaded.

**Parameters**

| | |
|---|---|
| *ac* | The number of arguments. |
| *av* | The arguments. |

### 5.1.3 Member Function Documentation

#### 5.1.3.1 run()

```
int Arcade::Core::Core::run (
            const std::string & libName )
```

Runs the arcade. This function will select (via a menu) the graphic/game lib to play with, and run the main loop.

**Parameters**

| | |
|---|---|
| *libName* | |

**Returns**

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

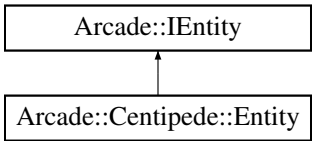
- include/core/Core.hpp

## 5.2 Arcade::Centipede::Entity Class Reference

A Centipede Entity.

```
#include <Entity.hpp>
```

Inheritance diagram for Arcade::Centipede::Entity:



## Public Member Functions

- Entity ()

  *Creates a new Entity.*
- Entity (std::vector< std::pair< float, float >> pos, std::pair< float, float > size, std::string texture, float rotation)

  *Creates a new Entity.*
- Entity (const Entity &entity)

  *Copy constructor.*
- void setPosition (std::vector< std::pair< float, float >> pos)

  *Sets the position of the Entity.*
- std::vector< std::pair< float, float > > getPosition () const override
- std::pair< float, float > getSize () const override
- std::string getTexture () const override
- float getRotation () const override
- void setSize (std::pair< float, float > size)

  *Sets the size of the Entity.*
- void setTexture (std::string texture)

  *Sets the texture of the Entity.*
- void setRotation (float rotation)

  *Sets the rotation of the Entity.*

### 5.2.1 Detailed Description

A Centipede Entity.

### 5.2.2 Constructor & Destructor Documentation

#### 5.2.2.1 Entity() [1/2]

```
Arcade::Centipede::Entity::Entity (
            std::vector< std::pair< float, float >> pos,
            std::pair< float, float > size,
            std::string texture,
            float rotation )
```

Creates a new Entity.

**Parameters**

| | |
|---|---|
| *pos* | The position of the Entity. |
| *size* | The size of the Entity. |
| *texture* | The texture of the Entity. |
| *rotation* | The rotation of the Entity. |

#### 5.2.2.2 Entity() [2/2]

```
Arcade::Centipede::Entity::Entity (
            const Entity & entity )
```

Copy constructor.

**Parameters**

| | |
|---|---|
| *entity* | The Entity to copy. |

### 5.2.3 Member Function Documentation

#### 5.2.3.1 getPosition()

```
std::vector<std::pair<float, float> > Arcade::Centipede::Entity::getPosition ( ) const  [override],
[virtual]
```

**See also**

IEntity::getPosition

Implements Arcade::IEntity.

#### 5.2.3.2 getRotation()

```
float Arcade::Centipede::Entity::getRotation ( ) const  [override], [virtual]
```

**See also**

IEntity::getRotation

Implements Arcade::IEntity.

#### 5.2.3.3 getSize()

```
std::pair<float, float> Arcade::Centipede::Entity::getSize ( ) const  [override], [virtual]
```

**See also**

IEntity::getSize

Implements Arcade::IEntity.

#### 5.2.3.4 getTexture()

```
std::string Arcade::Centipede::Entity::getTexture ( ) const  [override], [virtual]
```

**See also**

IEntity::getTexture

Implements Arcade::IEntity.

#### 5.2.3.5 setPosition()

```
void Arcade::Centipede::Entity::setPosition (
            std::vector< std::pair< float, float >> pos )
```

Sets the position of the Entity.

**Parameters**

| | |
|---|---|
| *pos* | The new position of the Entity. |

#### 5.2.3.6 setRotation()

```
void Arcade::Centipede::Entity::setRotation (
            float rotation )
```

Sets the rotation of the Entity.

**Parameters**

| | |
|---|---|
| *rotation* | The new rotation of the Entity. |

#### 5.2.3.7 setSize()

```
void Arcade::Centipede::Entity::setSize (
            std::pair< float, float > size )
```

Sets the size of the Entity.

**Parameters**

| | |
|---|---|
| *size* | The new size of the Entity. |

#### 5.2.3.8 setTexture()

```
void Arcade::Centipede::Entity::setTexture (
            std::string texture )
```

Sets the texture of the Entity.

**Parameters**

| | |
|---|---|
| *texture* | The new texture of the Entity. |

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

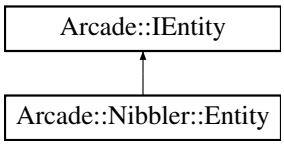
- include/games/centipede/Entity.hpp

## 5.3 Arcade::Nibbler::Entity Class Reference

A Nibbler Entity.

`#include <Entity.hpp>`

Inheritance diagram for Arcade::Nibbler::Entity:




### Public Member Functions

- Entity ()
  *Creates a new Entity.*
- Entity (std::vector< std::pair< float, float >> pos, std::pair< float, float > size, std::string texture, float rotation)
  *Creates a new Entity.*
- Entity (const Entity &entity)
  *Copy constructor.*
- std::vector< std::pair< float, float > > getPosition () const override
- std::pair< float, float > getSize () const override
- std::string getTexture () const override
- float getRotation () const override
- void setPosition (std::vector< std::pair< float, float >> pos)
  *Sets the position of the Entity.*
- void setSize (std::pair< float, float > size)
  *Sets the size of the Entity.*
- void setTexture (std::string texture)
  *Sets the texture of the Entity.*
- void setRotation (float rotation)
  *Sets the rotation of the Entity.*

### 5.3.1 Detailed Description

A Nibbler Entity.

### 5.3.2 Constructor & Destructor Documentation

#### 5.3.2.1 Entity() [1/2]

```
Arcade::Nibbler::Entity::Entity (
            std::vector< std::pair< float, float >> pos,
            std::pair< float, float > size,
            std::string texture,
            float rotation )
```

Creates a new Entity.

**Parameters**

| | |
|---|---|
| *pos* | The position of the Entity. |
| *size* | The size of the Entity. |
| *texture* | The texture of the Entity. |
| *rotation* | The rotation of the Entity. |

#### 5.3.2.2 Entity() [2/2]

```
Arcade::Nibbler::Entity::Entity (
            const Entity & entity )
```

Copy constructor.

**Parameters**

| | |
|---|---|
| *entity* | The Entity to copy. |

### 5.3.3 Member Function Documentation

#### 5.3.3.1 getPosition()

```
std::vector<std::pair<float, float> > Arcade::Nibbler::Entity::getPosition ( ) const  [override],
[virtual]
```

**See also**

IEntity::getPosition

Implements Arcade::IEntity.

#### 5.3.3.2 getRotation()

```
float Arcade::Nibbler::Entity::getRotation ( ) const  [override], [virtual]
```

**See also**

IEntity::getRotation

Implements Arcade::IEntity.

#### 5.3.3.3 getSize()

```
std::pair<float, float> Arcade::Nibbler::Entity::getSize ( ) const  [override], [virtual]
```

**See also**

IEntity::getSize

Implements Arcade::IEntity.

#### 5.3.3.4 getTexture()

```
std::string Arcade::Nibbler::Entity::getTexture ( ) const  [override], [virtual]
```

**See also**

IEntity::getTexture

Implements Arcade::IEntity.

#### 5.3.3.5 setPosition()

```
void Arcade::Nibbler::Entity::setPosition (
            std::vector< std::pair< float, float >> pos )
```

Sets the position of the Entity.

**Parameters**

| | |
|---|---|
| *pos* | The new position of the Entity. |

#### 5.3.3.6 setRotation()

```
void Arcade::Nibbler::Entity::setRotation (
            float rotation )
```

Sets the rotation of the Entity.

**Parameters**

| | |
|---|---|
| *rotation* | The new rotation of the Entity. |

#### 5.3.3.7 setSize()

```
void Arcade::Nibbler::Entity::setSize (
            std::pair< float, float > size )
```

Sets the size of the Entity.

**Parameters**

| | |
|---|---|
| *size* | The new size of the Entity. |

#### 5.3.3.8 setTexture()

```
void Arcade::Nibbler::Entity::setTexture (
            std::string texture )
```

Sets the texture of the Entity.

**Parameters**

| | |
|---|---|
| *texture* | The new texture of the Entity. |

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

- include/games/nibbler/Entity.hpp

## 5.4 Arcade::Graphics::Sdl::Font Class Reference

A SDL2 font.

`#include <Font.hpp>`

### Public Member Functions

- Font (const std::string &fontPath, int fontSize, bool bold=false)

  *Creates a new font.*
- TTF_Font * getRawFont () const

  *Gets the raw TTF_Font pointer.*

### 5.4.1 Detailed Description

A SDL2 font.

### 5.4.2 Constructor & Destructor Documentation

#### 5.4.2.1 Font()

```
Arcade::Graphics::Sdl::Font::Font (
            const std::string & fontPath,
            int fontSize,
            bool bold = false )
```

Creates a new font.

Once a font has been created, its size / boldness cannot be changed.

