ARCADE

Lucas HAUSZLER, Bastien GERARD & Thibault GUYONY

What is the ARCADE project ?

Arcade is a gaming plateform like an arcade terminal. The goal of this project is to implement with the C++ language, two arcade type games that can be played with several different graphic libraries. The user has the possibility to choose the game and the library loaded dynamically. This will allow him to change the library at any time during the game. This will be done by clicking on a definite button on the keyboard. Obviously when the player will change libraries he will return exactly the position of the game with the same score only with a different display.

One of the particularities of the project :

Indeed although this project is already very interesting as it is, another challenge is part of this project to make it more complex. This project was to be done in groups of three. But since the games and the graphic libraries are loaded in a dynamic way. It is imposed for the project to work with another group so that each group can launch the ARCADE project with the libraries of the other group.

ARCADE

Our games :

• **Snake :** (arcade_snake.so) Snake is an arcade game first released during the mid 1970s and has maintened popularity since then, becomeing something of a classic. After it became the standard pre-loaded game on Nokia phones in 1998, Snake found a massive audience. The simplicity and addictiveness of Snake made it available on almost every existing platform under various names.

• **Nibbler :** (arcade_nibbler.so) Nibbler is a simple arcade video game released in 1982. It looks like a Snake, but it isn't the same game. Nibbler itself was inspired by another great classic: Blockade, itself inspired from Tron Light Cycle.

Our graphics library :

• **Ncurses :** (arcade_ncurses.so)

•**SDL2 :** (arcade_sdl2.so)

• **SFML :** (arcade_sfml.so)

• **Libcaca :** (arcade_libcaca.so)

ARCADE

USAGE :

To launch our ARCADE you have to compile de project with « cmake ». Once the project is compiled, just run it with the following command «./arcade ./lib/arcade_ncurses.so ». You can replace the graphic libraries by the one you want and you will be redirected to the menu.

How to integrate your own libraries :

• **Graphic libraries :** To do this you just need to create a class with the name of your choice included in the namespace "acd" and which inherits from the class "AGraphicModule". Then you have to implement 5 functions:

• The **constructor** of your class (which will create your game window and load the font if needed)

• The **destructor** (which will clean and close the window)

• The **display** function which takes a map as parameter. This map represents the content of your window. It is divided into squares like a grid and has a size (x and y) and texts (score and name of the game). All this information can be retrieved with the getters (getGrid(), getSize() and getTexts()). Then you just have to display the squares according to the size (with two loops one for x and one for y). Then you just have to display the text by looping in the texts variable that you got with the getter.

• The function **setColors** which takes as parameter a color among the enum Colors. Depending on the color in parameter it returns the same color in the format of your library. And you can choose the color of the blocks in the display function.

• The **getInputs** function which will detect the keyboard events. It will then call the addInputs function and put in parameter the appropriate input depending on the keyboard button that has been pressed. The input is stored in our Inputs enum.

ARCADE

- **Game libraries :** To do this you just need to create a class with the name of your choice included in the namespace "acd" and which inherits from the class "AGameModule". Then you have to implement 8 functions.

- The **constructor** of your class who set the size of the map, set all of the boolean (_isPaused, _isGameOver, _movedOnce) to false. The position of the food (if there are). And the clock of your game. After you have to set all of your blocks in the map with the colors.

- The **_setGameOver** function who display the « Game Over » message and put the boolean _isGameOver to true.

- The function **update** who take the last input as parameter. This function go to update the clock and do the action according to the last key press on the keyboard

- **Pause** who put the is_paused boolean to true.

- **Play** who put the is_paused boolean to false.

- The **restart** function who reset the game to 0.

Thanks for reading !

Have a nice game.