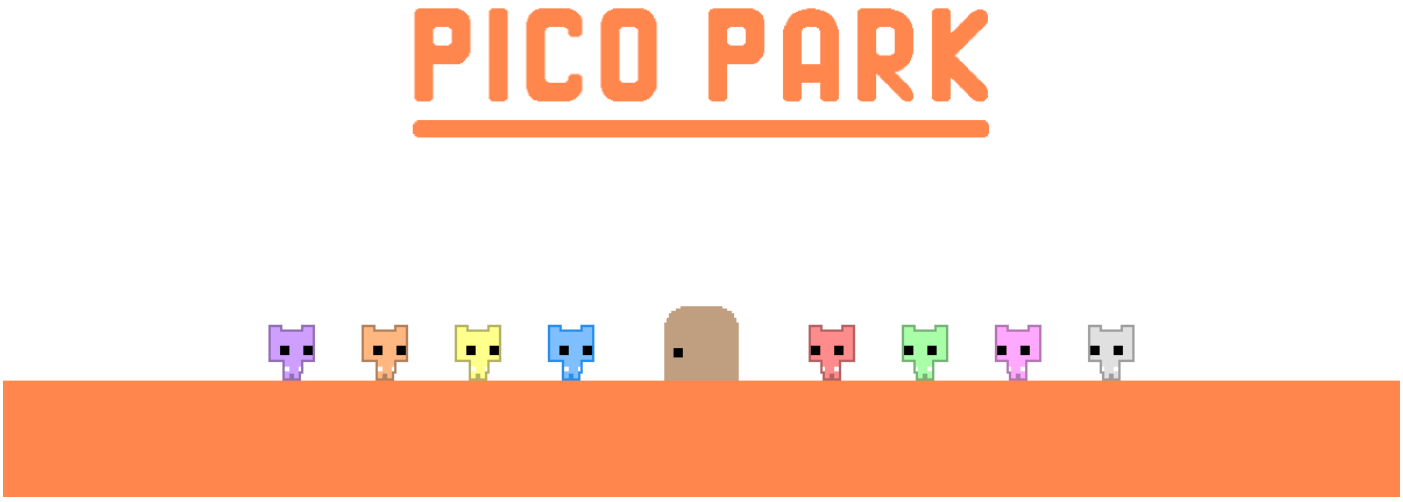
Université Abdelmalek Essaadi
Faculté ses Sciences et techniques de Tanger
Département Génie Informatique
LST GI S5
Algorithmique et POO C++

RAPPORT DU JEU :
PICO PARK (MOTEUR COCOS2D ,C++)



REALISE PAR : EL GHAZI LOUBNA ET EL MRABET MOHAMED
ENCADRE PAR : LOTFI ELAACHAK ET IKRAM BEN ABD LOUAHAB

Présentation du moteur du jeu et du langage de programmation utilises :

1. Cocos2d

Cocos2d est un framework logiciel libre. Il peut être utilisé pour créer des jeux, des applications et d'autres programmes interactifs basés sur une interface graphique multiplateforme. Cocos2d contient de nombreuses branches dont les plus connues sont Cocos2d-objc, Cocos2d-x, Cocos2d-html5 et Cocos2d-XNA. Il existe des éditeurs indépendants dans la communauté cocos2d, tels que ceux qui contribuent dans les domaines de l'édition de SpriteSheet, de l'édition de particules, de l'édition de polices et de l'édition de Tilemap, ainsi que des éditeurs mondiaux tels que SpriteBuilder et CocoStudio.

2. c++

C++ est un langage multiplateforme qui peut être utilisé pour créer des applications hautes performances.
C++ a été développé par Bjarne Stroustrup, comme une extension du langage C.
C++ donne aux programmeurs un haut niveau de contrôle sur les ressources système et la mémoire.
Le langage a été mis à jour 4 fois en 2011, 2014, 2017 et 2020 en C++11, C++14, C++17, C++20.

