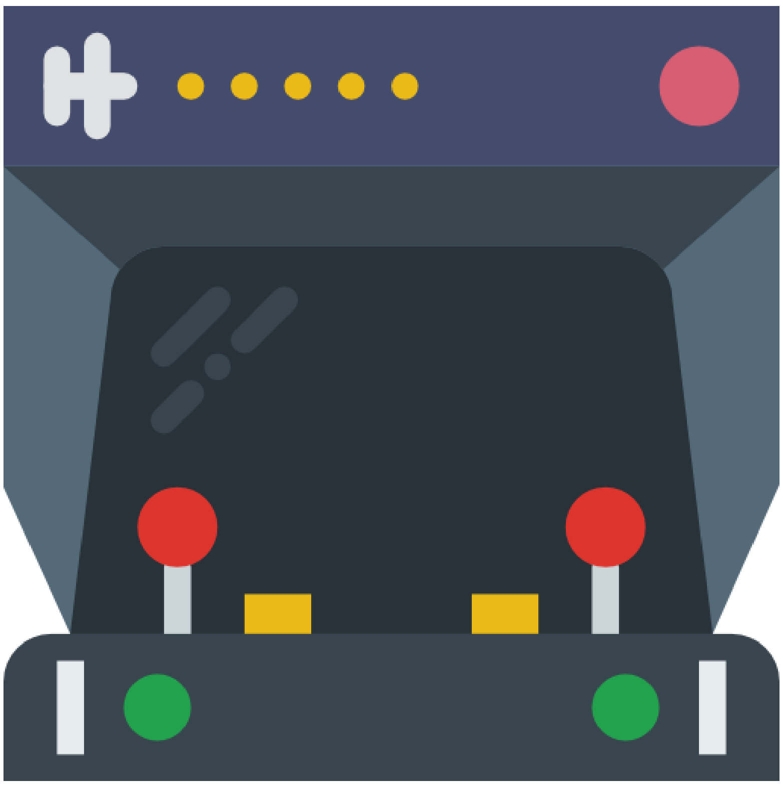
OOP Arcade 2018



Epitech
Promos 2022

Lanfranchini Grégoire
Dumont Clément
Gery Mathieu

Table of contents

Table of contents **2**

Introduction : **3**

To compile: **3**

Install Libraries and dependencies : **3**

Libraries and games : **4**

Launch the program : **4**

Add games and lib : **5**

- Games : 5
- Entities : 5
- Libs : 6

Introduction :

Arcade is a gaming platform: a program that lets the user choose a game to play and keeps a register of player scores. They are differents games like Pacman and Nibler the program also include the ability to choose differents graphical library to run your games and it's possible to change it at run time ! In your case we use SFM, SDL2 and Libcaca as graphicals library.

To compile:

In order to compile our program you must use the Makefile. You can use differents rules:

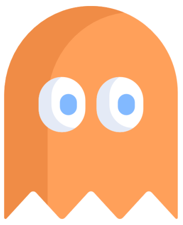
- core: it only build the core of your arcade (not the games nor the graphical libraries)
- games: it only build your games librairies
- graphicals: it must only build your graphical libraries

Install Libraries and dependencies :

You must install SFML by simply clone and launch *"./install-sfml.sh"* after cloning this repo : https://github.com/AlexMog/InstallSFMLScript

You must also install SDL2 and his dependencies :*"SDL2_image"* & *"SDL2_ttf"*, *"SDL2-devel"* with *"dnf install .."* command.

And finaly you need libcaca we can found here : https://github.com/cacalabs/libcaca by simply follow the Readme. And you must install the imlib2 by running this command : *"dnf install imlib2-devel"*



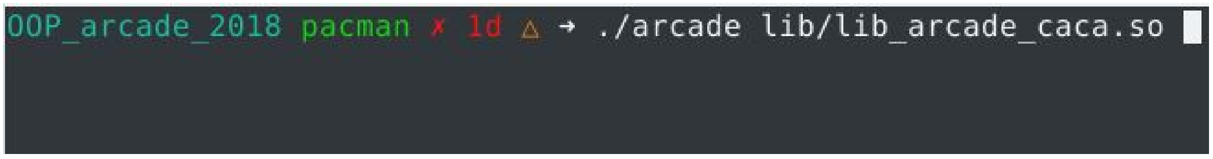
Libraries and games :

The libraries are stocks in “*/lib*” folder.
The games are stock in *“/games”* folder.
It’s in here where you can add you own libraries and games.

Launch the program :

To lunch **arcade** simply tape *“./arcade”* followed by the path of a graphical libraries in *“/lib”*.

Example :



```
OOP_arcade_2018 pacman ✗ 1d △ → ./arcade lib/lib_arcade_caca.so
```

Commands :

- **“Escape”** : can exit the program
- **“L”** : go to the next graphics libraries
- **“k”** : go to the previous graphics libraries
- **“Enter”** : select a graphical library or a game in the menu
- **“arrow keys”** : use to move into the games

Add games and lib :

Games :

Each games are implemented as shared libraries and they are load at run time. In order to code a game you class **must** implement the *"IGame"* interface include the following commands:

- `init(std::string username)` : to init the game with a username
- `bool refresh(ILib::event)` : call at each frames
- `std::vector<Entity> getEntities()` : get the Entities of the game

Entities :

In each of our games we use Entity, which are for examples : players, walls, monsters, ect …

This is the following Entity object :

```
struct vector2_t {
   float x;
   float y;
};

class Entity {
   public:
       Entity() = default;
       ~Entity() = default;
       Entity(size_t, size_t, size_t, size_t);
       explicit Entity(std::string path);
       explicit Entity(text_t text);

       Texture getTexture();
       vector2_t getPosition() const;
       vector2_t getSize() const;
       int getAngle();
       int getDepth();

       void setTexture(Texture texture);
       void setPosition(vector2_t);
       void setSize(vector2_t);
       void setAngle(int);
       void setDepth(int);
```

```
        std::string _menuMetaData = "";
        Texture _texture;
    protected:
    private:
        vector2_t _pos = {0, 0};
        vector2_t _size = {1, 1};
        int _angle = 0;
        int _depth = 0;
};
```

Entity can be create with a Path to a sprite, a text_t that contain std::string and color, it can also create with RGB and transparency parameters.

Libs :

If you want to create your own graphical library for arcade, you **must** implement the *"ILib"* interface.

This is the following Ilib Interface :

```
class ILib {
    public:
        typedef enum event {up, left, down, right, enter, escape, next_lib,
prev_lib, next_game, prev_game, changeMenuModule, restart, menu, help, none}
event;

        //window method
        virtual void initWindow() = 0;
        virtual void destroyWindow() = 0;
        virtual bool isWindowOpen() = 0;
        virtual void refreshWindow() = 0;

        //input method
        virtual event getEvent() = 0;

        //sprite and texture
        virtual bool getTexture(Entity entity) = 0;
        virtual void draw(Entity) = 0;
};
```

- `void initWindow()` : to init the window
- `void destroyWindow()` : to destroy the window
- `bool isWindowOpen()` : check if the window is open
- `void refreshWindow()` : refresh window
- `event getEvent()` : getEvent at each frames
- `bool getTexture(Entity entity)` : get texture of an Entity
- `void draw(Entity)` : draw and Entity