**Parameters**

| | |
|---|---|
| *fontPath* | The path to the font file (.ttf / .otf) |
| *fontSize* | The size of the font |
| *bold* | Whether the font should be bold or not |

### 5.4.3 Member Function Documentation

#### 5.4.3.1 getRawFont()

```
TTF_Font* Arcade::Graphics::Sdl::Font::getRawFont ( ) const
```

Gets the raw TTF_Font pointer.

**Returns**

The raw TTF_Font pointer.

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

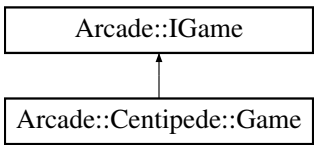
- include/libs/sdl/Font.hpp

## 5.5 Arcade::Centipede::Game Class Reference

The Centipede game class.

`#include <Centipede.hpp>`

Inheritance diagram for Arcade::Centipede::Game:




### Public Member Functions

- Game ()
  *Creates a new Centipede game.*
- void handleKeys (const std::vector< Key > &pressedKeys)
- void update (const std::string &username)
- IGameData & getGameData () const

### 5.5.1 Detailed Description

The Centipede game class.

### 5.5.2 Member Function Documentation

#### 5.5.2.1 getGameData()

`IGameData& Arcade::Centipede::Game::getGameData ( ) const [virtual]`

**See also**

IGame::getGameData

Implements Arcade::IGame.

#### 5.5.2.2 handleKeys()

```
void Arcade::Centipede::Game::handleKeys (
            const std::vector< Key > & pressedKeys )  [virtual]
```

**See also**

IGame::handleKeys

Implements Arcade::IGame.

#### 5.5.2.3 update()

```
void Arcade::Centipede::Game::update (
            const std::string & username )  [virtual]
```

**See also**

IGame::update

Implements Arcade::IGame.

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

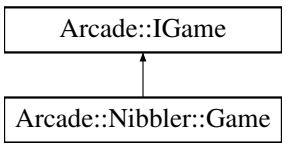
- include/games/centipede/Centipede.hpp

## 5.6 Arcade::Nibbler::Game Class Reference

The Nibbler game class.

`#include <Nibbler.hpp>`

Inheritance diagram for Arcade::Nibbler::Game:




### Public Member Functions

- Game ()
  *Creates a new Nibbler game.*
- void handleKeys (const std::vector< Key > &pressedKeys) override
- void update (const std::string &username) override
- IGameData & getGameData () const override

### 5.6.1 Detailed Description

The Nibbler game class.

### 5.6.2 Member Function Documentation

#### 5.6.2.1 getGameData()

```
IGameData& Arcade::Nibbler::Game::getGameData ( ) const  [override], [virtual]
```

**See also**

IGame::getGameData

Implements Arcade::IGame.

#### 5.6.2.2 handleKeys()

```
void Arcade::Nibbler::Game::handleKeys (
            const std::vector< Key > & pressedKeys )  [override], [virtual]
```

**See also**

IGame::handleKeys

Implements Arcade::IGame.

#### 5.6.2.3 update()

```
void Arcade::Nibbler::Game::update (
            const std::string & username )  [override], [virtual]
```

**See also**

IGame::update

Implements Arcade::IGame.

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

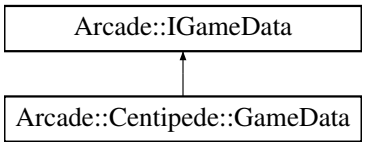
- include/games/nibbler/Nibbler.hpp

## 5.7 Arcade::Centipede::GameData Class Reference

The Centipede GameData.

`#include <GameData.hpp>`

Inheritance diagram for Arcade::Centipede::GameData:




### Public Member Functions

- GameData ()
  *Creates a new GameData.*
- std::map< std::string, int > getScores () const override
- std::string getGameName () const override
- std::vector< std::shared_ptr< IEntity > > & getEntities () override
- std::pair< int, int > getMapSize () const override
- bool isGameOver () const override
- const ControlMap & getControls () const override
- void addScore (std::string name, int score)
  *Adds a score to the GameData.*
- void addEntity (std::shared_ptr< IEntity > entity)
  *Adds an Entity to the GameData.*
- void removeEntities ()
  *Removes all the Entities from the GameData.*
- void setGameOver (int gameOver)
  *Sets the GameData as over.*

### 5.7.1 Detailed Description

The Centipede GameData.

### 5.7.2 Member Function Documentation

#### 5.7.2.1 addEntity()

```
void Arcade::Centipede::GameData::addEntity (
            std::shared_ptr< IEntity > entity )
```

Adds an Entity to the GameData.

**Parameters**

| | |
|---|---|
| *entity* | The Entity to add. |

#### 5.7.2.2 addScore()

```
void Arcade::Centipede::GameData::addScore (
            std::string name,
            int score )
```

Adds a score to the GameData.

**Parameters**

| | |
|---|---|
| *name* | The score name |
| *score* | The score value |

#### 5.7.2.3 getControls()

```
const ControlMap& Arcade::Centipede::GameData::getControls ( ) const  [inline], [override],
[virtual]
```

**See also**

IGameData::getControls

Implements Arcade::IGameData.

#### 5.7.2.4 getEntities()

```
std::vector<std::shared_ptr<IEntity> >& Arcade::Centipede::GameData::getEntities ( )  [override],
[virtual]
```

**See also**

IGameData::getEntities

Implements Arcade::IGameData.

#### 5.7.2.5 getGameName()

```
std::string Arcade::Centipede::GameData::getGameName ( ) const  [override], [virtual]
```

**See also**

IGameData::getGameName

Implements Arcade::IGameData.

#### 5.7.2.6 getMapSize()

```
std::pair<int, int> Arcade::Centipede::GameData::getMapSize ( ) const  [override], [virtual]
```

**See also**

IGameData::getMapSize

Implements Arcade::IGameData.

#### 5.7.2.7 getScores()

```
std::map<std::string, int> Arcade::Centipede::GameData::getScores ( ) const  [override], [virtual]
```

**See also**

IGameData::getScores

Implements Arcade::IGameData.

#### 5.7.2.8 isGameOver()

```
bool Arcade::Centipede::GameData::isGameOver ( ) const  [override], [virtual]
```

**See also**

IGameData::isGameOver

Implements Arcade::IGameData.

#### 5.7.2.9 setGameOver()

```
void Arcade::Centipede::GameData::setGameOver (
            int gameOver )
```

Sets the GameData as over.

**Parameters**

| | |
|---|---|
| *gameOver* | The new value of the gameover. |

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

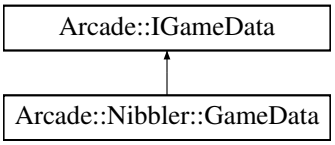
- include/games/centipede/GameData.hpp

## 5.8 Arcade::Nibbler::GameData Class Reference

The Nibbler GameData.

`#include <GameData.hpp>`

Inheritance diagram for Arcade::Nibbler::GameData:




### Public Member Functions

- bool isGameOver () const override
- std::map< std::string, int > getScores () const override
- std::string getGameName () const override
- std::vector< std::shared_ptr< IEntity > > & getEntities () override
- std::pair< int, int > getMapSize () const override
- const ControlMap & getControls () const override
- void addScore (std::string name, int score)

  *Adds a score to the GameData.*
- void addEntity (std::shared_ptr< IEntity > entity)

  *Adds an entity to the GameData.*
- void removeEntities ()

  *Removes all entities from the GameData.*
- void setGameOver (int gameOver)

  *Sets the game over state.*

### 5.8.1 Detailed Description

The Nibbler GameData.

### 5.8.2 Member Function Documentation

#### 5.8.2.1 addEntity()

```
void Arcade::Nibbler::GameData::addEntity (
            std::shared_ptr< IEntity > entity )
```

Adds an entity to the GameData.

**Parameters**

| | |
|---|---|
| *entity* | The entity to add |

#### 5.8.2.2 addScore()

```
void Arcade::Nibbler::GameData::addScore (
            std::string name,
            int score )
```

Adds a score to the GameData.

**Parameters**

| | |
|---|---|
| *name* | The score name |
| *score* | The score value |

#### 5.8.2.3 getControls()

```
const ControlMap& Arcade::Nibbler::GameData::getControls ( ) const  [inline], [override],
[virtual]
```

**See also**

IGameData::getControls

Implements Arcade::IGameData.

#### 5.8.2.4 getEntities()

```
std::vector<std::shared_ptr<IEntity> >& Arcade::Nibbler::GameData::getEntities ( )  [override],
[virtual]
```

**See also**

IGameData::getEntities

Implements Arcade::IGameData.

#### 5.8.2.5 getGameName()

```
std::string Arcade::Nibbler::GameData::getGameName ( ) const  [override], [virtual]
```

**See also**

IGameData::getGameName

Implements Arcade::IGameData.

#### 5.8.2.6 getMapSize()

```
std::pair<int, int> Arcade::Nibbler::GameData::getMapSize ( ) const  [override], [virtual]
```

**See also**

IGameData::getMapSize

Implements Arcade::IGameData.

#### 5.8.2.7 getScores()

```
std::map<std::string, int> Arcade::Nibbler::GameData::getScores ( ) const  [override], [virtual]
```

**See also**

IGameData::getScores

Implements Arcade::IGameData.

