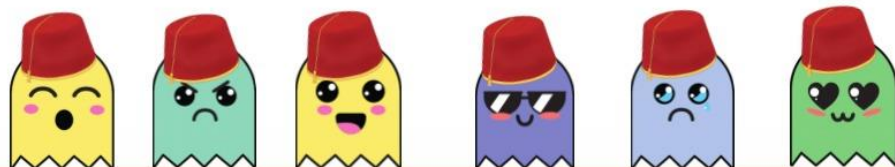


# PROJET DE MODULE

## C++

# PICO PARK



Prepared by : AYOUB ET-TOUBI && ABDERRAZZAK EL BOURKADI

Encadre par : EL ACHAK LOTF

PLAN .....

I. Introduction au cocos2d-x

II. Proj.win32

III. PICO PARK

IV. COTE DESIGN

V. Conclusion



---

## I. Introduction au cocos2d-x

### I) Definition :

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 et Js. En utilisant la version la plus récente de Cocos2D-x 3.6.2 , vous pouvez cibler Windows, Mac, Linux, iOS et Android.

Dans notre projet on travaille avec langage c++ .

## 2) Cocos histoire :

Retour en 2008, Cocos est venu à être, nommé d'après la ville de Los Cocos, Argentine, au cas où vous vous demandiez d'où exactement le nom vient. Il a commencé par un rassemblement de développeurs Python, vous l'avez deviné, Los Cocos. Comme vous pouvez le deviner du fait qu'il a été lancé par un groupe de développeurs Python, Cocos a commencé à être écrit en Python.

Puis est venu ce petit téléphone appelé l'iPhone, et une version de Cocos2D a été apporté à ObjectiveC pour une utilisation sur iOS, le cocos2d-iphone bien nommé. Un certain nombre d'applications Cocos2d-iphone développé a commencé à apparaître sur l'iPhone, y compris StickWars, qui a frappé le numéro 1 sur l'App Store.

Maintenant, nous touchons le bouton avance rapide sur l'histoire et voir que Cocos2d est porté sur un certain nombre de plates-formes. Un de ces ports était bien sûr Cocos2d-x, qui était un port de Cocos2D à C++, le sujet de notre tutoriel ici. Cocos2d-x lui-même a également engendré un certain nombre de ports, y compris HTML et XNA. En cours de route, un certain nombre d'outils ont également été développés, y compris un éditeur nommé CocosStudio (lui-même le spawn d'un certain nombre de projets enfants ) et CocosCodeIDE et IDE pour Lua et JavaScript script dans Cocos2d-x.

## 3) Cocos est Manipulation des scènes :

Sans fonctionnalité, se déplacer entre les scènes n'est pas possible et elles seraient inutiles. Cocos2d-x fournit d'excellentes méthodes pour passer d'une scène à l'autre. Cocos2d-x utilise une pile pour gérer les scènes, qui est un système Last-In First-Out (LIFO) qui exécute la dernière scène de la pile. Voici les principales méthodes utilisées :

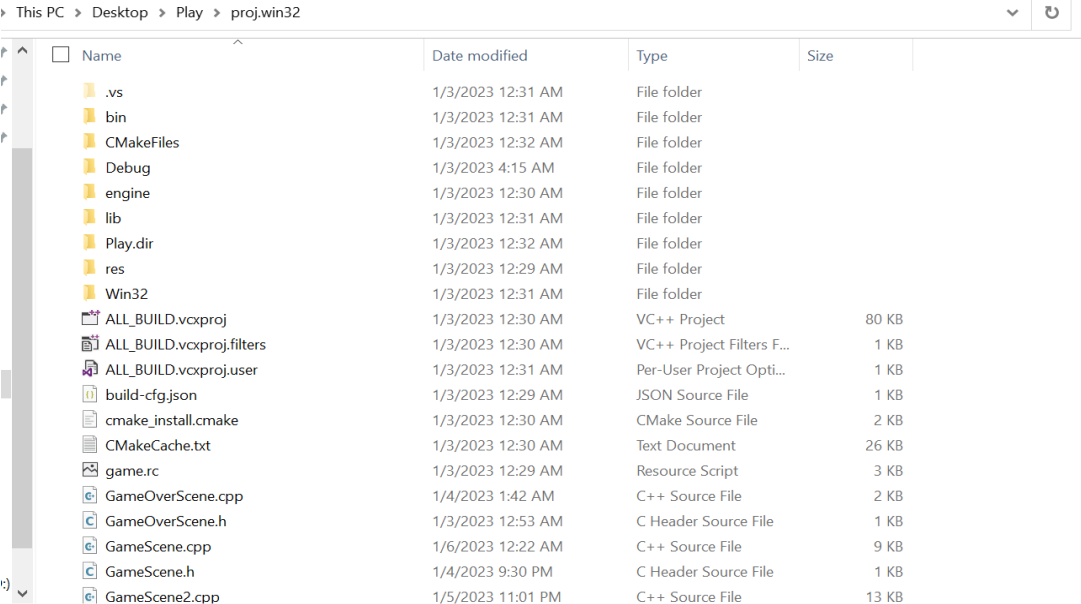
- **Pushing a scene (Pousser une scène)** : Cette méthode pousse une scène particulière sur la pile tout en conservant la scène courante mais met en pause son exécution. Un exemple réel est que lorsque vous cliquez sur un bouton de pause, la scène de pause sera

poussée sur la pile, alors que la scène de jeu existe toujours .

