Projet : Jeu 2D

Encadré par :

Lotfi EL AACHAK

Ikram BEN ABDELOUAHAB

Réalise par :

Imane KHOUSSI

Groupe :2

Imane DOUAS

Groupe :1

Remerciement :

Avant tout développement nous voulons commencer par exprimer nos remerciements à notre

Prof EL AACHAK qui nous a donné l'opportunité de faire un travail pratique de développer un jeu pour enrichir et développer nos connaissances pratiques en programmation orienté objet en c++.

Assurer vous Monsieur que nous étions très motivés durant la réalisation de ce projet afin d'être au niveau attendu de votre part.

Nous espérons que ce travail sera à la hauteur de vos attentes.

Sommaire

Bibliographie :

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

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

https://gamefromscratch.com/cocos2d-x-c-game-programming-tutorialseries/

https://www.raywenderlich.com/1848-cocos2d-x-tutorial-forbeginners#toc-anchor-001

Introduction :

Le sujet de ce projet est le développement du jeu ***' Pico Park '*** avec Cocos2D-X.

Le développement de jeux, ou Game Development est le processus de création du jeu. Le développement du jeu consiste à exploiter un concept pour le déployer, le programmer, le concevoir, l'enregistrer, le mixer, le produire, le tester, etc. jusqu'à l'obtention d'un produit fini. Les tâches du développeur relèvent de la programmation, du codage, du rendu, de l'ingénierie et des tests du jeu (et de tous ses éléments : son, niveaux, personnages, etc.).

On commence par expliquer les concepts de base du Cocos2D-X.

Définitions :

Cocos2d-x :

Cocos2d est un framework libre en Python, permettant de développer des applications ou des jeux vidéo2.

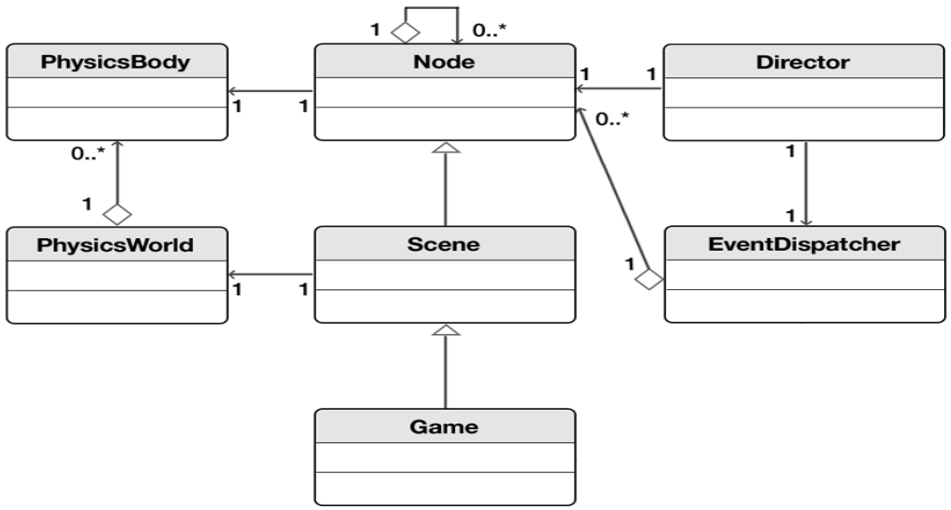
Des jeux comme FarmVille3, Geometry Dash ou Angry Birds Fight!4 ont été développés avec Cocos2D.

Logiciel Dev C++ :

Microsoft Visual Studio est une suite de logiciels de développement pour **Windows** et **mac OS** conçue par Microsoft.

Visual Studio est un ensemble complet d'outils de développement permettant de générer des applications web ASP.NET, des services web XML, des applications bureautiques et des applications mobiles. Visual Basic, Visual C++, Visual C# utilisent tous le même environnement de développement intégré (IDE), qui leur permet de partager des outils et facilite la création de solutions faisant appel à plusieurs langages. Par ailleurs, ces langages permettent de mieux tirer parti des fonctionnalités du framework .NET, qui fournit un accès à des technologies clés simplifiant le développement d'applications web ASP et de services web XML grâce à Visual Web Developer.

