Rapport de projet jeux 2D

ROLLER SPALT

---

auteur:

ISMAIL BOUZARHOUN

ISSA EL KHAOUA

encadrant:

Pr. ELAACHAk LOTFI

Pr. Ben Abdel ouahab Ikram

Licence Génie Informatique

2021-2022

https://github.com/Elkhaoua-issa/ROLLER_SPALT

Table de matière :

1-Introduction

2-1 objectif de projet

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 est roller Splat

2-2 Contexte général de projet

Le projet dont nous sommes en charge consiste a réaliser le jeux 2D Roller Splat en C++, avec le moteur de jeu cocos2d-x.

Le jeu commence par une interface simple qui consiste à parcourir des labyrinthes avec une boule pour les compléter en même temps que vous les coloriez.

2-Analyse de projet

2-1 outils et langage

A-partie graphiques : on a créé les labyrinthes avec canva et Photoshop et aussi on a utilisé ulistrator pour les buttons.

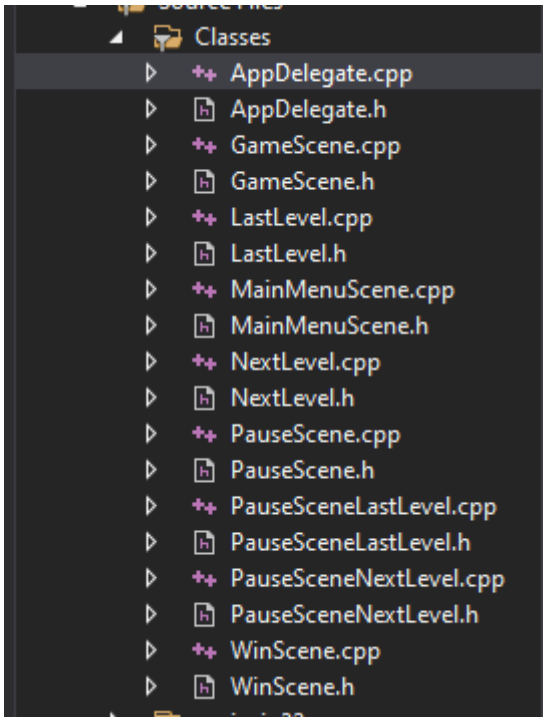
B- -version : on a utilisé la version cocos2d-x 3.17, et avec Visual studio on a utilisé la version 16 2019.

C-- langage : 100% C++

d-Platform de développement : on a utilisé Visual studio 2019 avec la bibliothèque cocos2d-x-3.17

2-2 réalisation :

A-classes



On a divisé le code en plusieurs partie chaque partie il a un code diffèrent chaque code la fait des taches et il a une scène différente

B-AppDelegate :

On a pas beaucoup modifier dans le code de AppDelegate seulement dons la fonction :

```
bool AppDelegate::applicationDidFinishLaunching() { ... }
```

Pour :

- aider à gérer les opérations sur les fichiers.
- obtient la taille de l'écran de l'appareil.
- crée un vecteur de chaînes pour stocker les répertoires d'images. Il y a plusieurs répertoires car le jeu se repliera s'il ne trouve pas l'image dans un répertoire à plus haute résolution.
- vérifiez quel type d'appareil est utilisé et utilisez les répertoires appropriés. Pour prendre en charge Android, l'instruction else est utilisée pour vérifier si l'appareil a une taille d'écran supérieure ou égale à 1080. Si ce n'est pas le cas, le jeu utilise des actifs de résolution inférieure, même pour les iPhones sans rétine.

• assigne les répertoires d'images aux chemins de recherche de l'utilitaire de fichiers afin que le jeu recherche l'image dans chaque répertoire.

C-MainMenu :

Ce code sera le code principal du jeu (Home) qui va commencer après le lancement du jeu.