Pourquoi utiliser C++ ??

C++ est l'un des langages de programmation les plus populaires au monde.
C++ se trouve dans les systèmes d'exploitation, les interfaces utilisateur graphiques et les systèmes embarqués d'aujourd'hui.
C++ est un langage de programmation orienté objet qui donne une structure claire aux programmes et permet de réutiliser le code, réduisant ainsi les coûts de développement.
C++ est portable et peut être utilisé pour développer des applications pouvant être adaptées à plusieurs plates-formes.

3.VScode

Visual Studio est un environnement de développement intégré (IDE) de Microsoft. Il est utilisé pour développer des programmes informatiques, notamment des sites Web, des applications Web, des services Web et des applications mobiles. Visual Studio utilise les plates-formes de développement de logiciels Microsoft telles que Windows API, Windows Forms, Windows Presentation Foundation, Windows Store et Microsoft Silverlight. Il peut produire à la fois du code natif et du code managé.

Le processus de développement & les options développées

Le jeu 2d développé se compose de 3 Levels(scènes) essentiels :

- Le 1 er level basique, le 2 -ème et le 3 -ème. Dont la difficulté de jeux croit en passant chaque level .
- A cote des SCENES :Game over , Win et main menu.
- Celle de main menu est la scène principale .On peut dire c'est le point de départ de notre jeux .
- L' appel de Gameover si le joueur échoue en jeux.
- Celle de Win est appelée à la fin du jeux (level 3) pour féliciter le joueur .

1-LA SCENE DE : APPDELEGATE

APPdelegate.h :

Contient la definition de la classe appdelegate et les fonctions utilisees en appdelegate cpp.

```
#pragma once

#include "cocos2d.h"

class  AppDelegate : private cocos2d::Application
{
public:
    AppDelegate();
    virtual ~AppDelegate();

    virtual bool applicationDidFinishLaunching();
    virtual void applicationDidEnterBackground();
    virtual void applicationWillEnterForeground();

};
```

Appdelegate.cpp

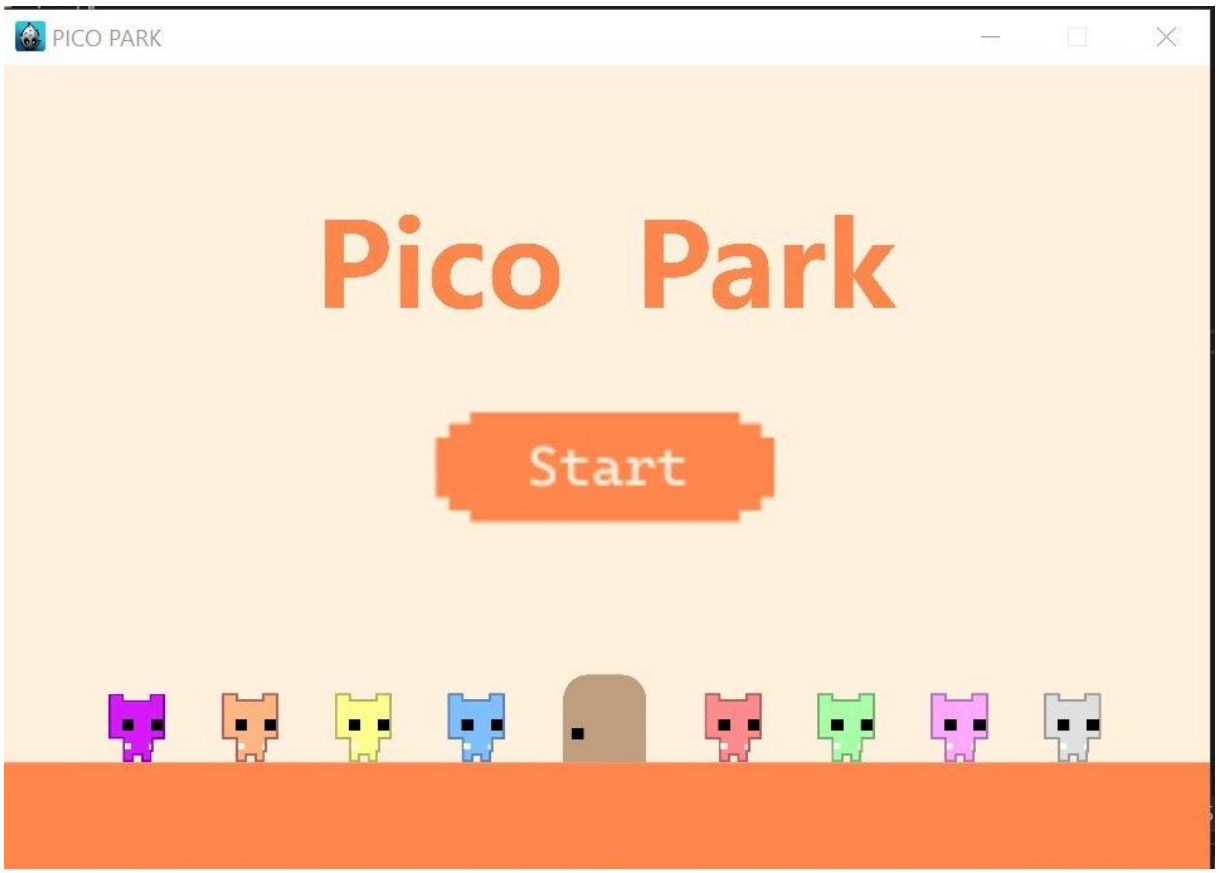
L appel de la classe et les fonctions utilisees :

