Arcade Project

Generated by Doxygen 1.9.7

1 Hierarchical Index 1

1.1 Class Hierarchy . . . 1

2 Class Index 3

2.1 Class List . . . 3

3 File Index 5

3.1 File List . . . 5

4 Class Documentation 7

4.1 IGrid::Cell Struct Reference . . . 7

4.1.1 Member Data Documentation . . . 7

4.1.1.1 color . . . 7

4.1.1.2 x . . . 8

4.1.1.3 y . . . 8

4.2 IClock Class Reference . . . 8

4.2.1 Constructor & Destructor Documentation . . . 9

4.2.1.1 ~IClock() . . . 9

4.2.2 Member Function Documentation . . . 9

4.2.2.1 getTimeElapsed() . . . 9

4.2.2.2 initClock() . . . 9

4.2.2.3 resetClock() . . . 9

4.2.2.4 startClock() . . . 10

4.3 IEntity Class Reference . . . 10

4.3.1 Member Enumeration Documentation . . . 11

4.3.1.1 Color . . . 11

4.3.2 Constructor & Destructor Documentation . . . 12

4.3.2.1 ~IEntity() . . . 12

4.3.3 Member Function Documentation . . . 12

4.3.3.1 destroy() . . . 12

4.3.3.2 displayEntity() . . . 12

4.3.3.3 setPosition() . . . 12

4.4 IGame Class Reference . . . 13

4.4.1 Constructor & Destructor Documentation . . . 13

4.4.1.1 ~IGame() . . . 13

4.4.2 Member Function Documentation . . . 14

4.4.2.1 getEventBinding() . . . 14

4.4.2.2 processGameTick() . . . 14

4.4.2.3 restart() . . . 15

4.5 IGraphicalFactory Class Reference . . . 15

4.5.1 Constructor & Destructor Documentation . . . 16

4.5.1.1 ~IGraphicalFactory() . . . 16

4.5.2 Member Function Documentation . . . 16

4.5.2.1 createIClock() . . . 16
4.5.2.2 createWindow() . . . 16
4.5.2.3 destroyRessource() . . . 17
4.5.2.4 loadResource() . . . 17
4.6 IGrid Class Reference . . . 17
4.6.1 Constructor & Destructor Documentation . . . 19
4.6.1.1 ~IGrid() . . . 19
4.6.2 Member Function Documentation . . . 19
4.6.2.1 create() . . . 19
4.6.2.2 updateCell() . . . 19
4.7 IMenu Class Reference . . . 20
4.7.1 Member Typedef Documentation . . . 21
4.7.1.1 LambdaCoreLoading . . . 21
4.7.2 Constructor & Destructor Documentation . . . 21
4.7.2.1 ~IMenu() . . . 21
4.7.3 Member Function Documentation . . . 21
4.7.3.1 getEventBinding() . . . 21
4.7.3.2 loadCoreActions() . . . 21
4.7.3.3 processMenuTick() . . . 22
4.7.3.4 restart() . . . 22
4.8 IText Class Reference . . . 23
4.8.1 Member Enumeration Documentation . . . 25
4.8.1.1 EntityType . . . 25
4.8.2 Constructor & Destructor Documentation . . . 25
4.8.2.1 ~IText() . . . 25
4.8.3 Member Function Documentation . . . 25
4.8.3.1 changeString() . . . 25
4.8.3.2 create() . . . 26
4.9 IWindow Class Reference . . . 26
4.9.1 Member Typedef Documentation . . . 27
4.9.1.1 EventCallBack . . . 28
4.9.1.2 EventHandler . . . 28
4.9.2 Member Enumeration Documentation . . . 28
4.9.2.1 EventType . . . 28
4.9.3 Constructor & Destructor Documentation . . . 28
4.9.3.1 ~IWindow() . . . 28
4.9.4 Member Function Documentation . . . 28
4.9.4.1 callEvent() . . . 29
4.9.4.2 clear() . . . 29
4.9.4.3 closeWindow() . . . 29
4.9.4.4 createIGrid() . . . 29
4.9.4.5 createIText() . . . 30

4.9.4.6 display() . . . 30
4.9.4.7 eventPollEvent() . . . 30
4.9.4.8 initWindow() . . . 30
4.9.4.9 loadEventBindings() . . . 31
4.9.4.10 windowIsOpen() . . . 31

5 File Documentation **33**

5.1 Interface/Game/IGame.hpp File Reference . . . 33
5.2 IGame.hpp . . . 34
5.3 Interface/Graphical/IClock.hpp File Reference . . . 34
5.4 IClock.hpp . . . 34
5.5 Interface/Graphical/IEntity.hpp File Reference . . . 35
5.6 IEntity.hpp . . . 36
5.7 Interface/Graphical/IGraphicalFactory.hpp File Reference . . . 37
5.8 IGraphicalFactory.hpp . . . 38
5.9 Interface/Graphical/IWindow.hpp File Reference . . . 38
5.10 IWindow.hpp . . . 39
5.11 Interface/Menu/IMenu.hpp File Reference . . . 41
5.12 IMenu.hpp . . . 41

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

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

IGrid::Cell 7
IClock 8
IEntity 10
    IGrid 17
    IText 23
IGame 13
IGraphicalFactory 15
IMenu 20
IWindow 26

Chapter 2

Class Index

2.1 Class List

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

IGrid::Cell 7
IClock 8
IEntity 10
IGame 13
IGraphicalFactory 15
IGrid 17
IMenu 20
IText 23
IWindow 26

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

Interface/Game/IGame.hpp 33
Interface/Graphical/IClock.hpp 34
Interface/Graphical/IEntity.hpp 35
Interface/Graphical/IGraphicalFactory.hpp 37
Interface/Graphical/IWindow.hpp 38
Interface/Menu/IMenu.hpp 41

Chapter 4

Class Documentation

4.1 IGrid::Cell Struct Reference

`#include <IEntity.hpp>`

Collaboration diagram for IGrid::Cell:

Public Attributes

- Color color
- size_t x
- size_t y

4.1.1 Member Data Documentation

4.1.1.1 color

`Color IGrid::Cell::color`

4.1.1.2 x

```
size_t IGrid::Cell::x
```

4.1.1.3 y

```
size_t IGrid::Cell::y
```

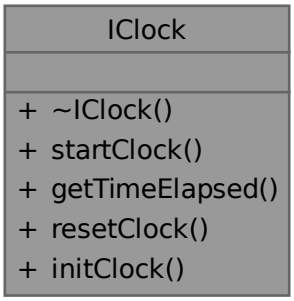
The documentation for this struct was generated from the following file:

- Interface/Graphical/IEntity.hpp

4.2 IClock Class Reference

```
#include <IClock.hpp>
```

Collaboration diagram for IClock:



Public Member Functions

- ~IClock ()=default
- virtual void startClock ()=0

  Start the clock used to measure time in the game or simulation. This is a pure virtual function that must be implemented by the derived class. Its purpose is to start the clock used to measure time in the game or simulation.
- virtual double getTimeElapsed ()=0

  Get the time elapsed since the start of the clock. Its purpose is to retrieve the time elapsed since the start of the clock used to measure time in the game or simulation.
- virtual void resetClock ()=0

  Reset the clock used to measure time in the game or simulation. Its purpose is to reset the clock used to measure time in the game or simulation to its initial state.
- virtual void initClock ()=0

  Initialize the clock used to measure time in the game or simulation. Its purpose is to initialize the clock used to measure time in the game or simulation, setting it to its initial state.

