Université Abdelmalek Essaâdi
Faculté des Sciences et Technique
de Tanger

RAPPORT DE PROJET

Réalisé par :

- Mohamed SERBOUT
- Basma EL BARKI
- Groupe : 2

Encadré par :

Pr. Ikram Ben abdel ouahab

Objective

Vieux générale de projet

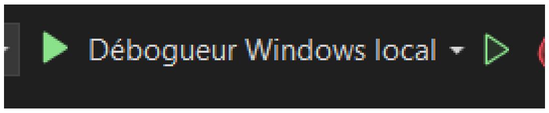
Dans ce projet on développer un jeu connue PICO_PARK à partir de logiciel sfml. L'objectif principal de ce projet est de maitriser la programmation orientée objet par la mise en place D'un jeu vidéo 2D, le jeu roller PICO_PARK, c'est un jeu qui a connu un grand succès dans les plateformes mobile.

INTRODUCTION

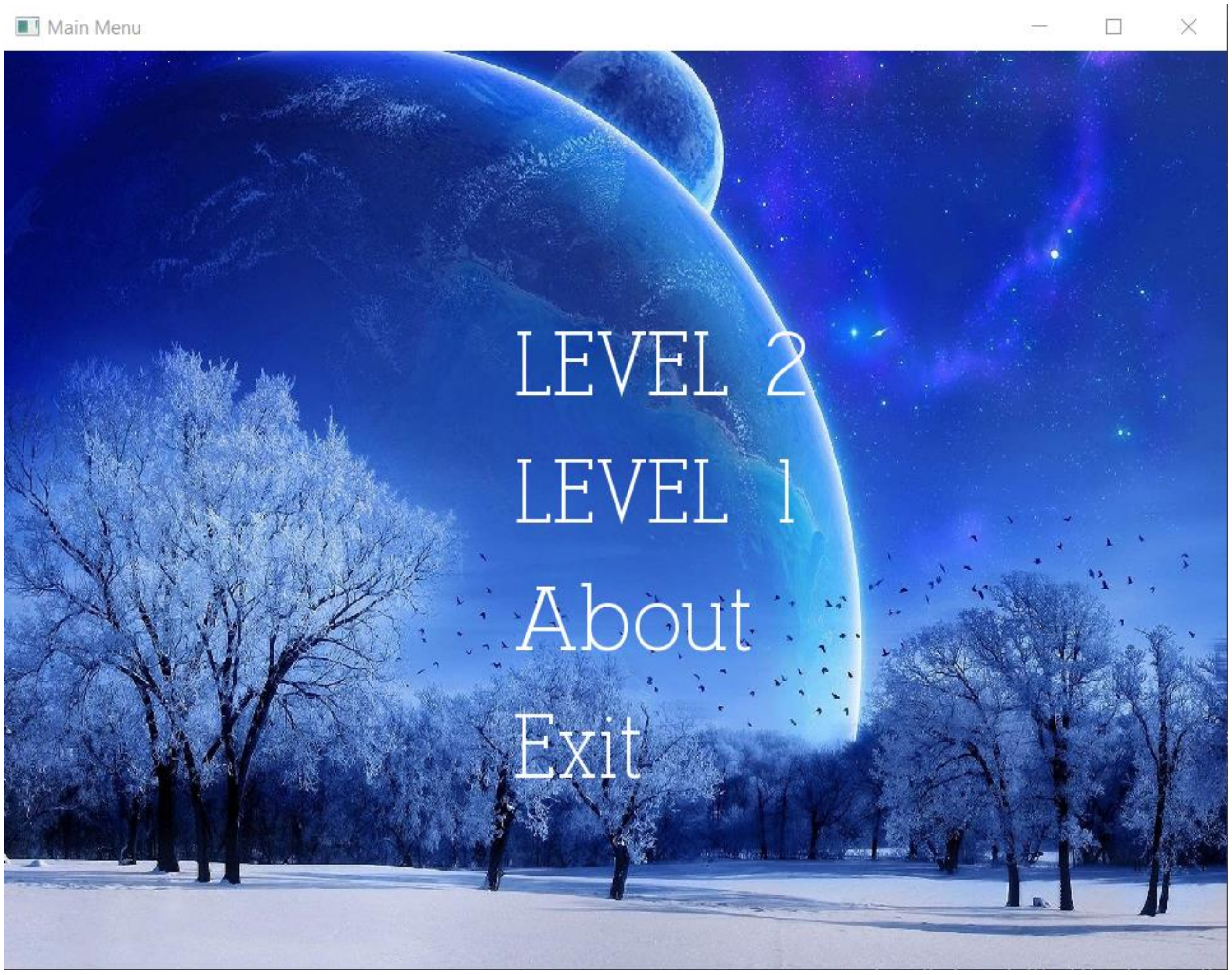
SFML est une interface de programmation destinée à construire des jeux vidéo ou des programmes interactifs. Elle est écrite en C++, mais également disponible dans divers langages comme C,D,Python,Ruby,OCaml ou Microsoft .NET. Elle a entre autres pour but de proposer une alternative orientée objet à la SDL.

Comment Jouer ?

Pour jouer double clic sur le dossier pico_park après sur fichier comporte le picoPark.sln . alors maintenant click sur Déboguer Windows local :

