RAPPORT DU PROJET

ALLIENS ATTACK

Préparé Par :

NADA BALKASSEM

YOUSSEF ESSLIMANI

LIEN GITHUB : https://github.com/Abderrahman-Rouas/Roller-splat.git

Encadré par :

Pr. EL AACHAK LOTFI

Pr. Ben Abdel ouahab Ikram

PLAN DE TRAVAIL :

- INTRODUCTION GENERAL
- DEFINITON DES ELEMENTS DU TRAVAIL
- le Menu
- Le premier niveau
- deuxième &troisième niveau
- Final scène
- CONCLUSION
- BIBLIOGRAPHIE

Introduction :

Tout d'abord nous devons connaître que veut dire cocos2dx ?

Cocos2d est un Framework libre en Python, permettant de développer des applications ou des jeux vidéo .

But de ce rapport :

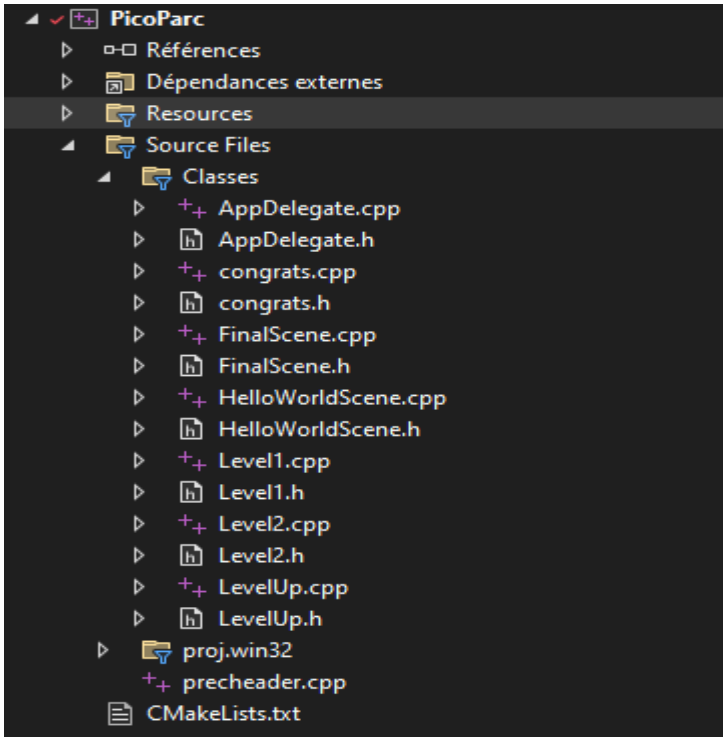
Nous souhaitons créer un jeu vidéo 2D nommée « Roller Splat », c'est un jeu qui a connu un grand succès dans les plateformes mobile.

NOTE :

Avant de passer au code, il est souhaitable de prendre un peu de recul et accorder

quelques instants à la compréhension de l'organisation de nos classes .

Apres la création de notre projet, on crée 12 classes.



HelloWordScene.cpp& HelloWordScene.h : Où on a créé notre menu qui va accueillir le joueur

level1.cpp&level1.h : Où on a développé le premier niveau de notre jeu

level2.cpp&level2.h: Où on a développé le deuxième niveau de notre jeu

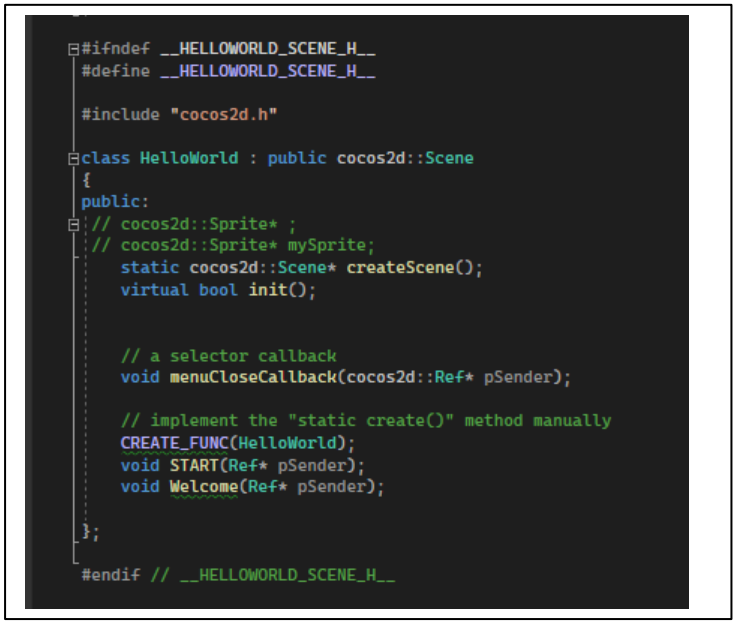
levelUp.cpp&levelUp.h: Où on a développé la scène qui félicite le joueur apres terminer le premier niveau

final.cpp&final.h : Où on a créé la dernière scène qui félicite le joueur

Premièrement : le Menu.

