

RAPPORT DE PROJET



SFML

Réalisé par :

- Mohamed SERBOUT
- Basma EL BARKI
- Groupe : 2

Encadré par :

Pr. Ikram Ben abdel ouahab

Objective

Vieux générale de projet

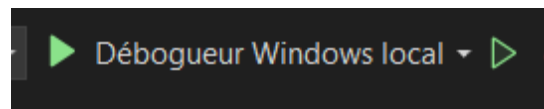
Dans ce projet on développer un jeu connue PICO_PARK à partir de logiciel sfml. L'objectif principal de ce projet est de maitriser la programmation orientée objet par la mise en place D'un jeu vidéo 2D, le jeu roller PICO_PARK, c'est un jeu qui a connu un grand succès dans les plateformes mobile.

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 Débogueur Windows local :

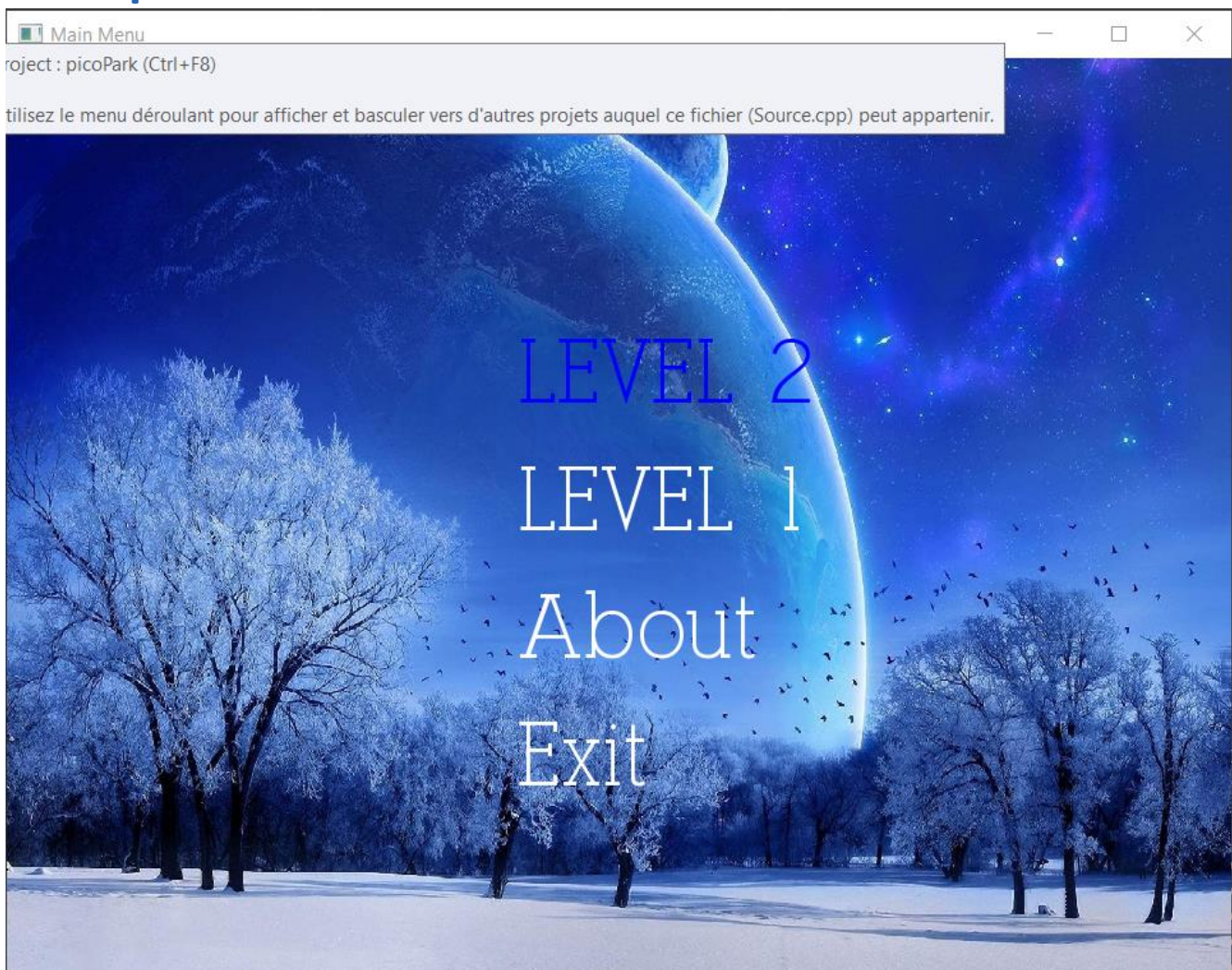


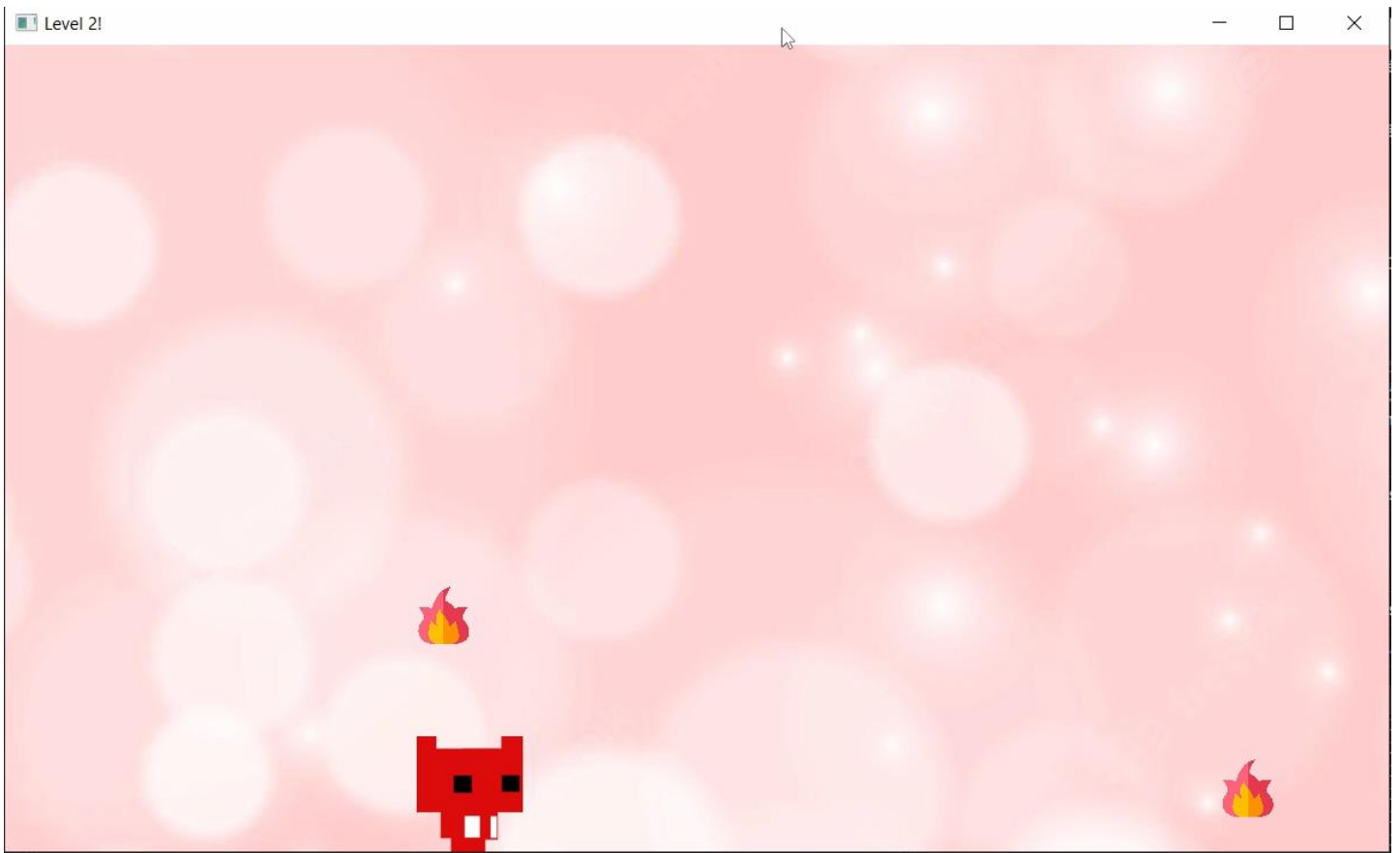
Alors vous affiche menu de jeux :

