PROJET DE MODULE
C++

Prpareer par :

AYOUB ET-TOUBI

ABDERRAZZAK EL BOURKADI

Encadre par : EL ACHAK LOTFI

Objectif : 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 proposé s'appelle PICO PARK, un jeu de type Platformer Puzzle Game.

C est quoi cocos2d-x

Cocos2d-x est le moteur de jeu open source le plus populaire au monde. Cocos2d-x intègre deux langages de programmation principaux, C++ et Lua. Ce rapport se concentrera sur l'implémentation C++. Il existe également une version JavaScript appelée Cocos2d-JS, qui prend également en charge le développement Web. Pour mettre les choses en perspective, quiconque a joué à un jeu mobile en aura joué un qui a très probablement été construit à l'aide Cocos2d-x. Les capacités du moteur de jeu s'étendent au-delà du développement de jeux, avec

fonctionnalités pour le développement d'applications générales. Cependant, l'aspect le plus important qui rend Cocos2d-x phénoménal est sa nature multiplateforme, permettant le développement pour toutes les principales plates-formes mobiles et de bureau.

Les etapes pour developer Jeux

Pour afficher le jeux avec une grande taille :

AppDelegate.cpp

Pour Menu :

```
#include "MainMenuScene.h"
```

C´est interface de notre jeux :

Nous faisond le background et le button play qui permet d'aller a level 1

```cpp
#include "MainMenuScene.h"
#include "GameScene.h"
#include "GameScene2.h"
#include "GameScene3.h"
USING_NS_CC;

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

}
bool MainMenu::init()
{
    ////////////////////////////
    // 1. super init first
    if (!Layer::init())
    {
        return false;
    }

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

    auto menuTitle =
        MenuItemImage::create("logo.jpg",
            "logo.jpg");
    auto playItem =
        MenuItemImage::create("play.png",
            "play.png",
            CC_CALLBACK_1(MainMenu::GoToGameScene, this));
    auto menu = Menu::create(menuTitle, playItem, NULL);
    menu->alignItemsVerticallyWithPadding(visibleSize.height / 4);
    this->addChild(menu);
    return true;
```

```cpp
#include "MainMenuScene.cpp"
```

```cpp
#ifndef __HELLOWORLD_SCENE_H__
#define __HELLOWORLD_SCENE_H__

#include "cocos2d.h"

class MainMenu: 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(MainMenu);
    void GoToGameScene(Ref* pSender);
};

#endif // __HELLOWORLD_SCENE_H__
```

Level 1 :'

Les bibliotheques qui sont utilisees pour le level 1 :

```
#include "GameScene.h"
#include "PauseScene.h"
#include "GameOverScene.h"
#include "MainMenuScene.h"
#include "physics/CCPhysicsBody.h"
#include "physics/CCPhysicsJoint.h"
#include "AudioEngine.h"
#include "GameScene2.h"
```

```
bool GameScreen::init()
{
    ////////////////////////////
    // 1. super init first
    if (!Layer::init())
    {
        return false;
    }


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

    //pause button
    auto pauseItem =
        MenuItemImage::create("Pause_Button.png",
            "Pause_Button.png",
            CC_CALLBACK_1(GameScreen::GoToPauseScene, this));
    pauseItem->setPosition(Point(pauseItem->getContentSize().width -
        (pauseItem->getContentSize().width / 4) + origin.x,
        visibleSize.height - pauseItem->getContentSize().height +
        (pauseItem->getContentSize().width / 4) + origin.y));
    auto menu = Menu::create(pauseItem, NULL);
    menu->setPosition(Point::ZERO);
    this->addChild(menu);


    //background
```

```cpp
USING_NS_CC;

Scene* GameScreen::createScene()
{
    auto scene = Scene::createWithPhysics();
    PhysicsWorld* world = scene->getPhysicsWorld();
    scene->getPhysicsWorld()->setDebugDrawMask(PhysicsWorld::DEBUGDRAW_ALL);
    scene->getPhysicsWorld()->setGravity(Vec2(0, -500));
    scene->getPhysicsWorld()->setDebugDrawMask(0);



    auto layer = GameScreen::create();
    scene->addChild(layer);
    return scene;

}
```

d

```cpp
    // Create a keyboard event listener
    auto keyboardListener = EventListenerKeyboard::create();
    keyboardListener->onKeyPressed = CC_CALLBACK_2(GameScreen::onKeyPressed, this);
    keyboardListener->onKeyReleased = CC_CALLBACK_2(GameScreen::onKeyPressed, this);

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

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

        if (KeyCode == EventKeyboard::KeyCode::KEY_UP_ARROW) {
            auto action1 = JumpBy::create(0.57f, Vec2(50, 100), 15.0f, 1);
            auto easeAction = EaseOut::create(action1, 2.0f);
            player->runAction(action1);

        }
        if (KeyCode == EventKeyboard::KeyCode::KEY_RIGHT_ARROW) {
            auto jump = JumpBy::create(0.5f, Vec2(50, 50), 20.0f, 1);
            MoveBy* moveAction = MoveBy::create(1.2, Vec2(70, 0));
            RepeatForever* repeatAction = RepeatForever::create(moveAction);
            player->runAction(repeatAction);

        }
        if (KeyCode == EventKeyboard::KeyCode::KEY_LEFT_ARROW) {
            auto jump = JumpBy::create(0.5f, Vec2(50, 50), 50.0f, 1);
            MoveBy* moveAction = MoveBy::create(1.2, Vec2(-70, 0));
            RepeatForever* repeatAction = RepeatForever::create(moveAction);
            player->runAction(repeatAction);

        }

    };
```