a-HelloWordScene.h

dans le header on a créé une classe HelloWord dans cette dernière nous allons définir toutes les fonctions de la classe principale .



```
#ifndef __HELLOWORLD_SCENE_H__
#define __HELLOWORLD_SCENE_H__

#include "cocos2d.h"

class HelloWorld : public cocos2d::Scene
{
public:
// cocos2d::Sprite* ;
// cocos2d::Sprite* mySprite;
    static cocos2d::Scene* createScene();
    virtual bool init();


    // a selector callback
    void menuCloseCallback(cocos2d::Ref* pSender);

    // implement the "static create()" method manually
    CREATE_FUNC(HelloWorld);
    void START(Ref* pSender);
    void Welcome(Ref* pSender);

};

#endif // __HELLOWORLD_SCENE_H__
```

b- HelloWordScene.cpp :

Dans la fonction init nous avons créé une sprite en utilisant « GIMP » qui va être le background de notre menu puis nous lui avons donné une position dans notre scène , Après nous avons créé une variable dont la quelle nous allons attribuer un item de menu nommé START, et autre item nommé QUIT et nous l'avons positionné par setPosition et ajouter à notre menu par addChild .

```cpp
Scene* HelloWorld::createScene()
{
    return HelloWorld::create();
}



// on "init" you need to initialize your instance
bool HelloWorld::init()
{
    //////////////////////////////
    // 1. super init first
    if (!Scene::init())
    {
        return false;
    }

    auto visibleSize = Director::getInstance()->getVisibleSize();
    Vec2 origin = Director::getInstance()->getVisibleOrigin();

    //-------------menu-background-------------------------//

    auto bg = Sprite::create("bg.jpg");
    bg->setPosition(Vec2(visibleSize.width / 2 + origin.x, visibleSize.height / 2 + origin.y));
    this->addChild(bg, -6);
    //---------------------start-button-----------------//
    auto menu_item_1 = MenuItemFont::create("START", CC_CALLBACK_1(HelloWorld::START, this));
    menu_item_1->setPosition(Vec2(visibleSize.width / 2 , (visibleSize.height / 7) + origin.y));
    //----------------------quit-button-------//
    auto menu_item_2 = MenuItemFont::create("QUIT", CC_CALLBACK_1(HelloWorld::menuCloseCallback, this));
    menu_item_2->setPosition(Vec2(visibleSize.width / 2 , (visibleSize.height / 12) + origin.y));

    auto * menu = Menu::create(menu_item_1, menu_item_2, NULL);
    menu->setPosition(Vec2(0, 0));
    this->addChild(menu);
```

Dehors de la fonction init nous faisons appel à la fonction START qui a été définie dans le header de cette scène, cette fonction va lier l'item START avec le premier niveau du jeu .

```cpp
void HelloWorld::START(cocos2d::Ref* pSender) {
    CCLOG("START");
    auto visibleSize = Director::getInstance()->getVisibleSize();
    Vec2 origin = Director::getInstance()->getVisibleOrigin();
    // Create a label to display the countdown
    auto label = Label::createWithTTF("3", "fonts/arial.ttf", 45);
    label->setPosition(Vec2(visibleSize.width / 2, visibleSize.height + origin.y - 50));
    this->addChild(label);

    // Run a sequence of actions to countdown from 3 to 1
    label->runAction(Sequence::create(
        CallFunc::create([label]() { label->setString("3"); }),
        DelayTime::create(1.0),
        CallFunc::create([label]() { label->setString("2"); }),
        DelayTime::create(1.0),
        CallFunc::create([label]() { label->setString("1"); }),
        DelayTime::create(1.0),
        CallFunc::create([label]() { label->setString("Let's Go!"); }),
        DelayTime::create(1.0),
        CallFunc::create([this]() {
            // Push the main menu scene onto the scene stack when the countdown is finished
            auto scene = MainMenu::createScene();
            Director::getInstance()->pushScene(scene);
            }),
        nullptr
```

Enfin on obtient ce résultat :

Deuxièmement : le premier niveau.

a-level1.h :

Dans la partie header du niveau 1 nous avons créé une classe MainMenu dont

laquelle nous allons définir toutes les classes du premier niveau du jeu .