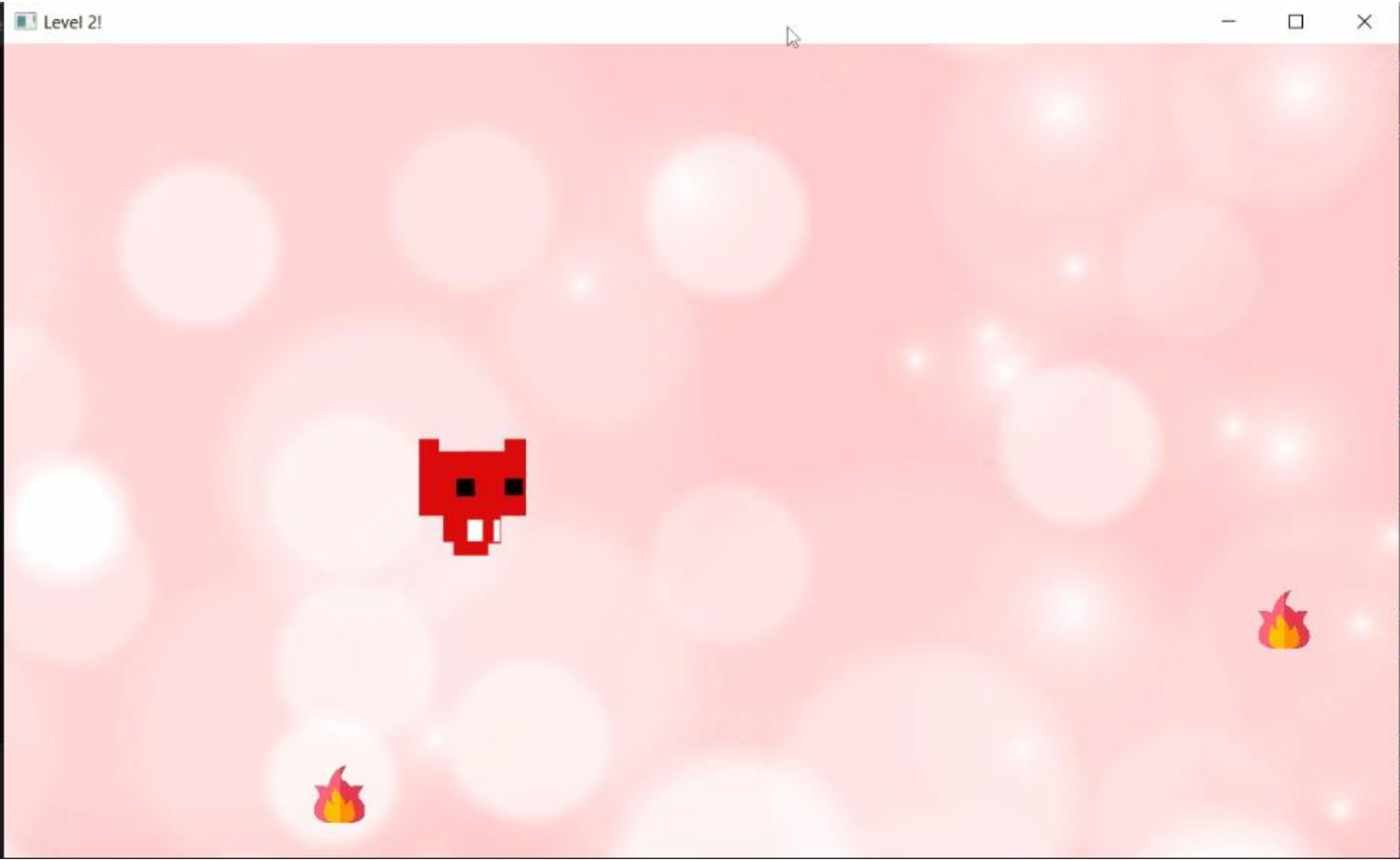


Et voilà maintenant vous devez choisir level 2 ou 1 ou quelqu'un avec clavier et clic sur entrer

Example : clic sur Level 2

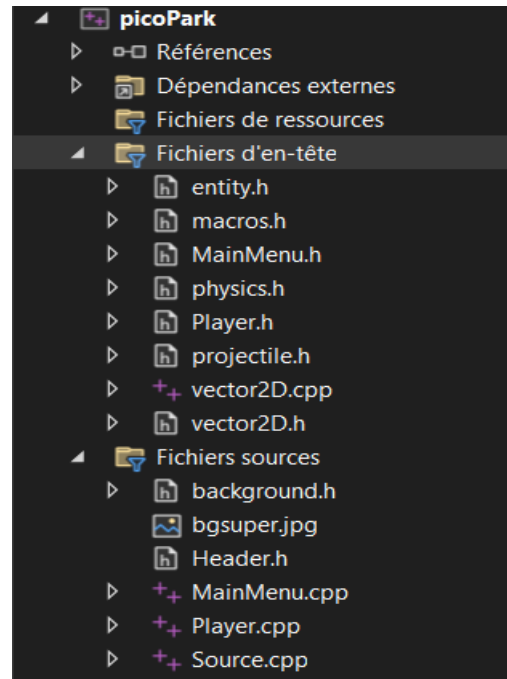






Code source :