#### 5.8.2.8 isGameOver()

```
bool Arcade::Nibbler::GameData::isGameOver ( ) const  [override], [virtual]
```

**See also**

IGameData::isGameOver

Implements Arcade::IGameData.

#### 5.8.2.9 setGameOver()

```
void Arcade::Nibbler::GameData::setGameOver (
            int gameOver )
```

Sets the game over state.

**Parameters**

| | |
|---|---|
| *gameOver* | The game over state |

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

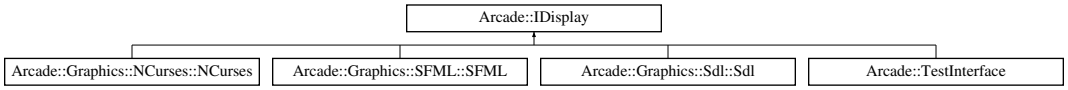
- include/games/nibbler/GameData.hpp

## 5.9 Arcade::IDisplay Class Reference

Interface for the display.

`#include <IDisplay.hpp>`

Inheritance diagram for Arcade::IDisplay:




### Public Member Functions

- virtual std::vector< Key > getPressedKeys ()=0

  *Get the pressed keys. To indicate an EXIT event, the display library must return a vector containing the Key::ESCAPE key. See Arcade::Key for the list of available keys.*
- virtual void render (IGameData &gameData)=0

  *Renders the game. This method should perform tne following actions:*
- virtual void renderMenu (const std::vector< std::string > &games, const std::vector< std::string > &graphics, int selectedGame, int selectedDisplay, const ControlMap &controls)=0

  *Renders the menu. This method should wait sufficient time to reach a static framerate.*

### 5.9.1 Detailed Description

Interface for the display.

For a display library to be compatible with the Arcade, it must contains the following symbols:

- "createDisplay" : a function that returns a pointer to an instance of Arcade::IDisplay
- "destroyDisplay" : a function that takes a pointer to an instance of Arcade::IDisplay as parameter and deletes it The only role of a display library is to render the game selection menu or the selected game, and to fetch the pressed keys. IT MUST NOT HANDLE ANY EVENTS (window resizing / closing, key pressed, etc.), NOR ANY GAME LOGIC. This actions are performed by the core.

### 5.9.2 Member Function Documentation

#### 5.9.2.1 getPressedKeys()

```
virtual std::vector<Key> Arcade::IDisplay::getPressedKeys ( )  [pure virtual]
```

Get the pressed keys. To indicate an EXIT event, the display library must return a vector containing the Key::↩ ESCAPE key. See Arcade::Key for the list of available keys.

**Note**

This method should handle EVERY possible pressed keys, as described in the Arcade::Key enum.

This method is called every frame.

**Returns**

A vector containing ALL the currently pressed keys

Implemented in Arcade::TestInterface, Arcade::Graphics::SFML::SFML, Arcade::Graphics::Sdl::Sdl, and Arcade::Graphics::NCurses::NCurses.

#### 5.9.2.2 render()

```
virtual void Arcade::IDisplay::render (
            IGameData & gameData )  [pure virtual]
```

Renders the game. This method should perform tne following actions:

- wait sufficient time to reach a static framerate
- draw every entities, scores, controls (described in the gameData parameter)

**Note**

This method is called every frame.

**Parameters**

| | |
|---|---|
| *gameData* | The game data, containing the entities, scores, controls, etc. |

Implemented in Arcade::TestInterface, Arcade::Graphics::SFML::SFML, Arcade::Graphics::Sdl::Sdl, and Arcade::Graphics::NCurses::NCurses.

#### 5.9.2.3 renderMenu()

```
virtual void Arcade::IDisplay::renderMenu (
            const std::vector< std::string > & games,
```

```
const std::vector< std::string > & graphics,
int selectedGame,
int selectedDisplay,
const ControlMap & controls )  [pure virtual]
```

Renders the menu. This method should wait sufficient time to reach a static framerate.

**Note**

This method is called every frame.

**Parameters**

| | |
|---|---|
| *games* | The list of available game libraries |
| *graphics* | The list of available graphics libraries |
| *selectedGame* | The index of the selected game (0 => first game) |
| *selectedDisplay* | The index of the selected display (0 => first display) |
| *controls* | A map, associating a key (as a string) to an action (as a string). It's only used to inform the user of the controls. |

Implemented in Arcade::TestInterface, Arcade::Graphics::SFML::SFML, Arcade::Graphics::NCurses::NCurses, and Arcade::Graphics::Sdl::Sdl.

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

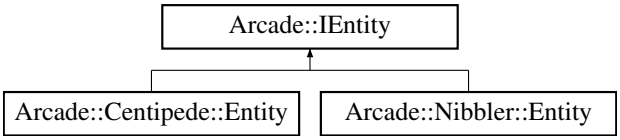
- include/IDisplay.hpp

## 5.10 Arcade::IEntity Class Reference

Interface of an entity. An entity is the building block of a game; That is to say, anything displayed by a graphical library is an entity (except for scores and controls). For instance, if you want to display walls, create a WallEntity and sets it every position where you want a wall to be. If you want particles, create a ParticleEntity with a small size and a specific texture and add to it as many positions as you want.

```
#include <GameInterfaces.hpp>
```

Inheritance diagram for Arcade::IEntity:



## Public Member Functions

- virtual std::vector< std::pair< float, float > > getPosition () const =0

  *Gets all the positions of the entity. These positions are expressed in terms of cell. The entity should be centered in the cell. For an entity in the middle of the screen, the position would be (map_size_x / 2, map_size_y / 2).*
- virtual std::pair< float, float > getSize () const =0

  *Gets the size of the entity. This size is expressed in terms of cell percentage. i.e, a size of (1, 1) means that the entity occupies the whole cell, while a size of (0.5, 0.5) means the entity occupies half of the cell.*
- virtual std::string getTexture () const =0

  *Gets the texture name of the entity. This method is used to fetch the adequate texture from the texture manager. Example: If the current display name is "libcaca", the game name is "nibbler", and the texture name is "player", the texture manager will look for the texture "assets/nibbler/libcaca/player". The textures should be correctly formatted for the display to be able to load them. If the texture could not be loaded, the behaviour is display-dependent.*
- virtual float getRotation () const =0

  *Gets the rotation of the entity. The rotation is expressed in degrees, and is clockwise.*

### 5.10.1 Detailed Description

Interface of an entity. An entity is the building block of a game; That is to say, anything displayed by a graphical library is an entity (except for scores and controls). For instance, if you want to display walls, create a WallEntity and sets it every position where you want a wall to be. If you want particles, create a ParticleEntity with a small size and a specific texture and add to it as many positions as you want.

### 5.10.2 Member Function Documentation

#### 5.10.2.1 getRotation()

```
virtual float Arcade::IEntity::getRotation ( ) const  [pure virtual]
```

Gets the rotation of the entity. The rotation is expressed in degrees, and is clockwise.

**Returns**

The rotation of the entity.

Implemented in Arcade::Nibbler::Entity, and Arcade::Centipede::Entity.

#### 5.10.2.2 getTexture()

```
virtual std::string Arcade::IEntity::getTexture ( ) const  [pure virtual]
```

Gets the texture name of the entity. This method is used to fetch the adequate texture from the texture manager. Example: If the current display name is "libcaca", the game name is "nibbler", and the texture name is "player", the texture manager will look for the texture "assets/nibbler/libcaca/player". The textures should be correctly formatted for the display to be able to load them. If the texture could not be loaded, the behaviour is display-dependent.

**Returns**

The texture name of the entity.

Implemented in Arcade::Nibbler::Entity, and Arcade::Centipede::Entity.

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

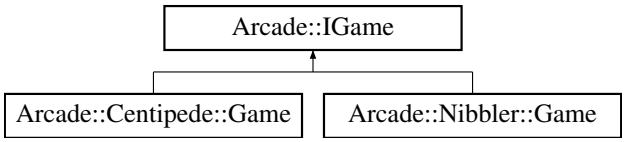
- include/GameInterfaces.hpp

## 5.11 Arcade::IGame Class Reference

The IGame class is the interface that all games must implement. For a game library to be compatible with the arcade, it must contains the following symbols:

`#include <GameInterfaces.hpp>`

Inheritance diagram for Arcade::IGame:




### Public Member Functions

- virtual void handleKeys (const std::vector< Key > &pressedKeys)=0

  *Handles the user input. This method may be used to set the direction of some entities, or to perform some actions. You may also want to store the pressed keys, so that the next time this method is called, you can check if a key was released.*
- virtual void update (const std::string &username)=0

  *Updates the game state. This method is the core of the game logic. Calls to this method should generally update the content of the game data. IMPORTANT: This method may not be called at a fixed rate, so it should update the game state according to the time elapsed since the last call.*
- virtual IGameData & getGameData () const =0

  *Gets the game data.*

### 5.11.1 Detailed Description

The IGame class is the interface that all games must implement. For a game library to be compatible with the arcade, it must contains the following symbols:

- "createGame" : A function that returns a pointer to an instance of Arcade::IGame.
- "destroyGame" : A function that takes a pointer to an instance of Arcade::IGame and deletes it. The only role of the game library is to perform the game logic, given the user input.

