Core

The `Core` part of the project is the gate between our graphic libraries and our game libraries.

This class is initialized inside the `main` function . The constructor of the `Core` class will parse the lib folder and the user arguments and create the `Core` class.

Role of the Core class

Parse the libraries

The main functions of the `Core` is to parse all the libraries and have methods to get informations about them:

```
void Core::getAllLib();
void Core::pushLib(std::string path, std::vector<std::string>& container);
```

Load the libraries

The game and graphic libraries are dynamic, so that they can be changed during runtime. The `Core` class uses a `DLLoader` so that it can open, close and load one graphic and one game library during the program's execution.

Then there are all the methods used to change dynamically the libraries:

```
void Core::loadNextGame();
void Core::loadNextGraph();
void Core::loadSpecificGame(std::string path);
void Core::loadSpecificGraph(std::string path);
```

Handle errors

If a library is not loaded correctly, of if during the execution of the games there is an issue, the `Core` will throw a custom exception called `ArcadeError` inheriting from the `std::exception` class. It will then stop the execution of the program in order to ensure the program's safety.

Handle events during execution

The `Core` handles the state of the program. This state depends on the user's input. At the launch of the program, the user will be on the menu, where he can choose his name, game and graphical library.

According to the user's actions, the `Core` will handle all events related to the program's execution. All events are triggered by key bindings in the program. You can check the list of keybindings availables in the `README` file at the root of the repository

Handle the program loop

As the 'gate' of our program, the Core has the responsibility to handle the execution of the program. The method `coreStateHandler()` handles the different scenes as well as gets the events and key pressed by the user. The methods

```
void Menu::menuLoopHandler(IGraphic& graphLib, Core& core);
void Core::gameLoopHandler();
```

handle the logic and display of the scenes, as well as the `menu` and `game` loops.