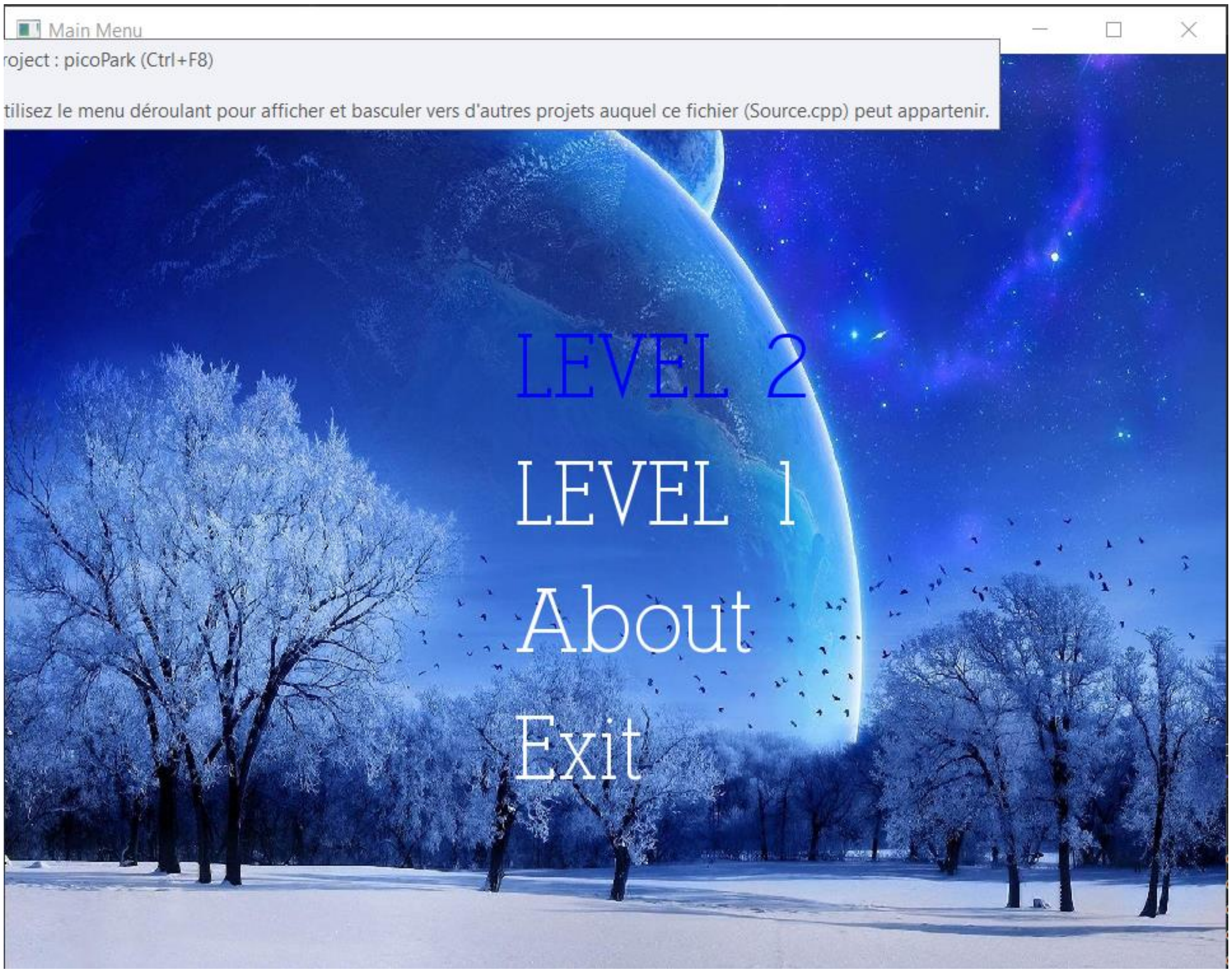


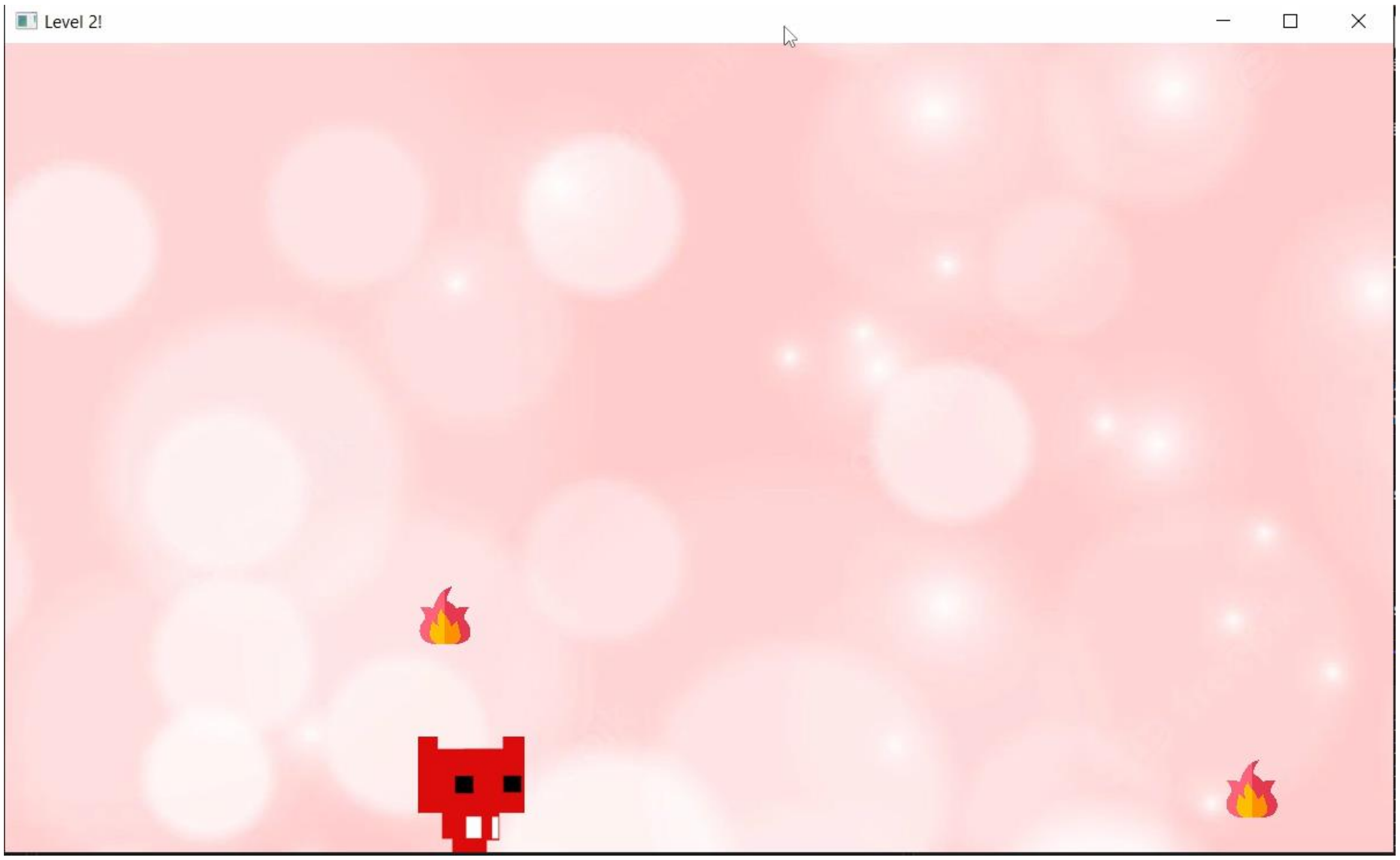
Alors vous affiche menu de jeux :

