

# RAPPORT DU PROJET FINAL

**POO**

Préparé Par

- SOULAIMAN ZAHOUANI
- HANAE EL M'RABET

encadré par :

MONSIEUR LOTFI EL AACHAK

# RS

## Roller Splat

cocos 2D



# PLAN DE TRAVAIL

INTRODUCTION GENERAL

DEVELOPEMENT

1. DEFINITION DES ELEMENTS DU TRAVAIL
2. METHODE DU TRAVAIL

CONCLUSION

BIBLIOGRAPHIE



PROGRAMMATION



# INTRODUCTION GENERAL

La programmation orienté objet est un paradigme au sein de la programmation informatique. Il s'agit d'une représentation des choses, un modèle cohérent partager à travers différents langages qui permettent leur usage.

Le c++ offre un mécanisme de classe rassemblant données et traitement, sont principe c'est de créer et faire interagir des briques logicielles que l'ont appelé objet, et l'objet représente une idée un concept ou on entité du monde physique.

**Cocos2d** est un framework libre en Python, permettant de développer des applications ou des jeux vidéo. Des jeux comme FarmVille, Geometry Dash ou Angry Birds Fight! ont été développés avec **Cocos2D**.





# But

L'objectif principal de ce projet est de maîtriser la programmation orientée objet par la mise en place d'un jeu vidéo 2D, le jeu proposé s'appelle roller Splat, c'est un jeu qui a connu un grand succès dans les plateformes mobile.





# Développement

## INSTALLATION DU COCOS 2D et Visual Studio

On a installé cocos2d-x version 4-0 du site [cocos2d-x.org](https://cocos2d-x.org), et python version 2.7.16 du [python.org](https://python.org).

L'installation du python est nécessaire pour le fonctionnement du cocos2d. Et on a installé aussi

**CMake**, est un système de construction logicielle multiplateforme qui permet cocos de créer plusieurs projets dans des différentes plateforme.

Après l'installation, on va ouvrir l'invite de commande et on tape plusieurs commandes :

`C:\Users\Hanae Elm'rabet\Desktop>python` : vérifie si python est bien installé

`C:\Users\Hanae Elm'rabet\Desktop>cmake` : aussi vérifie l'installation du cmake

`C:\Users\Hanae Elm'rabet\Desktop>cocos new Game -l cpp -p com.sonarsystems.game`: cette commande crée le dossier cocos avec le nom indiqué après new

`C:\Users\Hanae Elm'rabet\Desktop\game>cmake .. -G "VisualStudio 17-2022" -A Win32`: Pour le fonctionnement de proj.win, tous les fichiers de cocos et visual studio

## Les éléments de notre projets



L'objectif de Roller Splat est **de** recouvrir le sol d'un labyrinthe avec **de** la peinture dans chaque niveau amusant et captivant.

On a programmé ce jeux 2d en utilisant la programmation orienté objet.

D'abord on a utilisé les classes qui permettent de représenter des structure complexes dans un langage de programmation, qui a 2 composants (méthodes / attributs).

### **APPDELEGATE :**

Le délégué d'application est un objet qui reçoit des notifications lorsque l'objet UIApplication atteint certain état.

 AppDelegate  
 AppDelegate.h



- AppDelegate.h : son rôle c'est de définir les méthodes et les fonctions concernant cocos.
- AppDelegate.cpp : chargé de définir la taille, couleur, structure, Background .... De l'élément cocos

### **HELLOWORDSCENE :**

 HelloWorldScene  
 HelloWorldScene.h

- Hellowordscene.h : Pour la déclaration des fonctions qui crée les scènes
- Hellowordscene.cpp : le déplacement des éléments du menu

### **LEVELSCENE :**

 LevelScene  
 LevelScene.h



Responsable de créer le menu du les niveaux, et les scènes de chaque menu.

On a utiliser aussi plusieurs d'autre classe qu'on va les explique après .

Commençant par **helloworldscene**

On va déclarer les fonctions qui va nous créer des scènes, ces dernières c'est pour partir d'une page à l'autre .