Cocos2d-x offre Scene, Transition, Sprite, Menu, Sprite3D, Audio et plus encore.



Scene :

Scenes définissent les différents modes de votre jeu. Si vous avez un ‘main menu’, quelque niveaux… vous pouvez les organiser et les parcouriez facilement avec Scene.

Chaque cocos2d-x application nécessite au moins une Scene.

Scenes sont des nœuds invisibles dans Cocos2d-X. Vous ne pouvez pas en fait voir une scène, mais vous pouvez voir la plupart de ses enfants, comme les Sprites, boutons, arrière-plans, etc.

Layer :

Un autre type de nœud Cocos2d-X invisible. Vous pouvez. Avoir plusieurs couches dans une scène, mais pour la plupart, vous n’aurez besoin que d’une seule couche.

L’ordre z le plus élevé est en haut.

Node :

Layers, ainsi que Scenes, Sprites et tout le reste dans Cocos2d-X, dérivent de Node.

Nodes donnent à chaque élément de Cocos2d-X une commune fonctionnalité. Tous les nodes ont une position, par exemple, qui peut être définie afin de déplacer le Node. Nodes ont également des anchor points, cameras, z order, rotation, scaling, flipping, skewing et on/off visibility.

Un Node a un parent possible et de nombreux enfants possibles.

Cela organise tous les éléments de Cocos2d-X dans une hiérarchie. La scène est au sommet de la hiérarchie, puis vient layer, puis les Sprites, les backgrounds, etc.

Sprites :

Sprites sont les objets que vous déplacez autour de l'écran. Nous pouvons les manipuler. Dans notre jeu le Sprite est la balle.

Sprites sont faciles à créer et ils ont des propriétés configurables comme : position, rotation, scale, opacity, color et plus encore.

Event Dispatcher :

Event Dispatch est un mécanisme permettant de répondre aux événements des utilisateurs.

- Event Listeners encapsulent votre code de traitement d'événement
- Event Dispatcher informe les auditeurs des événements des utilisateurs.
- Event Object contiennent des informations sur l'événement ***5 types of event listeners.***

EventListenerTouch - répond aux événements de contact

EventListenerKeyboard - répond aux événements du clavier

EventListenerAcceleration - répond aux événements d'accéléromètre

EventListenMouse - répond aux événements de la souris

EventListenerCustom - répond aux événements personnalisés

Présentation du projet :

Objectif du projet :

L'objectif de ce projet est de maitriser la programmation orientée objet en C++ par la mise en place d'un jeu vidéo 2D en utilisant le platform COCOS2D-X.

Concept du projet :

Roller Splat est que ce n'est pas le jeu le plus compliqué. Il semble très simple et son but est facile à comprendre – vous devez couvrir entièrement un niveau de peinture

Périmètre du projet :

Eléments du périmètre :