### 5.11.2 Member Function Documentation

#### 5.11.2.1 getGameData()

```
virtual IGameData& Arcade::IGame::getGameData ( ) const  [pure virtual]
```

Gets the game data.

**Returns**

A reference to the game data (as described by Arcade::IGameData)

Implemented in Arcade::Nibbler::Game, and Arcade::Centipede::Game.

#### 5.11.2.2 handleKeys()

```
virtual void Arcade::IGame::handleKeys (
            const std::vector< Key > & pressedKeys )  [pure virtual]
```

Handles the user input. This method may be used to set the direction of some entities, or to perform some actions. You may also want to store the pressed keys, so that the next time this method is called, you can check if a key was released.

**Parameters**

| | |
|---|---|
| *pressedKeys* | The list of currently pressed keys |

Implemented in Arcade::Nibbler::Game, and Arcade::Centipede::Game.

#### 5.11.2.3 update()

```
virtual void Arcade::IGame::update (
            const std::string & username )  [pure virtual]
```

Updates the game state. This method is the core of the game logic. Calls to this method should generally update the content of the game data. IMPORTANT: This method may not be called at a fixed rate, so it should update the game state according to the time elapsed since the last call.

**Parameters**

| | |
|---|---|
| *username* | |

Implemented in Arcade::Nibbler::Game, and Arcade::Centipede::Game.

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

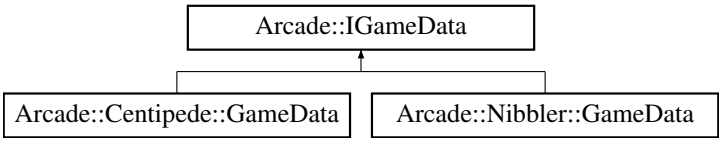
- include/GameInterfaces.hpp

## 5.12 Arcade::IGameData Class Reference

Interface for the game data. This interface contains all the method required to represent a game.

`#include <GameInterfaces.hpp>`

Inheritance diagram for Arcade::IGameData:



## Public Member Functions

- virtual std::map< std::string, int > getScores () const =0

  *Gets the differents scores of the game. The key of the map is the name of the score, and the value is the score. For example, a game with a score and a highscore would return a map with the following keys: "score" and "highscore". But, a game with a timer and a lives counter would return a map with the following keys: "timer" and "lives".*
- virtual std::string getGameName () const =0

  *Gets the name of the game. This method is used to (obviously) display the name of the game in the arcade, but it is also used to fetch the game's assets.*
- virtual std::vector< std::shared_ptr< Arcade::IEntity > > & getEntities ()=0

  *Gets the entities of the game. The entities are the objects that are displayed on the screen (see Arcade::IEntity).*
- virtual std::pair< int, int > getMapSize () const =0

  *Gets the size of the map, in terms of cell.*
- virtual const ControlMap & getControls () const =0

  *Gets the controls of the game. The controls are the keys that the user can use to play the game. The key of the map is the name of the control, and the value is the key that the user must press to perform the action. For example, a game with a "move left" and a "move right" control would return a map with the following keys: "move left" => "Q" and "move right" => "D".*
- virtual bool isGameOver () const =0

  *Gets the current state of the game.*

### 5.12.1 Detailed Description

Interface for the game data. This interface contains all the method required to represent a game.

### 5.12.2 Member Function Documentation

#### 5.12.2.1 getControls()

```
virtual const ControlMap& Arcade::IGameData::getControls ( ) const  [pure virtual]
```

Gets the controls of the game. The controls are the keys that the user can use to play the game. The key of the map is the name of the control, and the value is the key that the user must press to perform the action. For example, a game with a "move left" and a "move right" control would return a map with the following keys: "move left" => "Q" and "move right" => "D".

**Returns**

Implemented in Arcade::Nibbler::GameData, and Arcade::Centipede::GameData.

#### 5.12.2.2 getEntities()

```
virtual std::vector<std::shared_ptr<Arcade::IEntity> >& Arcade::IGameData::getEntities ( )
[pure virtual]
```

Gets the entities of the game. The entities are the objects that are displayed on the screen (see Arcade::IEntity).

**Returns**

A reference to a vector containing the entities of the game.

Implemented in Arcade::Nibbler::GameData, and Arcade::Centipede::GameData.

#### 5.12.2.3 getGameName()

```
virtual std::string Arcade::IGameData::getGameName ( ) const  [pure virtual]
```

Gets the name of the game. This method is used to (obviously) display the name of the game in the arcade, but it is also used to fetch the game's assets.

**Returns**

The name of the game.

Implemented in Arcade::Nibbler::GameData, and Arcade::Centipede::GameData.

#### 5.12.2.4 getMapSize()

```
virtual std::pair<int, int> Arcade::IGameData::getMapSize ( ) const  [pure virtual]
```

Gets the size of the map, in terms of cell.

**Returns**

A pair containing the size (x / y) of the map, in terms of cell.

Implemented in Arcade::Nibbler::GameData, and Arcade::Centipede::GameData.

#### 5.12.2.5 getScores()

```
virtual std::map<std::string, int> Arcade::IGameData::getScores ( ) const  [pure virtual]
```

Gets the differents scores of the game. The key of the map is the name of the score, and the value is the score. For example, a game with a score and a highscore would return a map with the following keys: "score" and "highscore". But, a game with a timer and a lives counter would return a map with the following keys: "timer" and "lives".

**Returns**

A map containing the scores of the game.

Implemented in Arcade::Nibbler::GameData, and Arcade::Centipede::GameData.

#### 5.12.2.6 isGameOver()

```
virtual bool Arcade::IGameData::isGameOver ( ) const  [pure virtual]
```

Gets the current state of the game.

**Returns**

True if the game is over, false otherwise.

Implemented in Arcade::Nibbler::GameData, and Arcade::Centipede::GameData.

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

- include/GameInterfaces.hpp

## 5.13 Arcade::Core::LibHandle Class Reference

A wrapper around a dynamic library handle.

```
#include <LibHandle.hpp>
```

### Public Member Functions

- LibHandle (const std::string &path)
  *Construct a new Library Handle.*
- **DELETE_COPY_MOVE** (LibHandle)
- operator bool () const
  *see LibHandle::isSet()*
- bool isSet () const
  *Check if the handle is valid.*
- bool symbolExists (const std::string &symbolName) const
  *Check if a symbol exists in the library.*
- template<typename T >
  T fetchSymbol (const std::string &symbolName) const
  *Fetch a symbol from the library.*

### 5.13.1 Detailed Description

A wrapper around a dynamic library handle.

### 5.13.2 Constructor & Destructor Documentation

#### 5.13.2.1 LibHandle()

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

Construct a new Library Handle.

**Parameters**

| | |
|---|---|
| *path* | The path to the dynamic library. |

### 5.13.3 Member Function Documentation

#### 5.13.3.1 fetchSymbol()

```
template<typename T >
T Arcade::Core::LibHandle::fetchSymbol (
            const std::string & symbolName ) const  [inline]
```

Fetch a symbol from the library.

**Template Parameters**

| | |
|---|---|
| *T* | The type of the symbol. |

**Parameters**

| | |
|---|---|
| *symbolName* | The name of the symbol. |

**Returns**

The symbol, null if can not be loaded.

#### 5.13.3.2 isSet()

```
bool Arcade::Core::LibHandle::isSet ( ) const
```

Check if the handle is valid.

**Returns**

True if the handle is valid, false otherwise.

#### 5.13.3.3 symbolExists()

```
bool Arcade::Core::LibHandle::symbolExists (
            const std::string & symbolName ) const
```

Check if a symbol exists in the library.

**Parameters**

| | |
|---|---|
| *symbolName* | The name of the symbol. |

**Returns**

True if the symbol exists, false otherwise.

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

- include/core/LibHandle.hpp

## 5.14 Arcade::Core::LibLoader Class Reference

The LibLoader for arcade-like libraries.

```
#include <LibLoader.hpp>
```

### Public Types

- enum LibType { **GRAPHICAL** , **GAME** , **ERROR** }

  *The type of the library.*

## Public Member Functions

- LibType getLibType (const std::string &path) const

  *Gets the type of the dynamic library.*
- std::string getLastError () const

  *Gets the last error that occurred.*
- IDisplay * loadGraphicalLib (const std::string &path)

  *Loads a graphical library.*
- void unloadGraphicalLib (IDisplay *lib)

  *Unloads a graphical library.*
- IGame * loadGameLib (const std::string &path)

  *Loads a game library.*
- void unloadGameLib (IGame *lib)

  *Unloads a game library.*

## Static Public Member Functions

- static LibLoader & getInstance ()

  *Gets the library loader.*

### 5.14.1 Detailed Description

The LibLoader for arcade-like libraries.

**Note**

This class is a singleton, and may be accessed using the getInstance() method.

### 5.14.2 Member Function Documentation

#### 5.14.2.1 getInstance()

```
static LibLoader& Arcade::Core::LibLoader::getInstance ( )  [inline], [static]
```

Gets the library loader.