4.2.1 Constructor & Destructor Documentation

4.2.1.1 ~IClock()

```
IClock::~IClock ( )  [default]
```

4.2.2 Member Function Documentation

4.2.2.1 getTimeElapsed()

```
virtual double IClock::getTimeElapsed ( )  [pure virtual]
```

Get the time elapsed since the start of the clock. Its purpose is to retrieve the time elapsed since the start of the clock used to measure time in the game or simulation.

Returns

The time elapsed, in seconds.

4.2.2.2 initClock()

```
virtual void IClock::initClock ( )  [pure virtual]
```

Initialize the clock used to measure time in the game or simulation. Its purpose is to initialize the clock used to measure time in the game or simulation, setting it to its initial state.

4.2.2.3 resetClock()

```
virtual void IClock::resetClock ( )  [pure virtual]
```

Reset the clock used to measure time in the game or simulation. Its purpose is to reset the clock used to measure time in the game or simulation to its initial state.

4.2.2.4 startClock()

```
virtual void IClock::startClock ( )  [pure virtual]
```

Start the clock used to measure time in the game or simulation. This is a pure virtual function that must be implemented by the derived class. Its purpose is to start the clock used to measure time in the game or simulation.

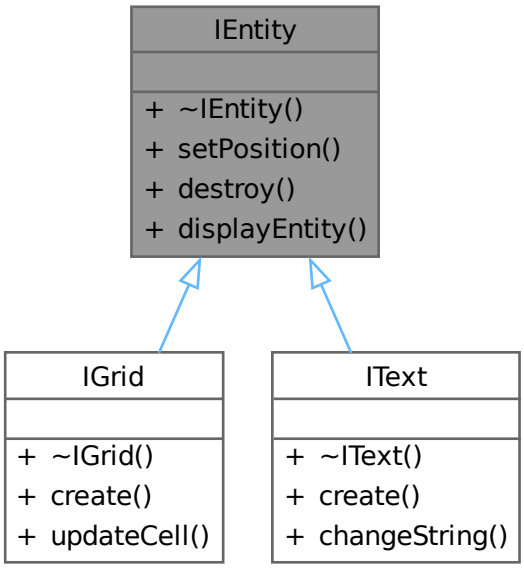
The documentation for this class was generated from the following file:

- Interface/Graphical/IClock.hpp

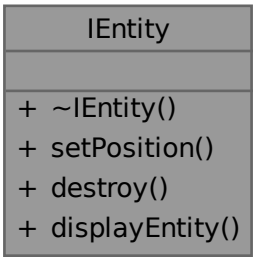
4.3 IEntity Class Reference

```
#include <IEntity.hpp>
```

Inheritance diagram for IEntity:



Collaboration diagram for IEntity:



Public Types

- enum class Color {
  Blue , Green , Orange , Red ,
  Brown , Yellow }

Public Member Functions

- ~IEntity ()=default
- virtual void setPosition (int x, int y)=0

  Sets the position of the entity. Its purpose is to set the position of the entity to the specified x and y coordinates. The function takes two integer parameters representing the x and y coordinates to set as the new position of the entity.
- virtual void destroy ()=0

  Destroys any resources used in the implementation. Its purpose is to destroy any resources used in the implementation, such as windows or textures.
- virtual void displayEntity ()=0

  Displays an entity on the screen. Its purpose is to display an entity on the screen, such as a sprite or image.

4.3.1 Member Enumeration Documentation

4.3.1.1 Color

```
enum class IEntity::Color  [strong]
```

Enumerator

Blue	
Green	
Orange	
Red	
Brown	
Yellow	

4.3.2 Constructor & Destructor Documentation

4.3.2.1 ~IEntity()

```
IEntity::~IEntity ( )  [default]
```

4.3.3 Member Function Documentation

4.3.3.1 destroy()

```
virtual void IEntity::destroy ( )  [pure virtual]
```

Destroys any resources used in the implementation. Its purpose is to destroy any resources used in the implementation, such as windows or textures.

4.3.3.2 displayEntity()

```
virtual void IEntity::displayEntity ( )  [pure virtual]
```

Displays an entity on the screen. Its purpose is to display an entity on the screen, such as a sprite or image.

4.3.3.3 setPosition()

```
virtual void IEntity::setPosition (
            int x,
            int y )  [pure virtual]
```

Sets the position of the entity. Its purpose is to set the position of the entity to the specified x and y coordinates. The function takes two integer parameters representing the x and y coordinates to set as the new position of the entity.

Parameters

x	
y	

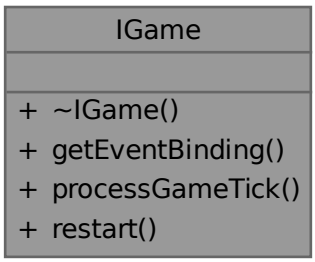
The documentation for this class was generated from the following file:

- Interface/Graphical/IEntity.hpp

4.4 IGame Class Reference

`#include <IGame.hpp>`

Collaboration diagram for IGame:



Public Member Functions

- ~IGame ()=default
- virtual IWindow::EventHandler & getEventBinding ()=0

  Returns the event handler object for the window. Its purpose is to return the event handler object for the window, allowing the client to modify or access event bindings. The function returns a reference to the IWindow::EventHandler object associated with the window.
- virtual bool processGameTick (IGrid &grid, IText &scoreText, IText &timeText, IClock &clock)=0

  Process a single tick of the game or simulation. Its purpose is to process a single tick of the game or simulation, updating the state of the game or simulation accordingly. The function takes four parameters: a reference to an IGrid object representing the game board or simulation space, a reference to an IText object representing the score or other information to be displayed, a reference to an IClock object representing the game or simulation timer, and a reference to an IText object representing the time or other information to be displayed. The function returns a boolean value indicating whether the game or simulation should continue running. If false, the game or simulation should end and any necessary cleanup should be performed.
- virtual void restart ()=0

  Restarts the implementation's game or simulation. Its purpose is to restart the game or simulation implemented by the class, resetting all necessary variables and objects to their initial state.