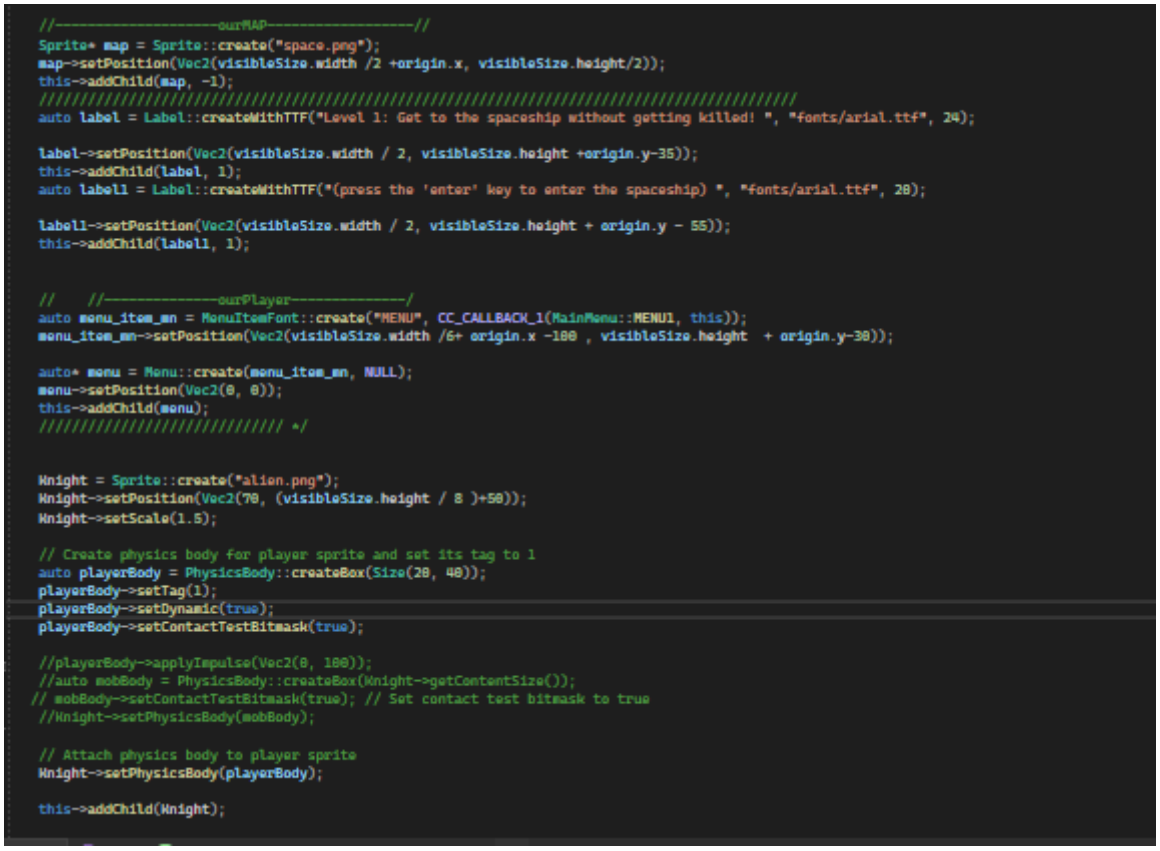
```
class MainMenu : public cocos2d::Scene
{
public:

    static cocos2d::Scene* createScene();
    virtual bool init();

    // a selector callback
    void menuCloseCallback(cocos2d::Ref* pSender);
    void menuCloseCallback(std::string message);
    void update1(float delta);
    int DirX = 0;
    int DirY = 0;
    cocos2d::Sprite* Knight;
    void update(float dt);

    // implement the "static create()" method manually
    CREATE_FUNC(MainMenu);
    void MENU1(Ref* pSender);

    //cocos2d::Sprite* mySprite;
private:
    cocos2d::PhysicsWorld* sceneWorld;


    void SetPhysicsWorld(cocos2d::PhysicsWorld* world) { sceneWorld = world; };

    bool onContactBegin(cocos2d::PhysicsContact& contact);
    bool onContactBegin1(cocos2d::PhysicsContact& contact);
};
```

b-level1.cpp :

Dans la fonction init nous avons créé une sprite (space.png) qui va être le background de notre stage

Nous avons créé une deuxième sprite qui est notre alien et nous l'avons affecté à une variable nommée Knight après nous lui avons donné une position dans la scène de notre premier stage.

Et aussi nous avons créé une variable nommée menu_item_mn pour créer un item du menu nommé Menu, son rôle est de permettre de revenir à la page scène principale du jeu, nous l'avons positionné dans notre MAP .



```
//--------------------ourMAP-----------------//
Sprite* map = Sprite::create("space.png");
map->setPosition(Vec2(visibleSize.width /2 +origin.x, visibleSize.height/2));
this->addChild(map, -1);
////////////////////////////////////////////////////////////////////////////////
auto label = Label::createWithTTF("Level 1: Get to the spaceship without getting killed! ", "fonts/arial.ttf", 24);

label->setPosition(Vec2(visibleSize.width / 2, visibleSize.height +origin.y-35));
this->addChild(label, 1);
auto label1 = Label::createWithTTF("(press the 'enter' key to enter the spaceship) ", "fonts/arial.ttf", 20);

label1->setPosition(Vec2(visibleSize.width / 2, visibleSize.height + origin.y - 55));
this->addChild(label1, 1);


//    //--------------ourPlayer--------------/
auto menu_item_mn = MenuItemFont::create("MENU", CC_CALLBACK_1(MainMenu::MENU1, this));
menu_item_mn->setPosition(Vec2(visibleSize.width /6+ origin.x -100 , visibleSize.height  + origin.y-30));

auto* menu = Menu::create(menu_item_mn, NULL);
menu->setPosition(Vec2(0, 0));
this->addChild(menu);
//////////////////////////// */


Knight = Sprite::create("alien.png");
Knight->setPosition(Vec2(70, (visibleSize.height / 8 )+50));
Knight->setScale(1.5);

// Create physics body for player sprite and set its tag to 1
auto playerBody = PhysicsBody::createBox(Size(20, 40));
playerBody->setTag(1);
playerBody->setDynamic(true);
playerBody->setContactTestBitmask(true);

//playerBody->applyImpulse(Vec2(0, 100));
//auto mobBody = PhysicsBody::createBox(Knight->getContentSize());
// mobBody->setContactTestBitmask(true); // Set contact test bitmask to true
//Knight->setPhysicsBody(mobBody);

// Attach physics body to player sprite
Knight->setPhysicsBody(playerBody);

this->addChild(Knight);
```

Dans cette partie nous avons créé une variable nommée menu_item_1 pour créer un item du menu nommé Menu, son rôle est de permettre de revenir à la page scène principale du jeu, nous l'avons positionné dans notre MAP