- Popping a scene ( **Éclatement d'une scène** ) : Cette méthode supprime la scène supérieure/courante de la pile. Un exemple du monde réel pour cela est lors de la reprise d'une scène en pause, la scène actuelle (la scène en pause) est supprimée et retourne à la scène de jeu
- Replacing a scene ( **Remplacer une scène** ) : Cette méthode remplace la scène actuelle par une nouvelle scène, essentiellement éclater la scène actuelle et puis pousser la nouvelle scène sur la pile. Un exemple réel pour cela est lorsque le joueur meurt et va à la scène Game Over.

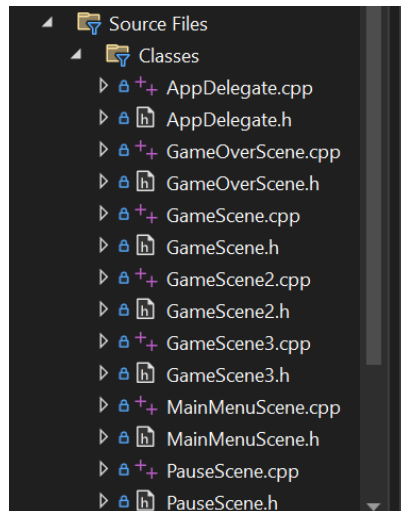
## II. Proj.win32

Comme vous pouvez le voir, chaque dossier contient tout le code spécifique à la plateforme, les ressources et surtout, les fichiers de projet pour chaque plateforme. Proj.win32 contient en outre des dossiers spécifiques à chaque plateforme.



Name	Date modified	Type	Size
.vs	1/3/2023 12:31 AM	File folder	
bin	1/3/2023 12:31 AM	File folder	
CMakeFiles	1/3/2023 12:32 AM	File folder	
Debug	1/3/2023 4:15 AM	File folder	
engine	1/3/2023 12:30 AM	File folder	
lib	1/3/2023 12:31 AM	File folder	
Play.dir	1/3/2023 12:32 AM	File folder	
res	1/3/2023 12:29 AM	File folder	
Win32	1/3/2023 12:31 AM	File folder	
ALL_BUILD.vcxproj	1/3/2023 12:30 AM	VC++ Project	80 KB
ALL_BUILD.vcxproj.filters	1/3/2023 12:30 AM	VC++ Project Filters F...	1 KB
ALL_BUILD.vcxproj.user	1/3/2023 12:31 AM	Per-User Project Opti...	1 KB
build-cfg.json	1/3/2023 12:29 AM	JSON Source File	1 KB
cmake_install.cmake	1/3/2023 12:30 AM	CMake Source File	2 KB
CMakeCache.txt	1/3/2023 12:30 AM	Text Document	26 KB
game.rc	1/3/2023 12:29 AM	Resource Script	3 KB
GameOverScene.cpp	1/4/2023 1:42 AM	C++ Source File	2 KB
GameOverScene.h	1/3/2023 12:53 AM	C Header Source File	1 KB
GameScene.cpp	1/6/2023 12:22 AM	C++ Source File	9 KB
GameScene.h	1/4/2023 9:30 PM	C Header Source File	1 KB
GameScene2.cpp	1/5/2023 11:01 PM	C++ Source File	13 KB

Le dossier ressource est un dépôt populaire de tous les différents actifs que votre jeu utilisera, tels que les graphiques, audio, etc. Peut-être que le dossier Classes est le plus important de tous, c'est là que votre code non plateforme spécifique va! Maintenant le contenu devrait ressembler à:



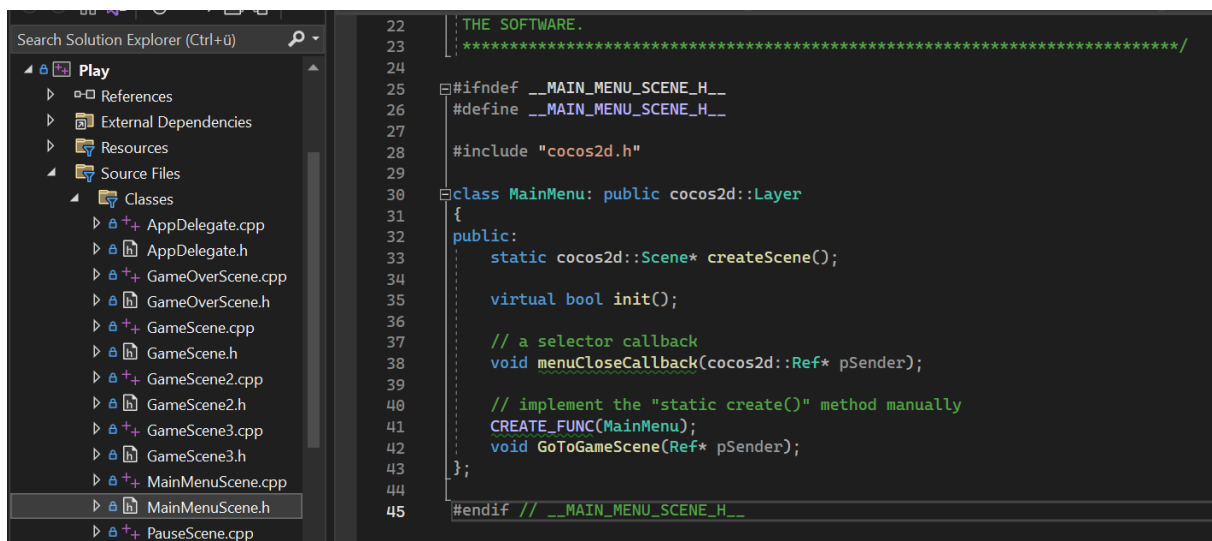
### III. PICO PARK

#### I. Ecran :

Alors 1<sup>er</sup> chose que n a fait change la taille d'affichage de jeux avec le code :

```
if (glview) {  
#if (CC_TARGET_PLATFORM == CC_PLATFORM_WIN32) || (CC_TARGET_PLATFORM == CC_PLATFORM_MAC) || (CC_TARGET_PLATFORM == CC_
```

Après ca en renommé les fichier HelloWorld.cpp et HelloWorld.h par MainMenuScene.h et MainMenuScene.h est en écrire les fonctions nécessaire de class MainMenu en MainMenuScene.h



Après nous allons écrire le code pour changer image d'écran pour ajouter Button pour aller à scène de level est lie avec la scène de level 1

```
#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();
}
```

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

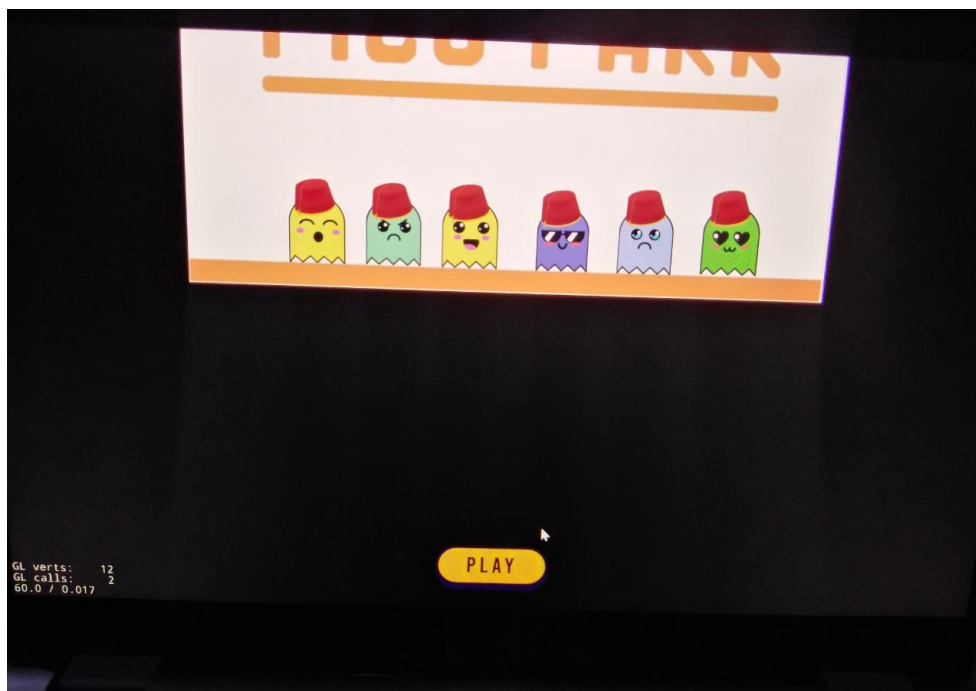
auto visibleSize = Director::getInstance()->getVisibleSize();
Vec2 origin = Director::getInstance()->getVisibleOrigin();