4.4.1 Constructor & Destructor Documentation

4.4.1.1 ~IGame()

`IGame::~IGame ( ) [default]`

4.4.2 Member Function Documentation

4.4.2.1 getEventBinding()

```
virtual IWindow::EventHandler & IGame::getEventBinding ( )  [pure virtual]
```

Returns the event handler object for the window. Its purpose is to return the event handler object for the window, allowing the client to modify or access event bindings. The function returns a reference to the IWindow::EventHandler object associated with the window.

Returns

IWindow::EventHandler&

4.4.2.2 processGameTick()

```
virtual bool IGame::processGameTick (
            IGrid & grid,
            IText & scoreText,
            IText & timeText,
            IClock & clock )  [pure virtual]
```

Process a single tick of the game or simulation. Its purpose is to process a single tick of the game or simulation, updating the state of the game or simulation accordingly. The function takes four parameters: a reference to an IGrid object representing the game board or simulation space, a reference to an IText object representing the score or other information to be displayed, a reference to an IClock object representing the game or simulation timer, and a reference to an IText object representing the time or other information to be displayed. The function returns a boolean value indicating whether the game or simulation should continue running. If false, the game or simulation should end and any necessary cleanup should be performed.

Parameters

grid	
scoreText	
timeText	
clock	

Returns

true

false

4.4.2.3 restart()

```
virtual void IGame::restart ( )  [pure virtual]
```

Restarts the implementation's game or simulation. Its purpose is to restart the game or simulation implemented by the class, resetting all necessary variables and objects to their initial state.

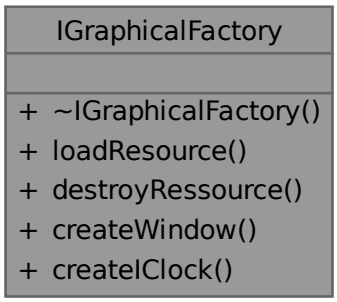
The documentation for this class was generated from the following file:

- Interface/Game/IGame.hpp

4.5 IGraphicalFactory Class Reference

```
#include <IGraphicalFactory.hpp>
```

Collaboration diagram for IGraphicalFactory:



Public Member Functions

- ~IGraphicalFactory ()=default
- virtual void loadResource ()=0

  Loads the necessary resources required for the implementation of the program. Its purpose is to load any required resources that are used in the implementation, such as textures, sounds, or fonts, into memory so they can be used later on.
- virtual void destroyRessource ()=0

  Releases the resources held by the implementation. Its purpose is to release any resources that have been allocated or loaded by the implementation, such as textures, sounds, or fonts that were loaded during the course of the program. It ensure that all resources are properly released to avoid any memory leaks or resource conflicts.
- virtual std::unique_ptr< IWindow > createWindow (std::string name, size_t width, size_t height)=0

  This function creates a window object required for the implementation.
- virtual std::unique_ptr< IClock > createIClock ()=0

  Creates an IClock object required for the implementation. Its purpose is to create a new IClock object, which is used to track time in the implementation.

4.5.1 Constructor & Destructor Documentation

4.5.1.1 ~IGraphicalFactory()

```
IGraphicalFactory::~IGraphicalFactory ( )  [default]
```

4.5.2 Member Function Documentation

4.5.2.1 createIClock()

```
virtual std::unique_ptr< IClock > IGraphicalFactory::createIClock ( )  [pure virtual]
```

Creates an IClock object required for the implementation. Its purpose is to create a new IClock object, which is used to track time in the implementation.

Parameters

width	
height	

Returns

std::unique_ptr<IClock> A unique_ptr<IClock> object pointing to the newly created IClock.

4.5.2.2 createWindow()

```
virtual std::unique_ptr< IWindow > IGraphicalFactory::createWindow (
            std::string name,
            size_t width,
            size_t height )  [pure virtual]
```

This function creates a window object required for the implementation.

Parameters

name	A string specifying the name of the window to be created.
width	A size_t specifying the width of the window to be created.
height	A size_t specifying the height of the window to be created.

Returns

std::unique_ptr<IWindow> A unique_ptr<IWindow> object pointing to the newly created window.

4.5.2.3 destroyRessource()

```
virtual void IGraphicalFactory::destroyRessource ( )  [pure virtual]
```

Releases the resources held by the implementation. Its purpose is to release any resources that have been allocated or loaded by the implementation, such as textures, sounds, or fonts that were loaded during the course of the program. It ensure that all resources are properly released to avoid any memory leaks or resource conflicts.

4.5.2.4 loadResource()

```
virtual void IGraphicalFactory::loadResource ( )  [pure virtual]
```

Loads the necessary resources required for the implementation of the program. Its purpose is to load any required resources that are used in the implementation, such as textures, sounds, or fonts, into memory so they can be used later on.

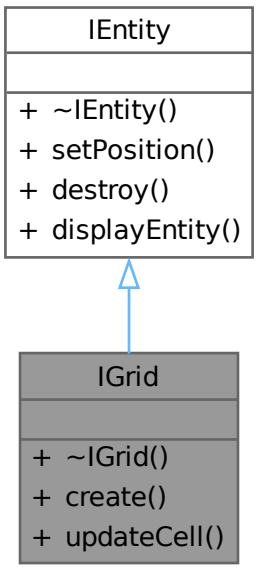
The documentation for this class was generated from the following file:

- Interface/Graphical/IGraphicalFactory.hpp

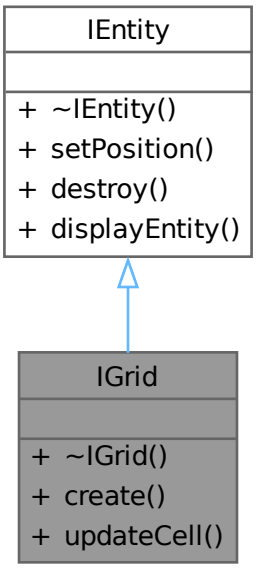
4.6 IGrid Class Reference

```
#include <IEntity.hpp>
```

Inheritance diagram for IGrid:



Collaboration diagram for IGrid:



Classes

- struct Cell

Public Member Functions

- ~IGrid ()=default
- virtual void create (int width, int heigth)=0

  Creates a new object in the implementation with the given width and height. Its purpose is to create a new object in the implementation with the given width and height. The function takes two integer parameters, representing the width and height of the new object to be created.
- virtual void updateCell (int x, int y, IEntity::Color)=0

  a pure virtual function that must be implemented by the derived class. Its purpose is to update the color of a cell in the implementation's grid, identified by its x and y coordinates. The function takes three parameters: two integer parameters representing the x and y coordinates of the cell, and a parameter of type IEntity::Color representing the new color to assign to the cell.

