Documentation Arcade

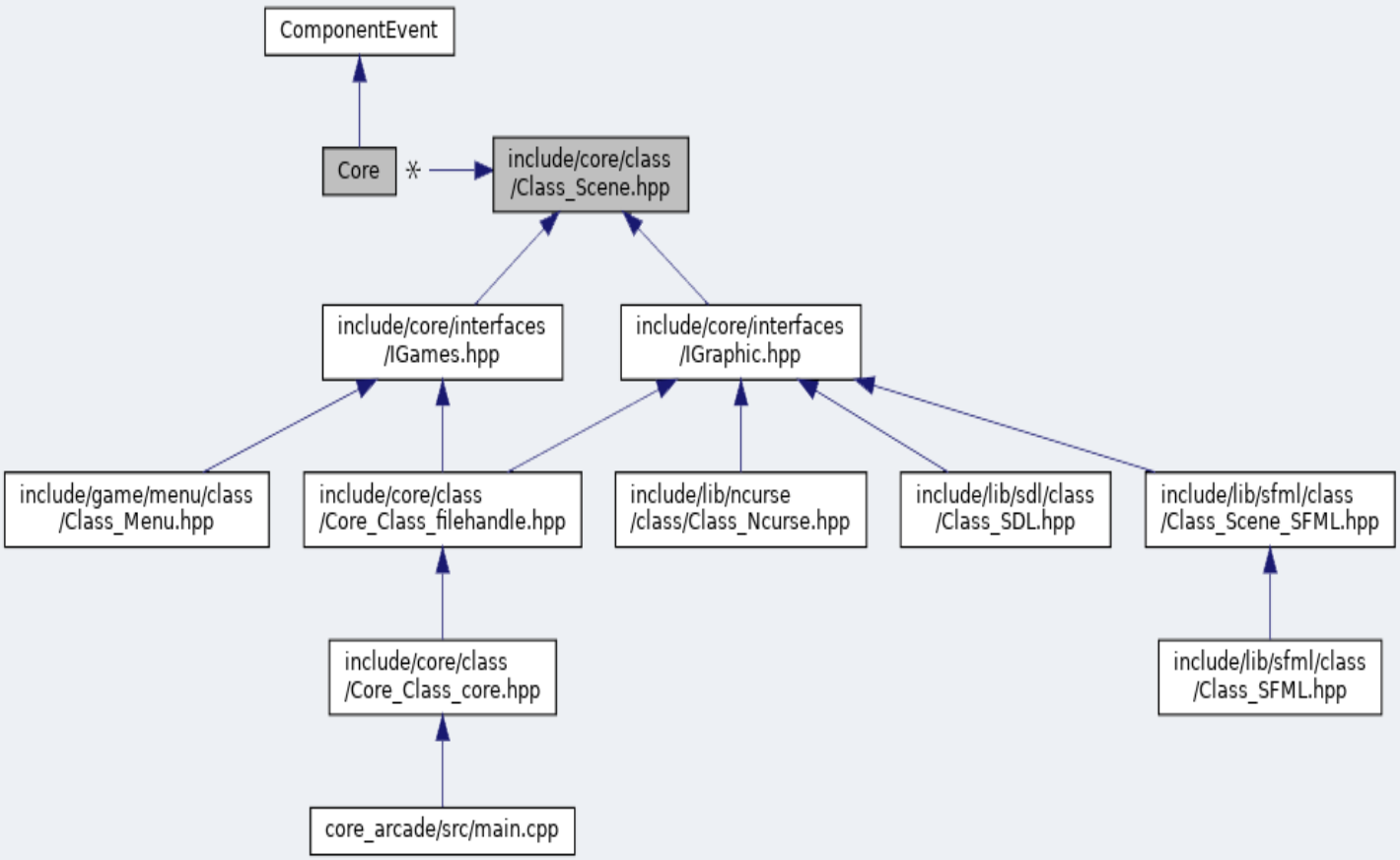
C++ (Oriented Object)
2nd year

Nicolas Da-silva
Clément Bérard

The Arcade's Interfaces project is a side-project to achieve the Epitech's 2nd year C++ project "Arcade" in a very modular way.

Description

The Arcade project aim to create an "old school arcade" by creating a Core program (the arcade), and loading Games and Graphic lib dynamically, and using theses in an abstracted way.
To achieve this, the Game and Graphic lib must never communicate directly, the communication must always be between the **Game ⇄ Core** and the **Core ⇄ Lib**.
Here is a schema to illustrate that:



How to run the program ?

Compilation of files:

Compilation is done with the Makefile at the base of the directory. It calls the other Makefiles in the "sub-folders", in particular to create the binaries for each lib.

→ “**make**”, in the directory

→ “**make re**”, if there were any modifications made upstream

Binary launch:

./arcade [lib]

Example : - ./arcade ./lib/lib_sdl2.so
- ./arcade ./lib/lib_sfml.so
- ./arcade ./lib/lib_ncurses.so

Interactions in the menu:

In the menu there are 6 possible interactions. In fact, in the menu you can then :

→ Choose a graphics library

→ Choose a game

→ Watch the “How to play” to find out how the games work

→ Start the game (after selecting a game)