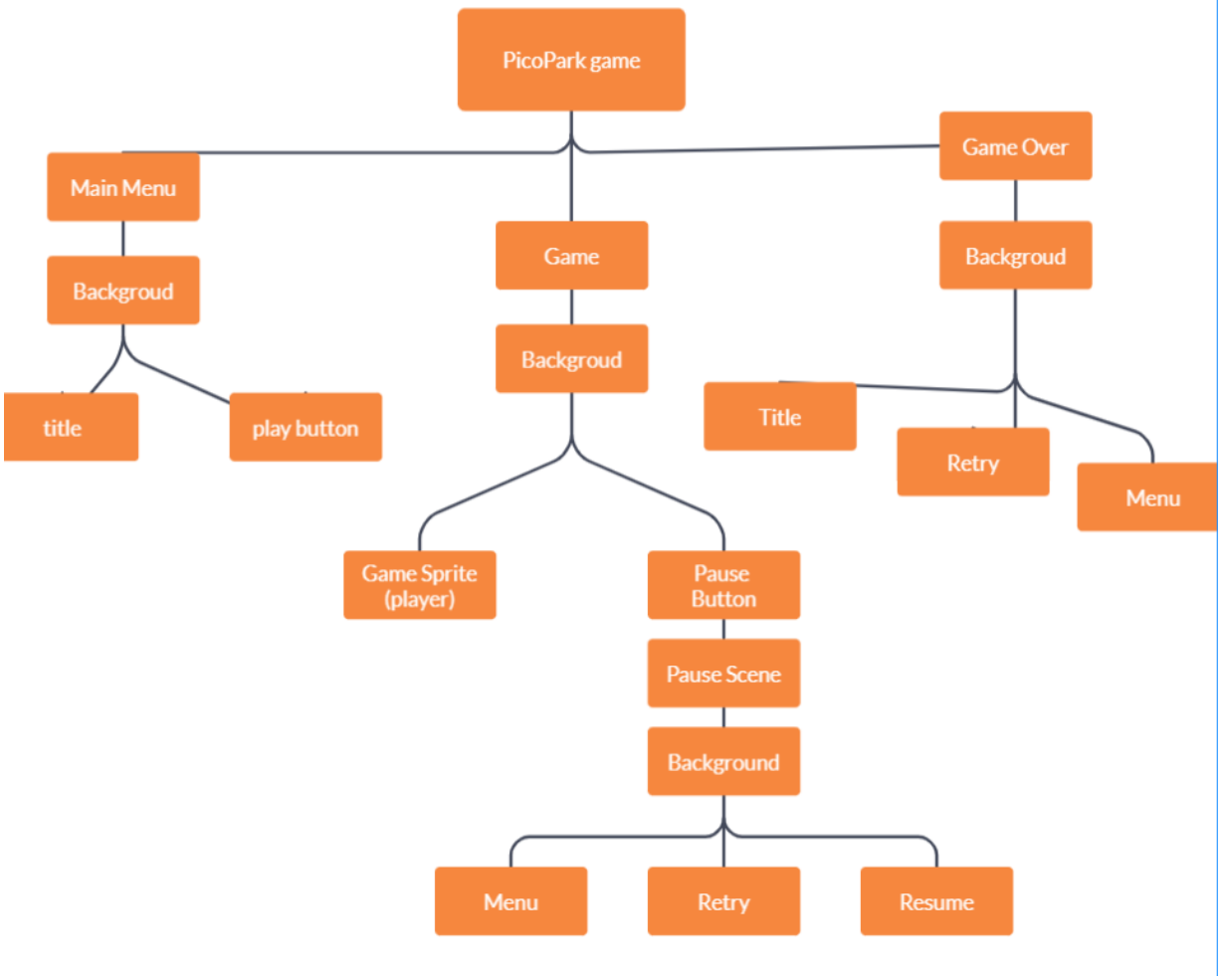
- ouvrer et exécuter la solution qui se trouve dans le dossier proj.win32

- Ajouter les images qu'on va utiliser dans le dossier recourses

- Changer le nom du programme HelloWorldScene.cpp/.h en GameScene.cpp/.h

- Ajouter des autres programmes (GameSprite.cpp/.h, GameOverScene.cpp/.h, MainMenuScene.cpp/.h, PauseScene.cpp/.h).cpp/.h, MainMenuScene.cpp/.h, PauseScene.cpp/.h)

