RAPPORT DU PROJET

Jeux 2d :Imposter Game

prépare par :

aya kammah

**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
* Game Over scène
* Final scène
* CONCLUSION

Introduction :

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

**Une image contenant texte, lumière

Description générée automatiquement**

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 « Imposter ».

**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 , après la création de mon projet , j’ai créé 19 classes:

Graphical user interface, text

Description automatically generated with medium confidence

HelloWorldScene.cpp/HelloworldScene.h : ou j’ai créé mon menu qui va accueillir le joueur

Definitions.h : ou j’ai met les définitions que J’ai utilise .

Level1Scene.cpp/Level1Scene.h : ou j’ai développé le premier niveau de mon jeu

Level2Scene.cpp/Level2Scene.h : ou j’ai développé le deuxième niveau de mon jeu

Level3Scene.cpp/Level3Scene.h : ou j’ai développé le troisième niveau de mon jeu

GameOverScene1.cpp/GameOverScene1.h – GameOverScene2.cpp/GameOverScene2.h- GameOverScene3.cpp/GameOverScene3.h : ou j’ai développé la scène d’échouement de joueur dans chaque niveau (le numéro indique le niveau )

YouWinScene.cpp/YouWinScene.h : ou j ai crée la dernière scène qui félicite le joueur .

Premièrement : Le Menu .

**a-HelloWorldScene.h :**

dans le header j ai cree une classe HelloWorldScene dans cette dernière nous allons définir toutes les fonctions de la classe principale .cette classe définit une scene de jeu de base avec un menu de depart et une fonction (GoToLevel1Scene) pour aller a la scene de priemier niveau (Level1Scene )

class HelloWorldScene : public cocos2d::Layer

{

public:

static cocos2d::Scene\* createScene();

virtual bool init();

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

CREATE\_FUNC(HelloWorldScene);

private:

void GoToLevel1Scene(cocos2d::Ref\* sender);

};

**b-HelloWorldScene.cpp :**

Dans la fonction init j ai crée une backgroundsprite en utilisant Photoshop qui va être le background de notre menu puis nous lui avons donne une position dans notre scène , après crée une variable dont laquelle nous allons attribuer un item de menu avec les sprites « START.png » ET « dark start.png » pour les états actif et inactif de l élément de menu . l élément de menu appelle la fonction « GoToLevel1Scene » lorsqu’ il est sélectionne en fin j ai appelé la fonction GoToLevel1Scene qui gère la transition vers la scène de niveau 1 lorsque l élément de menu est sélectionne

Scene\* HelloWorldScene::createScene()

{

auto scene = Scene::create();

auto layer = HelloWorldScene::create();

scene->addChild(layer);

return scene;

}

// on "init" you need to initialize your instance

bool HelloWorldScene::init()

{

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

// 1. super init first

if ( !Layer::init() )

{

return false;

}

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

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

auto backgroundSprite = Sprite::create("ROOBY (1).png");

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

this->addChild(backgroundSprite);

auto playItem = MenuItemImage::create("START.png", "dark start.png", CC\_CALLBACK\_1(HelloWorldScene::GoToLevel1Scene, this));

playItem->setPosition(Point(visibleSize.width / 2 + origin.x, visibleSize.height / 3 + origin.y));

auto menu = Menu::create(playItem, NULL);

menu->setPosition(Point::ZERO);

this->addChild(menu);

return true;

return true;

}

void HelloWorldScene::GoToLevel1Scene(cocos2d::Ref\* sender) {

auto scene = Level1Scene::createScene();

Director::getInstance()->replaceScene(TransitionFade::create(TRANSITION\_TIME, scene));

}

Enfin on obtient ce résultat : Graphical user interface, application

Description automatically generated

**Deuxièment : le premier niveau**

**a-Level1Scene.h :**

dans la partie header du niveau 1 nous avons créé une classe Lvel1Scene dont laquelle nous allons définir toutes les fonctions et variables membre associe a cette scène , telles que la mise a jour de l’état de la scène , la gestion des collisions et la transition vers d autre scènes **.**

class Level1Scene : public cocos2d::Layer

{

public:

static cocos2d::Scene\* createScene();

virtual bool init();