**Note**

This handle is unique, and will be destroyed when the program is exited.

**Returns**

The library loader.

#### 5.14.2.2 getLastError()

```
std::string Arcade::Core::LibLoader::getLastError ( ) const
```

Gets the last error that occurred.

**Returns**

The last error that occurred.

#### 5.14.2.3 getLibType()

```
LibType Arcade::Core::LibLoader::getLibType (
            const std::string & path ) const
```

Gets the type of the dynamic library.

**Parameters**

| | |
|---|---|
| *path* | The path to the dynamic library. |

**Returns**

The type of the dynamic library.

#### 5.14.2.4 loadGameLib()

```
IGame* Arcade::Core::LibLoader::loadGameLib (
            const std::string & path )
```

Loads a game library.

**Parameters**

| | |
|---|---|
| *path* | The path to the library. |

**Returns**

The game library, nullptr is not loadable (see getLastError).

#### 5.14.2.5 loadGraphicalLib()

```
IDisplay* Arcade::Core::LibLoader::loadGraphicalLib (
            const std::string & path )
```

Loads a graphical library.

**Parameters**

| | |
|---|---|
| *path* | The path to the library. |

**Returns**

The graphical library, nullptr is not loadable (see getLastError).

#### 5.14.2.6 unloadGameLib()

```
void Arcade::Core::LibLoader::unloadGameLib (
            IGame * lib )
```

Unloads a game library.

**Parameters**

| | |
|---|---|
| *lib* | The library to unload. |

#### 5.14.2.7 unloadGraphicalLib()

```
void Arcade::Core::LibLoader::unloadGraphicalLib (
            IDisplay * lib )
```

Unloads a graphical library.

**Parameters**

| | |
|---|---|
| *lib* | The library to unload. |

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

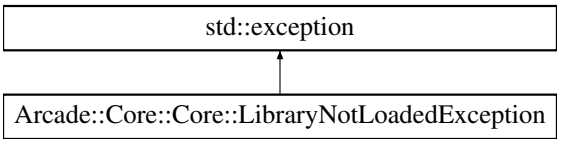
- include/core/LibLoader.hpp

## 5.15 Arcade::Core::Core::LibraryNotLoadedException Class Reference

Exception thrown when a library could not be loaded.

`#include <Core.hpp>`

Inheritance diagram for Arcade::Core::Core::LibraryNotLoadedException:




### Public Member Functions

- const char * **what** () const noexcept override

### 5.15.1 Detailed Description

Exception thrown when a library could not be loaded.

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

- include/core/Core.hpp

## 5.16 Arcade::Graphics::NCurses::Menu Class Reference

A ncurses menu.

`#include <Menu.hpp>`

### Public Member Functions

- Menu (Window *parent, const std::string &name, const Pos &pos, const std::vector< std::string > &items, int selectedItem, Size size={0, 0})

  *Creates a new menu.*
- std::string getName () const

  *Gets the name of the menu.*
- std::pair< int, int > getPos () const

  *Gets the position of the menu.*
- std::vector< std::string > getItems () const

  *Gets the items of the menu.*
- int getSelectedItem () const

  *Gets the selected item.*
- void setSelected (int selectedItem)

  *Sets the currently selected item.*
- void render ()

  *Renders the menu.*

### 5.16.1 Detailed Description

A ncurses menu.

A menu is a contains a name, a position, a list of items and a selected item, and a size.

### 5.16.2 Constructor & Destructor Documentation

#### 5.16.2.1 Menu()

```
Arcade::Graphics::NCurses::Menu::Menu (
            Window * parent,
            const std::string & name,
            const Pos & pos,
            const std::vector< std::string > & items,
            int selectedItem,
            Size size = {0, 0} )
```

Creates a new menu.

**Parameters**

| | |
|---|---|
| *parent* | The parent window. |
| *name* | The name of the menu. |
| *pos* | The position of the menu. |
| *items* | The items of the menu. |
| *selectedItem* | The selected item. |
| *size* | The size of the menu. |

### 5.16.3 Member Function Documentation

#### 5.16.3.1 getItems()

```
std::vector<std::string> Arcade::Graphics::NCurses::Menu::getItems ( ) const
```

Gets the items of the menu.

**Returns**

#### 5.16.3.2 getName()

```
std::string Arcade::Graphics::NCurses::Menu::getName ( ) const
```

Gets the name of the menu.

**Returns**

#### 5.16.3.3 getPos()

```
std::pair<int, int> Arcade::Graphics::NCurses::Menu::getPos ( ) const
```

Gets the position of the menu.

**Returns**

#### 5.16.3.4 getSelectedItem()

```
int Arcade::Graphics::NCurses::Menu::getSelectedItem ( ) const
```

Gets the selected item.

**Returns**

#### 5.16.3.5 render()

```
void Arcade::Graphics::NCurses::Menu::render ( )
```

Renders the menu.

This method draws the menu on the parent window, with a box around it.

#### 5.16.3.6 setSelected()

```
void Arcade::Graphics::NCurses::Menu::setSelected (
            int selectedItem )
```

Sets the currently selected item.

**Parameters**

| | |
|---|---|
| *selectedItem* | The selected item. |

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

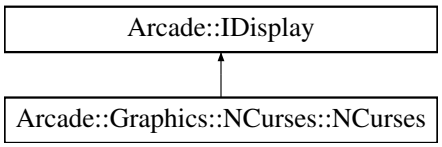
- include/libs/ncurses/Menu.hpp

## 5.17 Arcade::Graphics::NCurses::NCurses Class Reference

The NCurses graphical library.

`#include <NCurses.hpp>`

Inheritance diagram for Arcade::Graphics::NCurses::NCurses:




### Public Member Functions

- NCurses ()

  *Creates a new NCurses graphical library.*
- std::vector< Key > getPressedKeys () override
- void render (IGameData &gameData) override
- void renderMenu (const std::vector< std::string > &games, const std::vector< std::string > &graphics, int selectedGame, int selectedGraph, const ControlMap &map) override
- void setFramerateLimit (int fps)

  *Sets the framerate limit.*

### 5.17.1 Detailed Description

The NCurses graphical library.

### 5.17.2 Member Function Documentation

#### 5.17.2.1 getPressedKeys()

```
std::vector<Key> Arcade::Graphics::NCurses::NCurses::getPressedKeys ( )  [override], [virtual]
```

**See also**

Arcade::IDisplay::getPressedKeys()

Implements Arcade::IDisplay.

#### 5.17.2.2 render()

```
void Arcade::Graphics::NCurses::NCurses::render (
            IGameData & gameData )  [override], [virtual]
```

**See also**

Arcade::IDisplay::render()

Implements Arcade::IDisplay.

#### 5.17.2.3 renderMenu()

```
void Arcade::Graphics::NCurses::NCurses::renderMenu (
            const std::vector< std::string > & games,
            const std::vector< std::string > & graphics,
            int selectedGame,
            int selectedGraph,
            const ControlMap & map )  [override], [virtual]
```

**See also**

Arcade::IDisplay::renderMenu()

Implements Arcade::IDisplay.

#### 5.17.2.4 setFramerateLimit()

```
void Arcade::Graphics::NCurses::NCurses::setFramerateLimit (
            int fps )
```

Sets the framerate limit.

**Parameters**

| | |
|---|---|
| *fps* | The framerate limit. |

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

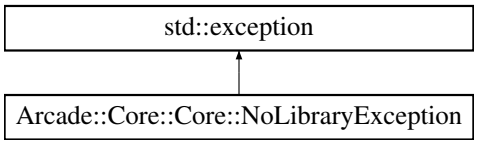
- include/libs/ncurses/NCurses.hpp

## 5.18 Arcade::Core::Core::NoLibraryException Class Reference

Exception thrown when no library is found.

`#include <Core.hpp>`

Inheritance diagram for Arcade::Core::Core::NoLibraryException:




### Public Member Functions

- **NoLibraryException** (LibLoader::LibType type)
- const char * **what** () const noexcept override

### 5.18.1 Detailed Description

Exception thrown when no library is found.

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

- include/core/Core.hpp

## 5.19 Arcade::Graphics::Sdl::RectangleShape Class Reference

A rectangle shape, with a size, a position and a color.

`#include <RectangleShape.hpp>`

## Public Member Functions

- RectangleShape ()

  *Creates a new rectangle shape, with a size of 0, a position of 0,0 and a color of black.*
- RectangleShape (SpriteSize size, SpriteSize position, const SDL_Color &fillColor)

  *Creates a new rectangle shape.*
- void setSize (const SpriteSize &size)

  *Sets the size of the rectangle.*
- void setPosition (const SpriteSize &position)

  *Sets the position of the rectangle.*
- void setFillColor (const SDL_Color &fillColor)

  *Sets the color of the rectangle.*
- SpriteSize getSize () const

  *Gets the size of the rectangle.*
- SpriteSize getPosition () const

  *Gets the position of the rectangle.*
- const SDL_Color & getFillColor () const

  *Gets the color of the rectangle.*

### 5.19.1 Detailed Description

A rectangle shape, with a size, a position and a color.

