Arcade

Generated by Doxygen 1.8.15

Contents

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

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

Chapter 2

Class Index

2.1 Class List

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

Chapter 3

File Index

3.1 File List

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

Chapter 4

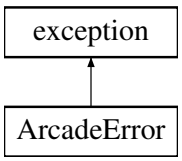
Class Documentation

4.1 ArcadeError Class Reference

Error handling class class we throw error with.

`#include <ArcadeError.hpp>`

Inheritance diagram for ArcadeError:



Public Member Functions

- ArcadeError (std::string msg)
  Constructor.
- const char * what () const throw ()
  convert the message string

4.1.1 Detailed Description

Error handling class class we throw error with.

class ArcadeError

4.1.2 Constructor & Destructor Documentation

4.1.2.1 ArcadeError()

```
ArcadeError::ArcadeError (
            std::string msg )  [inline]
```

Constructor.

Constructor for ArcadeError class

Parameters

msg	: error message

4.1.3 Member Function Documentation

4.1.3.1 what()

```
const char* ArcadeError::what ( ) const throw )   [inline]
```

convert the message string

convert message string into char * to print it

Returns

error message

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

- ArcadeError.hpp

4.2 DLLoader Class Reference

Loading library handle loading library dynamically while arcade is running.

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

Public Member Functions

- DLLoader ()
  Constructor.
- ~DLLoader ()
  Destructor.
- void initGame (const char *)
  loading first game
- void initGraph (const char *)
  loading first graphical lib
- void initListGame ()
  initialise the game's list
- void initListGraph ()
  initialise the graphicals lib's list
- void play ()
  launch games
- void close () const noexcept
  close all open libs
- void nextGame ()
  loading next game of the list
- void previousGame ()
  loading previous game of the list
- void nextGraph ()
  loading next graphical lib of the list
- void previousGraph ()
  loading previous graphical lib of the list

4.2.1 Detailed Description

Loading library handle loading library dynamically while arcade is running.

4.2.2 Constructor & Destructor Documentation

4.2.2.1 DLLoader()

```
DLLoader::DLLoader ( )
```

Constructor.

Constructor of DLLoader class

4.2.2.2 ~DLLoader()

```
DLLoader::~DLLoader ( )
```

Destructor.

Destructor of DLLoader class

4.2.3 Member Function Documentation

4.2.3.1 initGame()

```
void DLLoader::initGame (
            const char * lib_name )
```

loading first game

loading and initialising the first game

Parameters

game	lib's name

4.2.3.2 initGraph()

```
void DLLoader::initGraph (
```

```
            const char * lib_name )
```

loading first graphical lib

loading and initialising the first graphical lib

Parameters

graphical	lib's name

4.2.3.3 initListGame()

```
void DLLoader::initListGame ( )
```

initialise the game's list

reading into ./games and make a list of all games

4.2.3.4 initListGraph()

```
void DLLoader::initListGraph ( )
```

initialise the graphicals lib's list

reading into ./lib and make a list of all graphicals libs

4.2.3.5 play()

```
void DLLoader::play ( )
```

launch games

launch game and handle communciation between the game and the graphical lib

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

- DLLoader.hpp
- DLLoader.cpp

4.3 IGameHandler Class Reference

interface for games library to add a new game, you need to use this interface.

```
#include <IGameHandler.hpp>
```

Public Member Functions

- virtual void init ()=0
  init game env
- virtual void update (KeyInput::Key)=0
  update the game with the input given
- virtual void display (IGraphicHandler *)=0
  display games info
- virtual void **exitLib** ()=0

4.3.1 Detailed Description

interface for games library to add a new game, you need to use this interface.

4.3.2 Member Function Documentation

4.3.2.1 display()

```
virtual void IGameHandler::display (
            IGraphicHandler *  )  [pure virtual]
```

display games info

Parameters

the	interface of the graphical libs than you can make call to its function

4.3.2.2 update()

```
virtual void IGameHandler::update (
            KeyInput::Key  )  [pure virtual]
```

update the game with the input given

Parameters

key	: input key (see the enum) given by the core

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

- IGameHandler.hpp

4.4 IGraphicHandler Class Reference

interface for graphical library to add a new graphical lib, you need to use this interface.