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

```
}

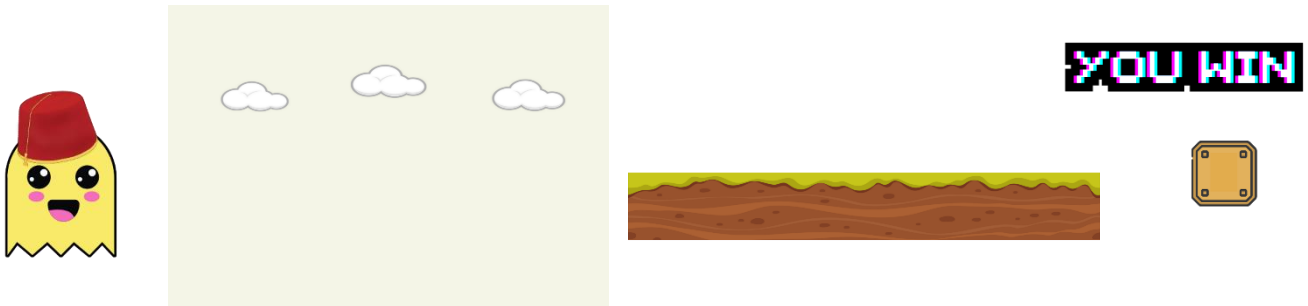
void MainMenu::GoToGameScene(Ref* pSender)
{
    auto scene = GameScreen3::createScene();

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



## 2. LEVEL

En design notre joueur et notre scene de level I



### a. Gestion des événements clavier :

nous créons un EventListener, dans ce cas un EventListenerKeyboard, implémentez le gestionnaire d'événement onKeyPressed. Le premier paramètre passe est l'enum EventKeyboard::KeyCode, qui représente la cle enfonce e. La deuxième valeur était la cible de l'événement, dans ce cas notre sprite. Nous utilisons le pointeur Event pour obtenir le Node cible et mettre à jour sa position dans une direction en fonction de la touche enfoncée. Enfin, nous câblons le \_eventDispatcher de notre scène pour recevoir les événements. Rien de vraiment inattendu ici.

```
// Create a keyboard event listener
auto keyboardListener = EventListenerKeyboard::create();
keyboardListener->onKeyPressed = CC_CALLBACK_2(GameScreen::onKeyPressed, this);
keyboardListener->onKeyReleased = CC_CALLBACK_2(GameScreen::onKeyReleased, 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(0.5f, Vec2(50, 100), 15.0f, 1);
        auto easeAction = EaseOut::create(action1, 2.0f);
        player->runAction(easeAction);
    }

    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), 20.0f, 1);
        MoveBy* moveAction = MoveBy::create(1.2, Vec2(-70, 0));
        RepeatForever* repeatAction = RepeatForever::create(moveAction);
        player->runAction(repeatAction);
    }
};

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();
    }
};
```



## b. Physique

On déclare les fonction de physique pour le scènes

```
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;
}
```

Nous ajoutons les physique pour le joueur et les obstacles

```
//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);
//physique floor 1
```

et en manipuler le size de physique avec le size () et la fonction  
setContactTestBitmask(1) pour détecte le contacte avec les obstacles

## c. Le contact et les évènements

```
//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 d'yal player
player->setName("player");

player->setTag(10);
this->addChild(player, 2);
//physique player
```

La fonction setTag(10) donne un valeur pour le contact de joueur

```

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);

```

En la en détecté qu'il contact doit exécuté

Si en a quelque objet avec fonction setTag(10) faire un contacte avec objet de même setTag(10) doit aller au scène de GameOver et si faire le contact avec un objet de fonction setTag(x) avec  $x \neq 0$  le joueur aller au GameScreen2 (Level 2)

### 3. PauseSceneMenu :

O créons class MainMenu dans MainMenuScene.h et en déclarons les fonctions nécessaire

```

#ifndef __PAUSE_SCENE_H__
#define __PAUSE_SCENE_H__

#include "cocos2d.h"

class PauseMenu : 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(PauseMenu);

    void Resume(Ref* pSender);
    void GoToMainMenuScene(Ref* pSender);
    void Retry(Ref* pSender);
};