### 5.19.2 Constructor & Destructor Documentation

#### 5.19.2.1 RectangleShape()

```
Arcade::Graphics::Sdl::RectangleShape::RectangleShape (
            SpriteSize size,
            SpriteSize position,
            const SDL_Color & fillColor )
```

Creates a new rectangle shape.

**Parameters**

| | |
|---|---|
| *size* | The size of the rectangle. |
| *position* | The position of the rectangle. |
| *fillColor* | The color of the rectangle. |

### 5.19.3 Member Function Documentation

#### 5.19.3.1 getFillColor()

```
const SDL_Color& Arcade::Graphics::Sdl::RectangleShape::getFillColor ( ) const
```

Gets the color of the rectangle.

**Returns**

The color of the rectangle.

#### 5.19.3.2 getPosition()

```
SpriteSize Arcade::Graphics::Sdl::RectangleShape::getPosition ( ) const
```

Gets the position of the rectangle.

**Returns**

The position of the rectangle.

#### 5.19.3.3 getSize()

```
SpriteSize Arcade::Graphics::Sdl::RectangleShape::getSize ( ) const
```

Gets the size of the rectangle.

**Returns**

The size of the rectangle.

#### 5.19.3.4 setFillColor()

```
void Arcade::Graphics::Sdl::RectangleShape::setFillColor (
            const SDL_Color & fillColor )
```

Sets the color of the rectangle.

**Parameters**

| | |
|---|---|
| *fillColor* | The new color of the rectangle. |

#### 5.19.3.5 setPosition()

```
void Arcade::Graphics::Sdl::RectangleShape::setPosition (
            const SpriteSize & position )
```

Sets the position of the rectangle.

**Parameters**

| | |
|---|---|
| *position* | The new position of the rectangle. |

#### 5.19.3.6 setSize()

```
void Arcade::Graphics::Sdl::RectangleShape::setSize (
            const SpriteSize & size )
```

Sets the size of the rectangle.

**Parameters**

| | |
|---|---|
| *size* | The new size of the rectangle. |

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

- include/libs/sdl/RectangleShape.hpp

## 5.20 Arcade::Graphics::Sdl::RenderWindow Class Reference

A window that can be drawn on.

`#include <RenderWindow.hpp>`

### Public Member Functions

- RenderWindow (int width, int height)
  *Creates a new window.*
- void clear ()
  *Clears the window.*
- void draw (const Sprite &sprite)
  *Draws a sprite on the window.*
- void draw (Text &text)
  *Draws a text on the window.*

- void draw (const RectangleShape &rect)

  *Draws a rectangle on the window.*
- void drawLine (const SpriteSize &start, const SpriteSize &end, const SDL_Color &color)

  *Draws a line on the window.*
- void display ()

  *Actually renders everything that has been drawn.*
- SDL_Renderer * getRenderer () const

  *Gets the SDL2 raw window pointer.*

## 5.20.1 Detailed Description

A window that can be drawn on.

## 5.20.2 Constructor & Destructor Documentation

### 5.20.2.1 RenderWindow()

```
Arcade::Graphics::Sdl::RenderWindow::RenderWindow (
            int width,
            int height )
```

Creates a new window.

**Parameters**

| | |
|---|---|
| *width* | The width of the window. |
| *height* | The height of the window. |

## 5.20.3 Member Function Documentation

### 5.20.3.1 draw() [1/3]

```
void Arcade::Graphics::Sdl::RenderWindow::draw (
            const RectangleShape & rect )
```

Draws a rectangle on the window.

**Parameters**

| | |
|---|---|
| *rect* | The rectangle to draw. |

#### 5.20.3.2 draw() [2/3]

```
void Arcade::Graphics::Sdl::RenderWindow::draw (
            const Sprite & sprite )
```

Draws a sprite on the window.

**Parameters**

| | |
|---|---|
| *sprite* | The sprite to draw. |

#### 5.20.3.3 draw() [3/3]

```
void Arcade::Graphics::Sdl::RenderWindow::draw (
            Text & text )
```

Draws a text on the window.

**Parameters**

| | |
|---|---|
| *text* | The text to draw. |

#### 5.20.3.4 drawLine()

```
void Arcade::Graphics::Sdl::RenderWindow::drawLine (
            const SpriteSize & start,
            const SpriteSize & end,
            const SDL_Color & color )
```

Draws a line on the window.

**Parameters**

| | |
|---|---|
| *start* | The start pos of the line. |
| *end* | The end pos of the line. |
| *color* | The color of the line. |

#### 5.20.3.5 getRenderer()

```
SDL_Renderer* Arcade::Graphics::Sdl::RenderWindow::getRenderer ( ) const
```

Gets the SDL2 raw window pointer.

**Returns**

The SDL2 raw window pointer.

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

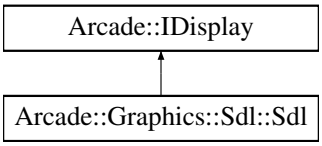
- include/libs/sdl/RenderWindow.hpp

## 5.21 Arcade::Graphics::Sdl::Sdl Class Reference

The SDL2 graphical library.

`#include <Sdl.hpp>`

Inheritance diagram for Arcade::Graphics::Sdl::Sdl:




### Public Member Functions

- Sdl ()

  *Creates a new SDL2 graphical library.*
- std::vector< Key > getPressedKeys () override
- void render (IGameData &gameData) override
- void renderMenu (const std::vector< std::string > &games, const std::vector< std::string > &graphics, int selectedGame, int selectedGraph, const ControlMap &controls) override

### Static Public Member Functions

- static void unloadTextures ()

  *Unloads all the textures.*

### 5.21.1 Detailed Description

The SDL2 graphical library.

### 5.21.2 Member Function Documentation

#### 5.21.2.1 getPressedKeys()

```
std::vector<Key> Arcade::Graphics::Sdl::Sdl::getPressedKeys ( )  [override], [virtual]
```

**See also**

Arcade::IDisplay::getPressedKeys()

Implements Arcade::IDisplay.

#### 5.21.2.2 render()

```
void Arcade::Graphics::Sdl::Sdl::render (
            IGameData & gameData )  [override], [virtual]
```

**See also**

Arcade::IDisplay::render()

Implements Arcade::IDisplay.

#### 5.21.2.3 renderMenu()

```
void Arcade::Graphics::Sdl::Sdl::renderMenu (
            const std::vector< std::string > & games,
            const std::vector< std::string > & graphics,
            int selectedGame,
            int selectedGraph,
            const ControlMap & controls )  [override], [virtual]
```

**See also**

Arcade::IDisplay::renderMenu()

Implements Arcade::IDisplay.

#### 5.21.2.4 unloadTextures()

```
static void Arcade::Graphics::Sdl::Sdl::unloadTextures ( )  [static]
```

Unloads all the textures.

This method is only called when the SDL2 library is unloaded (via deleteDisplay)

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

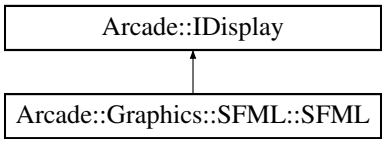
- include/libs/sdl/Sdl.hpp

# 5.22 Arcade::Graphics::SFML::SFML Class Reference

The SFML graphical library.

`#include <Sfml.hpp>`

Inheritance diagram for Arcade::Graphics::SFML::SFML:




## Public Member Functions

- SFML ()

  *Creates a new SFML graphical library.*
- std::vector< Key > getPressedKeys () override
- void render (IGameData &gameData) override
- void renderMenu (const std::vector< std::string > &games, const std::vector< std::string > &graphics, int selectedGame, int selectedGraph, const ControlMap &map) override

### 5.22.1 Detailed Description

The SFML graphical library.

### 5.22.2 Member Function Documentation

#### 5.22.2.1 getPressedKeys()

`std::vector<Key> Arcade::Graphics::SFML::SFML::getPressedKeys ( )  [override], [virtual]`

**See also**

Arcade::IDisplay::getPressedKeys()

**Returns**

Implements Arcade::IDisplay.

#### 5.22.2.2 render()

```
void Arcade::Graphics::SFML::SFML::render (
            IGameData & gameData )  [override], [virtual]
```

**See also**

Arcade::IDisplay::render()

Implements Arcade::IDisplay.

#### 5.22.2.3 renderMenu()

```
void Arcade::Graphics::SFML::SFML::renderMenu (
            const std::vector< std::string > & games,
            const std::vector< std::string > & graphics,
            int selectedGame,
            int selectedGraph,
            const ControlMap & map )  [override], [virtual]
```

**See also**

Arcade::IDisplay::renderMenu()

Implements Arcade::IDisplay.

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

- include/libs/sfml/Sfml.hpp

## 5.23 Arcade::Centipede::Snake Class Reference

A Centipede snake.

`#include <Snake.hpp>`

### Public Member Functions

- Snake ()
  *creates a new centipede*
- Snake (std::pair< int, int > head, std::pair< int, int > tail, std::vector< std::pair< int, int >> body, bool dir)
  *creates a new centipede*
- void folow (std::pair< int, int > pos)
  *Make the centipede goes to a position.*
- void move (char map[25][25])
  *Make the centipede move.*
- int touch (std::pair< int, int > pos, std::vector< Snake > &snakes)
  *Check if the centipede touch a position.*
- std::vector< std::pair< int, int > > getBody () const
  *Get the positions of the snake cells.*
- std::pair< int, int > getHead () const
  *Get the position of the head.*

### 5.23.1 Detailed Description

A Centipede snake.