Dehors de la fonction init nous avons créé une fonction appelée menu qui va lier par une clique item Menu avec le menu de notre jeu .

```
void MainMenu::MENU1(cocos2d::Ref* pSender) {
    CCLOG("MENU");
    auto scene = HelloWorld::createScene();
    Director::getInstance()->replaceScene(scene);
}
```

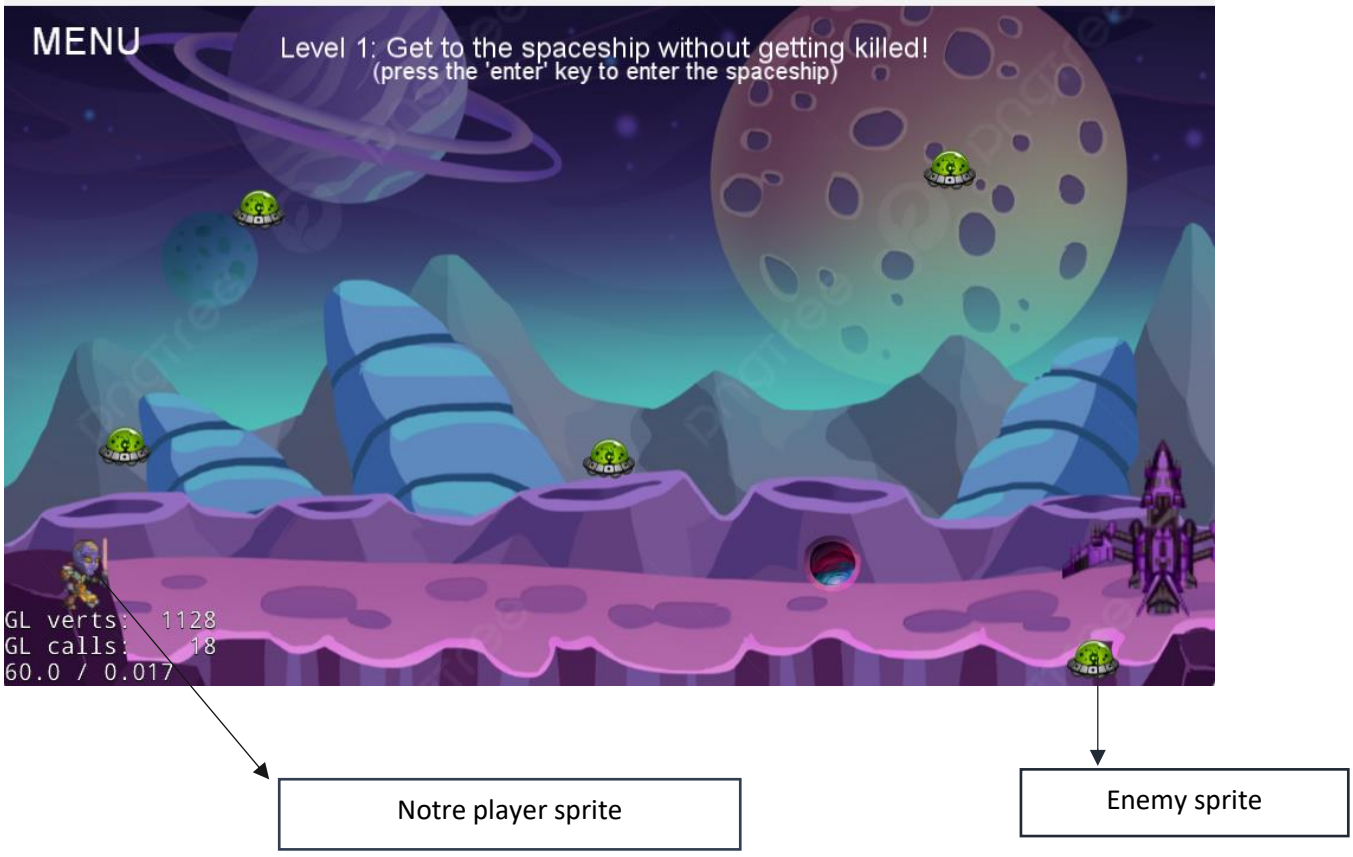
et aussi nous avons créé des autres players qui vont etre les enemies de notre player

```
{
    auto Enemy = Sprite::create("enemy.png");
    Enemy->setPosition(Vec2(origin.x + visibleSize.width / 2, origin.y + visibleSize.height));
    Enemy->setScale(1);

    // Create physics body for enemy sprite and set its tag to 2
    auto enemyBody = PhysicsBody::createBox(Size(40, 30));

    enemyBody->setContactTestBitmask(true);
    enemyBody->setDynamic(false);
    enemyBody->setTag(2);

    // Attach physics body to enemy sprite
    Enemy->setPhysicsBody(enemyBody);

    // Add enemy sprite to the scene
    this->addChild(Enemy);

    // Create the move action that moves the character from top to bottom
    auto moveToBottom = MoveTo::create(5, Vec2(origin.x + visibleSize.width / 2, origin.y+100)); // Move to bottom of the screen in 8 seconds
    auto moveToTop = MoveTo::create(5, Vec2(origin.x + visibleSize.width / 2, origin.y+200));
    // Create a sequence of actions that includes the move action
    auto sequence = Sequence::create(moveToBottom, moveToTop ,nullptr);

    // Create a repeat forever action that repeats the sequence of actions indefinitely
    auto repeatForever = RepeatForever::create(sequence);
    // Run actions on enemy sprite
    Enemy->runAction(repeatForever);
}
```