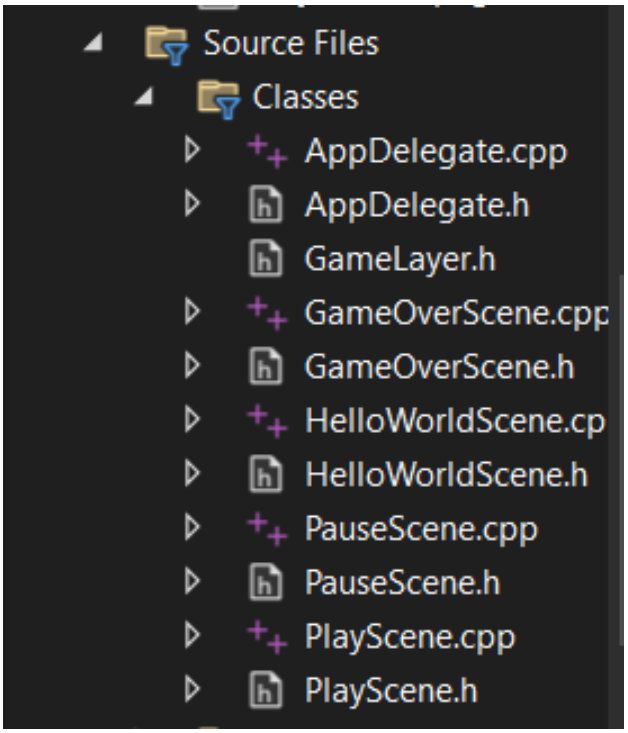
Spécifications fonctionnelles :



Spécifications techniques :

Compatibilité technique :

Le jeu est compatible à tout écran du PC.la taille est de 900 px*800 px ce qui rende le contenu plus visible et les instructions qui se trouvent à la rubrique droite lisible et claire.


Source Files
Classes
AppDelegate.cpp
AppDelegate.h
GameLayer.h
GameOverScene.cpp
GameOverScene.h
HelloWorldScene.cp
HelloWorldScene.h
PauseScene.cpp
PauseScene.h
PlayScene.cpp
PlayScene.h

Main Menu :

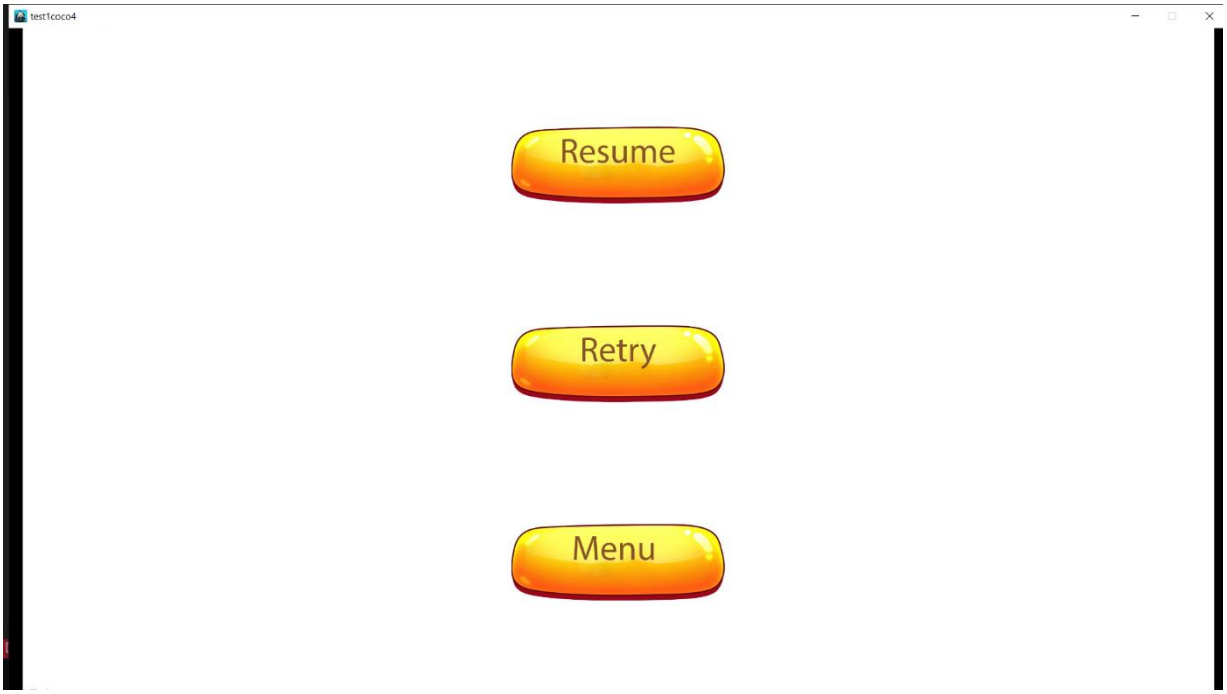
HelloWorldScene.h : on déclare la classe HelloWorld dans laquelle en implémente des attributs et des méthodes.

```
#ifndef __HELLOWORLD_SCENE_H__
#define __HELLOWORLD_SCENE_H__

#include "cocos2d.h"

class HelloWorld : 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(HelloWorld);
    void GoToPlayScene(Ref* pSender);

};

#endif // __HELLOWORLD_SCENE_H__
```