// a selector callback

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

// implement the "static create()" method manually

CREATE\_FUNC(Level1Scene);

private:

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

cocos2d::PhysicsWorld\* sceneWorld;

cocos2d::Sprite\* mySprite;

int DirX = 0;

int DirY = 0;

void update(float dt);

bool onContactBegin(cocos2d::PhysicsContact& contact);

void GoToHelloWorldScene(cocos2d::Ref\* sender);

};

**b-Level1Scene.cpp :**

**🡪** j ai définit une fonction qui crée une scène de jeu avec des élément de physique . la fonction commence par créer une scène en utilisant ‘’Scène ::createWithPhysics ‘’. Cette méthode crée une scène avec un monde physique intègre qui peut être utilise pour simuler des objets avec des corps physique et des forces .ensuite on a une fonction qui modifier la gravite de monde pour le joueur puisse tomber dans les vides

Scene\* Level1Scene::createScene()

{

auto scene = Scene::createWithPhysics();

//scene->getPhysicsWorld()->setDebugDrawMask(PhysicsWorld::DEBUGDRAW\_ALL);

scene->getPhysicsWorld()->setGravity(Vec2(0, -500));

auto layer = Level1Scene::create();

layer->SetPhysisicsWorld(scene->getPhysicsWorld());

scene->addChild(layer);

return scene;

}

🡪 dans cette partie j ai utilise la fonction « init » pour initialiser l’instance de la scène en ajoutant un background pour la scène et définissant un bord physique pour la scène , j ai crée aussi un Sprite pour le joueur qui est mySprite et j’ai lui attribuer un corps physique et j ai ajouter des élément de menu .

la fonction update est appelée a chaque frame et met a jour la position du Sprite du jouer en fonction des entrée clavier .

la fonction onContactBegin est appelée lorsqu’ il y a une collision entre deux cors physique ( jouer avec le feu , joueur avec la porte ) et permet de gère les séquences de la collision . la fonction GoToHelloWorldScene est appelée lorsque l élément de menu RETRY est sélectionne et permet de passer a la scène de jeu précédente .

bool Level1Scene::init()

{

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

// 1. super init first

if ( !Layer::init() )

{

return false;

}

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

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

//bachground de jeux

auto backgroundSprite = Sprite::create("LEVE1BG.png");

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

this->addChild(backgroundSprite);

//le cadre physiqye de jeux

//PHYSICSSHAPE\_MATERIAL\_DEFAULT( density, restitution,friction);

auto edgeBody = PhysicsBody::createEdgeBox(visibleSize, PHYSICSBODY\_MATERIAL\_DEFAULT, 3);

auto edgeNode = Node::create();

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

edgeNode->setPhysicsBody(edgeBody);

this->addChild(edgeNode);

// Creez un sprite pour le joueur et ajoutez - lie a la scene

mySprite = Sprite::create("PPPP.png");

mySprite->setPosition(Point(visibleSize.width / 8 + origin.x, visibleSize.height / 2 + origin.y));

addChild(mySprite);

// Creez un corps physique pour le joueur et attache-le au sprite

auto body = PhysicsBody::createBox(mySprite->getContentSize());

body->setCollisionBitmask(mySprite\_COLLISION\_BITMASK);

body->setContactTestBitmask(true);

mySprite->setPhysicsBody(body);

auto RetryItem = MenuItemImage::create("sign home.png", "home dark.png", CC\_CALLBACK\_1(Level1Scene::GoToHelloWorldScene, this));

RetryItem->setPosition(Vec2(23, 275));

auto menu = Menu::create(RetryItem, NULL);

menu->setPosition(Point::ZERO);

this->addChild(menu);

backgroundSprite = Sprite::create("rock.png");

backgroundSprite->setPosition(Vec2(383, 65));

addChild(backgroundSprite);

// Creez un corps physique pour la box1

auto box1 = PhysicsBody::createBox(backgroundSprite->getContentSize());

box1->setDynamic(false);

backgroundSprite->setPhysicsBody(box1);

backgroundSprite = Sprite::create("rock.png");

backgroundSprite->setPosition(Vec2(102, 65));

addChild(backgroundSprite);