Le principe de controler le joueur avec les fiches de clavier :

```
keyboardListener->onKeyReleased = [player](EventKeyboard::KeyCode KeyCode, Event* event)
{
    if (KeyCode == EventKeyboard::KeyCode::KEY_RIGHT_ARROW) {
        player->stopAllActions();
    }
    if (KeyCode == EventKeyboard::KeyCode::KEY_LEFT_ARROW) {
        player->stopAllActions();
    }

};
```

Pour ajouter les evenements ou cas ou le joueur gangner ou lose

```
auto contactListener =
    EventListenerPhysicsContact::create();
contactListener->onContactBegin = [](PhysicsContact& contact)
{
    auto shapeA = contact.getShapeA();
    auto bodyA = shapeA->getBody();
    auto shapeB = contact.getShapeB();
    auto bodyB = shapeB->getBody();

    auto nodeA = contact.getShapeA()->getBody()->getNode();
    auto nodeB = contact.getShapeB()->getBody()->getNode();

    if (nodeA->getTag() == 10 && nodeB->getTag() == 10)
    {


        Director::getInstance()->replaceScene(GameOver::createScene());


    }
    else {
        Director::getInstance()->replaceScene(GameScreen2::createScene());
    }
    return true;
};
this->getEventDispatcher()->addEventListenerWithSceneGraphPriority(contactListener, this);


//win
```

Le principe pour ajouter un joueur et les opstacles :

```
//background


auto backgroundSprite = Sprite::create
("back2.png");
backgroundSprite->setPosition(Point((visibleSize.width / 2) +
    origin.x, (visibleSize.height / 2) + origin.y));
this->addChild(backgroundSprite, -1);

//add floor
auto *floor = Sprite::create("floor2.png");
floor->setAnchorPoint(Vec2(0, 0));
floor->setPosition(Vec2(1, 0));
floor->setScale(0.425);
this->addChild(floor, 1);
//add floor2
auto* floor2 = Sprite::create("floor2.png");
floor2->setAnchorPoint(Vec2(0, 0));
floor2->setPosition(Vec2(280, 0));
floor2->setScale(0.425);
this->addChild(floor2, 1);

//add floor3
auto* floor3 = Sprite::create("floor2.png");
floor3->setAnchorPoint(Vec2(0, 0));
floor3->setPosition(Vec2(210, 0));
floor3->setScale(0.090);
floor3->setTag(10);
this->addChild(floor3, 3);
```

Pou ajouter les physiques :

```
physicsBody1->applyForce(force);
//physique floor 1
auto physicsBody_floor = PhysicsBody::createBox(Size(500.0f, 60.0f), PhysicsMaterial(100.0f, 0.1f, 0.6

physicsBody_floor->setDynamic(false);
physicsBody_floor->setCollisionBitmask(1);
physicsBody_floor->setCategoryBitmask(1);
floor->setPhysicsBody(physicsBody_floor);



//physique floor 2
auto physicsBody_floor2 = PhysicsBody::createBox(Size(500.0f, 60.0f), PhysicsMaterial(100.0f, 0.1f, 0.

physicsBody_floor2->setDynamic(false);
physicsBody_floor2->setCollisionBitmask(1);
physicsBody_floor2->setCategoryBitmask(1);
floor2->setPhysicsBody(physicsBody_floor2);
//physique floor 3
auto physicsBody_floor3 = PhysicsBody::createBox(floor3->getContentSize(), PhysicsMaterial(100.0f, 0.1

physicsBody_floor3->setDynamic(false);
physicsBody_floor3->setCollisionBitmask(3);
physicsBody_floor3->setCategoryBitmask(1);
physicsBody_floor3->setContactTestBitmask(1);
floor3->setPhysicsBody(physicsBody_floor3);

//physique floor 4
auto physicsBody_floor4 = PhysicsBody::createBox(Size(100.0f, 100.0f), PhysicsMaterial(100.0f, 0.1f, 0
```

Pour deplacer entre les scenes

```
bool GameScreen::onContactBegin(cocos2d::PhysicsContact& contact) {



    return true;
}
void GameScreen::GoToPauseScene(cocos2d::Ref* pSender)
{
    auto scene = PauseMenu::createScene();

    Director::getInstance()->pushScene(scene);
}
void GameScreen::GoToGameOverScene(cocos2d::Ref* pSender)
{
    auto scene = GameOver::createScene();

    Director::getInstance()->replaceScene(scene);
}
```

Level 2 :

Le meme principe que level 1 pour ajouter un joueur et les obstacles et backgroubd .