On va déclarer d'abord le menu avec ces éléments dans **helloworldscene.h**

```
void menuCloseCallback(cocos2d::Ref* pSender);

// implement the "static create()" method manually
CREATE_FUNC(HelloWorld);

cocos2d::Sprite* mySprite;

/* cocos2d::Menu* menu2; */
void Play(Ref* pSender);
void Settings(Ref* pSender);
void Help(Ref* pSender);
```

et après dans **cpp** on va afficher notre menu et déclarer des fonctions pour indiquer les éléments de ce dernier. Ainsi que déclarer les vecteurs qui sont responsable du déplacement des éléments du menu .

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

// Print useful error message instead of segfaulting when files are not there.
static void problemLoading(const char* filename)
{
    printf("Error while loading: %s\n", filename);
    printf("Depending on how you compiled you might have to add 'Resources/'
in front of filenames in HelloWorldScene.cpp\n");
}
```



```

}

// 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();

    auto mySprite = Sprite::create("back ground2.png");

```

Et aussi on va remplir les éléments du menu avec leur sous menu.

```

    auto menu_item_1 = MenuItemImage::create("play.png", "play2.png",
CC_CALLBACK_1(HelloWorld::Play, this));
    auto menu_item_2 = MenuItemImage::create("settings.png", "settings2.png",
CC_CALLBACK_1(HelloWorld::Settings, this));
    auto menu_item_3 = MenuItemImage::create("help.png", "help2.png",
CC_CALLBACK_1(HelloWorld::Help, this));
    auto* menu = Menu::create(menu_item_1, menu_item_2, menu_item_3, NULL);
    /*menu->setPosition(Point(0, 0));*/
    menu->alignItemsVertically();
    this->addChild(menu);

    return true;
}

```

Dans **levelscene.h**, on va lié cocos avec notre scène :

```

#include "cocos2d.h"

```

Et après on va crée un class hérité par cocos 2d contient une scène qui est une fonction de cocos :

```

class LevelScene : public cocos2d::Scene
{

```



Une même classe héritée va créer d'autre scène :

```
public:
    static cocos2d::Scene* createScene();
```

et on va créer aussi les nouveau menu qui vont être ouverte après cliquer sur Play.

```
void menuCloseCallback(cocos2d::Ref* pSender);
```

et dans **levelscene.cpp** on va déclarer les fonction utiles

```
#include "LevelScene.h"
#include "HelloWorldScene.h"
#include "level1.h"
#include "Level2.h"
#include "Level3.h"
```

Après créer une scène nommé levelscène qui va être enregistrer, déclarer une fonction qui va afficher si j'ai un erreur dans la création :

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

// Print useful error message instead of segfaulting when files are not there.
static void problemLoading(const char* filename)
{
    printf("Error while loading: %s\n", filename);
    printf("Depending on how you compiled you might have to add 'Resources/'
in front of filenames in HelloWorldScene.cpp\n");
}
```



On va aussi créer des sous menus , chaque sous menus contient un sous menu qui vont être sous forme de sprite . finalement créer une fonction qui nous aide si j'ai choisi un level il va le créer et l'afficher .

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

auto mySprite = Sprite::create("back ground2.png");

mySprite->setPosition(Point((visibleSize.width / 2) + origin.x,
(visibleSize.height / 2) + origin.y));
this->addChild(mySprite);

auto menu_item_1 = MenuItemFont::create("Level1",
CC_CALLBACK_1(LevelScene::Level1, this));
auto menu_item_2 = MenuItemFont::create("Level2",
CC_CALLBACK_1(LevelScene::Level2, this));
auto menu_item_3 = MenuItemFont::create("Level3",
CC_CALLBACK_1(LevelScene::Level3, this));
auto menu_item_4 = MenuItemFont::create("GoBack",
CC_CALLBACK_1(LevelScene::GoBack, this));

auto* menu = Menu::create(menu_item_1, menu_item_2, menu_item_3,
menu_item_4, NULL);
//menu->setPosition(Point(0, 0));
menu->alignItemsVertically();
this->addChild(menu);

return true;
}

void LevelScene::Level1(cocos2d::Ref* pSender)
{
    CCLOG("Level1");
    auto scene = Level1::scene();
    Director::getInstance()->pushScene(TransitionFade::create(2, scene));
}
```



Et maintenant dans l'élément `spritgame.h` on va créer tous les probabilités qu'on peut utiliser dans la fonction Sprite qui se trouve déjà dans cocos