// Creez un corps physique pour la box2

auto box2 = PhysicsBody::createBox(backgroundSprite->getContentSize());

box2->setDynamic(false);

backgroundSprite->setPhysicsBody(box2);

backgroundSprite = Sprite::create("door.png");

backgroundSprite->setPosition(Vec2(465, 159));

addChild(backgroundSprite);

auto door = PhysicsBody::createBox(backgroundSprite->getContentSize());

door->setCollisionBitmask(DOOR\_COLLISION\_BITMASK);

door->setContactTestBitmask(true);

door->setDynamic(false);

backgroundSprite->setPhysicsBody(door);

//Fire obstacle

backgroundSprite = Sprite::create("FIRE1.png");

backgroundSprite->setPosition(Vec2(243, 45));

addChild(backgroundSprite);

auto fire = PhysicsBody::createBox(backgroundSprite->getContentSize());

fire->setCollisionBitmask(FIRE\_COLLISION\_BITMASK);

fire->setContactTestBitmask(true);

fire->setDynamic(false);

backgroundSprite->setPhysicsBody(fire);

auto keyboardListener = EventListenerKeyboard::create();

keyboardListener->onKeyPressed = [=](EventKeyboard::KeyCode keyCode, Event\* event)

{

switch (keyCode)

{

case EventKeyboard::KeyCode::KEY\_W:

case EventKeyboard::KeyCode::KEY\_UP\_ARROW:

DirY += 3.0f;

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 -= 3.0f;

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:

DirY -= 3.0f;

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 += 3.0f;

break;

case EventKeyboard::KeyCode::KEY\_D:

case EventKeyboard::KeyCode::KEY\_RIGHT\_ARROW:

DirX -= 3.0f;

break;

}

};

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

this->scheduleUpdate();

auto contactListener = EventListenerPhysicsContact::create();

contactListener->onContactBegin = CC\_CALLBACK\_1(Level1Scene::onContactBegin, this);

Director::getInstance()->getEventDispatcher()->addEventListenerWithSceneGraphPriority(contactListener, this);

return true;

}

🡪J ai appelée cette fonction pour remplacée la scène Level1Scene par la scène HelloWorldScene quant le joueur appui sur sight home en utilisant une transition de type ‘TransitionFade’ , la durée de la transition est définie par la constante TRANSITION\_TIME qui est définie dans la classe ‘Definitions.h’

void Level1Scene::GoToHelloWorldScene(cocos2d::Ref\* sender) {

auto scene = HelloWorldScene::createScene();

Director::getInstance()->replaceScene(TransitionFade::create(TRANSITION\_TIME, scene));

}

🡪j ai appelée la méthode update qui censée être exécuter a chaque frame de l application . la méthode calcule une nouvelle position pour le jouer mySprite en utilisant sa position actuelle et les valeurs de ‘DirX’ et ‘DirrY’. La nouvelle position est ensuite définir sur le Sprite en utilisant ‘setposition’. Cela permet de déplacer le prite a chaque frame de l application

void Level1Scene::update(float dt)

{

float newPosX = Level1Scene::mySprite->getPositionX() + (DirX);

float newPosY = Level1Scene::mySprite->getPositionY() + (DirY);

Level1Scene::mySprite->setPosition(newPosX, newPosY);

}

* Ce code gère les collisions entre le joueur est le feu et le joueur est la porte , lorsqu’ une collision est détectée , le code vérifie les masques de collision des deux objets en contact et agit en conséquence . si le joueur entre en contact avec le feu de masque de collision ‘FIRE\_COLLISION\_BITMASK’ , la scène de fin de jeu(GameOverScene1) est affichée . si le joueur entre en contact avec la porte de masque de collision ‘DOOR\_COLLISION\_BITMASK’ ,la scène de deuxième niveau s affichée (Level2Scene).

bool Level1Scene::onContactBegin(cocos2d::PhysicsContact& contact)

{

PhysicsBody\* a = contact.getShapeA()->getBody();

PhysicsBody\* b = contact.getShapeB()->getBody();

// v�rifiez si le joueur entre en contact avec une pierre

if ((a->getCollisionBitmask() == mySprite\_COLLISION\_BITMASK && b->getCollisionBitmask() == FIRE\_COLLISION\_BITMASK) ||