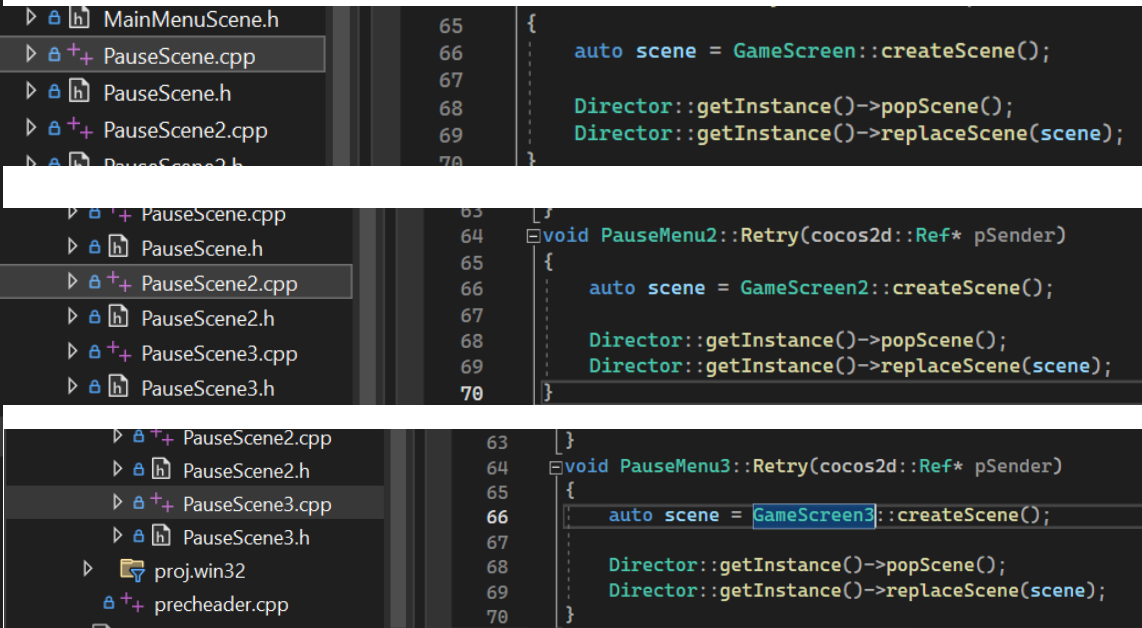
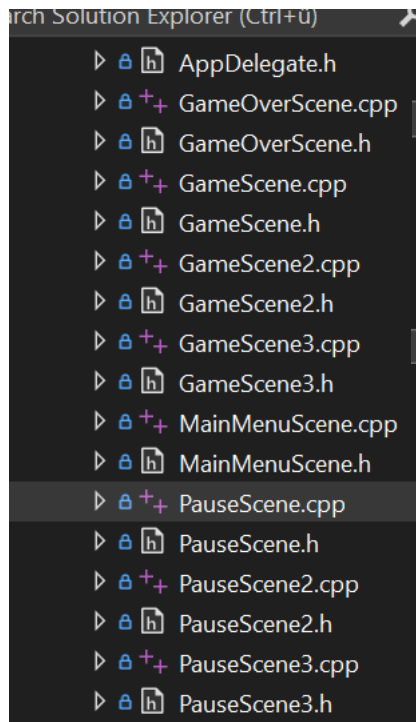
#endif // __HELLOWORLD_SCENE_H__

```

Et a MainMenuScene.cpp en faire l'interface de PauseMenu et les Button pour aller a MainMenu ou pour continue ou pour redémarre level

PauseScene.cpp	MainMenuScene.cpp	GameScene3.cpp
<pre> 4 5 6 7 USING_NS_CC; 8 9 Scene* PauseMenu::createScene() 10 { 11     auto scene = Scene::create(); 12     auto layer = PauseMenu::create(); 13     scene-&gt;addChild(layer); 14     return scene; 15 } 16 17 18 19 bool PauseMenu::init() 20 { 21     //////////////// 22     // 1. super init first 23     if (!Layer::init()) 24     { 25         return false; 26     } 27 28     auto visibleSize = Director::getInstance()-&gt;getVisibleSize(); 29     Vec2 origin = Director::getInstance()-&gt;getVisibleOrigin(); 30 31 32 33 </pre>	<pre> auto resumeItem =     MenuItemImage::create("Retry.png",         "Retry.png",         CC_CALLBACK_1(PauseMenu::Resume, this)); auto retryItem =     MenuItemImage::create("Retry.png",         "Retry.png",         CC_CALLBACK_1(PauseMenu::Retry, this)); auto