DOCUMENTATION-ARCADE

How to Install ?

To run this project you need ton install few tools. First : gpp to compile c++ Make: Use makefiles Graphical Librairies : LibCaca, Sdl2, Ncurses

UBUNTU / DEBIAN

```
apt install gpp make
apt-get install libncurses5-dev libncursesw5-dev
apt-get install libsdl2-dev libsdl2-ttf-dev
apt-get install libcaca-dev
```

How to launch ?

Run `make` and `./arcade lib/arcade_ncurses.so`.

Add another Graphic Library !

You need to respect this Interface :

```
class IGraphicLib {

public:

    virtual ~IGraphicLib() = default;
    virtual void clearwin() = 0;
    virtual int getKey() = 0;
    virtual void printText(int x, int y, std::string text) = 0;
    virtual void refresh() = 0;
    virtual void printbox(int x, int y, int h, int w) = 0;
    virtual void drawMenu(std::vector<std::string> libraries, std::vector<std::string> games, int curr, int act, std::string name) =
    virtual char libType() = 0;

protected:
private:};
```

Remember to include IGraphicLib.hpp !

Each library need to respect a class like that.

There is an example for the lib caca library header file.

```
class libcaca: IGraphicLib {
public:

    libcaca();
    ~libcaca();
    int getKey();
    void clearwin();
    void printText(int x, int y, std::string text);
    void refresh();
    void printbox(int x, int y, int h, int w);
    void drawMenu(std::vector<std::string> libraries, std::vector<std::string> games, int curr, int act, std::string name);
    char libType() { return ('L');}

private:};
```

Let's review each function : `int getkey()` Is the function that must return the entire **keyboard input** in **int**.
`void clearwin()` This function is used to **erase all** the content on the window.
`void printText(int x, it y, std::string text)` This function **displays** the content of the variable **text** at the **x** and **y** positions given in parameters.
`void refresh()` This function **updates** the **entire** window.
`void printbox(int x, int x, int h, int w)` This function allows to display a box at **x** and **y** positions with also a variable **height** and **width** thanks to the parameters **h** for **height** and **w** for **width**
`char libType()` This function will return **L** because it's a graphical Library.
`libraries, std::vector games, int curr, int act, std::string name)>` This function is used to display the menu, it takes as parameters a **std::string** vector containing the name of all **graphic libraries** and a **std::string** vector containing the name of all games. This function also takes as parameters an **int curr** of the value of the currently loaded **graphic library**, an **int act** of the currently loaded **game library** and a **std::string name** containing the **pseudo** of the user.

You must initialize all all you'r library (window, fonts, pics) in the contructor of the library, and destroy them in the destructor.

For your library to work and be interpreted by the core you must compile it in **.so** and place it in the **libs** folder in the root.

Add another Game !

You need to respect this Interface :

```
class Player{

public:
    Player(int x, int y, std::string symbol)
    {
        this->x = x;
        this->y = y;
        this->symbol = symbol;
    };
    int x;
    int y;
    std::string symbol;};


class IGames {

public:
    virtual ~IGames() = default;
    virtual int run(IGraphicLib *GraphicLib, std::string name) = 0;
    virtual void set_input(int a) = 0;
    virtual char libType() = 0;

protected:
private:};
```

Remember to include IGames.hpp !

There is a Player class which allows you to manage your player easily, it contains only an x and y position and a std::string symbol containing the symbol of the player.
For ascii games this allows to manage easily the player.
It simply allows you to create a player variable and give it attributes:
`Player player(10, 10, "P");`
This initialization will create a player at position 10, 10 with symbol P.

For your library to work and be interpreted by the core you must compile it in **.so** and place it in the **libs** folder in the root.

KEYS

Some keys are reserved for the operation of the core.

Keys **1**, **2** to change the graphic library (**F1**,**F2** for libcaca)

Keys **3**,**4** to change Game(**F3**,**F4** for libCaca)
Keys **5**, **6** to launch the game, or return to the menu(**F5**, **F6** for libCaca)
Key **7** to stop and quit the program (**F7** for libCaca)

The keys from **A**to **Z** in the menu are used to enter the nickname (**Space** not included)
In game only the **ZQSD** and **Space** keys are mapped and sent to the game for movement.