Source.cpp :



```
Source.cpp  [picoPark] (Portée globale)
1  #include <SFML/Graphics.hpp>
2  #include <SFML/Audio.hpp>
3  #include "macros.h"
4  #include "Player.h"
5  #include "entity.h"
6  #include "projectile.h"
7  #include <vector>
8  #include "background.h"
9  #include "MainMenu.h"
10
11
12
13
14 void addfire(std::vector<Projectile>& fireobs, std::vector<sf::Texture>& textures, float speed, bool isfireDown = true) {
15     fireobs.emplace_back(Projectile());
16     Projectile& fireob = fireobs[fireobs.size() - 1];
17     fireob.speed = speed;
18     fireob.setSprite(textures[1]);
19     fireob.physics.location.x = WINDOW_SIZE_X;
20     if (isfireDown) {
21         fireob.physics.location.y = WINDOW_SIZE_Y - 80;
22     }
23     else {
24         fireob.physics.location.y = WINDOW_SIZE_Y - 230;
25     }
26 }
27
28
29 void adfire(std::vector<Projectile>& fireobs, std::vector<sf::Texture>& textures, float speed, bool isfireDown = true)
30 {
31     fireobs.emplace_back(Projectile());
32     Projectile& fireob = fireobs[fireobs.size() - 1];
33     fireob.speed = speed;
34     fireob.setSprite(textures[1]);
35     fireob.physics.location.x = WINDOW_SIZE_X;
36
37     fireob.physics.location.y = WINDOW_SIZE_Y - 100;
38
39 }
40
```

```

43 bool addBackgroundTexture(const std::string& filename, std::vector<sf::Texture>& textures)
44 {
45     {
46         textures.emplace_back(sf::Texture());
47         sf::Image curImage;
48         if (!curImage.loadFromFile(filename))
49             return false;
50         else {
51             constexpr unsigned int margin = 0.5;
52             constexpr float transition = 3;
53             for (unsigned int y = 0; y < curImage.getSize().y; y++)
54             {
55                 for (unsigned int x = 0; x < margin; x++)
56                 {
57                     sf::Color color = curImage.getPixel(x, y);
58                     unsigned int delta = float(margin - x) * transition;
59                     if (delta > 255)
60                         delta = 255;
61                     if (color.r < 255 - delta)
62                         color.r += delta;
63                     else
64                         color.r = 255;
65                     if (color.g < 255 - delta)
66                         color.g += delta;
67                     else
68                         color.g = 255;
69                     if (color.b < 255 - delta)
70                         color.b += delta;
71                     else
72                         color.b = 255;
73                     curImage.setPixel(x, y, color);
74                 }
75                 for (unsigned int x = curImage.getSize().x - margin; x < curImage.getSize().x; x++)
76                 {
77                     sf::Color color = curImage.getPixel(x, y);
78                     unsigned int delta = float(x - (curImage.getSize().x - margin)) * transition;
79                     if (x * 5 > 255)
80                         delta = 255;
81                     else
82                         delta = x * 5;
83                     if (color.r < 255 - delta)
84                         color.r += delta;
85                     else
86                         color.r = 255;
87                     if (color.g < 255 - delta)
88                         color.g += delta;
89                     else
90                         color.g = 255;
91                     if (color.b < 255 - delta)
92                         color.b += delta;
93                     else
94                         color.b = 255;
95                     curImage.setPixel(x, y, color);
96                 }
97             }
98             textures[textures.size() - 1].loadFromImage(curImage);
99         }
100     }
101 }
102
103
104
105
106
107
108
109
110
111
112
113
114

```

```

115 int main()
116 {
117     {
118         //make a main window
119         RenderWindow MENU(VideoMode(960, 720), "Main Menu", Style::Default);
120         MainMenu mainMenu(MENU.getSize().x, MENU.getSize().y);
121         // set BACKGROUND
122         RectangleShape background;
123         background.setSize(Vector2f(960, 720));
124         Texture MainTexture;
125         MainTexture.loadFromFile("media/winter2.jpg");
126         background.setTexture(&MainTexture);
127
128         // Photo to the game
129         RectangleShape Pbackground;
130         Pbackground.setSize(Vector2f(960, 720));
131         Texture back_texture;
132         back_texture.loadFromFile("media/padoruMenu.jpg");
133         Pbackground.setTexture(&back_texture);
134
135         // Photo to the Option
136         RectangleShape Obackground;
137         Obackground.setSize(Vector2f(960, 720));
138         Texture Optiontexture;
139         Optiontexture.loadFromFile("media/winter2.jpg");
140         Obackground.setTexture(&Optiontexture);
141
142         // Photo to the Option
143         /*RectangleShape ABbackground;
144         ABbackground.setSize(Vector2f(960, 720));
145         Texture Abouttexture;
146         Optiontexture.loadFromFile("media/winter3.jpg");
147         ABbackground.setTexture(&Abouttexture);*/
148
149         while (MENU.isOpen())
150         {
151             Event event;
152             while (MENU.pollEvent(event))
153             {
154                 if (event.type == Event::Closed)
155                 {
156                     MENU.close();
157                 }
158                 if (event.type == Event::KeyReleased)
159                 {
160                     if (event.key.code == Keyboard::Up)
161                     {
162                         mainMenu.MoveUp();
163                         break;
164                     }
165                     if (event.key.code == Keyboard::Down)
166                     {
167                         mainMenu.MoveDown();
168                         break;
169                     }
170                 }
171             }
172         }
173     }
174 }