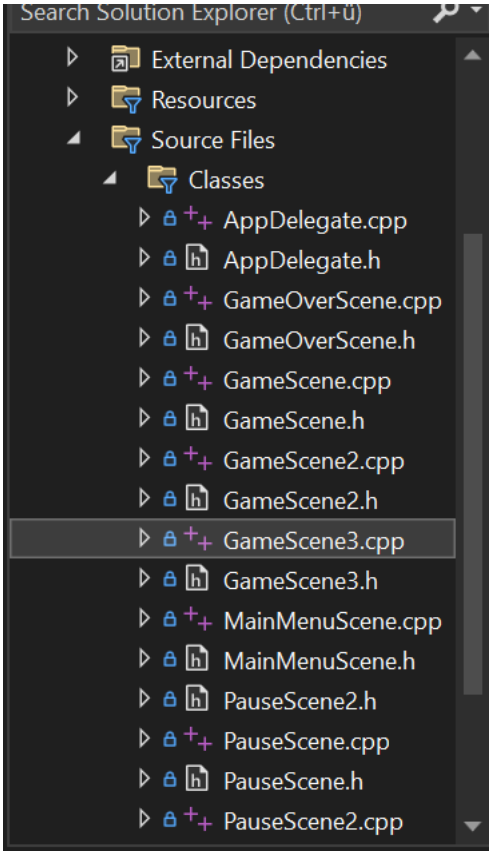
La valeur ajoutee dans ce level est le mouvement des obstacles

```
    //scrool

    auto* acc = MoveBy::create(0.07 * visibleSize.width, Point(-visibleSize.width * 4, 0));
    floor->runAction(acc->clone());
    floor1->runAction(acc->clone());
    floor4->runAction(acc->clone());
    floor3->runAction(acc->clone());
    floor2->runAction(acc->clone());
    floor5->runAction(acc->clone());
    floor6->runAction(acc->clone());
    floor7->runAction(acc->clone());
    floor8->runAction(acc->clone());
    floor9->runAction(acc->clone());
    floor10->runAction(acc->clone());
    floor11->runAction(acc->clone());
    floor12->runAction(acc->clone());
```

Level 3 :

```cpp
USING_NS_CC;
Scene* GameScreen3::createScene()

{
    auto scene = Scene::createWithPhysics();
    PhysicsWorld* world = scene->getPhysicsWorld();
    scene->getPhysicsWorld()->setDebugDrawMask(PhysicsWorld::DEBUGDRAW_ALL);
    scene->getPhysicsWorld()->setGravity(Vec2(0, -500));
    scene->getPhysicsWorld()->setDebugDrawMask(1);


    auto layer = GameScreen3::create();
    scene->addChild(layer);
    return scene;

}


bool GameScreen3::init()
{
    ////////////////////////////
    // 1. super init first
    if (!Layer::init())
    {
        return false;
    }


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

    auto pauseItem =
```

```cpp
    //add player
auto player = Sprite::create("play1.png");
player->setAnchorPoint(Vec2(0.5, 0.5));
player->setPosition(Vec2(40, 180));
player->setScale(0.2); //scale dyal player
player->setName("player");
player->setTag(10);
this->addChild(player, 2);

//Physique player
auto physicsBody1 = PhysicsBody::createBox(Size(140.0f, 135.0f), PhysicsMaterial(5000.0f, 0.5f, 0.5f));
physicsBody1->setDynamic(true);
physicsBody1->setContactTestBitmask(1);
physicsBody1->setRotationEnable(false);
physicsBody1->setCollisionBitmask(1);
player->setPhysicsBody(physicsBody1);
Vec2 force = Vec2(0, -physicsBody1->getMass() * 1.8f);
physicsBody1->applyForce(force);

//add floor
auto* floor = Sprite::create("floor2.png");
floor->setAnchorPoint(Vec2(0, 0));
floor->setPosition(Vec2(1, 0));
floor->setScale(0.425);
this->addChild(floor, 1);

auto physicsBody_floor = PhysicsBody::createBox(Size(500.0f, 60.0f), PhysicsMaterial(100.0f, 0.1f, 0.009f));

physicsBody_floor->setDynamic(false);
physicsBody_floor->setCollisionBitmask(1);
physicsBody_floor->setCategoryBitmask(1);
floor->setPhysicsBody(physicsBody_floor);

//add floor1 obstacl1 1
auto* floor1 = Sprite::create("9.png");
floor1->setAnchorPoint(Vec2(0, 0));
```


Search Solution Explorer (Ctrl+ü)
External Dependencies
Resources
Source Files
Classes
AppDelegate.cpp
AppDelegate.h
GameOverScene.cpp
GameOverScene.h
GameScene.cpp
GameScene.h
GameScene2.cpp
GameScene2.h
GameScene3.cpp
GameScene3.h
MainMenuScene.cpp
MainMenuScene.h
PauseScene2.h
PauseScene.cpp
PauseScene.h
PauseScene2.cpp