```
bool iscolored;  
GameSprite();  
virtual ~GameSprite();  
static GameSprite* gameSpriteinit();
```

et dans `spritegame.cpp` on va initialiser le constructeur et déclarer un destructeur vide, ainsi que créer des nouveau Sprite et remplir la fonction choisis

```
GameSprite::GameSprite(void) {  
    this->iscolored = false;  
};  
GameSprite::~~GameSprite(void) {  
  
}  
GameSprite* GameSprite::gameSpriteinit() {  
    auto sprite = new GameSprite();  
    sprite->initWithFile("tileset.png");  
    return sprite;  
}  
  
void GameSprite::changeColor(Color3B _color) {
```

## Pour créer le niveau 1

On va d'abord appeler cocos et sprit dans `leve1.h`

```
#include "cocos2d.h"
#include "SpriteGame.h"
```

Et déclarer les variables qui contrôle le déplacement du map, la direction des fonctions du cocos.

```
private:
    Vector<GameSprite*> _spritevector;
    Vector<GameSprite*> _collidsprites;
    bool ismoving = false;
    int _direction_x = 0;
    int _direction_y = 0;
    int end;
    int laith = 0;
public:
    static
        //int count;
        Director* director;
```

Après dans `level1.cpp` o, va positionne le mur du jeu et donner une direction 0 au vecteur x y pour commencer, ainsi qu'on va déclarer un destructeur vide, et donner une condition d'arrêt.

```
int array2[32][2] = {
{2,2},{2,3},{2,4},{2,5},{2,6},{2,7},{2,8},{2,9},{2,10},{2,11},{3,2},{3,6},{3,1
0},{4,2},{4,4},{4,6},{4,8},{4,10},{4,12},{5,2},{5,4},{5,6},{5,8},{5,10},{6,2},
{6,4},{6,6},{6,8},{6,10},{6,11},{7,4},{7,8} };
    Level1::Level1() {
        ismoving = false;
        _direction_x = 0;
        _direction_y = 0;
    }
    Level1::~~Level1() {
    }
    void Level1::update(float dt) {
        if (ismoving == true) {
            _ball->setPosition(_ball->getPosition().x + _direction_x, _ball-
>getPosition().y + _direction_y);
        }
    }
```

```

    for (auto _sprites : _collidsprites) {
        if (_direction_x == 0 && _direction_y == 1) {
            if (_ball->getPositionX() == _sprites->getPositionX() && _ball->getPositionY() == _sprites->getPositionY() - 10) {
                ismoving = false;
            }
        }
        if (_direction_x == 0 && _direction_y == -1) {
            if (_ball->getPositionX() == _sprites->getPositionX() && _ball->getPositionY() == _sprites->getPositionY() + 10) {
                ismoving = false;
            }
        }
        if (_direction_x == 1 && _direction_y == 0) {
            if (_ball->getPositionX() == _sprites->getPositionX() - 10 && _ball->getPositionY() == _sprites->getPositionY()) {
                ismoving = false;
            }
        }
        if (_direction_x == -1 && _direction_y == 0) {
            if (_ball->getPositionX() == _sprites->getPositionX() + 10 && _ball->getPositionY() == _sprites->getPositionY()) {
                ismoving = false;
            }
        }
    }

    for (auto _mysprites : _spritevector) {
        if (_ball->getPosition() == _mysprites->getPosition() && _mysprites->iscolored == false) {
            _mysprites->changeColor(Color3B(0, 0, 0)); //tri9 d kora
            laith++;
        }
    }