et un sprite nommé « spaceship » qui sera la solution pour compléter le niveau 1 lorsque notre joueur l'atteindra sans se faire tuer

```
    {
        auto spaceship = Sprite::create("spaceship.png");
        spaceship->setPosition(Vec2(980, 135));
        auto spaceshipBody = PhysicsBody::createBox(Size(20, 70));

        spaceship->setScale(1.7);
        // Create physics body for enemy sprite and set its tag to 2

        spaceshipBody->setTag(3);
        spaceshipBody->setDynamic(false);
        spaceshipBody->setContactTestBitmask(true);
        // Attach physics body to enemy sprite
        spaceship->setPhysicsBody(spaceshipBody);

        // Add enemy sprite to the scene
        this->addChild(spaceship);


    }
```

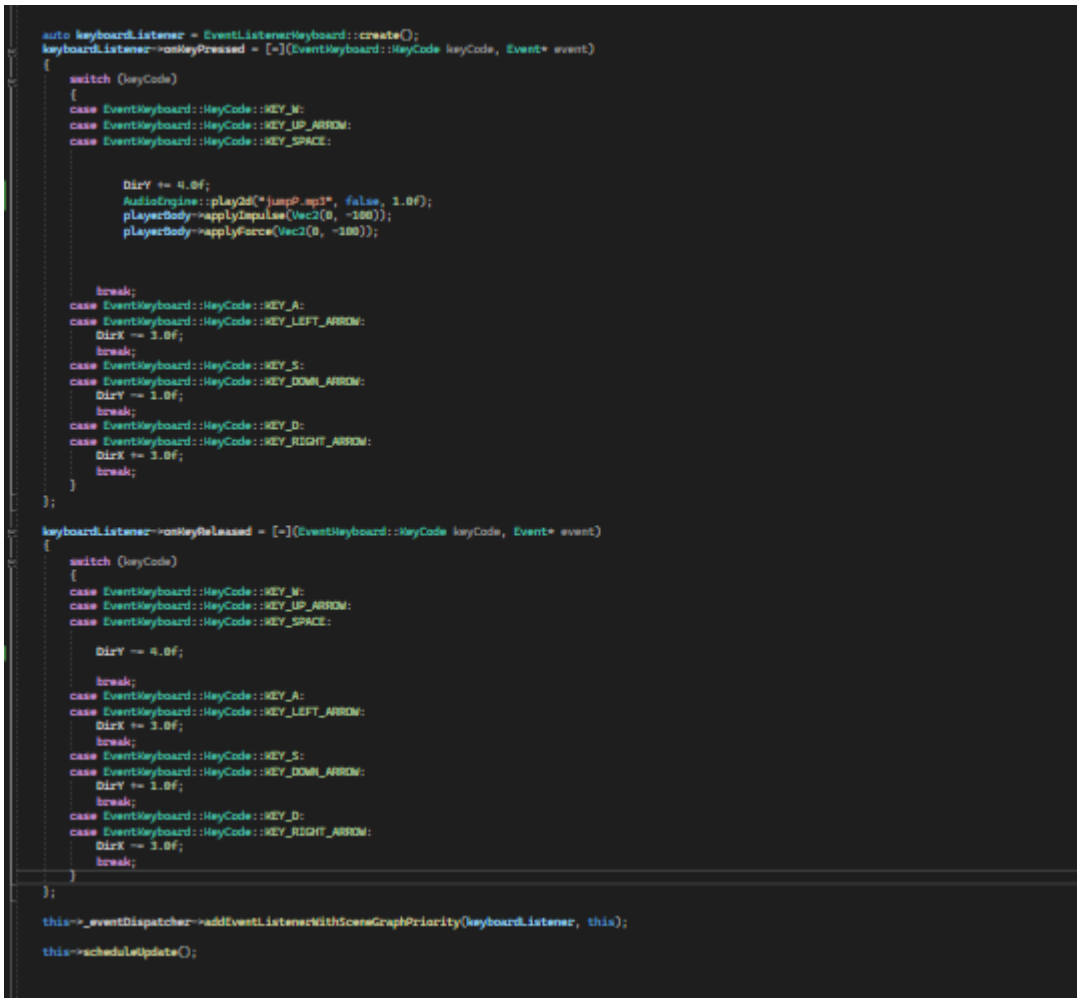
Voilà le résultat :



Dans cette partie, on va parler sur le mouvement de Player,. Pour manipuler et le déplacer dans le map, nous avons utilisé des classes et des fonctions déjà définie,

Cocos2d-x prend en charge les événements clavier. La classe EventListenerKeyboard Permet de fais ça.