Public Member Functions inherited from IEntity

- ~IEntity ()=default
- virtual void setPosition (int x, int y)=0

  Sets the position of the entity. Its purpose is to set the position of the entity to the specified x and y coordinates. The function takes two integer parameters representing the x and y coordinates to set as the new position of the entity.
- virtual void destroy ()=0

  Destroys any resources used in the implementation. Its purpose is to destroy any resources used in the implementation, such as windows or textures.
- virtual void displayEntity ()=0

  Displays an entity on the screen. Its purpose is to display an entity on the screen, such as a sprite or image.

Additional Inherited Members

Public Types inherited from IEntity

- enum class Color {
  Blue , Green , Orange , Red ,
  Brown , Yellow }

4.6.1 Constructor & Destructor Documentation

4.6.1.1 ~IGrid()

```
IGrid::~IGrid ( )  [default]
```

4.6.2 Member Function Documentation

4.6.2.1 create()

```
virtual void IGrid::create (
            int width,
            int heigth )  [pure virtual]
```

Creates a new object in the implementation with the given width and height. Its purpose is to create a new object in the implementation with the given width and height. The function takes two integer parameters, representing the width and height of the new object to be created.

Parameters

width	
heigth	

4.6.2.2 updateCell()

```
virtual void IGrid::updateCell (
            int x,
            int y,
            IEntity::Color  )  [pure virtual]
```

a pure virtual function that must be implemented by the derived class. Its purpose is to update the color of a cell in the implementation's grid, identified by its x and y coordinates. The function takes three parameters: two integer parameters representing the x and y coordinates of the cell, and a parameter of type IEntity::Color representing the new color to assign to the cell.

Parameters

x	
y	

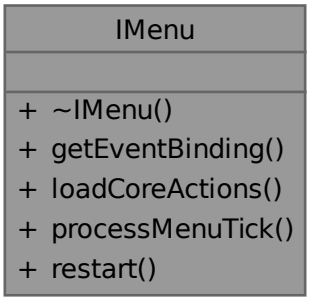
The documentation for this class was generated from the following file:

- Interface/Graphical/IEntity.hpp

4.7 IMenu Class Reference

`#include <IMenu.hpp>`

Collaboration diagram for IMenu:



Public Types

- using LambdaCoreLoading = std::function< void(std::string, std::string)>

Public Member Functions

- ~IMenu ()=default
- virtual IWindow::EventHandler & getEventBinding ()=0

  Retrieves the event binding object for the window. This function returns a reference to an object of type IWindow::EventHandler, which is used to handle events for the window.
- virtual void loadCoreActions (LambdaCoreLoading)=0

  Loads the core actions required for the implementation.
- virtual bool processMenuTick (IText &, IText &, std::array< std::reference_wrapper< IText >, 3 >, std::array< std::reference_wrapper< IText >, 3 >)=0

  Processes a tick in the game menu.
- virtual void restart ()=0

  Restarts the game.

4.7.1 Member Typedef Documentation

4.7.1.1 LambdaCoreLoading

```
using IMenu::LambdaCoreLoading = std::function<void(std::string, std::string)>
```

4.7.2 Constructor & Destructor Documentation

4.7.2.1 ~IMenu()

```
IMenu::~IMenu ( )  [default]
```

4.7.3 Member Function Documentation

4.7.3.1 getEventBinding()

```
virtual IWindow::EventHandler & IMenu::getEventBinding ( )  [pure virtual]
```

Retrieves the event binding object for the window. This function returns a reference to an object of type IWindow::EventHandler, which is used to handle events for the window.

Returns

IWindow::EventHandler& A reference to an object of type IWindow::EventHandler.

4.7.3.2 loadCoreActions()

```
virtual void IMenu::loadCoreActions (
            LambdaCoreLoading  )  [pure virtual]
```

Loads the core actions required for the implementation.

This function is used to load the core actions required for the implementation using a LambdaCoreLoading parameter.

Parameters

LambdaCoreLoading	A parameter of type LambdaCoreLoading that is used to load the core actions.

4.7.3.3 processMenuTick()

```
virtual bool IMenu::processMenuTick (
            IText & ,
            IText & ,
            std::array< std::reference_wrapper< IText >, 3 > ,
            std::array< std::reference_wrapper< IText >, 3 >  )  [pure virtual]
```

Processes a tick in the game menu.

This function is used to process a tick in the game menu, using several parameters of type IText and an array of references to IText objects. It returns a boolean value indicating whether or not the menu should continue to be processed.

Parameters

IText	A parameter of type IText that is used to process the game menu.
IText	A parameter of type IText that is used to process the game menu.
std::array<std::reference_wrapper<IText>,3>	An array of references to IText objects.
std::array<std::reference_wrapper<IText>,3>	An array of references to IText objects.

Returns

true The menu should continue to be processed.

false The menu should not continue to be processed.

4.7.3.4 restart()

```
virtual void IMenu::restart ( )  [pure virtual]
```

Restarts the game.

This function is used to restart the game. It is declared as a pure virtual function, so it must be implemented by any derived classes.

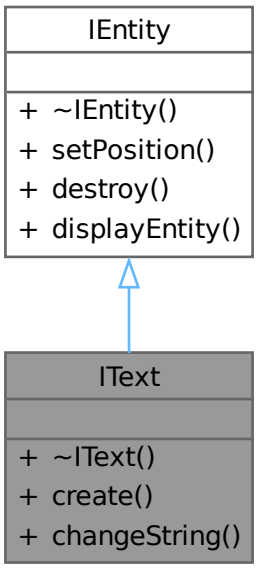
The documentation for this class was generated from the following file:

- Interface/Menu/IMenu.hpp

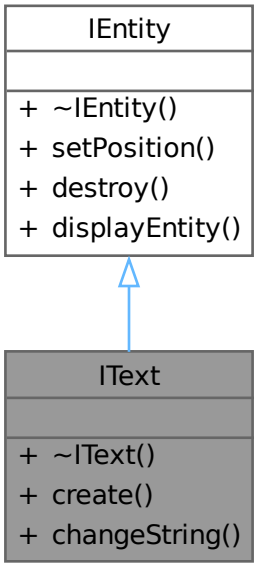
4.8 IText Class Reference

`#include <IEntity.hpp>`

Inheritance diagram for IText:



Collaboration diagram for IText:



Public Types

- enum class EntityType { GridEntity , SpriteEntity , TextEntity }

Public Types inherited from IEntity

- enum class Color {
  Blue , Green , Orange , Red ,
  Brown , Yellow }

Public Member Functions

- ~IText ()=default
- virtual void create (std::string str)=0

  Creates a new object in the implementation. Its purpose is to create a new object in the implementation based on the given string parameter. The function take a string as parameter, which can be used to determine the type or properties of the object to be created.
- virtual void changeString (std::string str)=0

  Changes the string property of the object in the implementation. Its purpose is to change the string property of the object in the implementation. The function takes a string parameter, which represents the new value for the string property of the object.