Premièrement, on a créé un background (avec sa position et sa taille) par la méthode create qui trouve dans la classe Sprite

```
auto backgroundSprite = Sprite::create("MainMenuScreen_Background.png");
backgroundSprite->setPosition(Point((visibleSize.width / 2) +
    origin.x, (visibleSize.height / 2) + origin.y));
backgroundSprite->setScale(0.55);
this->addChild(backgroundSprite, -1);
```

Le titre doit être sous forme d'image et intégrer dans le menu

```
auto menuTitle = MenuItemImage::create("Game_Title.png","Game_Title.png");
menuTitle->setScale(0.7);
```

Et aussi on a créé un bouton start pour aller au niveau 1 par la fonction GoToGameScene:

```
auto playItem = MenuItemImage::create("Play_Button.png","Play_Button.png",CC_CALLBACK_1(MainMenu::GoToGameScene, this));
playItem->setScale(0.16);

auto menu = Menu::create(menuTitle, playItem, NULL);
menu->alignItemsVerticallyWithPadding(visibleSize.height / 4);
this->addChild(menu);
```

Cette fonction sert a remplacer la scene actuelle par une autre (GameScene):

```
void MainMenu::GoToGameScene(Ref* pSender)
{

    auto scene = GameScreen::createScene();

    Director::getInstance()->replaceScene(TransitionFade::create(2, scene));
}
```

D-GameScene :

Ce code source fait pour créer le premier niveau

Premièrement, on créer un menu qui contient bouton pause et bouton next pour aller a le niveau suivant ; et aussi on cre er la background et la ball.

```
//---------Menu------------
//Pause
auto pauseItem = MenuItemImage::create("Pause_Button.png","Pause_Button.png",CC_CALLBACK_1(GameScreen::GoToPauseScene, t
pauseItem->setScale(0.07);
pauseItem->setPosition(Vec2(17, visibleSize.height  + origin.y - 25));
//le niveau prochaine
auto nextItem = MenuItemImage::create("next.png","next.png", CC_CALLBACK_1(GameScreen::GoToNextLevel, this));
nextItem->setScale(0.5);
nextItem->setPosition(Vec2(280, visibleSize.height + origin.y - 30));

auto menu = Menu::create(pauseItem,nextItem, NULL);
menu->setPosition(Point::ZERO);
this->addChild(menu);

//Background
auto sprite = Sprite::create("bg.png");
sprite->setPosition(Vec2(visibleSize.width / 2 + origin.x, visibleSize.height / 2 + origin.y));
sprite->setScale(0.37);
this->addChild(sprite, -1);

//ball
auto my_sprite = Sprite::create("ball.png");
my_sprite->setPosition(Vec2(286, 95));
my_sprite->setScale(0.18);
this->addChild(my_sprite, 20);
```

Mais le intéressant dans ce code c'est la fonction Mouvement qui intéresse de la cote physique (mouvement, coloration).