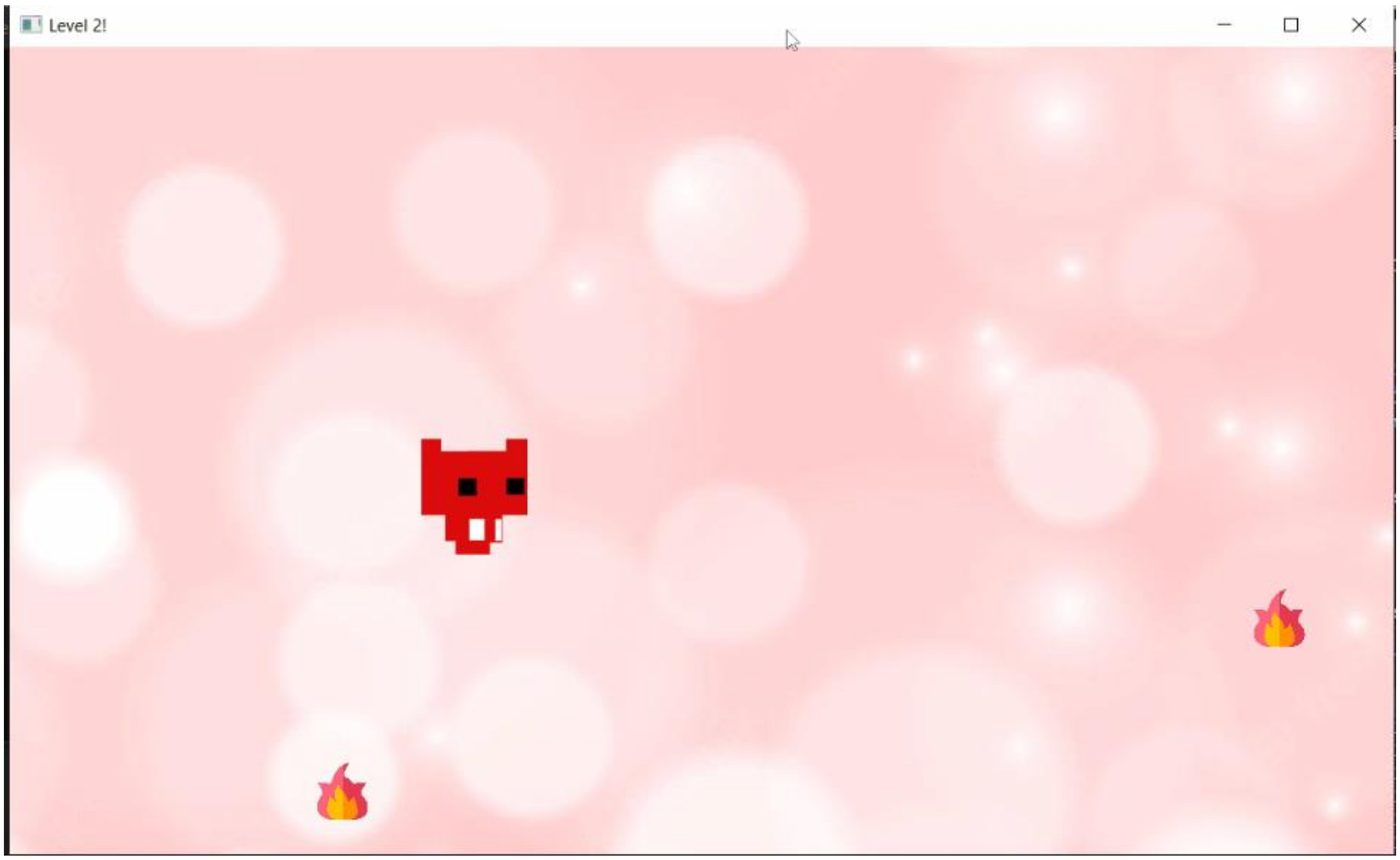


Et voilà maintenant vous devez choisir level 2 ou 1 o quelqu'un avec clavier et clic sur entrer

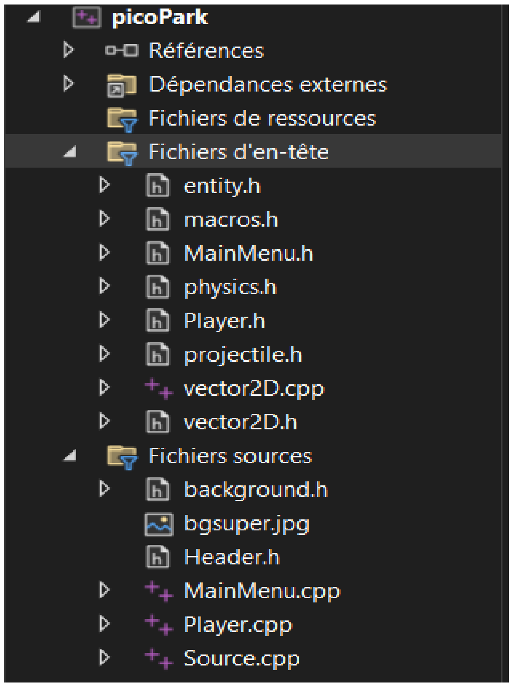
Example : clic sur Level 2


Main Menu
roject : picoPark (Ctrl+F8)
tilisez le menu déroulant pour afficher et basculer vers d'autres projets auquel ce fichier (Source.cpp) peut appartenir.
LEVEL 2
LEVEL 1
About
Exit

