Arcade

- Antoine-Zachary Khalidy (Project Manager, Core and Graphics Libraries)
- Yoan Gerrard (Creator of Pacman)
- Louis Lejaille (Creator of Snake)

Summary

- Interfaces
- Makefile
- Game, Graphics, and Audio Implementations

Interfaces

There are multiple interfaces in the Arcade project:

- IAudio
- IGraphical
- IGame
- IData
- IEntity

Each one is specific to a part of the project and is mostly independent.

IAudio: As the name suggests, IAudio is the audio part of the games. It allows adding music to make the game more lively.

IGraphical: IGraphical takes care of the graphical part of the games and allows obtaining a visual rendering, whether it be in Terminal mode, 2D or even 3D.

IGame: IGame takes care of the main part of the game, everything related to user inputs management must be done in it and must fill IData so that IGraphical displays what has been done in IGame.

IData: IData contains the main data of the game, whether it be the score, the number of lives, the level, or the map. All this data must be provided to IData so that the core can use it later on.

IEntity: IEntity is each part of the IData map. For example, a wall is an entity, meaning that it can have a color, a sprite, a position, etc. IEntity allows having a flexible map, whether it be in 2D, 3D or in Terminal mode.

Makefile

The Makefile is quite simple to use despite what one might think:

- At the very top, there are the names of the shared libraries, in our case arcade_[lib].so.
- Below that, there is the location of the .cpp and .hpp source files required for the use of the library.
- Then, there are the OBJ, which are the moments when they are transformed so that our PC can read them.
- Now, the flags. Most libraries require flags to be used, this is where you should put your flags.
- Finally, all the rules are below.

Implementations

The implementations in the Arcade project are divided into three main parts: games, graphical libraries, and audio libraries.

1. To implement a game, you need to create a new folder in the "Game" directory and name it as you wish. Then, create an hpp and a cpp file. In the hpp file, create a class that inherits from the IGame interface and implement the associated functions. In the cpp file, implement the logical part of the game using the vector of integers for user inputs, IData for data to display, and time to manage movements or animations. Finally, in the Makefile, add the necessary information for it to work, including the name, path, and flags. Then, copy-paste the lines into the "games" rules and modify them accordingly.

2. For graphical libraries, create a new folder in the "Graphical" directory and name it as you wish. Then, create a .hpp and a

.cpp file. In the .hpp file, create a class that inherits from the IGraphical interface and implement the associated functions. Then, fill the getInputs function so that the core receives a value every time the user presses a key. Finally, create a double loop using IData as a source to display each entity according to our needs (characters for a terminal-based graphical library, sprites for a 2D or 3D graphical library). In the Makefile, add the necessary information for it to work, including the name, path, and flags. Then, copy-paste the lines into the "graphicals" rules and modify them accordingly.

3. For audio libraries, follow the same process as for graphical libraries. Create a new folder in the "Audio" directory and name it as you wish. Then, create an hpp and a cpp file. In the cpp file, generate our music from the path and play it. In the Makefile, add the necessary information for it to work, including the name, path, and flags. Then, copy-paste the lines into the "audios" rules and modify them accordingly.

This is how to implement libraries in my Arcade project.