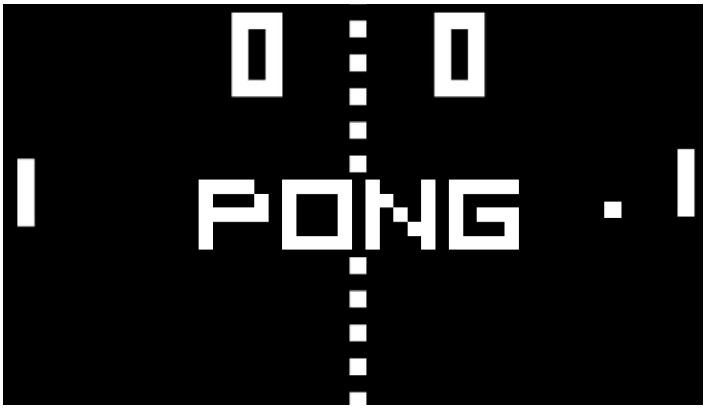
B4 - Object-Oriented Programming

B-OOP-400

Arcade

Documentation



Arcade

binary name: arcade
language: C++
compilation: via Makefile, including re, clean and fclean rules

- The totality of your source files, except all useless files (binary, temp files, obj files,...), must be included in your delivery.
- All the bonus files (including a potential specific Makefile) should be in a directory named *bonus*.
- Error messages have to be written on the error output, and the program should then exit with the 84 error code (0 if there is no error).

Arcade is a gaming platform that lets the user choose a game to play and keeps a register of
player scores.
The elements of our gaming plate-form, our graphics libraries and our games dynamic libraries was loaded at run-time.
Each GUI available for the program are shared libraries that are loaded at run-time.

COMPILATION

You need to compile your grahic library and game so it create a shared library (.so) and only a shared library (.so) for each of them.

All the assets of the graphic library need to be stored in the memory of the library to avoid multiple files for a single library

All the libraries need to be in a lib folder at the root of the arcade binary

INFORMATION

Here are all the enum used in the arcade project:

```
namespace Enum {
    enum class ObjectType {
        PLAYER,
        PLAYER_PART,
        ENEMY,
        ITEM,
        BORDER
    };

    enum class Color {
        RED,
        GREEN,
        BLUE,
        YELLOW,
        WHITE,
        BLACK
    };

    enum libType {
        GRAPHIC,
        GAME
    };
}
```

The ObjectType enum is used to know what kind of object is displayed.
The Color enum is used to know what color is used for the background and the foreground of the text.
The libType enum is used to know what kind of library is loaded.

Graphic Libraries

You should clear the window at the start of the displayObjects function and only render your objects in the display function to avoid graphical bugs

Graphic libraries needs to contain the following functions for working properly:

entryPoint function that return a IDisplayModule object (object of your current library class), the function need to be wrapped in a extern “C”:

Terminal

```
~/B-OOP-400>
extern "C"
IDisplayModule *entryPoint()
{
      return new MyGraphicLibObject();
}
```

InitWindow function that initialize the window, and all the need to the graphical library to run correctly, such as font, or sprite:

Terminal

```
~/B-OOP-400>
void GraphicName::InitWindow()
{

}
```

FiniWindow function that close and free all elements used by the graphical library and the window (basically it returns in the state before the InitWindow function):

Terminal

```
~/B-OOP-400>
void GraphicName::FiniWindow()
{

}
```

displayObjects function that display all the object in the map based on the ObjectType enum at the position given by the pair of int (x, y):

Terminal

```
~/B-OOP-400>
void GraphicName::displayObjects(std::map<int, std::pair<Enum::ObjectType,
std::pair<int, int>> _ObjectData)
{

}
```

displayScore function that display a integer at the pos given in parameter:

Terminal

```
~/B-OOP-400>
void GraphicName::displayScore(int _Score, int x, int y)
{

}
```

displayText function that display a string at a given position with the front color and back color given by the enum Color:

Terminal

```
~/B-OOP-400>
void GraphicName::displayText(std::string _String, std::pair<int, int> _Pos,
Enum::Color FrontFont, Enum::Color BackFont)
{

}
```

GetLibType function that return the type of the library stored in a libType enum for letting the core know if its a graphic library or a game library:

Terminal

```
~/B-OOP-400>
Enum::libType GraphicName::GetLibType()
{

}
```

GetWindowSize function that return a pair of the height and width of the window:

Terminal

```
~/B-OOP-400>
std::pair<int, int> LibrarySFML::GetWindowSize()
{

}
```

getUserInput function that based on the current graphic library get the user input (key pressed by the user) and return it:

Terminal

```
~/B-OOP-400>
char GraphicName::getUserInput()
{

}
```

display function that render all current drawed object to the window (do not render object in other function it could cause blinking screen):

Terminal

```
~/B-OOP-400>
void GraphicName::display()
{

}
```

Game Libraries

Do not make infinite loop, the core already make it

It the handleUserInput function job to slow down the game loop

Here is the Objects map that need to be filled by the handleUserInput function:

```
std::map<int, std::pair<Enum::ObjectType, std::pair<int, int>>> _ObjectData;
```

Game libraries need to contain the following functions for working properly:

handleUserInput function that take a key in parameter (the key got by the getUserInput function from the current graphic library) and do action based on this key, like moving the player. This function should modify the score, the status of the game (game over) and filling the Objects map (a main function like for the Game library):

Terminal

```
~/B-OOP-400>
void GameName::handleUserInput(char key)
{

}
```

getScore function that return the current score of the game:

Terminal

```
~/B-OOP-400>
int GameName::getScore()
{

}
```

getStatus function that return the current status of the game (true for game over):

Terminal

```
~/B-OOP-400>
bool GameName::getStatus()
{

}
```

GetLibType function that return the type of the library stored in a libType enum for letting the core know if its a graphic library or a game library:

Terminal

```
~/B-OOP-400>
Enum::libType GameName::GetLibType()
{

}
```

ResetGame function that resets the game to it's initial state:

Terminal

```
~/B-OOP-400>
void GameName::ResetGame()
{

}
```

You can use the base graphic and base game library to make your own graphic and game library !