(b->getCollisionBitmask() == mySprite\_COLLISION\_BITMASK && a->getCollisionBitmask() == FIRE\_COLLISION\_BITMASK))

{

// affichez la sc�ne de fin de jeu (game over) ici

auto scene = GameOverScene1::createScene();

Director::getInstance()->replaceScene(TransitionFade::create(TRANSITION\_TIME, scene));

}

// v�rifiez si le joueur entre dans la porte

if ((a->getCollisionBitmask() == mySprite\_COLLISION\_BITMASK && b->getCollisionBitmask() == DOOR\_COLLISION\_BITMASK) ||

(b->getCollisionBitmask() == mySprite\_COLLISION\_BITMASK && a->getCollisionBitmask() == DOOR\_COLLISION\_BITMASK))

{

// affichez la sc�ne de victoire ici

auto scene = Level2Scene::createScene();

Director::getInstance()->replaceScene(TransitionFade::create(TRANSITION\_TIME, scene));

}

return true;

}

Voila notre scène Level1Scene avec une vue physique qui nous montre les corps physique qu’ on a crée : A picture containing shape

Description automatically generated

Vu normal : A screenshot of a video game

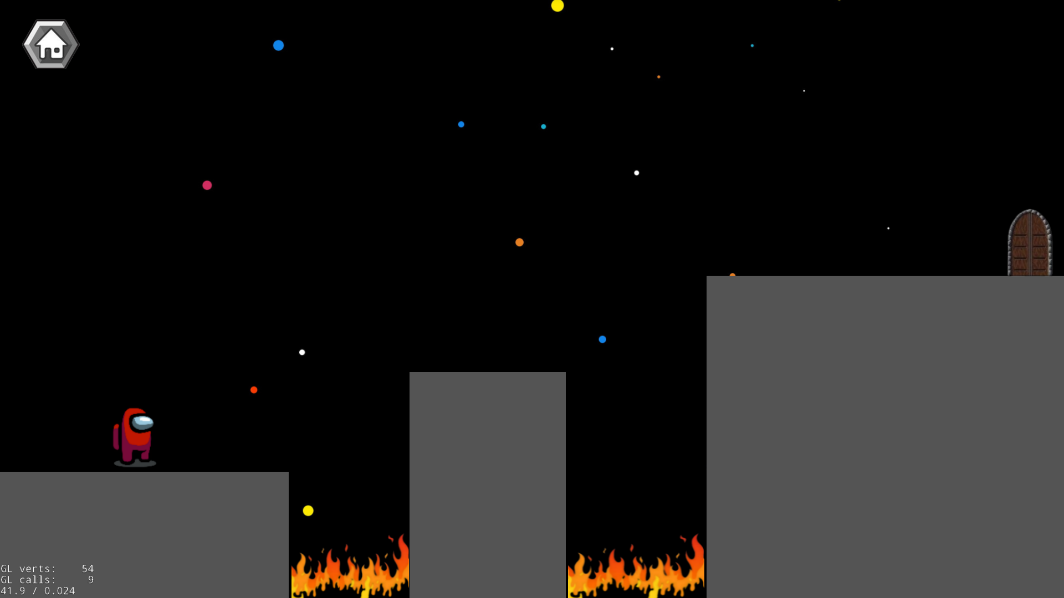
Description automatically generated

**Troisièment : le deuxième et le troisième niveau :**

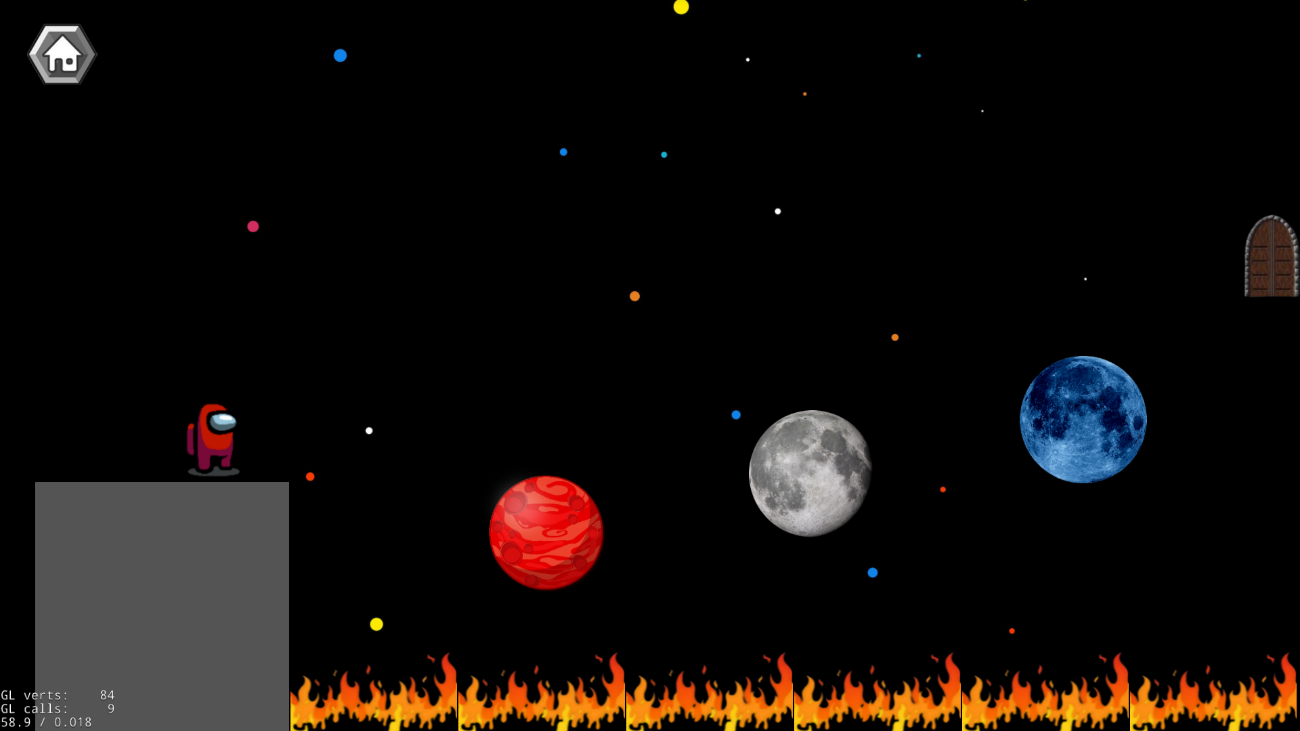
Suivant les mêmes étapes et les mêmes fonctions précédentes j ai simplement juste changer les maps et les positions de notre roche et obstacles .

Voila le resultat :

Level 2 :



Level 3 :