HelloWorldScene.cpp : on initialise les méthodes de la classe MainMenu.

```
auto menuTitle = MenuItemImage::create("Game_Title.png", "HelloWorld.png");
    auto playItem = MenuItemImage::create("Play_Button.jpg", "play.jpg",
CC_CALLBACK_1(HelloWorld::GoToPlayScene, this));

    menuTitle->setScale(2.5);
    playItem->setScale(2.5);

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



    // add "HelloWorld" splash screen"
    auto background = Sprite::create("backgroundmenu.jpg");
    if (background == nullptr)
    {
        problemLoading("'backgroundmenu.jpg'");
        auto label = Label::createWithTTF("Hello World", "fonts/Marker Felt.ttf",
24);
        label->setPosition(Vec2(origin.x + visibleSize.width / 2,
            origin.y + visibleSize.height - label->getContentSize().height));
        this->addChild(label, 1);
    }
    else
    {
        // position the sprite on the center of the screen
        background->setPosition(Vec2(visibleSize.width/2 + origin.x,
visibleSize.height/2 + origin.y));
        background->setScaleY(6.0);
        background->setScaleX(10);
        // add the sprite as a child to this layer
        this->addChild(background, -1);
    }
```

Pause Scene :



De même on a créé les fichiers PauseScene.h /.cpp.

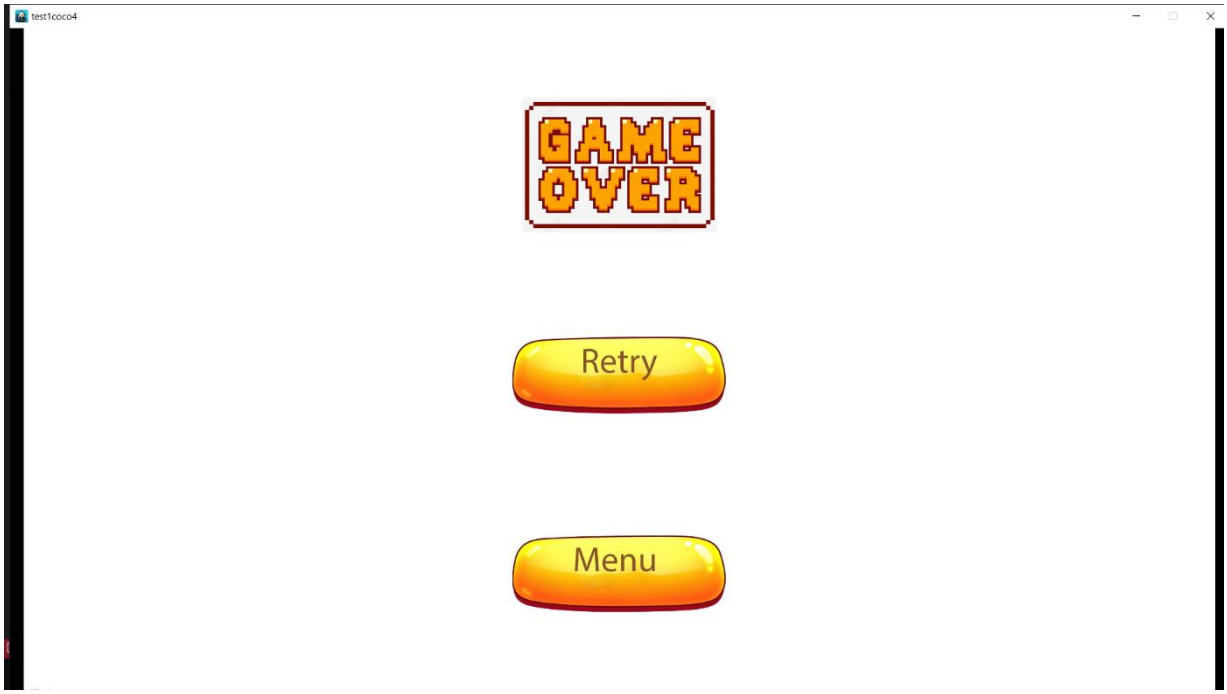
PauseScene.h : on déclare la classe Pause dans laquelle en implémente des attributs et des méthodes.

```
#ifndef __PAUSE_SCENE_H__
#define __PAUSE_SCENE_H__

#include "cocos2d.h"

class PauseScene : public cocos2d::Scene
{
public:
    // there's no 'id' in cpp, so we recommend returning the class instance
pointer
    static cocos2d::Scene* createScene();

    // Here's a difference. Method 'init' in cocos2d-x returns bool, instead of
returning 'id' in cocos2d-iphone
    virtual bool init();

    CREATE_FUNC(PauseScene);
    void Resume(Ref* pSender);//will be called when player clicks on resume button
from the Pause scene to pop the Pause scene off the stack.
    void GoToHelloWodldScene(Ref* pSender); //will be called when player  clicks
on menu button from the Pause scene to replace the current scene with the Main
Menu scene while popping all the scenes off the stack.
    void Retry(Ref* pSender); //will be called when the player clicks on the retry
button from the Pause scene to replace the current scene with the Game scene while
popping all the scenes off the stack
};

#endif
```