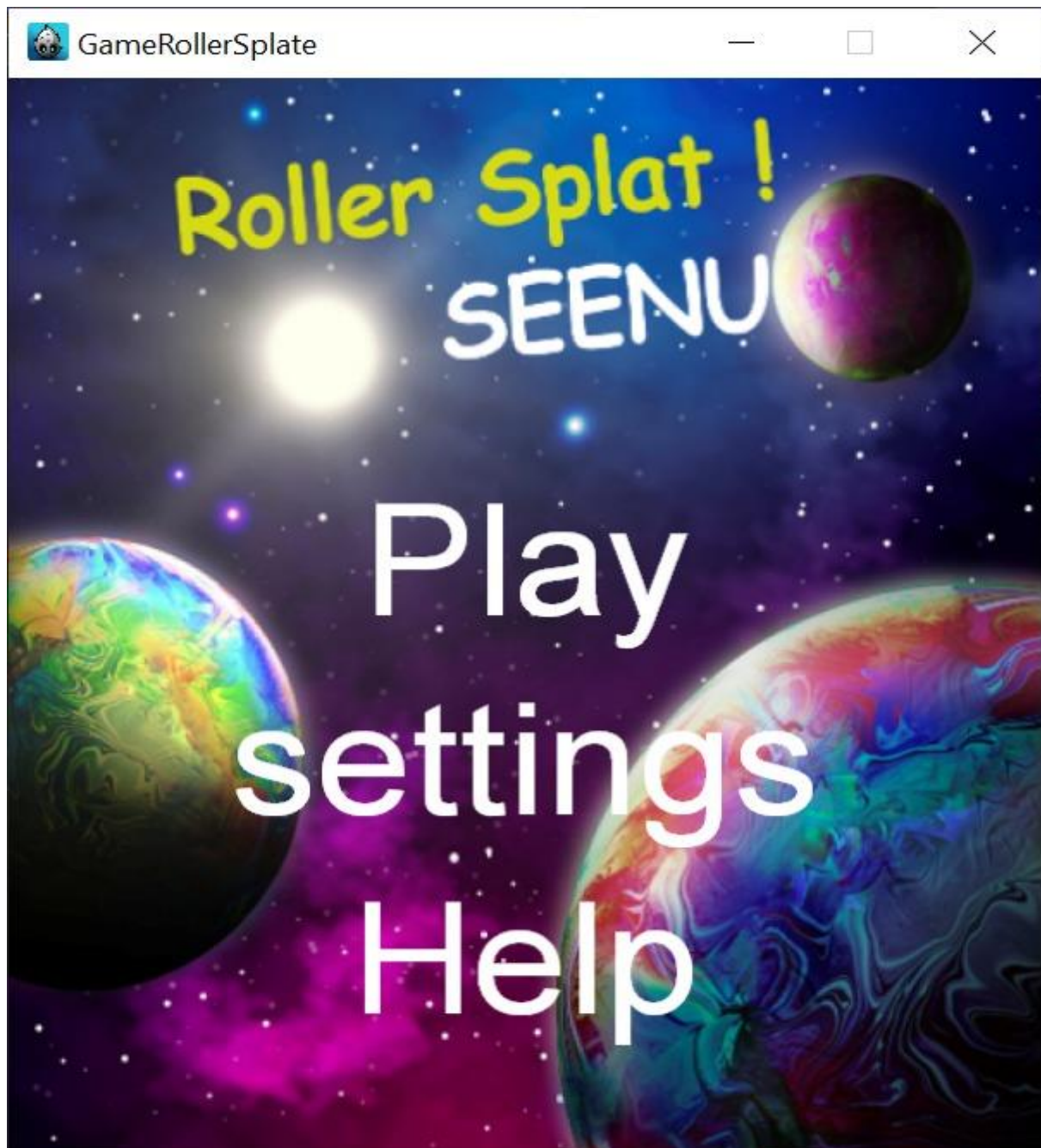
    if (laith == end) {
        //count++;
        auto scene = Scene::create();
        auto layer = Level1::create();
        scene->addChild(layer);