Voilà le code source :



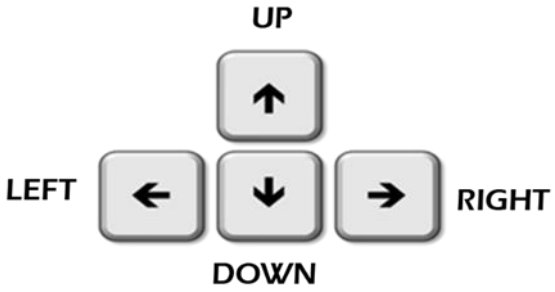
```
auto keyboardListener = EventListenerKeyboard::create();
keyboardListener->onKeyPressed = [=](EventKeyboard::KeyCode keyCode, Event* event)
{
    switch (keyCode)
    {
    case EventKeyboard::KeyCode::KEY_W:
    case EventKeyboard::KeyCode::KEY_UP_ARROW:
    case EventKeyboard::KeyCode::KEY_SPACE:


            DirY += 4.0f;
            AudioEngine::play2d("jumpP.mp3", false, 1.0f);
            playerBody->applyImpulse(Vec2(0, -100));
            playerBody->applyForce(Vec2(0, -100));


        break;
    case EventKeyboard::KeyCode::KEY_A:
    case EventKeyboard::KeyCode::KEY_LEFT_ARROW:
        DirX -= 3.0f;
        break;
    case EventKeyboard::KeyCode::KEY_S:
    case EventKeyboard::KeyCode::KEY_DOWN_ARROW:
        DirY -= 1.0f;
        break;
    case EventKeyboard::KeyCode::KEY_D:
    case EventKeyboard::KeyCode::KEY_RIGHT_ARROW:
        DirX += 3.0f;
        break;
    }
};

keyboardListener->onKeyReleased = [=](EventKeyboard::KeyCode keyCode, Event* event)
{
    switch (keyCode)
    {
    case EventKeyboard::KeyCode::KEY_W:
    case EventKeyboard::KeyCode::KEY_UP_ARROW:
    case EventKeyboard::KeyCode::KEY_SPACE:

        DirY -= 4.0f;

        break;
    case EventKeyboard::KeyCode::KEY_A:
    case EventKeyboard::KeyCode::KEY_LEFT_ARROW:
        DirX += 3.0f;
        break;
    case EventKeyboard::KeyCode::KEY_S:
    case EventKeyboard::KeyCode::KEY_DOWN_ARROW:
        DirY += 1.0f;
        break;
    case EventKeyboard::KeyCode::KEY_D:
    case EventKeyboard::KeyCode::KEY_RIGHT_ARROW:
        DirX -= 3.0f;
        break;
    }
};

this->_eventDispatcher->addEventListenerWithSceneGraphPriority(keyboardListener, this);

this->scheduleUpdate();
```

Nous avons créé un EventListener, dans ce cas un EventListenerKeyboard, implémente le gestionnaire d'événement onKeyPressed. Le premier paramètre transmis est l'énumération EventKeyboard::KeyCode, est une valeur qui représente la touche qui a été enfoncée.

La deuxième valeur était l’event de l'événement. C’est notre sprite. Nous utilisons le pointeur d'événement

pour obtenir le nœud event, et mettre à jour sa position dans une direction en fonction de la touche enfoncée .

Nous avons utilisé le switch avec 4 case , chaque case représente une touche directionnelle de clavier :



La position initiale donné à la balle est (350,170) , Pour le mouvement du balle de sa position initiale , il y'en a une seul possibilité c'est de la déplacer vers la position (135,170) en cliquant sur la touche LEFT.

Nous avons utilisé un "switch" pour tester la valeur de la touche enfoncée et effectue des actions en conséquence. Par exemple, si la touche "W", "flèche haut" ou "espace" est enfoncée, une variable "DirY" est augmentée de 4, un son est joué, et une force et une impulsion sont appliquées au corps du joueur.

De manière similaire, lorsqu'une touche est relâchée, une autre fonction lambda est appelée et effectue des actions en conséquence.

Enfin, l'écouteur d'événements du clavier est ajouté à l'objet "EventDispatcher" du "Node" courant, et la fonction scheduleUpdate est appelée sur ce "Node". Cela signifie que la fonction "update" (par défaut vide) sera appelée régulièrement sur ce "Node"

Les passages entre les Scènes :