PauseScene.cpp : on initialise les méthodes de la classe

```
auto background = Sprite::create("bggame.jpg");
	background->setPosition(Vec2(visibleSize.width / 2 + origin.x,
visibleSize.height / 2 + origin.y));
	background->setScaleY(9.0);
	background->setScaleX(9.0);
	this->addChild(background, -4);
	auto resumeItem =
		MenuItemImage::create("resume.png",
			"resume.png",
			CC_CALLBACK_1(PauseScene::Resume, this));
	auto retryItem =
		MenuItemImage::create("retry.png",
			"resume.png",
			CC_CALLBACK_1(PauseScene::Retry, this));
	auto mainMenuItem =
		MenuItemImage::create("menu.png",
			"menu.png",
			CC_CALLBACK_1(PauseScene::GoToHelloWodldScene, this));
	auto menu = Menu::create(resumeItem, retryItem, mainMenuItem,
		NULL);
	menu->alignItemsVerticallyWithPadding(visibleSize.height / 4);
	this->addChild(menu);

	retryItem->setScale(2.5);
		mainMenuItem->setScale(2.5);
		resumeItem->setScale(2.5);
```

Game Over Scene :



On a créé les fichiers GameOverScene.h /.cpp.

GameOverScene.h : on déclare la classe GameOver dans laquelle en implémente des attributs et des méthodes.

```
#ifndef __GAMEOVER_SCENE_H__
#define __GAMEOVER_SCENE_H__

#include "cocos2d.h"

class GameOver : public cocos2d::Scene
{
public:
    // there's no 'id' in cpp, so we recommend returning the class instance
pointer
    static cocos2d::Scene* createScene();

    // Here's a difference. Method 'init' in cocos2d-x returns bool, instead of
returning 'id' in cocos2d-iphone
    virtual bool init();
    CREATE_FUNC(GameOver);
    void GoToPlayScene(Ref* pSender);//will be called when the player clicks on
the retry button from the game over s to replace it with game s
    void GoToHelloWorldScene(Ref* pSender);//will be called when the player clicks
on the menu button from the game over s to replace it with menu s
};

#endif // __PLAY_SCENE_H__
```