Public Member Functions inherited from IEntity

- ~IEntity ()=default
- virtual void setPosition (int x, int y)=0

  Sets the position of the entity. Its purpose is to set the position of the entity to the specified x and y coordinates. The function takes two integer parameters representing the x and y coordinates to set as the new position of the entity.
- virtual void destroy ()=0

  Destroys any resources used in the implementation. Its purpose is to destroy any resources used in the implementation, such as windows or textures.
- virtual void displayEntity ()=0

  Displays an entity on the screen. Its purpose is to display an entity on the screen, such as a sprite or image.

4.8.1 Member Enumeration Documentation

4.8.1.1 EntityType

```
enum class IText::EntityType  [strong]
```

Enumerator

GridEntity	
SpriteEntity	
TextEntity	

4.8.2 Constructor & Destructor Documentation

4.8.2.1 ~IText()

```
IText::~IText ( )  [default]
```

4.8.3 Member Function Documentation

4.8.3.1 changeString()

```
virtual void IText::changeString (
            std::string str )  [pure virtual]
```

Changes the string property of the object in the implementation. Its purpose is to change the string property of the object in the implementation. The function takes a string parameter, which represents the new value for the string property of the object.

Parameters

str	

4.8.3.2 create()

```
virtual void IText::create (
            std::string str )  [pure virtual]
```

Creates a new object in the implementation. Its purpose is to create a new object in the implementation based on the given string parameter. The function take a string as parameter, which can be used to determine the type or properties of the object to be created.

Parameters

str	

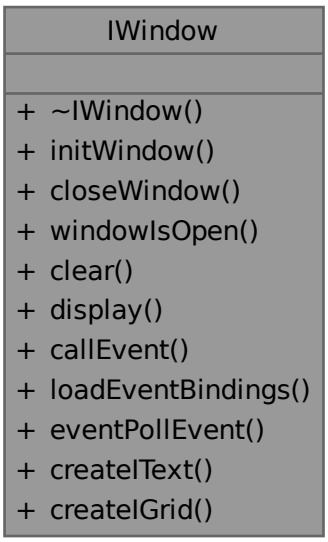
The documentation for this class was generated from the following file:

- Interface/Graphical/IEntity.hpp

4.9 IWindow Class Reference

```
#include <IWindow.hpp>
```

Collaboration diagram for IWindow:



Public Types

- enum class EventType {
  UP_pressed , DOWN_pressed , LEFT_pressed , RIGHT_pressed ,
  ENTER_pressed , QUIT , PAUSE , RESUME ,
  NEXT_GAME , NEXT_LIB , RESTART , GO_TO_MENU }
- using EventCallBack = std::function< void()>
- using EventHandler = std::map< IWindow::EventType, EventCallBack >

Public Member Functions

- ~IWindow ()=default
- virtual void initWindow (std::string name, size_t width, size_t height)=0

  Initializes the window object required for the implementation. Its purpose is to initialize the window object with the specified name, width, and height.
- virtual void closeWindow ()=0

  loses the window object used in the implementation. Its purpose is to close and destroy the window object used in the implementation.
- virtual bool windowIsOpen ()=0

  Checks whether the window object used in the implementation is currently open. Its purpose is to check whether the window object used in the implementation is currently open.
- virtual void clear ()=0

  Clears the rendering buffer used in the implementation. Its purpose is to clear the rendering buffer used in the implementation. This is typically done at the beginning of each frame to ensure that the previous frame's rendering is not visible in the current frame.
- virtual void display ()=0

  Displays the rendered content on the window object used in the implementation. Its purpose is to display the rendered content on the window object used in the implementation. This is typically done at the end of each frame to show the newly rendered content on the screen.
- virtual void callEvent (const IWindow::EventType)=0

  Calls the event handler function for the specified event type. Its purpose is to call the event handler function for the specified event type. This function is typically called by the application's main loop to handle user input or other events.
- virtual void loadEventBindings (EventHandler &)=0

  oads the event bindings for the specified event handler object. Its purpose is to load the event bindings for the specified event handler object. This function is typically called at the beginning of the application to register event handlers for various types of events (e.g. mouse clicks, key presses, etc.).
- virtual void eventPollEvent ()=0

  Polls for any pending events on the window object used in the implementation. Its purpose is to poll for any pending events on the window object used in the implementation. This function is typically called by the application's main loop to handle user input or other events.
- virtual std::unique_ptr< IText > createIText ()=0

  Creates a new instance of the IText interface for the implementation. Its purpose is to create a new instance of the IText interface for the implementation. The returned object can be used to draw text on the window.
- virtual std::unique_ptr< IGrid > createIGrid ()=0

  Creates a new instance of the IGrid interface for the implementation. Its purpose is to create a new instance of the IGrid interface for the implementation. The returned object can be used to draw a grid on the window.

4.9.1 Member Typedef Documentation

4.9.1.1 EventCallBack

```
using IWindow::EventCallBack = std::function<void()>
```

4.9.1.2 EventHandler

```
using IWindow::EventHandler = std::map<IWindow::EventType, EventCallBack>
```

4.9.2 Member Enumeration Documentation

4.9.2.1 EventType

```
enum class IWindow::EventType  [strong]
```

Enumerator

UP_pressed	
DOWN_pressed	
LEFT_pressed	
RIGHT_pressed	
ENTER_pressed	
QUIT	
PAUSE	
RESUME	
NEXT_GAME	
NEXT_LIB	
RESTART	
GO_TO_MENU	

4.9.3 Constructor & Destructor Documentation

4.9.3.1 ~IWindow()

```
IWindow::~IWindow ( )  [default]
```

4.9.4 Member Function Documentation

4.9.4.1 callEvent()

```
virtual void IWindow::callEvent (
            const IWindow::EventType  )  [pure virtual]
```

Calls the event handler function for the specified event type. Its purpose is to call the event handler function for the specified event type. This function is typically called by the application's main loop to handle user input or other events.