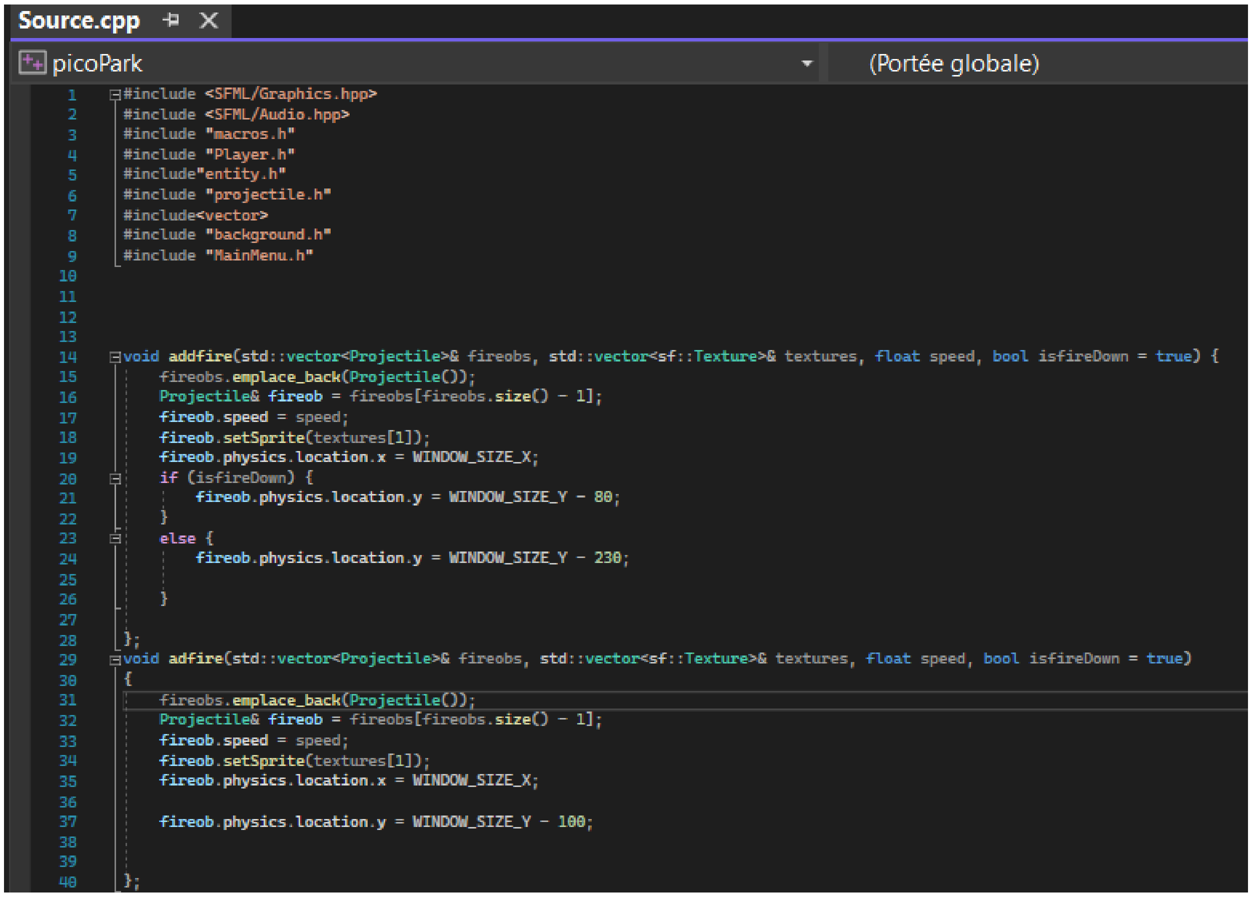

Level 2!


Level 2!

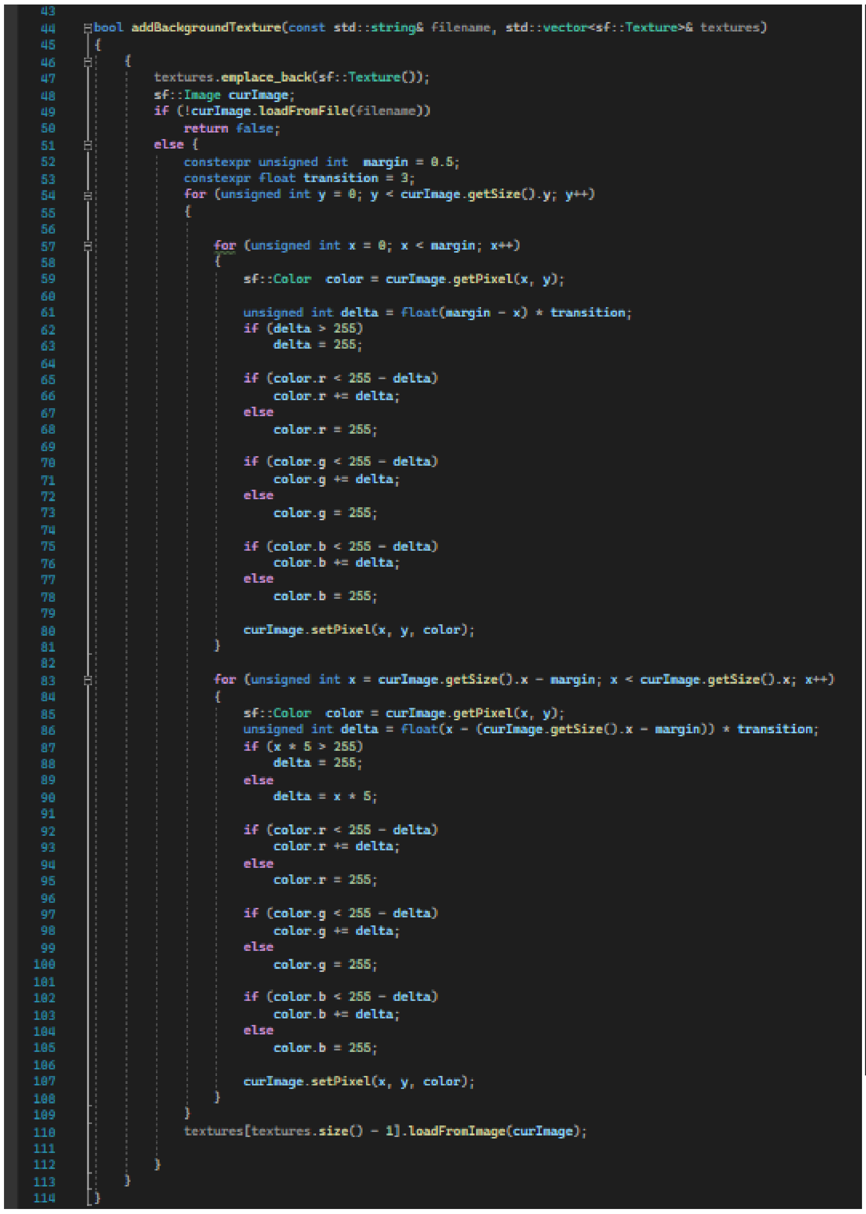
Code source :