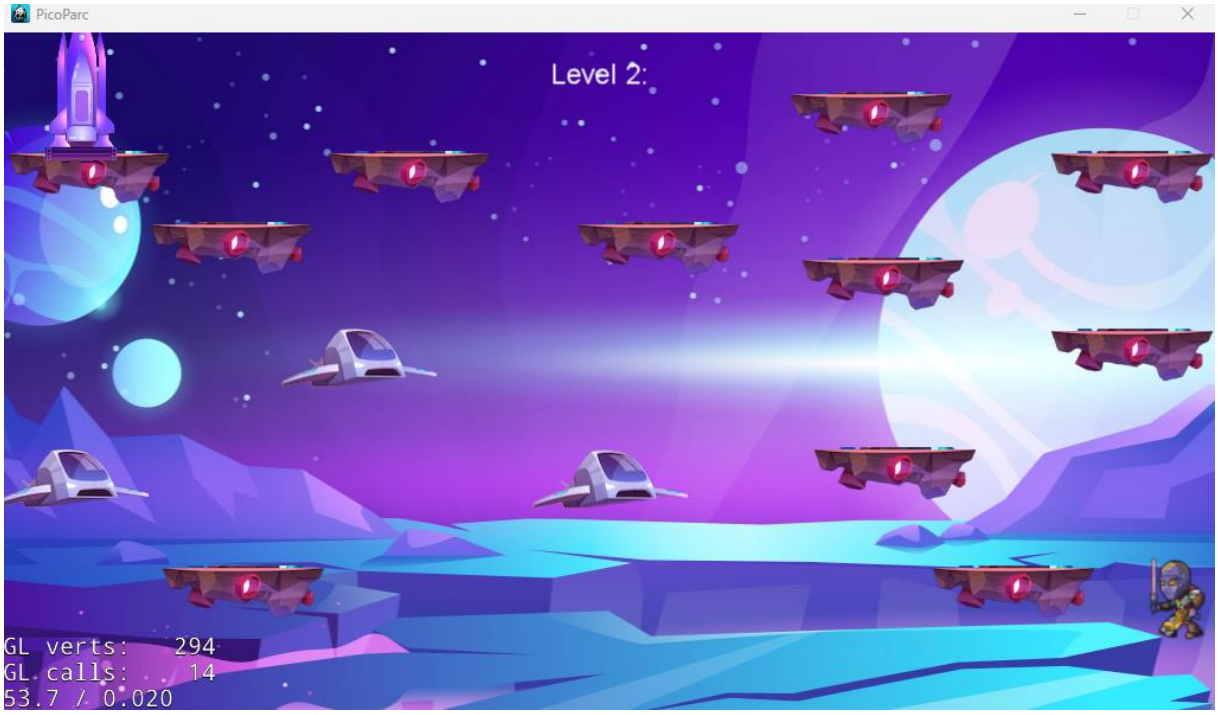
Pour passe de level1 vers level2 , nous avons utilisé les propriétés de class DIRECTOR , Le Director contrôle tous les aspects de votre jeu. Ce qui est affiché à l'écran . Donc on va changer level1 par level2 , mais après du vérification de condition . atteindre le vaisseau spatial

```
if ((bodyA->getTag() == 1 && bodyB->getTag() == 2) ||
    (bodyA->getTag() == 2 && bodyB->getTag() == 1))
{
    // Collision between player and enemy
    // Player takes damage

   AudioEngine::play2d("stop.mp3", false, 1.0f);

    auto visibleSize = Director::getInstance()->getVisibleSize();
    Vec2 origin = Director::getInstance()->getVisibleOrigin();//variable is not in scope or has not been declared.
    // Create a message label
    auto label = Label::createWithTTF("You failed!", "fonts/arial.ttf", 40);
    label->setPosition(Vec2(visibleSize.width / 2, visibleSize.height / 2));
    this->addChild(label, 10);

    // Pause the game for 2 seconds
    auto delay = DelayTime::create(1.0f);

    // Replace the scene
    auto replace = CallFunc::create([&]() {
        auto scene = FinalScene::createScene();
        Director::getInstance()->replaceScene(scene);
        });
```

Troisièmement : le deuxième niveau .

Suivant les mêmes étapes et les mêmes fonctions précédentes nous allons tout simplement changer les maps et les positions de notre boule.

Voilà le résultat :



Quatrièmement : final scène & gameOver scène

a-congrats.h& congrats.cpp

Dans la partie header de la dernière scène nous avons créé une classe FINAL dont laquelle nous allons définir toutes les classes de la dernière scène

```h
#ifndef __CONGRATS_H__
#define __CONGRATS_H__

#include "cocos2d.h"

class Congrats : public cocos2d::Scene
{
public:

    static cocos2d::Scene* createScene();
    virtual bool init();


    // a selector callback
    void menuCloseCallback(cocos2d::Ref* pSender);

    // implement the "static create()" method manually
    CREATE_FUNC(Congrats);
    void MENU(Ref* pSender);
};

#endif // __FINAL_SCENE_H__
```