mainMenuItem =     MenuItemImage::create("Menu.png",         "Menu.png",         CC_CALLBACK_1(PauseMenu::GoToMainMenuScene, this)); auto menu = Menu::create(resumeItem, retryItem, mainMenuItem,     NULL); menu-&gt;alignItemsVerticallyWithPadding(visibleSize.height / 4); this-&gt;addChild(menu); return true; }  void PauseMenu::Resume(cocos2d::Ref* pSender) {     Director::getInstance()-&gt;popScene(); }  void PauseMenu::GoToMainMenuScene(cocos2d::Ref* pSender) {     auto scene = MainMenu::createScene();      Director::getInstance()-&gt;popScene();     Director::getInstance()-&gt;replaceScene(scene); }  void PauseMenu::Retry(cocos2d::Ref* pSender) {     auto scene = GameScreen::createScene(); </pre>	<pre> auto resumeItem =     MenuItemImage::create("Retry.png",         "Retry.png",         CC_CALLBACK_1(PauseMenu::Resume, this)); auto retryItem =     MenuItemImage::create("Retry.png",         "Retry.png",         CC_CALLBACK_1(PauseMenu::Retry, this)); auto mainMenuItem =     MenuItemImage::create("Menu.png",         "Menu.png",         CC_CALLBACK_1(PauseMenu::GoToMainMenuScene, this)); auto