### 5.23.2 Constructor & Destructor Documentation

#### 5.23.2.1 Snake()

```
Arcade::Centipede::Snake::Snake (
            std::pair< int, int > head,
            std::pair< int, int > tail,
            std::vector< std::pair< int, int >> body,
            bool dir )
```

creates a new centipede

**Parameters**

| | |
|---|---|
| *head* | Position of the head |
| *tail* | Position of the tail |
| *body* | Position of the body |
| *dir* | Direction of the snake |

### 5.23.3 Member Function Documentation

#### 5.23.3.1 folow()

```
void Arcade::Centipede::Snake::folow (
            std::pair< int, int > pos )
```

Make the centipede goes to a position.

**Parameters**

| | |
|---|---|
| *pos* | Position to go |

#### 5.23.3.2 getBody()

```
std::vector<std::pair<int, int> > Arcade::Centipede::Snake::getBody ( ) const
```

Get the positions of the snake cells.

**Returns**

List of the positions

#### 5.23.3.3 getHead()

```
std::pair<int, int> Arcade::Centipede::Snake::getHead ( ) const
```

Get the position of the head.

**Returns**

Position of the head

#### 5.23.3.4 move()

```
void Arcade::Centipede::Snake::move (
            char map[25][25] )
```

Make the centipede move.

**Parameters**

| | |
|---|---|
| *map* | Map of the game |

#### 5.23.3.5 touch()

```
int Arcade::Centipede::Snake::touch (
            std::pair< int, int > pos,
            std::vector< Snake > & snakes )
```

Check if the centipede touch a position.

**Parameters**

| | |
|---|---|
| *pos* | Position to check |
| *snakes* | List of the snakes |

**Returns**

0 if the centipede is not dead, 1 if the centipede is dead, 2 if the centipede is dead and the player win

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

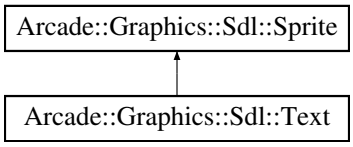
- include/games/centipede/Snake.hpp

## 5.24 Arcade::Graphics::Sdl::Sprite Class Reference

A sprite, with a texture, a size, a position and a texture rect.

`#include <Sprite.hpp>`

Inheritance diagram for Arcade::Graphics::Sdl::Sprite:




### Public Member Functions

- Sprite (const Texture &texture)

  *Creates a new sprite, with the size of the texture.*
- Sprite ()

  *Creates a new sprite, with no texture.*
- void setTexture (const Texture &texture)

  *Sets the texture of the sprite.*
- void setSize (const SpriteSize &size)

  *Sets the size of the sprite.*
- void setTextureRect (const TextureRect &rect)

  *Sets the zone of the texture to use.*
- void setPosition (const SpriteSize &position)

  *Sets the position of the sprite.*
- virtual const Texture & getTexture () const

  *Gets the texture of the sprite.*
- virtual SpriteSize getSize () const

  *Gets the size of the sprite.*
- virtual const TextureRect & getTextureRect () const

  *Gets the texture rect of the sprite.*
- const SpriteSize & getPosition () const

  *Gets the position of the sprite.*

### Protected Attributes

- const Texture * **_texture**
- SpriteSize **_size**
- SpriteSize **_position**
- TextureRect **_textureRect**

### 5.24.1 Detailed Description

A sprite, with a texture, a size, a position and a texture rect.

### 5.24.2 Constructor & Destructor Documentation

#### 5.24.2.1 Sprite()

```
Arcade::Graphics::Sdl::Sprite::Sprite (
            const Texture & texture )  [explicit]
```

Creates a new sprite, with the size of the texture.

**Parameters**

| | |
|---|---|
| *texture* | The texture of the sprite. |

### 5.24.3 Member Function Documentation

#### 5.24.3.1 getPosition()

```
const SpriteSize& Arcade::Graphics::Sdl::Sprite::getPosition ( ) const
```

Gets the position of the sprite.

**Returns**

The position of the sprite.

#### 5.24.3.2 getSize()

```
virtual SpriteSize Arcade::Graphics::Sdl::Sprite::getSize ( ) const  [virtual]
```

Gets the size of the sprite.

**Returns**

The size of the sprite.

Reimplemented in Arcade::Graphics::Sdl::Text.

#### 5.24.3.3 getTexture()

```
virtual const Texture& Arcade::Graphics::Sdl::Sprite::getTexture ( ) const  [virtual]
```

Gets the texture of the sprite.

**Returns**

The texture of the sprite.

#### 5.24.3.4 getTextureRect()

```
virtual const TextureRect& Arcade::Graphics::Sdl::Sprite::getTextureRect ( ) const  [virtual]
```

Gets the texture rect of the sprite.

**Returns**

The texture rect of the sprite.

#### 5.24.3.5 setPosition()

```
void Arcade::Graphics::Sdl::Sprite::setPosition (
            const SpriteSize & position )
```

Sets the position of the sprite.

**Parameters**

| | |
|---|---|
| *position* | The new position of the sprite. |

#### 5.24.3.6 setSize()

```
void Arcade::Graphics::Sdl::Sprite::setSize (
            const SpriteSize & size )
```

Sets the size of the sprite.

**Parameters**

| | |
|---|---|
| *size* | The new size of the sprite. |

#### 5.24.3.7 setTexture()

```
void Arcade::Graphics::Sdl::Sprite::setTexture (
            const Texture & texture )
```

Sets the texture of the sprite.

The size of the sprite will be set to the size of the texture.

**Parameters**

| | |
|---|---|
| *texture* | The new texture of the sprite. |

#### 5.24.3.8 setTextureRect()

```
void Arcade::Graphics::Sdl::Sprite::setTextureRect (
            const TextureRect & rect )
```

Sets the zone of the texture to use.

**Parameters**

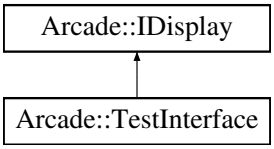
| | |
|---|---|
| *rect* | The new texture rect of the sprite. |

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

- include/libs/sdl/Sprite.hpp

## 5.25 Arcade::TestInterface Class Reference

Inheritance diagram for Arcade::TestInterface:




### Public Member Functions

- std::vector< Key > getPressedKeys () override

*Get the pressed keys. To indicate an EXIT event, the display library must return a vector containing the Key::ESCAPE key. See Arcade::Key for the list of available keys.*

- void render (IGameData &gameData) override

  *Renders the game. This method should perform tne following actions:*

- void renderMenu (const std::vector< std::string > &games, const std::vector< std::string > &graphics, int selectedGame, int selectedGraph, const ControlMap &map) override

  *Renders the menu. This method should wait sufficient time to reach a static framerate.*

## 5.25.1 Member Function Documentation

### 5.25.1.1 getPressedKeys()

```
std::vector<Key> Arcade::TestInterface::getPressedKeys ( )  [override], [virtual]
```

Get the pressed keys. To indicate an EXIT event, the display library must return a vector containing the Key::↩ ESCAPE key. See Arcade::Key for the list of available keys.

**Note**

This method should handle EVERY possible pressed keys, as described in the Arcade::Key enum.

This method is called every frame.

**Returns**

A vector containing ALL the currently pressed keys

Implements Arcade::IDisplay.

### 5.25.1.2 render()

```
void Arcade::TestInterface::render (
            IGameData & gameData )  [override], [virtual]
```

Renders the game. This method should perform tne following actions:

- wait sufficient time to reach a static framerate
- draw every entities, scores, controls (described in the gameData parameter)

**Note**

This method is called every frame.

**Parameters**

| | |
|---|---|
| *gameData* | The game data, containing the entities, scores, controls, etc. |

Implements Arcade::IDisplay.

#### 5.25.1.3 renderMenu()

```
void Arcade::TestInterface::renderMenu (
            const std::vector< std::string > & games,
            const std::vector< std::string > & graphics,
            int selectedGame,
            int selectedDisplay,
            const ControlMap & controls )  [override], [virtual]
```

Renders the menu. This method should wait sufficient time to reach a static framerate.

**Note**

This method is called every frame.

**Parameters**

| | |
|---|---|
| *games* | The list of available game libraries |
| *graphics* | The list of available graphics libraries |
| *selectedGame* | The index of the selected game (0 => first game) |
| *selectedDisplay* | The index of the selected display (0 => first display) |
| *controls* | A map, associating a key (as a string) to an action (as a string). It's only used to inform the user of the controls. |

Implements Arcade::IDisplay.

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

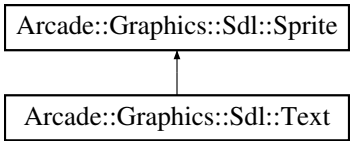
- include/libs/test/test.hpp

## 5.26 Arcade::Graphics::Sdl::Text Class Reference

A text, with a font, a color and a text.