GameOverScene.cpp : on initialise les méthodes de la classe GameOver

```
#include "GameOverScene.h"
```

```cpp
#include "HelloWorldScene.h"
#include "PlayScene.h"

USING_NS_CC;

Scene* GameOver::createScene()
{
    return GameOver::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 GameOver::init()
{

    if (!Scene::init())
    {
        return false;
    }
    CCLOG("HEy welcome to gameover scene");
    Size visibleSize = Director::getInstance()->getVisibleSize();
    Point origin = Director::getInstance()->getVisibleOrigin();


    // hna atji mktoba game over
    auto menuTitle = MenuItemImage::create("game_over.png",
                                           "game_over.png");
   //hna atji photo d retry
    auto retryItem =  MenuItemImage::create("HelloWorld.png",

"GameOverScreen/Retry_Button(Click).png",

        CC_CALLBACK_1(GameOver::GoToPlayScene, this));
    //hna atji photo li atsardo n menu
    auto mainMenuItem =
        MenuItemImage::create("GameOverScreen/Menu_Button.png",
                              "GameOverScreen/Menu_Button(Click).png",
                              CC_CALLBACK_1(GameOver::GoToHelloWorldScene,
this));
    auto menu = Menu::create(menuTitle, retryItem, mainMenuItem,
        NULL);
    menu->alignItemsVerticallyWithPadding(visibleSize.height / 4);
    this->addChild(menu);

    menuTitle->setScale(2.5);
    retryItem->setScale(2.5);
    mainMenuItem->setScale(2.5);

    return true;

}
void GameOver::GoToPlayScene(cocos2d::Ref* pSender)
{
    auto scene = PlayScene::createScene();

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

```
}
void GameOver::GoToHelloWorldScene(cocos2d::Ref* pSender)
{
    auto scene = HelloWorld::createScene();

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

Play Scene :

On définit dans PlayScene.h les méthodes dans la classe PlayScene et on les initialiser dans PlayScene.cpp, ainsi que tous les autres Classe

Utilisation de on key pressed pour deplacer le joueur

```
eventListener->onKeyPressed = [](EventKeyboard::KeyCode keyCode, Event* event) {

        bool leftkeypressed = false;
        bool rightkeypressed = false;

        Vec2 loc = event->getCurrentTarget()->getPosition();
        auto newPosition = loc + Vec2(0, 30);
        auto jumpHeight = 100;
        auto jumpDuration = 0.5f;
        auto jumpActionright = MoveTo::create(jumpDuration, loc + Vec2(30,
jumpHeight));
        auto jumpActionleft = MoveTo::create(jumpDuration, loc + Vec2(-30,
jumpHeight));

        auto downDuration = 0.5f;
        auto downActionright = MoveTo::create(downDuration, loc + Vec2(30, 0));
        auto downActionleft = MoveTo::create(downDuration, loc + Vec2(-30, 0));
        auto sequenceright = Sequence::create(jumpActionright, downActionright,
NULL);
        auto sequenceleft = Sequence::create(jumpActionleft, downActionleft,
NULL);
        switch (keyCode) {
        case EventKeyboard::KeyCode::KEY_LEFT_ARROW:
        case EventKeyboard::KeyCode::KEY_A:
            event->getCurrentTarget()->setScaleX(-1);
```

```
            event->getCurrentTarget()->setPosition(loc.x -30, loc.y);

            break;
        case EventKeyboard::KeyCode::KEY_RIGHT_ARROW:
        case EventKeyboard::KeyCode::KEY_D:

            event->getCurrentTarget()->setScaleX(1);
            event->getCurrentTarget()->setPosition(loc.x +30, loc.y);
            break;


        case EventKeyboard::KeyCode::KEY_SPACE:
        case EventKeyboard::KeyCode::KEY_UP_ARROW:
        case EventKeyboard::KeyCode::KEY_W:
            if(event->getCurrentTarget()->getScaleX() == 1)
                event->getCurrentTarget()->runAction(sequenceright);


            //event->getCurrentTarget()->runAction(sequence);;
            else
                event->getCurrentTarget()->runAction(sequenceleft);

            break;
        }

    };
```

On a 3 cas

Le clic sur le bouton up ou bien w ou bien espace pour sauter

Le clic sur le bouton droit ou d pour se deplacer a droit

Le clic sur le bouton Gauche pour se deplacer a