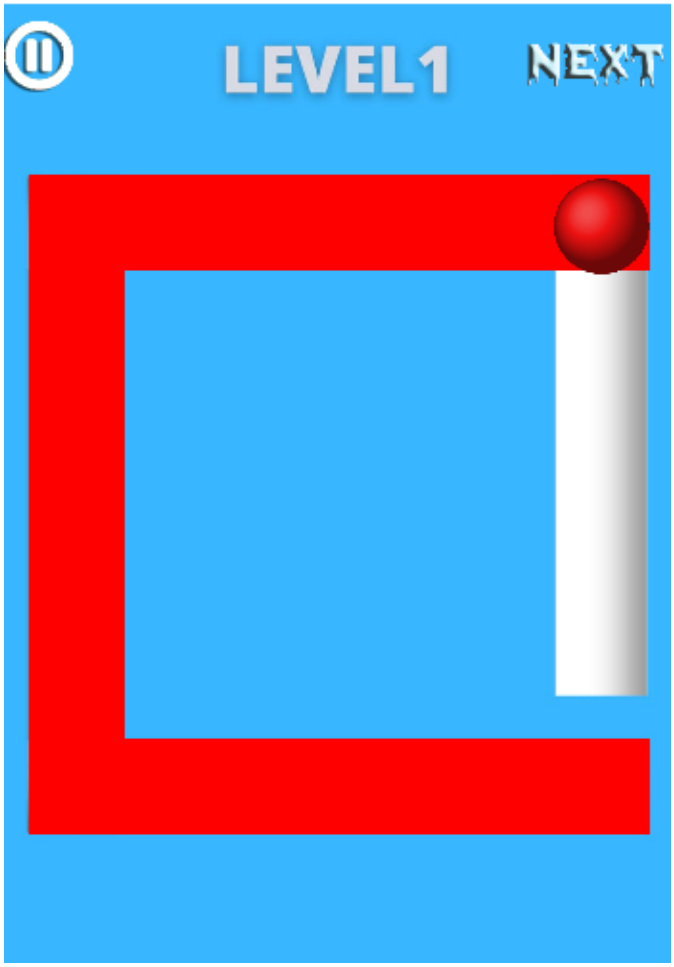
On a donner deux argument pour la fonction, la premiere Keycode c'est la bouton qui'on clique dans le clavier et un evenement event, pour nous ce evenement c'est la ball. Dans cette fonction an a fait des switch pour preciser les bouton convenable dans le clavier, et ce switch la trouve dans un boucle if pour verifier la position de la ball ; si les condition est verifier on fait quelque instruction, faire une action MoveTo pour placer la ball d ;une position a autre, une boucle for pour colorer le chemin de la ball avec les sons effects

```
void GameScreen::Mouvement(EventKeyboard::KeyCode keyCode, Event* event) {
    int i;
    drawNode = DrawNode::create();
    Vec2 loc = event->getCurrentTarget()->getPosition();

    if (loc == Vec2(286, 95))
    {
        switch (keyCode) {
        case EventKeyboard::KeyCode::KEY_LEFT_ARROW:
        case EventKeyboard::KeyCode::KEY_A:
            auto move = MoveTo::create(0.08, Vec2(loc.x - 250, loc.y));
            event->getCurrentTarget()->runAction(move);
            for ( i = 0; i <= 250; i++)
            {
                   drawNode->drawPoint(Vec2(286 - i, loc.y), 50.0f, Color4F::RED);
            }
            this->addChild(drawNode);
            break;
        }
    }
    if (loc == Vec2(36, 95))
    {
        switch (keyCode) {
        case EventKeyboard::KeyCode::KEY_RIGHT_ARROW:
        case EventKeyboard::KeyCode::KEY_D:
            auto move = MoveTo::create(0.08, Vec2(loc.x + 250, loc.y));
            event->getCurrentTarget()->runAction(move);
            break;
        }
    }
    if (loc == Vec2(36, 95))
    {
        switch (keyCode) {
        case EventKeyboard::KeyCode::KEY_UP_ARROW:
        case EventKeyboard::KeyCode::KEY_W:
            auto move = MoveTo::create(0.08, Vec2(loc.x, 267 + loc.y));
            event->getCurrentTarget()->runAction(move);
            for (i = 0; i < 270; i++)
            {
                drawNode->drawPoint(Vec2(loc.x, i + loc.y), 50.0f, Color4F::RED);
            }
```

```
            }
            this->addChild(drawNode);
            break;
        }
    }


    if (loc == Vec2(36, 362))
    {
        switch (keyCode) {
        case EventKeyboard::KeyCode::KEY_DOWN_ARROW:
        case EventKeyboard::KeyCode::KEY_S:
            auto move = MoveTo::create(0.08, Vec2(loc.x, loc.y - 267));
            event->getCurrentTarget()->runAction(move);
            break;
        }
    }
    if (loc == Vec2(36, 362))
    {
        switch (keyCode) {
        case EventKeyboard::KeyCode::KEY_RIGHT_ARROW:
        case EventKeyboard::KeyCode::KEY_D:
            auto move = MoveTo::create(0.08, Vec2(loc.x + 250, loc.y));
            event->getCurrentTarget()->runAction(move);
            for (i = 0; i <= 250; i++)
            {
                drawNode->drawPoint(Vec2(loc.x + i, loc.y+2), 50.0f, Color4F::RED);
            }
            this->addChild(drawNode);
            break;
        }
    }

    if (loc == Vec2(286, 362))
    {
        switch (keyCode) {
        case EventKeyboard::KeyCode::KEY_LEFT_ARROW:
        case EventKeyboard::KeyCode::KEY_A:
            auto move = MoveTo::create(0.08, Vec2(loc.x - 250, loc.y));
            event->getCurrentTarget()->runAction(move);
            break;
```

