B4 - Object-Oriented Programming

B-OOP-400

Arcade

A Retro Platform

1. IDisplayModule

If you want to implement a graphic library, here is the interface you have to inherit from :

```
class IDisplayModule {
    public:
        virtual ~IDisplayModule() = default;

        virtual void init() = 0;
        virtual void stop() = 0;

        virtual const std::string &getName() const = 0;

        virtual void playMusic(std::string) = 0;
        virtual void stopCurrentMusic() = 0;

        virtual bool isOpen() = 0;
        virtual void clear() = 0;
        virtual void display() = 0;

        virtual void drawText(int x, int y,
            const std::string &str, int fontSize, Color color) = 0;
        virtual void drawShape(int x, int y,
            std::vector< std::vector<Color> > shape) = 0;

        virtual float getTime() = 0;
        virtual void restartTime() = 0;

        virtual std::map<Input, bool> catchInput() = 0;
};
```

If you want to add a lib, place your folder architecture in the **lib** folder and fill the **Makefile** in it.

IDisplayModule - Method's Explanation

```
void init();
```

This method is called when the Arcade wants to load the library.

```
void stop();
```

This method is called when the Arcade wants to unload the library.

```
const std::string &getName();
```

This method returns the name of the library to display it in the main menu.

```
void playMusic(std::string musicName);
```

This method starts a music named ***musicName****. It should be placed in the "mucis" folder.*

Notes : Won't work with nCurses library.

```
void stopCurrentMusic();
```

This method stops the current playing music.

```
bool isOpen();
```

This method returns TRUE if the window is opened and FALSE if it is not.

```
void clear();
```

This method is called at the beginning of the main loop, it clears the window of all displayed elements.

```
void display();
```

This method is called at the end of the main loop, to display every elements the library has asked for before.

```
void drawText(int x, int y, const std::string &str, int
fontSize, Color color);
```

---

*This method displays the text **str** at the position **{x, y}** of size **fontSize** and of color **color.***

```
void drawShape(int x, int y, std::vector< std::vector<Color> >
shape);
```

---

*This method displays the asked **shape** which is a vector of vectors of Colors, at the position **{x, y}***

```
float getTime();
```

This method returns the time elapsed since the last call in float.

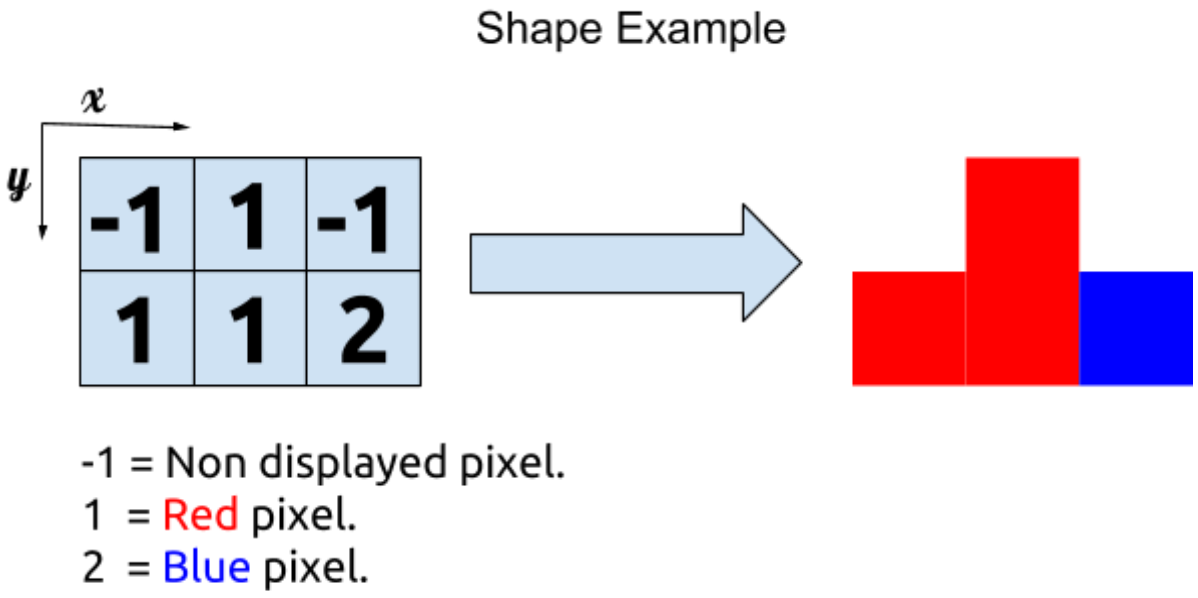
```
void restartTime();
```

This method restarts the time handled by the library.

```
std::map<Input, bool> catchInput();
```

This method returns a map containing each inputs pressed at actual frame.

Example : `map[RETURN_KEY] = TRUE;`



Inputs Class

```
enum Input {
    LEFT_ARROW_KEY,
    RIGHT_ARROW_KEY,
    UP_ARROW_KEY,
    DOWN_ARROW_KEY,
    SPACE_KEY,
    RETURN_KEY,
    ERASE_KEY,
    ESCAPE_KEY,
    A_KEY,
    B_KEY,
    C_KEY,
     .
     .
     .
    X_KEY,
    Y_KEY,
    Z_KEY,
    ONE_KEY,
    TWO_KEY,
    THREE_KEY,
    FOUR_KEY,
    FIVE_KEY,
    SIX_KEY,
    SEVEN_KEY,
    EIGHT_KEY,
    NINE_KEY,
    ZERO_KEY,
};
```

Color Class

```
enum Color {
    NONE = -1,
    BLACK = 0,
    RED = 1,
    GREEN = 2,
    YELLOW = 3,
    BLUE = 4,
    MAGENTA = 5,
    CYAN = 6,
    WHITE = 7
};
```

2. IGameModule

If you want to implement a game library, here is the interface you have to inherit from :

```
class IGameModule {
    public:
        virtual ~IGameModule() = default;

        virtual void init(const std::string &playerName,
          const int &highScore) = 0;
        virtual void stop() = 0;

        virtual const std::string &getName() const = 0;

        virtual void restart() const = 0;
        virtual void run(IDisplayModule *library,
          std::map<Input, bool> &inputs) = 0;
        virtual int getHighScore() const = 0;
};
```

If you want to add a game, place your folder architecture in the **games** folder and fill the **Makefile** in it.

IGameModule - Method's Explanation

```
void init(const std::string &playerName, const int &highScore);
```

This method is called when the Arcade wants to load the game.
playerName *is the name of the player who did the* ***highScore*** *for this game.*

```
void stop();
```

This method is called when the Arcade wants to unload the game.

```
const std::string &getName();
```

This method returns the name of the game to display it in the main menu.

```
void restart();
```

This method restarts the running game.

```
void run(IDisplayModule *library, std::map<Input, bool> &inputs);
```

This method is called in the main loop after the DisplayModule clear and before the DisplayModule display.
***library** is the currently running graphic library, it allows the game to access to all his methods to display the game's objects.*
***inputs** are the current frame's pressed buttons map.*

```
int getHighScore();
```

This method returns the highestScore done by the user. It is called after the gameModule stop.