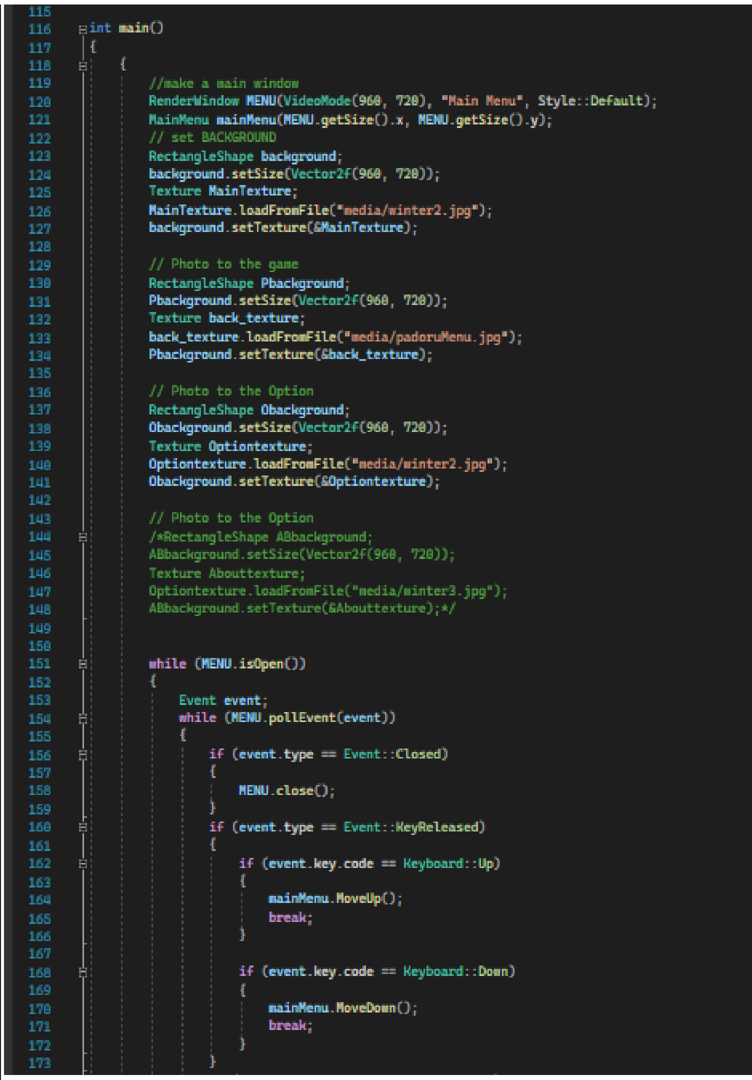
Source.cpp :



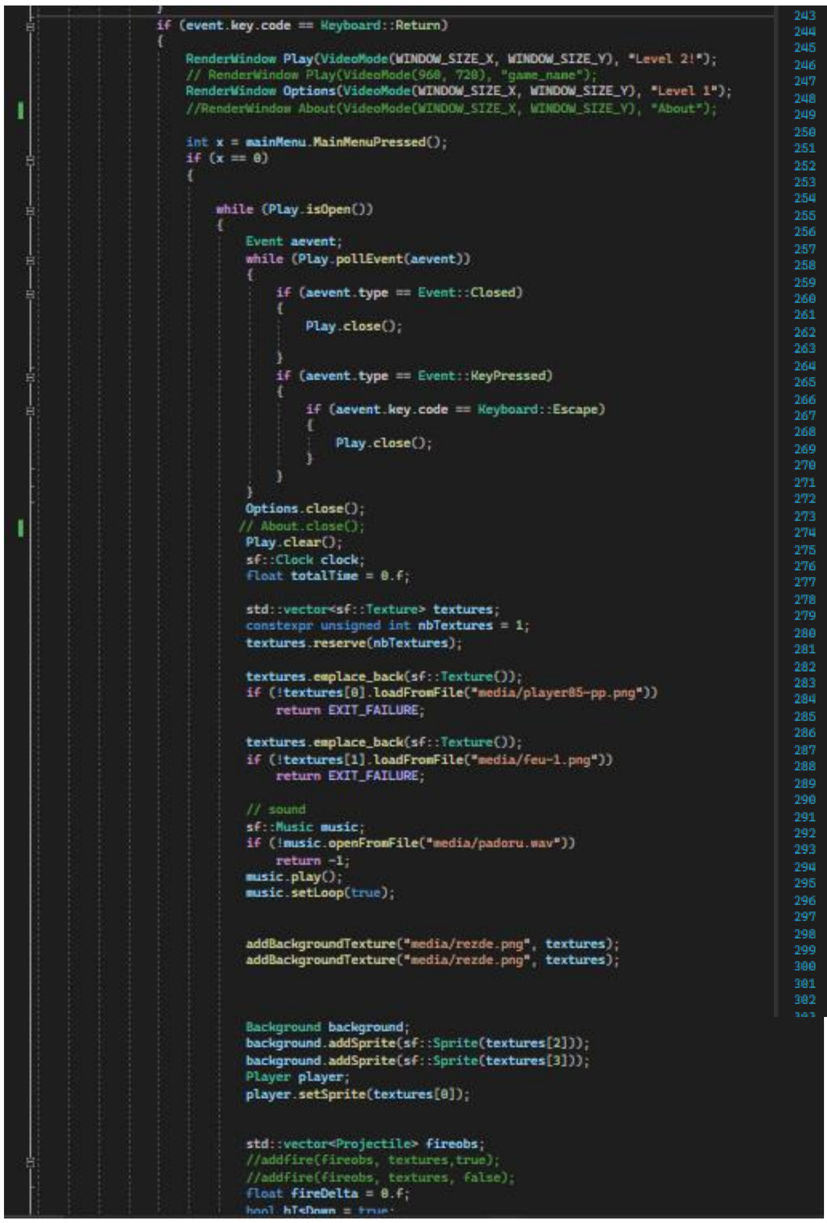
```cpp
#include <SFML/Graphics.hpp>
#include <SFML/Audio.hpp>
#include "macros.h"
#include "Player.h"
#include"entity.h"
#include "projectile.h"
#include<vector>
#include "background.h"
#include "MainMenu.h"




void addfire(std::vector<Projectile>& fireobs, std::vector<sf::Texture>& textures, float speed, bool isfireDown = true) {
    fireobs.emplace_back(Projectile());
    Projectile& fireob = fireobs[fireobs.size() - 1];
    fireob.speed = speed;
    fireob.setSprite(textures[1]);
    fireob.physics.location.x = WINDOW_SIZE_X;
    if (isfireDown) {
        fireob.physics.location.y = WINDOW_SIZE_Y - 80;
    }
    else {
        fireob.physics.location.y = WINDOW_SIZE_Y - 230;

    }

};
void adfire(std::vector<Projectile>& fireobs, std::vector<sf::Texture>& textures, float speed, bool isfireDown = true)
{
    fireobs.emplace_back(Projectile());
    Projectile& fireob = fireobs[fireobs.size() - 1];
    fireob.speed = speed;
    fireob.setSprite(textures[1]);
    fireob.physics.location.x = WINDOW_SIZE_X;

    fireob.physics.location.y = WINDOW_SIZE_Y - 100;


};
```