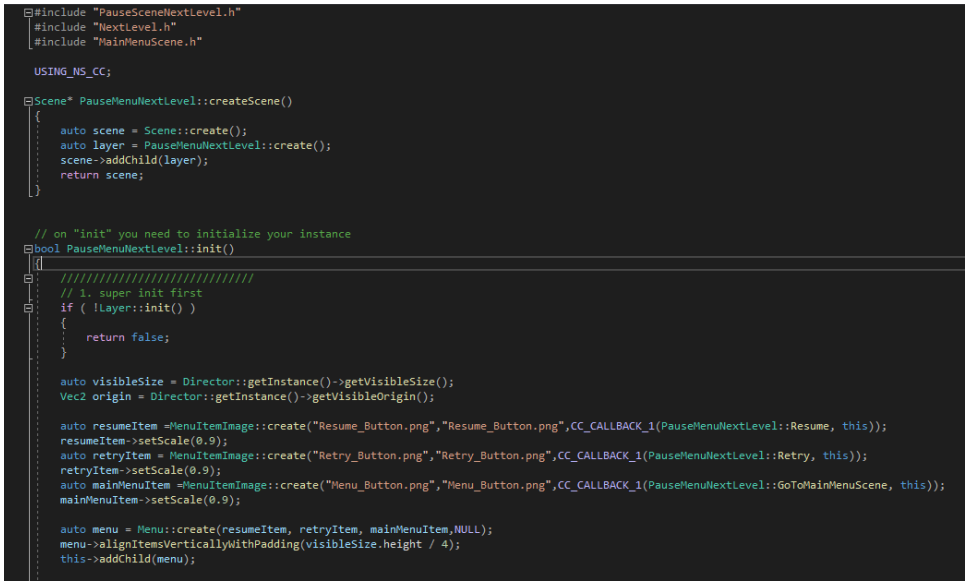
Pour le dernier déplacement (fin de niveau) on appelle une fonction qui remplace la scène actuelle par une autre (niveau prochaine)

```
    }
    if (loc == Vec2(286, 362))
    {
        switch (keyCode) {
        case EventKeyboard::KeyCode::KEY_DOWN_ARROW:
        case EventKeyboard::KeyCode::KEY_S:
            auto move = MoveTo::create(0.08, Vec2(loc.x, loc.y - 200));
            event->getCurrentTarget()->runAction(move);
            for (i = 0; i <= 200; i++)
            {
                drawNode->drawPoint(Vec2(loc.x , loc.y - i), 50.0f, Color4F::RED);
            }
            this->addChild(drawNode);
            auto scene = NextLevel::createScene();
            Director::getInstance()->replaceScene(TransitionFade::create(2, scene));
            break;
        }


    }
```



E-PauseScene :

Dans chaque niveau on a créé un class pour le bouton pause et chaque classe trois bouton et chaque bouton il a trois local scène un pour rejouer et un pour résumé et l'autre pour aller au MainMenu (Home)



```cpp
#include "PauseSceneNextLevel.h"
#include "NextLevel.h"
#include "MainMenuScene.h"

USING_NS_CC;

Scene* PauseMenuNextLevel::createScene()
{
    auto scene = Scene::create();
    auto layer = PauseMenuNextLevel::create();
    scene->addChild(layer);
    return scene;
}


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

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

    auto resumeItem =MenuItemImage::create("Resume_Button.png","Resume_Button.png",CC_CALLBACK_1(PauseMenuNextLevel::Resume, this));
    resumeItem->setScale(0.9);
    auto retryItem = MenuItemImage::create("Retry_Button.png","Retry_Button.png",CC_CALLBACK_1(PauseMenuNextLevel::Retry, this));
    retryItem->setScale(0.9);
    auto mainMenuItem =MenuItemImage::create("Menu_Button.png","Menu_Button.png",CC_CALLBACK_1(PauseMenuNextLevel::GoToMainMenuScene, this));
    mainMenuItem->setScale(0.9);

    auto menu = Menu::create(resumeItem, retryItem, mainMenuItem,NULL);
    menu->alignItemsVerticallyWithPadding(visibleSize.height / 4);
    this->addChild(menu);
```

3-Conclusion

Dans ce projet nous avons permet d'appliquer ce qu'on a vu dans le langage C++ et surtout les classes.

Et on a comprend les étapes pour créer un jeu avec cocos2d-x et aussi commet utilise cette dernière. Et comment travaille un projet dans une équipe.