: **Quatriement : Game Over scene**

**a-GameOverScene1.h**

🡪 dans le header j ai déclarer la classe GameOverScene1 qui contient plusieurs fonctions membres publique , notamment une fonction de création ‘createscene’ , une fontion d initialisation virtuelle appelée ‘init’ , une fonction de rappel de sélecteur appelée ‘menuCloseCallback’, la classe contient comme membre privées ‘GameOverScene1’, ‘GoToLevel2Scene’et ‘GoToHelloWorldScene’ ces fonction utilisées pour naviguer vers d autre scènes lorsque l utilisateur interagit avec l interface .

class GameOverScene1 : public cocos2d::Layer

{

public:

static cocos2d::Scene\* createScene();

virtual bool init();

// a selector callback

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

// implement the "static create()" method manually

CREATE\_FUNC(GameOverScene1);

private:

//membre des classes

void GoToGameScene(cocos2d::Ref\* sender);

void GoToLevel1Scene(cocos2d::Ref\* sender);

void GoToHelloWorldScene(cocos2d::Ref\* sender);

};

**b- GameOverScene1.cpp**

🡪 dans cette partie j ai juste crée le background de la scène avec les boutons qui son interactive (menuitems ) comme RETRY vous mené a la scène Level1Scene et QUIT qui vous mené a la scène HelloWorldScene

bool GameOverScene1::init()

{

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

// 1. super init first

if (!Layer::init())

{

return false;