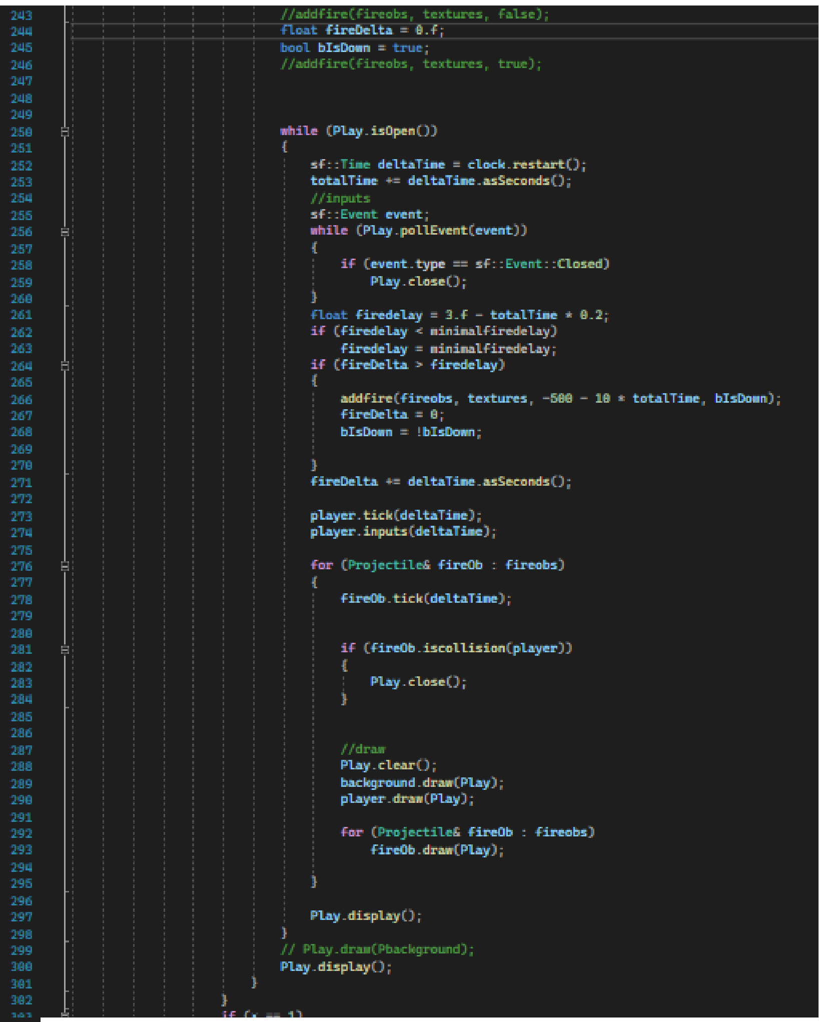
```cpp

bool addBackgroundTexture(const std::string& filename, std::vector<sf::Texture>& textures)
{
    {
        textures.emplace_back(sf::Texture());
        sf::Image curImage;
        if (!curImage.loadFromFile(filename))
            return false;
        else {
            constexpr unsigned int  margin = 0.5;
            constexpr float transition = 3;
            for (unsigned int y = 0; y < curImage.getSize().y; y++)
            {

                for (unsigned int x = 0; x < margin; x++)
                {
                    sf::Color  color = curImage.getPixel(x, y);

                    unsigned int delta = float(margin - x) * transition;
                    if (delta > 255)
                        delta = 255;

                    if (color.r < 255 - delta)
                        color.r += delta;
                    else
                        color.r = 255;

                    if (color.g < 255 - delta)
                        color.g += delta;
                    else
                        color.g = 255;

                    if (color.b < 255 - delta)
                        color.b += delta;
                    else
                        color.b = 255;

                    curImage.setPixel(x, y, color);
                }

                for (unsigned int x = curImage.getSize().x - margin; x < curImage.getSize().x; x++)
                {
                    sf::Color  color = curImage.getPixel(x, y);
                    unsigned int delta = float(x - (curImage.getSize().x - margin)) * transition;
                    if (x * 5 > 255)
                        delta = 255;
                    else
                        delta = x * 5;

                    if (color.r < 255 - delta)
                        color.r += delta;
                    else
                        color.r = 255;

                    if (color.g < 255 - delta)
                        color.g += delta;
                    else
                        color.g = 255;

                    if (color.b < 255 - delta)
                        color.b += delta;
                    else
                        color.b = 255;

                    curImage.setPixel(x, y, color);
                }
            }
            textures[textures.size() - 1].loadFromImage(curImage);

        }
    }
}
```