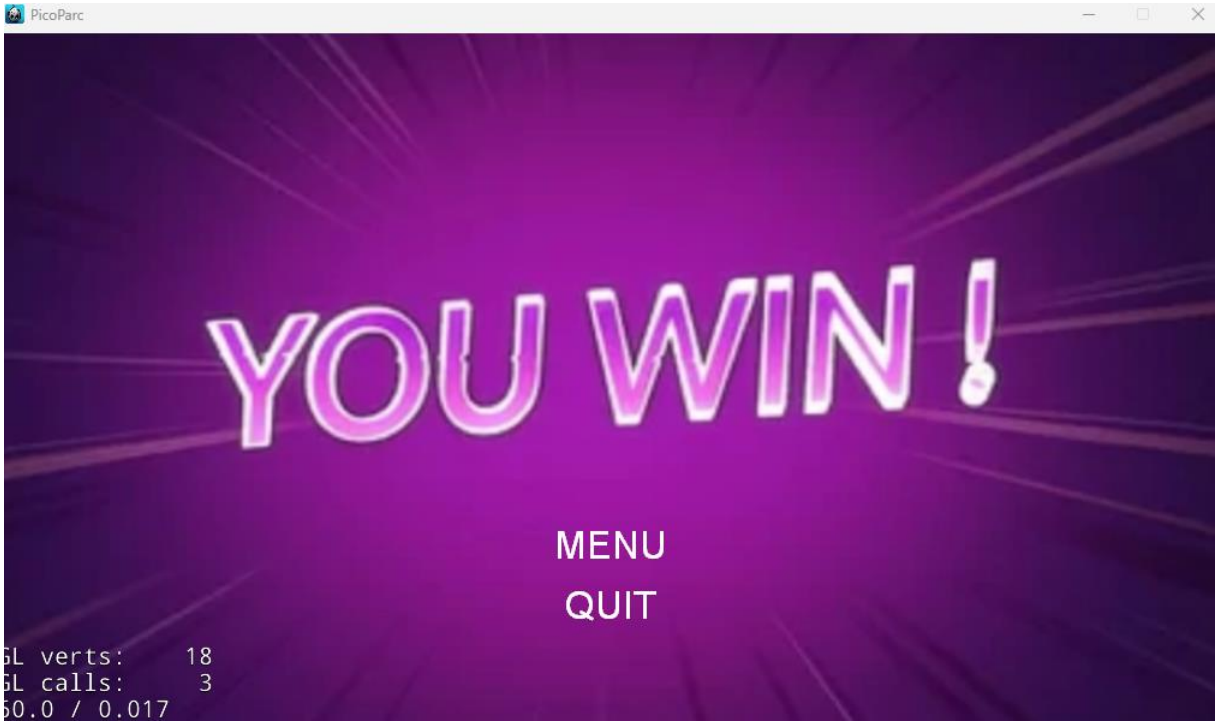
dans la fonction init de la dernier scène nous avons créé un Sprite qui va être le background de cette scène puis nous avons créé une variable pour créer un item du menu nommé Menu, son rôle est de permettre de revenir à la page du scène principale du jeu, nous l'avons positionné dans notre background

```cpp
    if ( !Scene::init() )
    {
        return false;
    }

    auto visibleSize = Director::getInstance()->getVisibleSize();
    Vec2 origin = Director::getInstance()->getVisibleOrigin();

    ////////////////////////////


    auto g = Sprite::create("uwin.png");
    g->setPosition(Vec2(visibleSize.width / 2 + origin.x, visibleSize.height / 2 + origin.y));
    this->addChild(g, -7);
    //----------------------start-button-----------------//
    auto menu_item_go = MenuItemFont::create("MENU", CC_CALLBACK_1(Congrats::MENU, this));
    menu_item_go->setFontName("Futura");
    menu_item_go->setPosition(Vec2(visibleSize.width / 2,150));

    auto menu_item_2 = MenuItemFont::create("QUIT", CC_CALLBACK_1(Congrats::menuCloseCallback, this));
    menu_item_2->setPosition(Vec2(visibleSize.width / 2,100));


    auto* menu1 = Menu::create(menu_item_go, menu_item_2, NULL);
    menu1->setPosition(Vec2(0, 0));
    this->addChild(menu1,200);

    return true;
}
```

Dehors de la fonction init nous avons créé une fonction appelée menu qui va lier item Menu avec le menu de notre jeu par une clique ceux qui va permettre de rejouer le jeu

```
void Congrats::MENU(cocos2d::Ref* pSender) {
    //CCLOG("RETRY");

          auto scene = HelloWorld::createScene();
            Director::getInstance()->pushScene(scene);

}
void Congrats::menuCloseCallback(cocos2d::Ref* pSender) {


    Director::getInstance()->end();


}
```

Enfin on obtient ce résultat :



b-finalScene.h&finalScene.cpp

Dans la partie header de la dernière scène nous avons créé une classe dont laquelle nous allons définir toutes les classes de la GameOver scène

```
#ifndef __FINAL_SCENE_H__
#define __FINAL_SCENE_H__

#include "cocos2d.h"

class FinalScene : public cocos2d::Scene
{
public:

    static cocos2d::Scene* createScene();
    virtual bool init();


    // a selector callback
    void menuCloseCallback(cocos2d::Ref* pSender);

    // implement the "static create()" method manually
    CREATE_FUNC(FinalScene);
    void RETRY(Ref* pSender);
};
```

dans la fonction init de GameOver scène nous avons créé un Sprite qui va être le background de cette scène puis nous avons créé une variable pour créer un item du menu nommé RETRY, son rôle est de permettre de revenir à la page du scène principale du level1 , nous l'avons positionné dans notre background

```
    auto g = Sprite::create("GameOver.png");
    g->setPosition(Vec2(visibleSize.width / 2 + origin.x, visibleSize.height / 2 + origin.y));
    this->addChild(g, -7);
    //----------------------start-button-----------------//
    auto menu_item_go = MenuItemFont::create("RETRY", CC_CALLBACK_1(FinalScene::RETRY, this));
    menu_item_go->setFontName("Futura");

    menu_item_go->setPosition(Vec2(visibleSize.width / 2 +origin.x-10, (visibleSize.height / 8) +origin.y));
    //----------------------quit-button-------//
    //auto menu_item_go1 = MenuItemImage::create("closeB.png", "closeB.png", CC_CALLBACK_1(FinalScene::menuCloseCallback, this));
    //menu_item_go1->setPosition(Vec2(visibleSize.width / 2, (visibleSize.height / 8) * 2));


    auto* menu1 = Menu::create(menu_item_go, NULL);
    menu1->setPosition(Vec2(0, 0));
    this->addChild(menu1);




    return true;
}
```

Enfin on obtient ce résultat :

CONCLUSION GENERAL

Tout au long de la préparation de notre jeu, nous avons essayé de pratiquer les connaissances requises durant notre cours du programmation orienté objet, et aussi notre connaissance sur ce qu'on a trouvé concernant cocos2d.

L'objectif c'est de concevoir et programmer un jeu, il nous a donné la possibilité de maitriser et découvrir une nouvelle approche de la programmation.

BIBLIOGRAPHIE

https://www.cocos.com/en/

https://docs.cocos.com/cocos2d-x/manual/en/

https://github.com/cocos2d/cocos2d-x