menu = Menu::create(resumeItem, retryItem, mainMenuItem,     NULL); menu-&gt;alignItemsVerticallyWithPadding(visibleSize.height / 4); this-&gt;addChild(menu); return true; }  void PauseMenu::Resume(cocos2d::Ref* pSender) {     Director::getInstance()-&gt;popScene(); }  void PauseMenu::GoToMainMenuScene(cocos2d::Ref* pSender) {     auto scene = MainMenu::createScene();      Director::getInstance()-&gt;popScene();     Director::getInstance()-&gt;replaceScene(scene); }  void PauseMenu::Retry(cocos2d::Ref* pSender) {     auto scene = GameScreen::createScene(); </pre>

Nous flairons trois pauseScene (Avec même code source Just déferent a scène de redémarre level )pour lie chaque PauseScene avec leur SceneScéen (Level) Par ce que en chaque level besoin action déferent de Pause



#### 4. GameOver

Cette scène exécuter si le joueur perd afficher photo qui indique le perd et button pour aller au Menu et button pour essayer un autre une fois

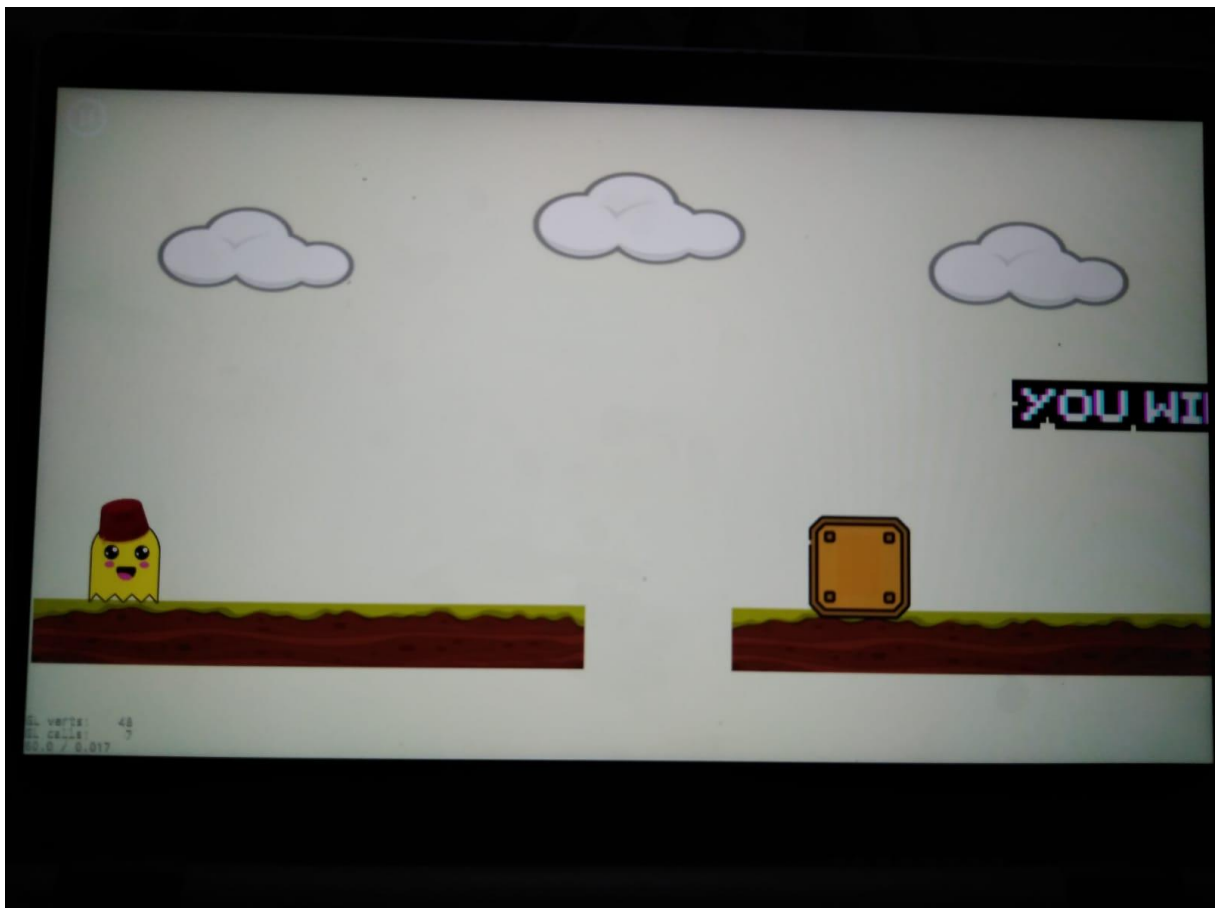


## 5. Mouvement des objets

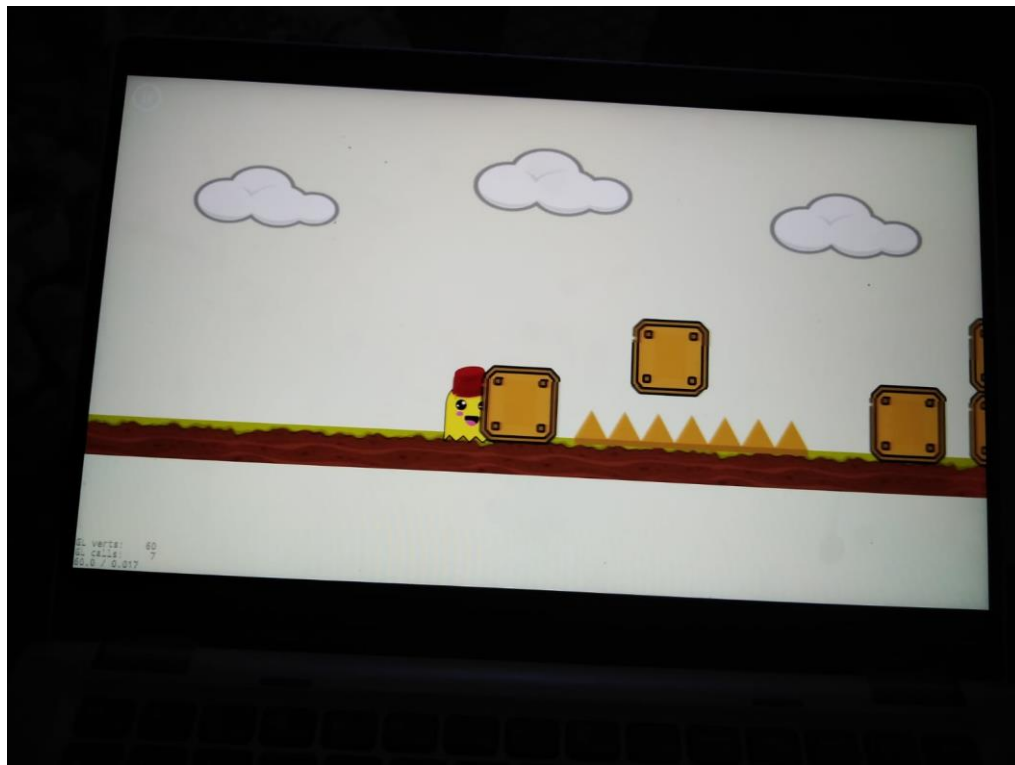
```
//scroll  
  