}

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

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

auto backgroundSprite = Sprite::create("BACKGROUND GAME OVER.png");

backgroundSprite->setPosition(Point(visibleSize.width / 2 + origin.x, visibleSize.height / 3 + origin.y));

this->addChild(backgroundSprite);

backgroundSprite = Sprite::create("GameOver.png");

backgroundSprite->setPosition(Vec2(250, 240));

addChild(backgroundSprite);

auto RetryItem = MenuItemImage::create("RETRY.png", "dark retry.png", CC\_CALLBACK\_1(GameOverScene1::GoToLevel1Scene, this));

RetryItem->setPosition(Vec2(250, 170));

auto menu = Menu::create(RetryItem, NULL);

menu->setPosition(Point::ZERO);

this->addChild(menu);

auto QuiteItem = MenuItemImage::create("QUIT.png", "dark quit.png", CC\_CALLBACK\_1(GameOverScene1::GoToHelloWorldScene, this));

QuiteItem->setPosition(Vec2(250, 140));

menu = Menu::create(QuiteItem, NULL);

menu->setPosition(Point::ZERO);

this->addChild(menu);

return true;

}

🡪J ai déjà explique le fonctionnement de cette partie dans la scène Level1Scene

void GameOverScene1::GoToLevel1Scene(cocos2d::Ref\* sender) {

auto scene = Level1Scene::createScene();

Director::getInstance()->replaceScene(TransitionFade::create(TRANSITION\_TIME, scene));

}

void GameOverScene1::GoToHelloWorldScene(cocos2d::Ref\* sender) {

auto scene = HelloWorldScene::createScene();

Director::getInstance()->replaceScene(TransitionFade::create(TRANSITION\_TIME, scene));

}

**Le resultat:**

Graphical user interface

Description automatically generated

**Cinqiuement: final scene** .

**a-YouWinScene.h:**

dans la partie header de la dernière scène nous avons crée une classe final dont laquelle nous allons définie toutes les classes de la dernière scène

class YouWinScene : public cocos2d::Layer

{

public:

static cocos2d::Scene\* createScene();

virtual bool init();

// a selector callback

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

// implement the "static create()" method manually

CREATE\_FUNC(YouWinScene);

private:

//membre des classes

void GoToGameScene(cocos2d::Ref\* sender);

void GoToLevel1Scene(cocos2d::Ref\* sender);

void GoToHelloWorldScene(cocos2d::Ref\* sender);

};

**b-YouWinScene.cpp :**

dans la fonction init de la dernier scène j ai créé backgroundsprite qui va être le background de cette scène puis on a crée des boutons qui sont des variables pour cree un menu , son rôle est de permettre de revenir a la page principale ou pour rejouer le jeu

bool YouWinScene::init()

{

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

// 1. super init first

if (!Layer::init())

{

return false;

}

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

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

auto backgroundSprite = Sprite::create("GP3.png");

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

this->addChild(backgroundSprite);

auto RetryItem = MenuItemImage::create("RETRY.png", "RETRY.png", CC\_CALLBACK\_1(YouWinScene::GoToLevel1Scene, this));

RetryItem->setPosition(Vec2(180, 130));

auto menu = Menu::create(RetryItem, NULL);

menu->setPosition(Point::ZERO);

this->addChild(menu);

auto QuiteItem = MenuItemImage::create("QUIT.png", "dark quit.png", CC\_CALLBACK\_1(YouWinScene::GoToHelloWorldScene, this));

QuiteItem->setPosition(Vec2(180, 100));

menu = Menu::create(QuiteItem, NULL);

menu->setPosition(Point::ZERO);

this->addChild(menu);

return true;

}

void YouWinScene::GoToLevel1Scene(cocos2d::Ref\* sender) {

auto scene = Level1Scene::createScene();

Director::getInstance()->replaceScene(TransitionFade::create(TRANSITION\_TIME, scene));

}

void YouWinScene::GoToHelloWorldScene(cocos2d::Ref\* sender) {

auto scene = HelloWorldScene::createScene();

Director::getInstance()->replaceScene(TransitionFade::create(TRANSITION\_TIME, scene));

}

Enfin on obtient ce résultat :



Conclusion :

Tout au long de la préparation de mon jeu , j ai essaye de pratiques les connaissances requises durant le coure de du programmation oriente objet , et aussi notre connaissance sur ce qu’ on a trouve concernant cocos2d .

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

Et aussi de faire un backup de chaque version de code .