```



```

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/player05-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, 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 < minalfiredelay)
        firedelay = minalfiredelay;
    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)

```

```

303 if (x == 1)
304 {
305     while (Options.isOpen())
306     {
307         Event aevent;
308         while (Options.pollEvent(aevent))
309         {
310             if (aevent.type == Event::Closed)
311             {
312                 Options.close();
313             }
314             if (aevent.type == Event::KeyPressed)
315             {
316                 if (aevent.key.code == Keyboard::Escape)
317                 {
318                     Options.close();
319                 }
320             }
321         }
322         Play.close();
323         Options.clear();
324
325         sf::Clock clock;
326         float totalTime = 0.f;
327
328         std::vector<sf::Texture> textures;
329         constexpr unsigned int nbTextures = 1;
330         textures.reserve(nbTextures);
331
332         textures.emplace_back(sf::Texture());
333         if (!textures[0].loadFromFile("media/player85-pp.png"))
334             return EXIT_FAILURE;
335
336         textures.emplace_back(sf::Texture());
337         if (!textures[1].loadFromFile("media/feu-1.png"))
338             return EXIT_FAILURE;
339
340         // sound
341         sf::Music music;
342         if (!music.openFromFile("media/padoru.wav"))
343             return -1;
344         music.play();
345         music.setLoop(true);
346
347         addBackgroundTexture("media/rezde.png", textures);
348         addBackgroundTexture("media/rezde.png", textures);
349
350         Background background;
351         background.addSprite(sf::Sprite(textures[2]));
352         background.addSprite(sf::Sprite(textures[3]));
353         Player player;
354         player.setSprite(textures[0]);
355
356         std::vector<Projectile> fireobs;
357         //addfire(fireobs, textures, true);
358         //addfire(fireobs, textures, false);
359         float fireDelta = 0.f;
360         bool bIsDown = true;
361         //addfire(fireobs, textures, true);
362

```

```

368 while (Options.isOpen())
369 {
370     sf::Time deltaTime = clock.restart();
371     totalTime += deltaTime.asSeconds();
372     //inputs
373     sf::Event event;
374     while (Options.pollEvent(event))
375     {
376         if (event.type == sf::Event::Closed)
377             Options.close();
378     }
379     float firedelay = 3.f - totalTime * 0.1;
380     if (firedelay < minimalfiredelay)
381         firedelay = minimalfiredelay;
382     if (fireDelta > firedelay)
383     {
384         addfire(fireobs, textures, -500 - 10 * totalTime, bIsDown);
385         fireDelta = 0;
386         bIsDown = !bIsDown;
387     }
388     fireDelta += deltaTime.asSeconds();
389
390     player.tick(deltaTime);
391     player.inputs(deltaTime);
392
393     for (Projectile& fireOb : fireobs)
394     {
395         fireOb.tick(deltaTime);
396
397         if (fireOb.iscollision(player))
398         {
399             Options.close();
400         }
401
402         //draw
403         Options.clear();
404         background.draw(Options);
405         player.draw(Options);
406
407         for (Projectile& fireOb : fireobs)
408             fireOb.draw(Options);
409
410     }
411     Options.display();
412
413     // Options.draw(0background);
414     //About.close();
415
416     Options.display();
417 }
418
419 if (x == 3)
420 {
421     MENU.close();
422     break;
423 }
424
425 MENU.clear();
426 MENU.draw(background);
427 mainMenu.draw1(MENU);
428 MENU.display();
429
430 }
431
432 // sf::RenderWindow window_play(sf::VideoMode(WINDOW_SIZE_X, WINDOW_SIZE_Y), "SFML works!");
433
434 return 0;
435

```

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>