auto* acc = MoveBy::create(0.07 * visibleSize.width, Point(-visibleSize.width * 4, 0));  
floor->runAction(acc->clone());  
floor1->runAction(acc->clone());  
floor4->runAction(acc->clone());  
floor3->runAction(acc->clone());  
floor2->runAction(acc->clone());  
floor5->runAction(acc->clone());  
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());
```

## IV. LEVELS

### 1.Level 1



### 2.Level 2



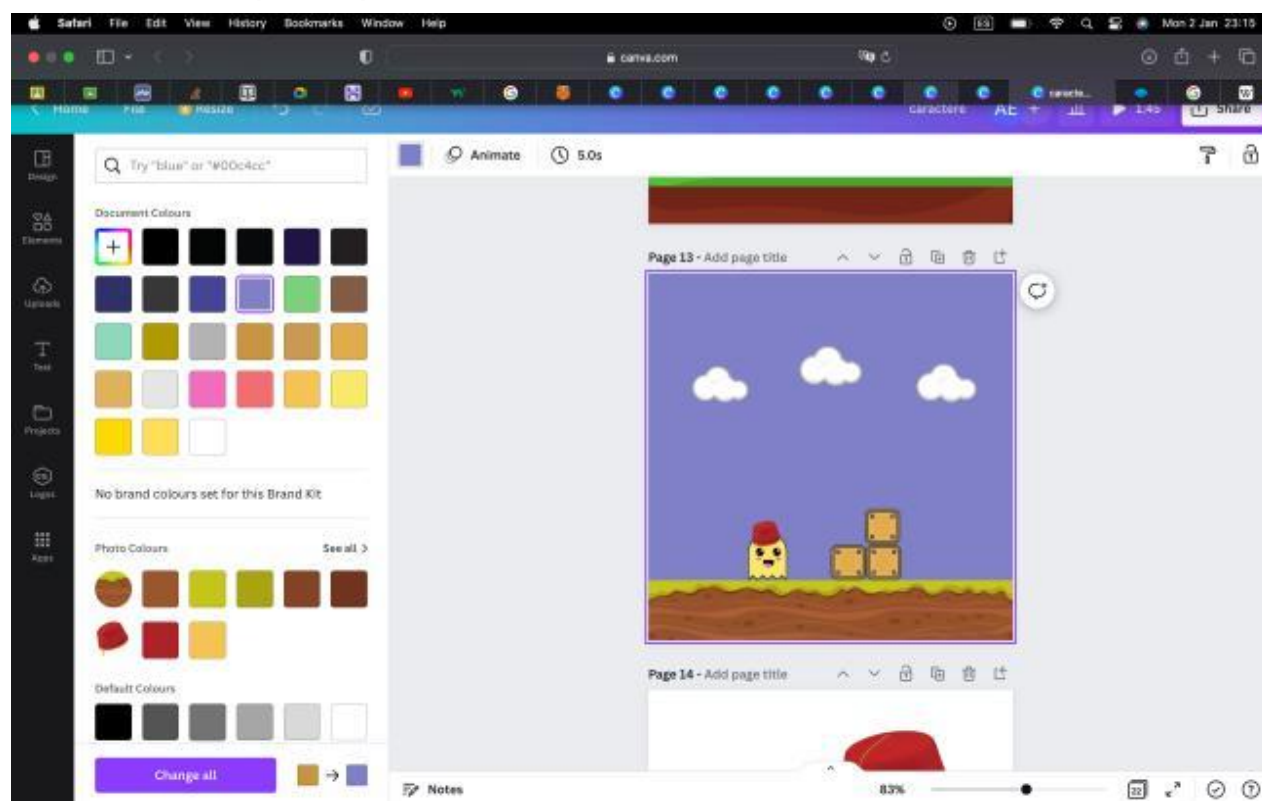
### 3.Level 3



## V. COTE DESIGN

Tous le design est un effort personnel .

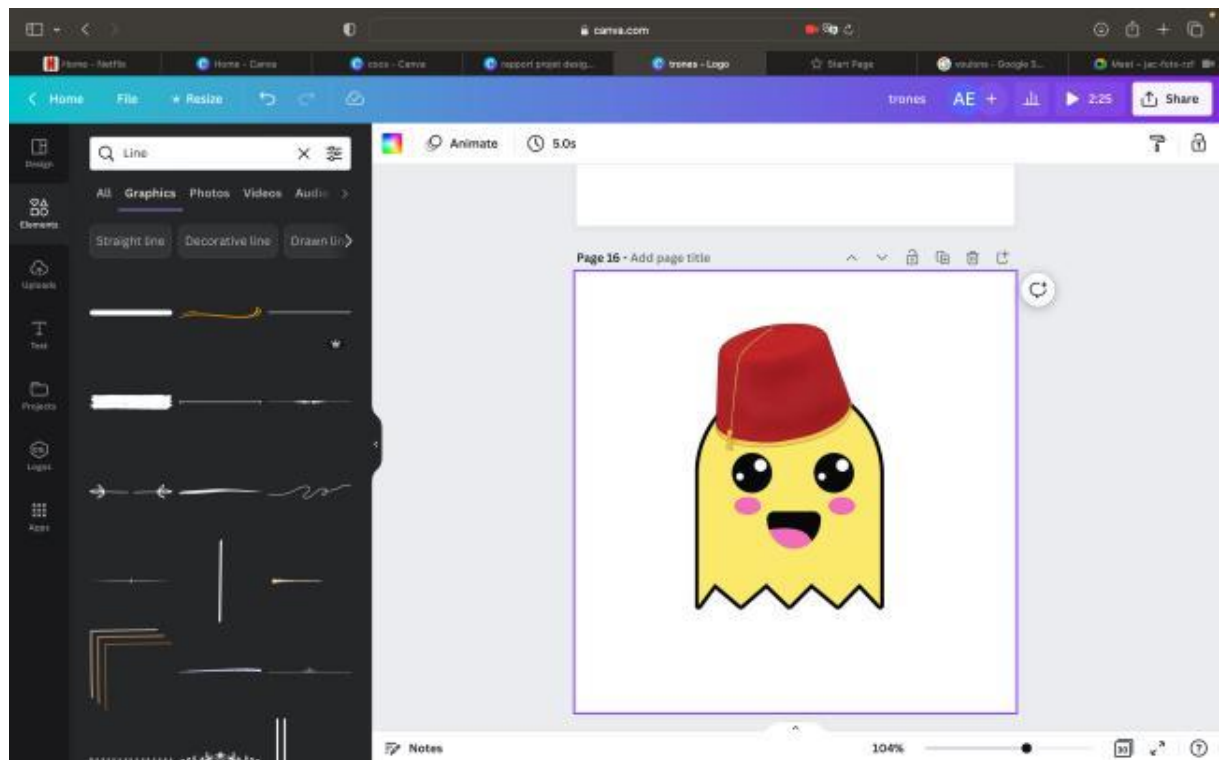
nous utilisons dans ce projet canva premium qui est est une société australienne d'informatique, qui conçoit en 2010 et quidifuse le logiciel de conception **graphique** en ligne Canva et ses dérivés payants (Canva Pro, Canva for Enterprise) ou non (Canva for Education, Canva for Nonprofits).



Nous voulions d'ajouter une petite touche marocaine traditionnel sur notre jeux c'est pour ça nous avons ajoute un "**tarbouche**" marocain sur le caractere

:





## VI. Conclusion

dans ce projet Nous avons découvert le domaine de programmation des jeux, nous avons appris comment faire les scènes et comment manipuler les scène est plusieurs chose de valeur ajouter a notre parcours scolaire