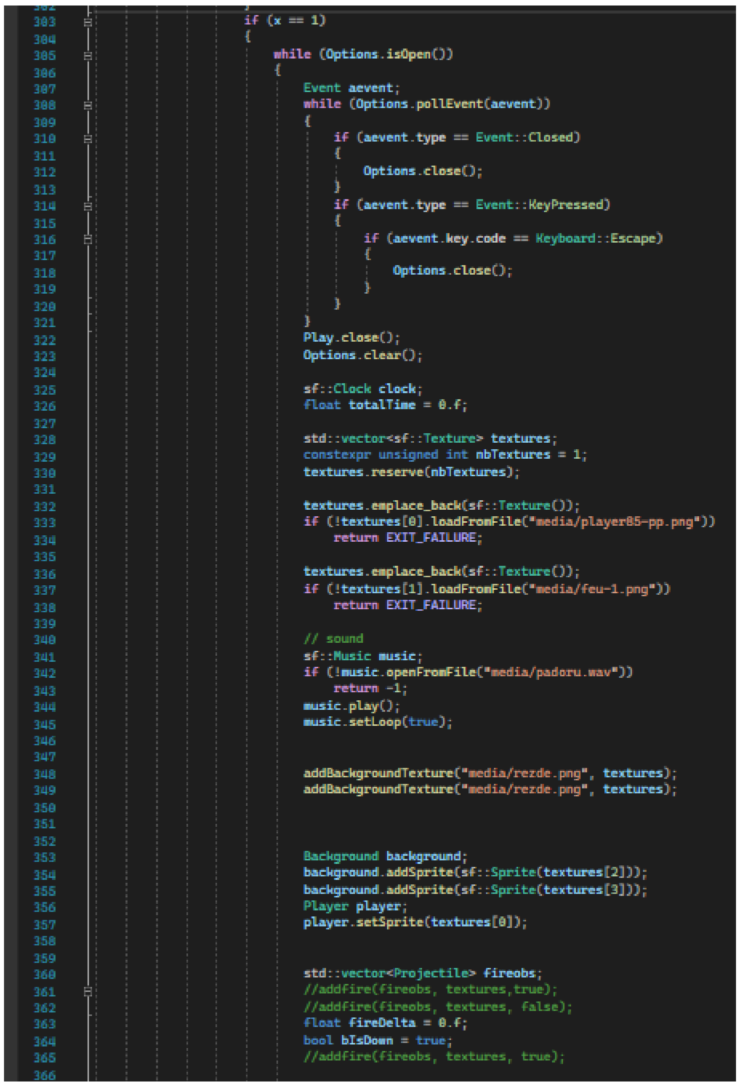
```cpp

int main()
{
    {
        //make a main window
        RenderWindow MENU(VideoMode(960, 720), "Main Menu", Style::Default);
        MainMenu mainMenu(MENU.getSize().x, MENU.getSize().y);
        // set BACKGROUND
        RectangleShape background;
        background.setSize(Vector2f(960, 720));
        Texture MainTexture;
        MainTexture.loadFromFile("media/winter2.jpg");
        background.setTexture(&MainTexture);

        // Photo to the game
        RectangleShape Pbackground;
        Pbackground.setSize(Vector2f(960, 720));
        Texture back_texture;
        back_texture.loadFromFile("media/padoruMenu.jpg");
        Pbackground.setTexture(&back_texture);

        // Photo to the Option
        RectangleShape Obackground;
        Obackground.setSize(Vector2f(960, 720));
        Texture Optiontexture;
        Optiontexture.loadFromFile("media/winter2.jpg");
        Obackground.setTexture(&Optiontexture);

        // Photo to the Option
        /*RectangleShape ABbackground;
        ABbackground.setSize(Vector2f(960, 720));
        Texture Abouttexture;
        Optiontexture.loadFromFile("media/minter3.jpg");
        ABbackground.setTexture(&Abouttexture);*/


        while (MENU.isOpen())
        {
            Event event;
            while (MENU.pollEvent(event))
            {
                if (event.type == Event::Closed)
                {
                    MENU.close();
                }
                if (event.type == Event::KeyReleased)
                {
                    if (event.key.code == Keyboard::Up)
                    {
                        mainMenu.MoveUp();
                        break;
                    }

                    if (event.key.code == Keyboard::Down)
                    {
                        mainMenu.MoveDown();
                        break;
                    }
                }
```