`#include <Text.hpp>`

Inheritance diagram for Arcade::Graphics::Sdl::Text:



## Public Member Functions

- Text (SDL_Renderer *renderer, const Font *font=nullptr, const std::string &text="", SDL_Color color={255, 255, 255, 255})

  *Creates a new text.*
- void setText (const std::string &text)

  *Sets the text.*
- void setFont (const Font &font)

  *Sets the font.*
- void setColor (SDL_Color color)

  *Sets the color.*
- SpriteSize getSize () const override

  *Gets the size of the text.*
- SDL_Texture * getRawTexture ()

  *Gets the raw SDL_Texture pointer.*

## Additional Inherited Members

### 5.26.1 Detailed Description

A text, with a font, a color and a text.

### 5.26.2 Constructor & Destructor Documentation

#### 5.26.2.1 Text()

```
Arcade::Graphics::Sdl::Text::Text (
            SDL_Renderer * renderer,
            const Font * font = nullptr,
            const std::string & text = "",
            SDL_Color color = {255, 255, 255, 255} )  [explicit]
```

Creates a new text.

**Parameters**

| | |
|---|---|
| *renderer* | The SDL2 rendered to use |
| *font* | The font to use |
| *text* | The text to display |
| *color* | The color of the text |

### 5.26.3 Member Function Documentation

#### 5.26.3.1 getRawTexture()

```
SDL_Texture* Arcade::Graphics::Sdl::Text::getRawTexture ( )
```

Gets the raw SDL_Texture pointer.

**Note**

You should call this method to update the content of the text.

**Returns**

The raw SDL_Texture pointer.

#### 5.26.3.2 getSize()

```
SpriteSize Arcade::Graphics::Sdl::Text::getSize ( ) const  [override], [virtual]
```

Gets the size of the text.

**Returns**

The size of the text

Reimplemented from Arcade::Graphics::Sdl::Sprite.

#### 5.26.3.3 setColor()

```
void Arcade::Graphics::Sdl::Text::setColor (
            SDL_Color color )
```

Sets the color.

**Parameters**

| | |
|---|---|
| *color* | The new color |

#### 5.26.3.4 setFont()

```
void Arcade::Graphics::Sdl::Text::setFont (
            const Font & font )
```

Sets the font.

**Parameters**

| | |
|---|---|
| *font* | The new font |

#### 5.26.3.5 setText()

```
void Arcade::Graphics::Sdl::Text::setText (
            const std::string & text )
```

Sets the text.

**Parameters**

| | |
|---|---|
| *text* | The new text |

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

- include/libs/sdl/Text.hpp

## 5.27 Arcade::Graphics::NCurses::Texture Class Reference

A ncurses texture.

```
#include <Texture.hpp>
```

### Public Member Functions

- Texture (const std::string &path, int width, int height)
  *Creates a new texture.*
- std::string getContent () const
- Color **getTextColor** () const
- Color **getBackgroundColor** () const
- void **setSize** (int width, int height)
- void **setTextColor** (Color color)
- void **setBackgroundColor** (Color color)
- void **setContent** (char c)
- short **getColorPair** () const

### Static Public Member Functions

- static bool **createColorPair** (Color fg, Color bg)
- static void **removeColorPair** (Color fg, Color bg)

### 5.27.1 Detailed Description

A ncurses texture.

A texture is a character with a background color, a text color, and a size.

### 5.27.2 Constructor & Destructor Documentation

#### 5.27.2.1 Texture()

```
Arcade::Graphics::NCurses::Texture::Texture (
            const std::string & path,
            int width,
            int height )
```

Creates a new texture.

A valid texture must be formatted the following way:

text: <single char> bg-color: <color> text-color: <color>

Color must be among the following:

- black
- red
- green
- yellow
- blue
- magenta
- cyan
- white
- none

**Parameters**

| | |
|---|---|
| *path* | Path to the texture file. |
| *width* | Width of the texture. |
| *height* | Height of the texture. |

### 5.27.3 Member Function Documentation

#### 5.27.3.1 getContent()

```
std::string Arcade::Graphics::NCurses::Texture::getContent ( ) const
```

**Returns**

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

- include/libs/ncurses/Texture.hpp

## 5.28 Arcade::Graphics::Sdl::Texture Class Reference

A wrapper around a SDL2 texture.

```
#include <Texture.hpp>
```

### Public Member Functions

- Texture (const std::string &texturePath, SDL_Renderer *renderer)
  *Creates a new texture.*
- TextureSize getSize () const
  *Gets the size of the texture.*
- SDL_Texture * getRawTexture () const
  *Gets the raw SDL_Texture pointer.*

### 5.28.1 Detailed Description

A wrapper around a SDL2 texture.

### 5.28.2 Constructor & Destructor Documentation

#### 5.28.2.1 Texture()

```
Arcade::Graphics::Sdl::Texture::Texture (
            const std::string & texturePath,
            SDL_Renderer * renderer )  [explicit]
```

Creates a new texture.

**Parameters**

| | |
|---|---|
| *texturePath* | The path to the texture file. |
| *renderer* | The renderer to use to create the texture. |

### 5.28.3 Member Function Documentation

#### 5.28.3.1 getRawTexture()

```
SDL_Texture* Arcade::Graphics::Sdl::Texture::getRawTexture ( ) const
```

Gets the raw SDL_Texture pointer.

**Returns**

The raw SDL_Texture pointer.

#### 5.28.3.2 getSize()

```
TextureSize Arcade::Graphics::Sdl::Texture::getSize ( ) const
```

Gets the size of the texture.

**Returns**

The size of the texture.

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

- include/libs/sdl/Texture.hpp

## 5.29 Arcade::Graphics::Sdl::TextureRect Struct Reference

### Public Attributes

- int **left**
- int **top**
- int **width**
- int **height**

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

- include/libs/sdl/Texture.hpp

# 5.30 Arcade::Graphics::NCurses::Window Class Reference

A ncurses window.

`#include <Window.hpp>`

## Public Member Functions

- Window ()

  *Creates a new window, without any parents, positioned at (0, 0) and taking the whole screen.*
- Window (Window *parent, const Pos &pos, const Size &size)

  *Creates a new window.*
- Pos getPos () const

  *Gets the position of the window.*
- Size getSize ()

  *Gets the size of the window.*
- void drawBox ()

  *Draw a box around the window.*
- void draw (const std::string &text, const Pos &pos)

  *Draws a string (centered) at the given position.*
- void draw (const Texture &texture, Pos pos)

  *Draws a texture at the given position.*
- void clear ()

  *Clears the window.*

## Static Public Member Functions

- static Arcade::Key getKey ()

  *Get the next pressed key.*

### 5.30.1 Detailed Description

A ncurses window.

### 5.30.2 Constructor & Destructor Documentation

#### 5.30.2.1 Window()

```
Arcade::Graphics::NCurses::Window::Window (
            Window * parent,
            const Pos & pos,
            const Size & size )
```

Creates a new window.

**Parameters**

| | |
|---|---|
| *parent* | The parent window. |
| *pos* | The position of the window. |
| *size* | The size of the window. |

### 5.30.3 Member Function Documentation

#### 5.30.3.1 draw() [1/2]

```
void Arcade::Graphics::NCurses::Window::draw (
            const std::string & text,
            const Pos & pos )
```

Draws a string (centered) at the given position.

**Parameters**

| | |
|---|---|
| *text* | The string to draw. |
| *pos* | The position to draw the string at. |

#### 5.30.3.2 draw() [2/2]

```
void Arcade::Graphics::NCurses::Window::draw (
            const Texture & texture,
            Pos pos )
```

Draws a texture at the given position.

**Parameters**

| | |
|---|---|
| *texture* | The texture to draw. |
| *pos* | The position to draw the texture at. |

#### 5.30.3.3 getKey()

```
static Arcade::Key Arcade::Graphics::NCurses::Window::getKey ( )  [static]
```

Get the next pressed key.

**Returns**

#### 5.30.3.4 getPos()

```
Pos Arcade::Graphics::NCurses::Window::getPos ( ) const
```

Gets the position of the window.

**Returns**

#### 5.30.3.5 getSize()

```
Size Arcade::Graphics::NCurses::Window::getSize ( )
```

Gets the size of the window.

**Returns**

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

- include/libs/ncurses/Window.hpp

## 5.31 Arcade::XDisplay Class Reference

A wrapper around the X11 display.

```
#include <XDisplay.hpp>
```

### Static Public Member Functions

- static void setInputDelay (int ms)

  *Set the input delay.*
- static int getInputDelay ()

  *Get the input delay.*

### 5.31.1 Detailed Description

A wrapper around the X11 display.

### 5.31.2 Member Function Documentation

#### 5.31.2.1 getInputDelay()

```
static int Arcade::XDisplay::getInputDelay ( )  [static]
```

Get the input delay.

**Returns**

The input delay in milliseconds.

#### 5.31.2.2 setInputDelay()

```
static void Arcade::XDisplay::setInputDelay (
            int ms )  [static]
```

Set the input delay.

**Parameters**

| | |
|---|---|
| *ms* | The delay in milliseconds. |

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

- include/XDisplay.hpp