→ Press “ESC” to return to the menu

→ Exit the Arcade

The Core

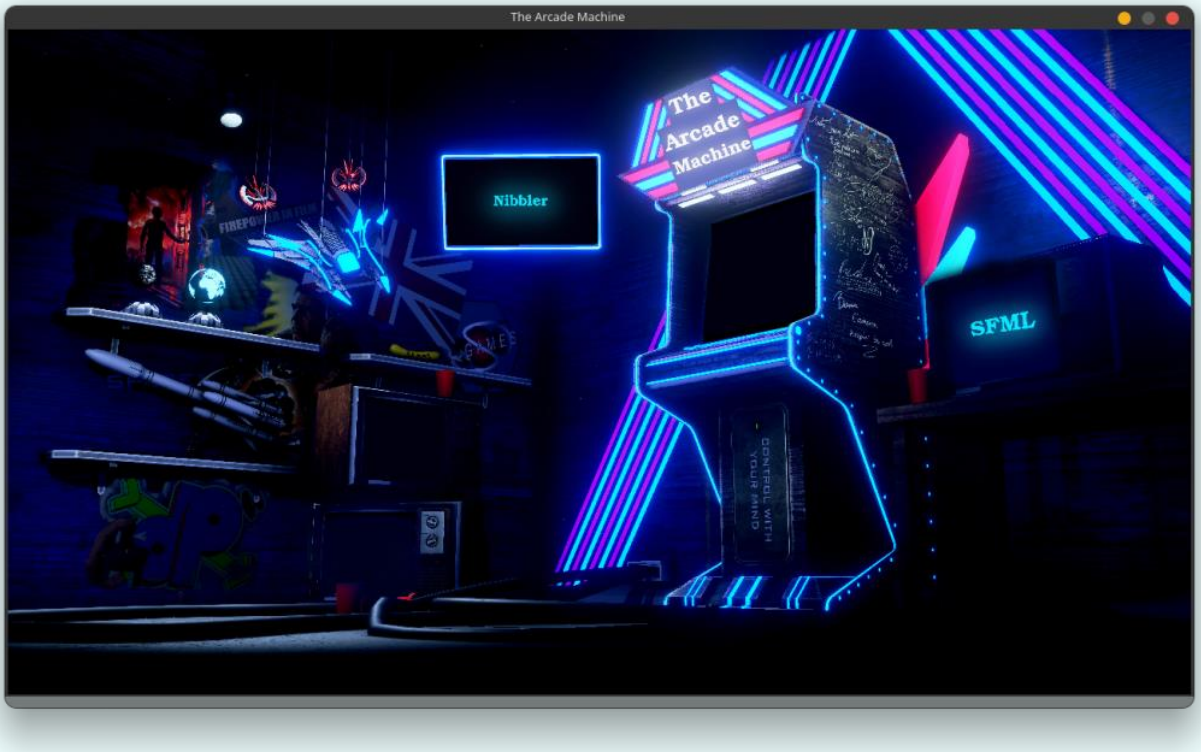
- **Explanations:**

→ All the files that make up the **Core** of our program are in "**/core_arcade**".

→ The **Core** is the interface which will make the link between the graphic libraries located in the "**/lib**" folder and the games located in the "**/games**" folder.

→ However, you can implement whatever libraries and games you want yourself ! To do this, follow the different steps available to you :

- **Adding a graphics library**
- **Adding a game**



Adding a graphics library

Choose a library to implement:

→ Install a graphics library (ex: Libcaca…)

→ Create a folder for the new library

→ To be usable, the name of your library binary must be in the form of : "lib_arcade_" + **name of your library** + ".so"

Implement a new graphic library:

- The first step is that your class must implement the **IGraphic** interface.
- The Interface contains 11 functions that you must implement in order to get your graphic library working with this project.
- You can find the interface file in "./include/core/interfaces/IGraphics.hpp" when you are at the root of the project directory.

Here are the 13 methods and their usage :

→ **~IGraphic** : will erase, delete, destroy... All the textures you have to set, the window that you created, the fonts… Whatever you have loaded !

→ **getEvent** : is one of the most important method in all the libs. It retrieves all the events created by the user. That method is there to catch all the keyboard inputs we want.

→ **getMousePosition** : get the position of the mouse on the screen.

→ **initializeWindow** : initializes the window with its size and name.

→ **windowIsOpen** : check if the window is open.

→ **closeWindow** : close the window.

→ **setTitle** : define a new name for the window.

→ **display** : is obviously the function that allows you to display, indeed you will call it to display all of the assets that you have set.

→ **getNameGraphic** : get the name of a graphics library (SFML, SDL, NCURSES …).

→ **inTerminal** : determine if this class render in a window or in a terminal.

→ **initScene** : initializes all the elements of the game in the chosen graphics library.

→ **updateScene** : all the elements of the coreScene will be updated in the chosen graphic library.

→ **clearGraphicalScene** : is the method to clear the screen. That is very useful when you have to draw because it clears all the screen.

Adding a game

Choose a game to implement:

→ Create a folder for the new game

→ Create a folder in “include” linked to the name of your game

→ To be usable, the name of your library binary must be in the form of: “lib_arcade_” + **name of your game** + “.so”

Implement a new game:

- The first step is that your class must implement the **IGames** interface.
- The Interface contains 4 functions that you must implement in order to get your game working with this project.
- You can find the interface file in “./include/core/interfaces/IGames.hpp” when you are at the root of the project directory.

Here are the 4 methods and their usage :

→ **~IGames** : will erase, delete, destroy... Whatever you have loaded!

→ **getNameGame** : get the name of the chosen game (Nibble, Pacman …).

→ **initScene** : initializes all the elements of the game.

→ **updateScene** : all the elements will be updated in the chosen game.

Obviously, you can implement many more functions/methods if you want ! But be careful they must not change the operation program. These methods must work just for your libs, and we don't have to modify the **IGraphics** and **IGames** interfaces to make it work.

Don't forget to modify the Makefiles with your additions !

Good Luck !