Parameters

eventType	An object of type IWindow::EventType specifying the type of the event to be handled.

4.9.4.2 clear()

```
virtual void IWindow::clear ( )  [pure virtual]
```

Clears the rendering buffer used in the implementation. Its purpose is to clear the rendering buffer used in the implementation. This is typically done at the beginning of each frame to ensure that the previous frame's rendering is not visible in the current frame.

4.9.4.3 closeWindow()

```
virtual void IWindow::closeWindow ( )  [pure virtual]
```

loses the window object used in the implementation. Its purpose is to close and destroy the window object used in the implementation.

4.9.4.4 createIGrid()

```
virtual std::unique_ptr< IGrid > IWindow::createIGrid ( )  [pure virtual]
```

Creates a new instance of the IGrid interface for the implementation. Its purpose is to create a new instance of the IGrid interface for the implementation. The returned object can be used to draw a grid on the window.

Returns

std::unique_ptr<IGrid>

4.9.4.5 createIText()

```
virtual std::unique_ptr< IText > IWindow::createIText ( )  [pure virtual]
```

Creates a new instance of the IText interface for the implementation. Its purpose is to create a new instance of the IText interface for the implementation. The returned object can be used to draw text on the window.

Returns

std::unique_ptr<IText>

4.9.4.6 display()

```
virtual void IWindow::display ( )  [pure virtual]
```

Displays the rendered content on the window object used in the implementation. Its purpose is to display the rendered content on the window object used in the implementation. This is typically done at the end of each frame to show the newly rendered content on the screen.

4.9.4.7 eventPollEvent()

```
virtual void IWindow::eventPollEvent ( )  [pure virtual]
```

Polls for any pending events on the window object used in the implementation. Its purpose is to poll for any pending events on the window object used in the implementation. This function is typically called by the application's main loop to handle user input or other events.

4.9.4.8 initWindow()

```
virtual void IWindow::initWindow (
            std::string name,
            size_t width,
            size_t height )  [pure virtual]
```

Initializes the window object required for the implementation. Its purpose is to initialize the window object with the specified name, width, and height.

Parameters

name	
width	
height	

4.9.4.9 loadEventBindings()

```
virtual void IWindow::loadEventBindings (
            EventHandler &  )  [pure virtual]
```

oads the event bindings for the specified event handler object. Its purpose is to load the event bindings for the specified event handler object. This function is typically called at the beginning of the application to register event handlers for various types of events (e.g. mouse clicks, key presses, etc.).

Parameters

handler	An object of type EventHandler that contains the event bindings to be loaded.

4.9.4.10 windowIsOpen()

```
virtual bool IWindow::windowIsOpen ( )  [pure virtual]
```

Checks whether the window object used in the implementation is currently open. Its purpose is to check whether the window object used in the implementation is currently open.

Returns

true

false

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

- Interface/Graphical/IWindow.hpp

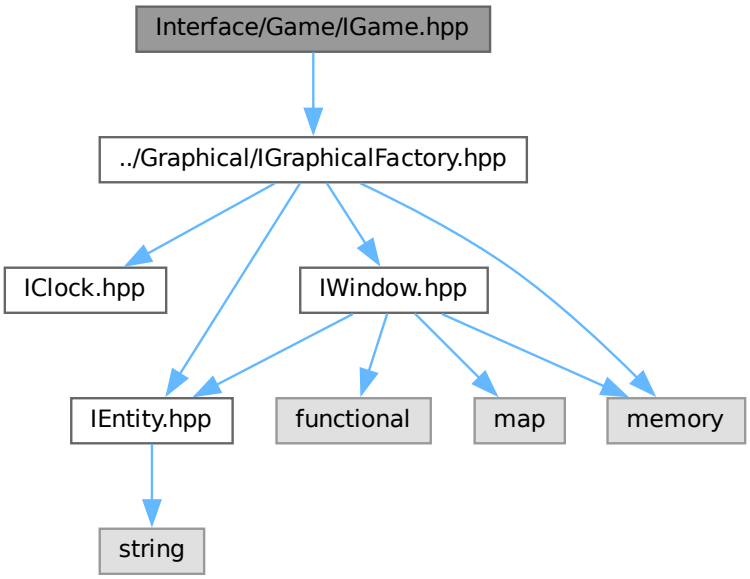
Chapter 5

File Documentation

5.1 Interface/Game/IGame.hpp File Reference

```
#include "../Graphical/IGraphicalFactory.hpp"
```

Include dependency graph for IGame.hpp:



Classes

- class IGame

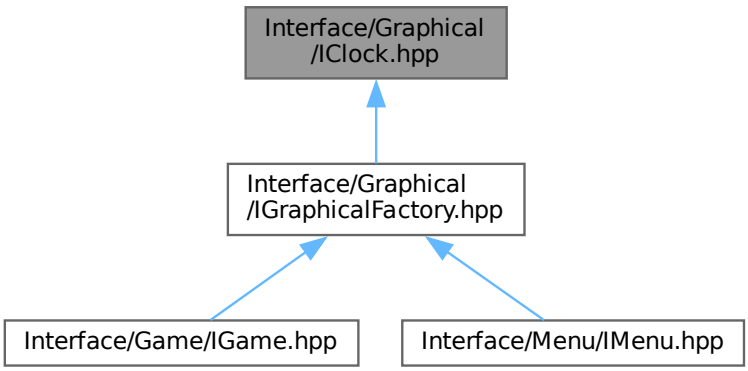
5.2 IGame.hpp

Go to the documentation of this file.

```
00001 /*
00002 ** EPITECH PROJECT, 2023
00003 ** B-OOP-400-BDX-4-1-arcade-leopold.sallan-gemard
00004 ** File description:
00005 ** IGame
00006 */
00007
00008 #pragma once
00009 #include "../Graphical/IGraphicalFactory.hpp"
00010
00011 class IGame
00012 {
00013     public:
00014         ~IGame() = default;
00015
00024         virtual IWindow::EventHandler &getEventBinding() = 0;
00025
00043         virtual bool processGameTick(IGrid &grid, IText &scoreText, IText &timeText,
00044                                      IClock &clock) = 0;
00045
00051         virtual void restart() = 0;
00052 };
```

5.3 Interface/Graphical/IClock.hpp File Reference

This graph shows which files directly or indirectly include this file:



Classes

- class IClock

5.4 IClock.hpp

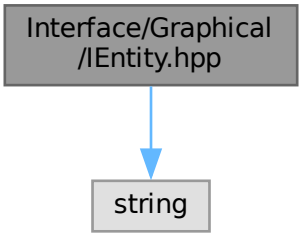
Go to the documentation of this file.

```
00001 /*
00002 ** EPITECH PROJECT, 2023
00003 ** arcade
```

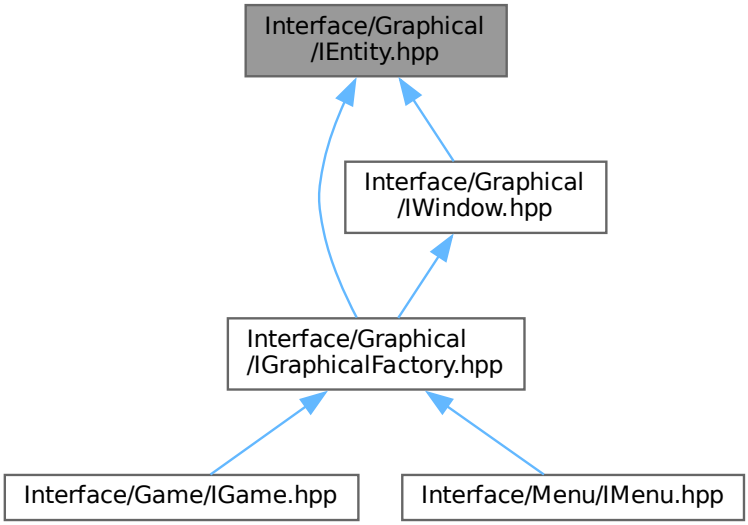
```
00004 ** File description:
00005 ** IClock
00006 */
00007
00008 #pragma once
00009
00010 class IClock
00011 {
00012     public:
00013         ~IClock() = default;
00014
00020         virtual void startClock() = 0;
00021
00028         virtual double getTimeElapsed() = 0;
00029
00035         virtual void resetClock() = 0;
00036
00042         virtual void initClock() = 0;
00043 };
```

5.5 Interface/Graphical/IEntity.hpp File Reference

`#include <string>`
Include dependency graph for IEntity.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class IEntity
- class IText
- class IGrid
- struct IGrid::Cell

5.6 IEntity.hpp

Go to the documentation of this file.

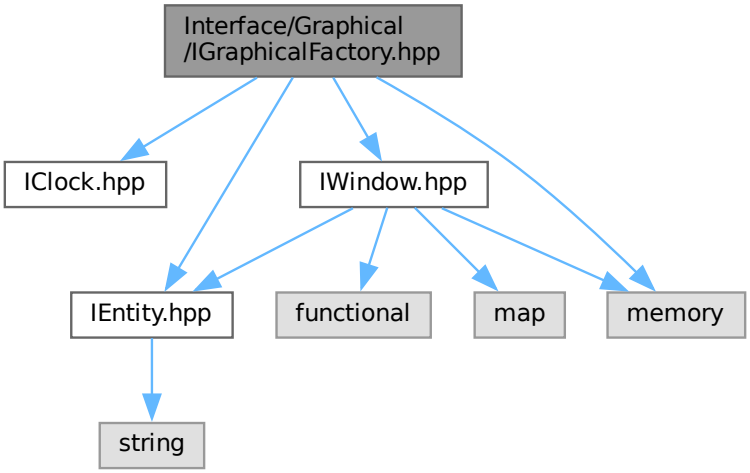
```
00001 /*
00002 ** EPITECH PROJECT, 2023
00003 ** arcade
00004 ** File description:
00005 ** IEntity
00006 */
00007
00008 #pragma once
00009 #include <string>
00010
00011 class IEntity
00012 {
00013     public:
00014         enum class Color {
00015             Blue,
00016             Green,
00017             Orange,
00018             Red,
00019             Brown,
00020             Yellow,
00021         };
00022
00023         ~IEntity() = default;
00024
00033         virtual void setPosition(int x, int y) = 0;
00034
00040         virtual void destroy() = 0;
```

```
00041
00046         virtual void displayEntity() = 0;
00047 };
00048
00049 class IText : public IEntity
00050 {
00051     public:
00052         enum class EntityType {
00053             GridEntity,
00054             SpriteEntity,
00055             TextEntity,
00056         };
00057         ~IText() = default;
00058
00067         virtual void create(std::string str) = 0;
00068
00076         virtual void changeString(std::string str) = 0;
00077 };
00078
00079 class IGrid : public IEntity
00080 {
00081     public:
00082         struct Cell {
00083                 Color color;
00084                 size_t x;
00085                 size_t y;
00086         };
00087         ~IGrid() = default;
00088
00098         virtual void create(int width, int heigth) = 0;
00099
00110         virtual void updateCell(int x, int y, IEntity::Color) = 0;
00111 };
```

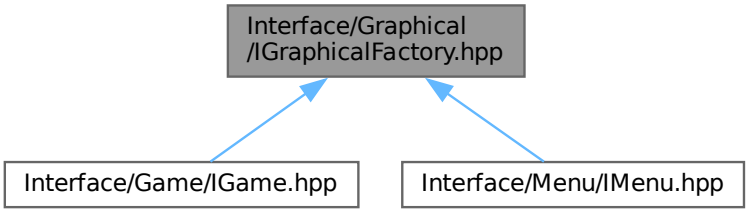
5.7 Interface/Graphical/IGraphicalFactory.hpp File Reference

```
#include "IClock.hpp"
#include "IEntity.hpp"
#include "IWindow.hpp"
#include <memory>
```

Include dependency graph for IGraphicalFactory.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class IGraphicalFactory

5.8 IGraphicalFactory.hpp

Go to the documentation of this file.