`#include <GraphicElements.hpp>`

Public Member Functions

- virtual void init ()=0
  init graphical env
- virtual void createWindow (t_Vector2i)=0
  creating lib's window
- virtual void **cleanScreen** ()=0
- virtual void **putPixel** (t_Vector2i, t_Color)=0
- virtual KeyInput::Key **getInput** ()=0
- virtual void **drawRectangle** (t_Vector2i, t_Vector2i, t_Color)=0
- virtual void **writeText** (t_Vector2i, std::string)=0
- virtual void **refresh** ()=0
- virtual void **exitLib** ()=0

4.4.1 Detailed Description

interface for graphical library to add a new graphical lib, you need to use this interface.

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

- GraphicElements.hpp

4.5 s_Color Struct Reference

Public Member Functions

- **s_Color** (int r=255, int g=255, int b=255)

Public Attributes

- int **r**
- int **g**
- int **b**
- int **a**

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

- GraphicElements.hpp
- GraphicElements.cpp

4.6 s_Vector2i Struct Reference

Public Member Functions

- **s_Vector2i** (int x=0, int y=0)
- bool **operator==** (s_Vector2i)

Public Attributes

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

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

- GraphicElements.hpp
- GraphicElements.cpp

4.7 t_Color Struct Reference

color struct

`#include <GraphicElements.hpp>`

4.7.1 Detailed Description

color struct

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

- GraphicElements.hpp

4.8 t_Vecto2i Struct Reference

vector of int to handle sizes

`#include <GraphicElements.hpp>`

4.8.1 Detailed Description

vector of int to handle sizes

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

- GraphicElements.hpp

Chapter 5

File Documentation

5.1 ArcadeError.hpp File Reference

```
#include <stdexcept>
```

Classes

- class ArcadeError

  Error handling class class we throw error with.

5.1.1 Detailed Description

class for error handling

Author

flavian.feugueur@epitech.eu

Version

1.0

5.2 DLLoader.hpp File Reference

arcade's core

```
#include <algorithm>
#include <list>
#include <iostream>
#include <map>
#include "Input.hpp"
#include "IGameHandler.hpp"
#include "GraphicElements.hpp"
```

Classes

- class DLLoader

  Loading library handle loading library dynamically while arcade is running.

Macros

- #define **WIDTH** 50
- #define **HEIGHT** 50

5.2.1 Detailed Description

arcade's core

Author

flavian.feugueur@epitech.eu

Version

1.0

5.3 GraphicElements.hpp File Reference

grapic handling tool

```
#include <string>
#include "Input.hpp"
```

Classes

- struct s_Vector2i
- struct s_Color
- class IGraphicHandler

  interface for graphical library to add a new graphical lib, you need to use this interface.

Typedefs

- typedef struct s_Vector2i **t_Vector2i**
- typedef struct s_Color **t_Color**

5.3.1 Detailed Description

grapic handling tool

Author

leo.colin-vimard@epitech.eu

Version

1.0

5.4 IGameHandler.hpp File Reference

interface for games

```
#include "GraphicElements.hpp"
```

Classes

- class IGameHandler

  interface for games library to add a new game, you need to use this interface.

5.4.1 Detailed Description

interface for games

Author

leo.colin-vimard@epitech.eu

Version

1.0

5.5 Input.hpp File Reference

key input enum

Enumerations

- enum KeyInput::Key {
  UNKNOWN, **A**, **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**,
  Quote, **Slash**, **BackSlash**, **Tilde**,
  Equal, **Dash**, **Space**, **Return**,
  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 }

5.5.1 Detailed Description

key input enum

Author

hugo.tallineau@epitech.eu

Version

1.0

5.5.2 Enumeration Type Documentation

5.5.2.1 Key

`enum KeyInput::Key`

contain all key input a player can input

Index