        Director::getInstance()->replaceScene(TransitionFade::create(1, scene));
    }
}

```

# PROJET FINAL

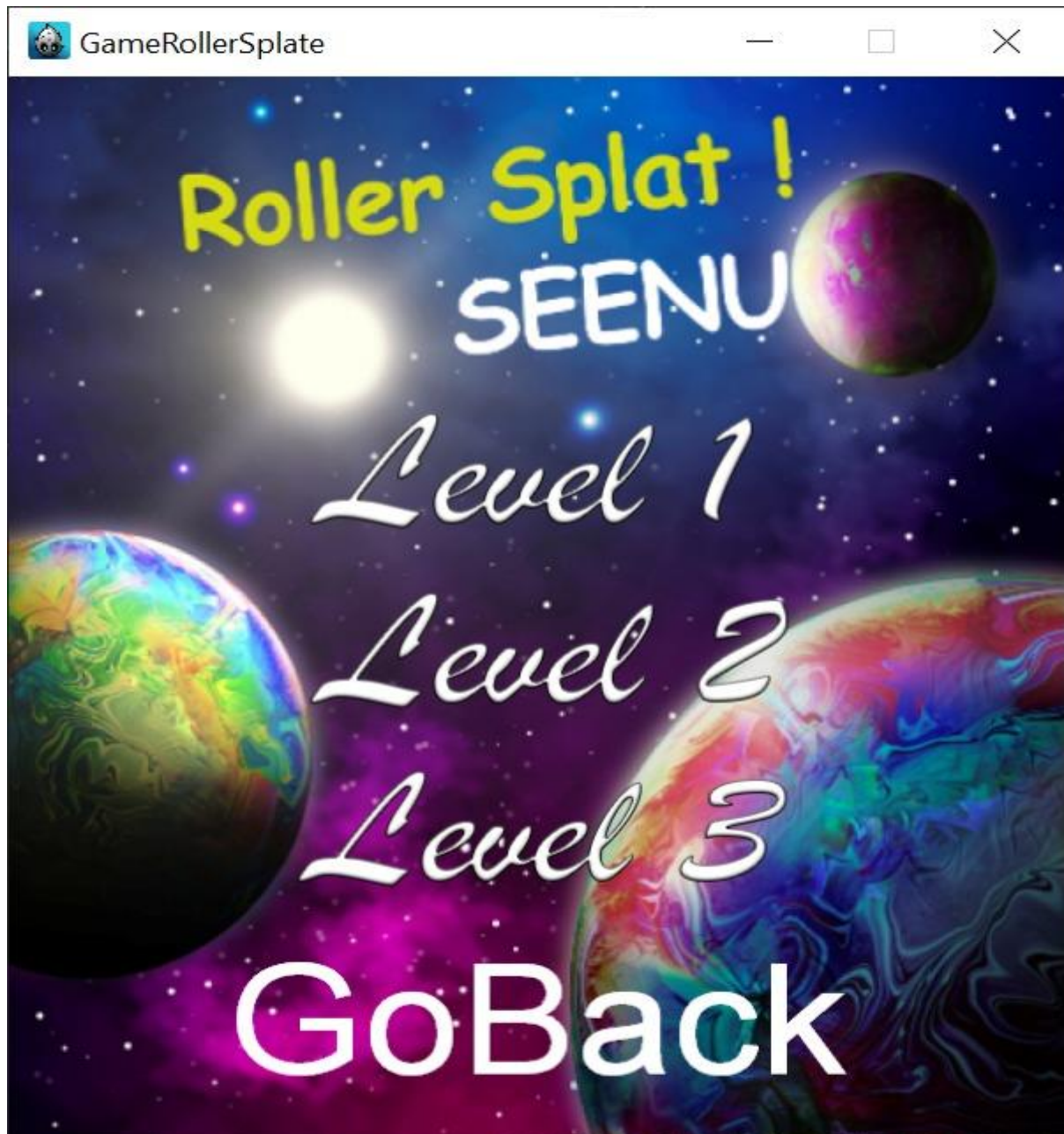
Voilà l'arrières du jeu :







Après qu'on clique sur Play s'affiches ses sous menus :







Exemple du premier niveau :







## CONCLUSION GENERAL

Tout au long de la préparation de notre jeu, nous avons essayé de pratiquer les connaissances requises durant notre cours de 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 maîtriser et découvrir une nouvelle approche de la programmation.





# BIBLIOGRAPHIE

- 🔗 VOTRE POLYCOPIE
- 🔗 <https://www.youtube.com/watch?v=dYTfFeJdfdY>
- 🔗 <https://www.youtube.com/watch?v=dYTfFeJdfdY>