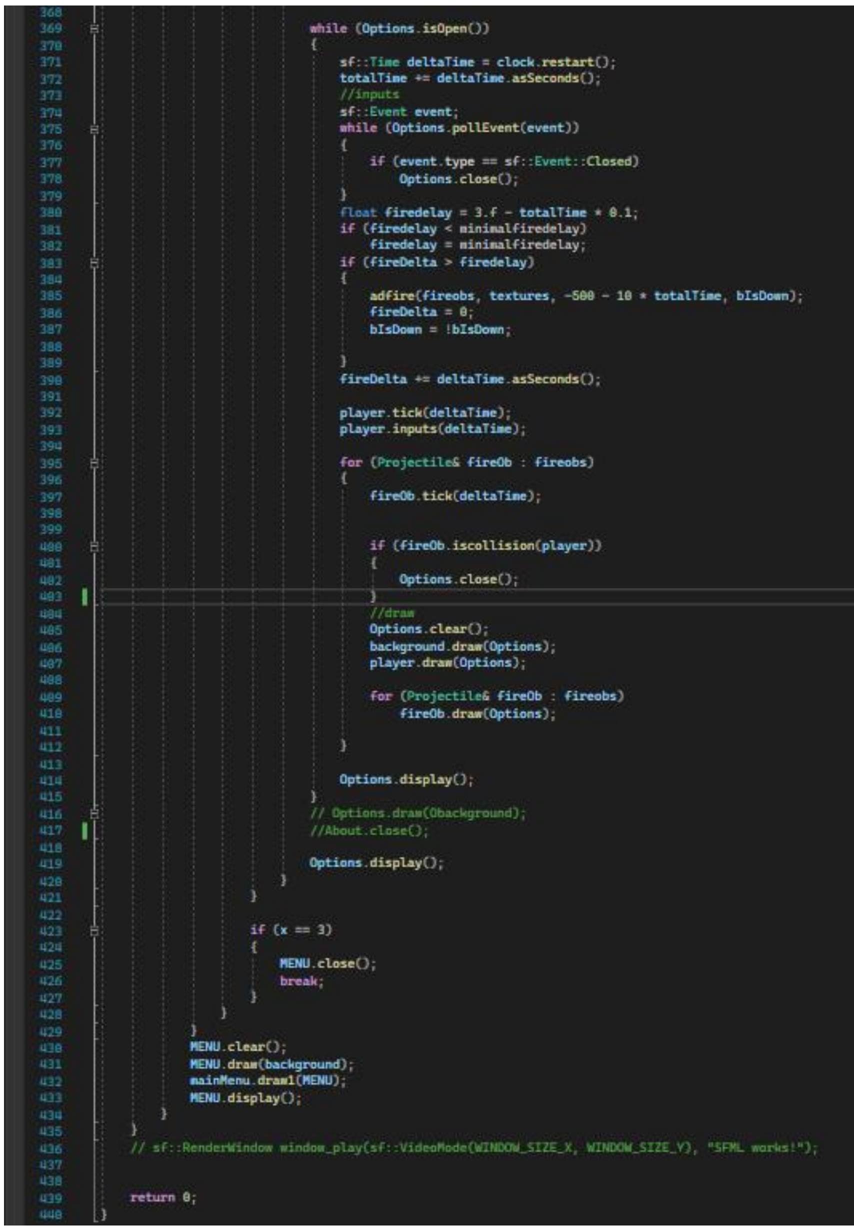
```cpp
if (event.key.code == Keyboard::Return)
{
    RenderWindow Play(VideoMode(WINDOW_SIZE_X, WINDOW_SIZE_Y), "Level 2!");
    // RenderWindow Play(VideoMode(960, 720), "game_name");
    RenderWindow Options(VideoMode(WINDOW_SIZE_X, WINDOW_SIZE_Y), "Level 1");
    //RenderWindow About(VideoMode(WINDOW_SIZE_X, WINDOW_SIZE_Y), "About");

    int x = mainMenu.MainMenuPressed();
    if (x == 0)
    {

        while (Play.isOpen())
        {
            Event aevent;
            while (Play.pollEvent(aevent))
            {
                if (aevent.type == Event::Closed)
                {
                    Play.close();

                }
                if (aevent.type == Event::KeyPressed)
                {
                    if (aevent.key.code == Keyboard::Escape)
                    {
                        Play.close();
                    }
                }
            }
            Options.close();
           // About.close();
            Play.clear();
            sf::Clock clock;
            float totalTime = 0.f;

            std::vector<sf::Texture> textures;
            constexpr unsigned int nbTextures = 1;
            textures.reserve(nbTextures);

            textures.emplace_back(sf::Texture());
            if (!textures[0].loadFromFile("media/player85-pp.png"))
                return EXIT_FAILURE;

            textures.emplace_back(sf::Texture());
            if (!textures[1].loadFromFile("media/feu-1.png"))
                return EXIT_FAILURE;

            // sound
            sf::Music music;
            if (!music.openFromFile("media/padoru.wav"))
                return -1;
            music.play();
            music.setLoop(true);


            addBackgroundTexture("media/rezde.png", textures);
            addBackgroundTexture("media/rezde.png", textures);



            Background background;
            background.addSprite(sf::Sprite(textures[2]));
            background.addSprite(sf::Sprite(textures[3]));
            Player player;
            player.setSprite(textures[0]);


            std::vector<Projectile> fireobs;
            //addfire(fireobs, textures,true);
            //addfire(fireobs, textures, false);
            float fireDelta = 0.f;
            bool bIsDown = true;
```



```cpp
            //addfire(fireobs, textures, false);
            float fireDelta = 0.f;
            bool bIsDown = true;
            //addfire(fireobs, textures, true);



            while (Play.isOpen())
            {
                sf::Time deltaTime = clock.restart();
                totalTime += deltaTime.asSeconds();
                //inputs
                sf::Event event;
                while (Play.pollEvent(event))
                {
                    if (event.type == sf::Event::Closed)
                        Play.close();
                }
                float firedelay = 3.f - totalTime * 0.2;
                if (firedelay < minimalfiredelay)
                    firedelay = minimalfiredelay;
                if (fireDelta > firedelay)
                {
                    addfire(fireobs, textures, -500 - 10 * totalTime, bIsDown);
                    fireDelta = 0;
                    bIsDown = !bIsDown;

                }
                fireDelta += deltaTime.asSeconds();

                player.tick(deltaTime);
                player.inputs(deltaTime);

                for (Projectile& fireOb : fireobs)
                {
                    fireOb.tick(deltaTime);


                    if (fireOb.iscollision(player))
                    {
                        Play.close();
                    }


                    //draw
                    Play.clear();
                    background.draw(Play);
                    player.draw(Play);

                    for (Projectile& fireOb : fireobs)
                        fireOb.draw(Play);

                }

                Play.display();
            }
            // Play.draw(Pbackground);
            Play.display();
        }
    }
    if (x == 1)
```



```cpp
if (x == 1)
{
    while (Options.isOpen())
    {
        Event aevent;
        while (Options.pollEvent(aevent))
        {
            if (aevent.type == Event::Closed)
            {
                Options.close();
            }
            if (aevent.type == Event::KeyPressed)
            {
                if (aevent.key.code == Keyboard::Escape)
                {
                    Options.close();
                }
            }
        }
        Play.close();
        Options.clear();

        sf::Clock clock;
        float totalTime = 0.f;

        std::vector<sf::Texture> textures;
        constexpr unsigned int nbTextures = 1;
        textures.reserve(nbTextures);

        textures.emplace_back(sf::Texture());
        if (!textures[0].loadFromFile("media/player85-pp.png"))
            return EXIT_FAILURE;

        textures.emplace_back(sf::Texture());
        if (!textures[1].loadFromFile("media/feu-1.png"))
            return EXIT_FAILURE;

        // sound
        sf::Music music;
        if (!music.openFromFile("media/padoru.wav"))
            return -1;
        music.play();
        music.setLoop(true);


        addBackgroundTexture("media/rezde.png", textures);
        addBackgroundTexture("media/rezde.png", textures);



        Background background;
        background.addSprite(sf::Sprite(textures[2]));
        background.addSprite(sf::Sprite(textures[3]));
        Player player;
        player.setSprite(textures[0]);


        std::vector<Projectile> fireobs;
        //addfire(fireobs, textures,true);
        //addfire(fireobs, textures, false);
        float fireDelta = 0.f;
        bool bIsDown = true;
        //addfire(fireobs, textures, true);

```



```cpp

                        while (Options.isOpen())
                        {
                            sf::Time deltaTime = clock.restart();
                            totalTime += deltaTime.asSeconds();
                            //inputs
                            sf::Event event;
                            while (Options.pollEvent(event))
                            {
                                if (event.type == sf::Event::Closed)
                                    Options.close();
                            }
                            float firedelay = 3.f - totalTime * 0.1;
                            if (firedelay < minimalfiredelay)
                                firedelay = minimalfiredelay;
                            if (fireDelta > firedelay)
                            {
                                adfire(fireobs, textures, -500 - 10 * totalTime, bIsDown);
                                fireDelta = 0;
                                bIsDown = !bIsDown;

                            }
                            fireDelta += deltaTime.asSeconds();

                            player.tick(deltaTime);
                            player.inputs(deltaTime);

                            for (Projectile& fireOb : fireobs)
                            {
                                fireOb.tick(deltaTime);


                                if (fireOb.iscollision(player))
                                {
                                    Options.close();
                                }
                                //draw
                                Options.clear();
                                background.draw(Options);
                                player.draw(Options);

                                for (Projectile& fireOb : fireobs)
                                    fireOb.draw(Options);

                            }

                            Options.display();
                        }
                        // Options.draw(Obackground);
                        //About.close();

                        Options.display();
                    }
                }

                if (x == 3)
                {
                    MENU.close();
                    break;
                }
            }
        }
        MENU.clear();
        MENU.draw(background);
        mainMenu.draw1(MENU);
        MENU.display();
    }
}
// sf::RenderWindow window_play(sf::VideoMode(WINDOW_SIZE_X, WINDOW_SIZE_Y), "SFML works!");


return 0;
}
```

Conclusion

En fin on a réalisé le projet qui nous permet à voir la réalité et l'utilisation de la programmation orienté objet avec le langage c++ . D'après la réalisation de ce projet on a constaté que le développement se base sur la recherche et la pratiques et de tomber dans mer d'erreurs et les corrigées en fin on a beaucoup apprendre la patience, la recherche, esprit d'équipe.

Bibliothèques

- https://www.sfml-dev.org/
- https://www.sfmldev.org/download/sfml/2.5.1/
- https://www.sfml-dev.org/tutorials/2.5/
- https://www.sfml-dev.org/learn.php