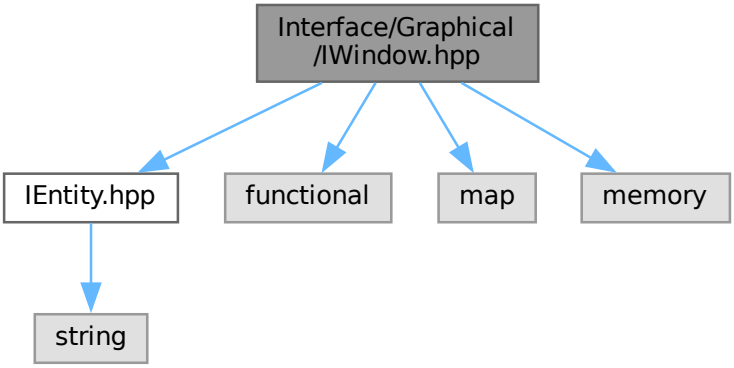
```
00001 /*
00002 ** EPITECH PROJECT, 2023
00003 ** B-OOP-400-BDX-4-1-arcade-leopold.sallan-gemard
00004 ** File description:
00005 ** IGraphicalFactorys
00006 */
00007
00008 #pragma once
00009 #include "IClock.hpp"
00010 #include "IEntity.hpp"
00011 #include "IWindow.hpp"
00012 #include <memory>
00013
00014 class IGraphicalFactory
00015 {
00016     public:
00017         ~IGraphicalFactory() = default;
00018
00024         virtual void loadResource() = 0;
00025
00034         virtual void destroyRessource() = 0;
00035
00044         virtual std::unique_ptr<IWindow> createWindow(std::string name, size_t width,
00045                                                       size_t height) = 0;
00046
00056         virtual std::unique_ptr<IClock> createIClock() = 0;
00057 };
```

5.9 Interface/Graphical/IWindow.hpp File Reference

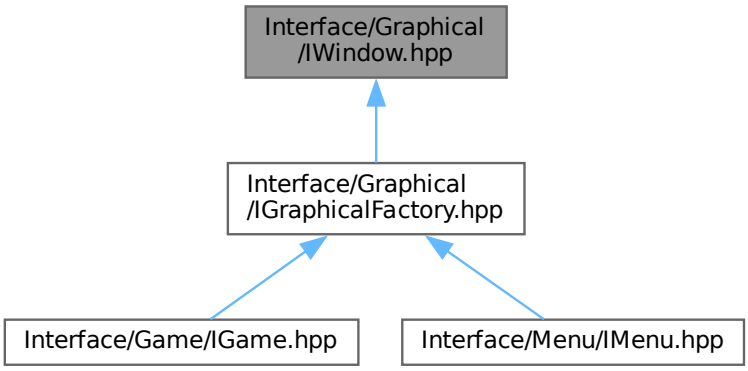
```
#include "IEntity.hpp"
#include <functional>
#include <map>
```

```
#include <memory>
```

Include dependency graph for IWindow.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class IWindow

5.10 IWindow.hpp

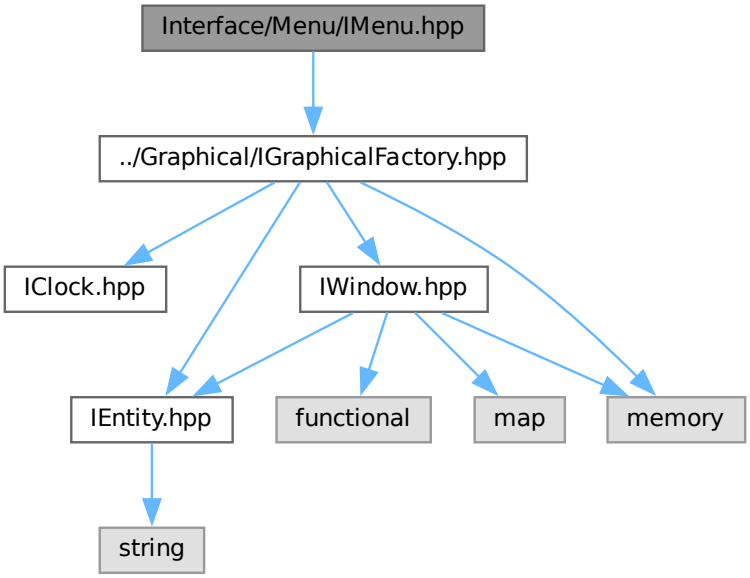
Go to the documentation of this file.

```
00001 /*
00002 ** EPITECH PROJECT, 2023
00003 ** arcade
00004 ** File description:
```

```
00005 ** IWindow
00006 */
00007
00008 #pragma once
00009 #include "IEntity.hpp"
00010 #include <functional>
00011 #include <map>
00012 #include <memory>
00013
00014 class IWindow
00015 {
00016     public:
00017         ~IWindow() = default;
00018
00019         //event
00020         enum class EventType {
00021             UP_pressed,
00022             DOWN_pressed,
00023             LEFT_pressed,
00024             RIGHT_pressed,
00025             ENTER_pressed,
00026             QUIT,
00027             PAUSE,
00028             RESUME,
00029             NEXT_GAME,
00030             NEXT_LIB,
00031             RESTART,
00032             GO_TO_MENU,
00033         };
00034         using EventCallBack = std::function<void()>;
00035         using EventHandler = std::map<IWindow::EventType, EventCallBack>;
00036
00045         virtual void initWindow(std::string name, size_t width, size_t height) = 0;
00046
00051         virtual void closeWindow() = 0;
00052
00060         virtual bool windowIsOpen() = 0;
00061
00068         virtual void clear() = 0;
00069
00076         virtual void display() = 0;
00077
00086         virtual void callEvent(const IWindow::EventType) = 0;
00087
00096         virtual void loadEventBindings(EventHandler &) = 0;
00097
00104         virtual void eventPollEvent() = 0;
00105
00112         virtual std::unique_ptr<IText> createIText() = 0;
00113
00120         virtual std::unique_ptr<IGrid> createIGrid() = 0;
00121 };
```

5.11 Interface/Menu/IMenu.hpp File Reference

`#include "../Graphical/IGraphicalFactory.hpp"`
Include dependency graph for IMenu.hpp:



Classes

- class IMenu

5.12 IMenu.hpp

Go to the documentation of this file.

```
00001 /*
00002 ** EPITECH PROJECT, 2023
00003 ** B-OOP-400-BDX-4-1-arcade-leopold.sallan-gemard
00004 ** File description:
00005 ** IMenu
00006 */
00007
00008 #pragma once
00009 #include "../Graphical/IGraphicalFactory.hpp"
00010
00015 class IMenu
00016 {
00017
00018     public:
00019         using LambdaCoreLoading = std::function<void(std::string, std::string)>;
00020
00021         ~IMenu() = default;
00022
00029         virtual IWindow::EventHandler &getEventBinding() = 0;
00030
00040         virtual void loadCoreActions(LambdaCoreLoading) = 0;
00041
00058         virtual bool processMenuTick(IText &, IText &, std::array<std::reference_wrapper<IText>, 3>,
00059                                      std::array<std::reference_wrapper<IText>, 3>) = 0;
00060
00067         virtual void restart() = 0;
00068 };
```