Role : resolution des dimensions de l ecran, modifer le nom du HELLO WORLD a Pico Park pour prendre le nom du jeu cree .

```
#include "AppDelegate.h"
#include "HelloWorldScene.h"

USING_NS_CC;

AppDelegate::AppDelegate() {

}

AppDelegate::~AppDelegate()
{
}

bool AppDelegate::applicationDidFinishLaunching() {
    auto director = Director::getInstance();
```

```
    auto glview = director->getOpenGLView();
    if (!glview) {
        //glview = GLViewImpl::create("Pico Park");
        //glview->setFrameSize(480, 320);
        glview = GLViewImpl::createWithRect("Pico Park", Rect(0, 0, 480, 320),2.0);
        director->setOpenGLView(glview);
    }

    auto scene = HelloWorld::createScene();
    director->runWithScene(scene);

    return true;
}
void AppDelegate::applicationDidEnterBackground()
{
}

void AppDelegate::applicationWillEnterForeground() {

}
```

La scene 1 : MAIN MENU (SCENE DE HELLO WORLD)




```
#include "cocos2d.h"
#include "SCENE1.h" //pour pouvoir switcher vers
#include "AudioEngine.h" //biblio pour importer les audioa
#include "HelloWorldScene.h"
#include <iostream>

USING_NS_CC;

Scene* HelloWorld::createScene()
{
    // 'scene' is an autorelease object
    auto scene = Scene::create();
    auto layer = HelloWorld::create();

    scene->addChild(layer);
```

```
    return scene;
}

bool HelloWorld::init()
{
    if (!Layer::init()) {
        return false;
    }

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

    ////////////////////menu background

    auto menuback = Sprite::create("menu_background.png");
    menuback->setPosition(Point((visibleSize.width / 2) + origin.x, (visibleSize.height / 2) +
origin.y));
    menuback->setScale(0.125);

    ////////////////Pico Park LOGO

    this->addChild(menuback);
    auto* menu = Menu::create();
    menu->setPosition(Point(0, 0));
    this->addChild(menu);
    auto start_button = Sprite::create("start_button.png");
    auto start_button_selected = Sprite::create("start_button_selected.png");

    //////////////Play button

    MenuItemImage* menu_play = MenuItemImage::create();
    menu_play->setNormalImage(start_button);
    menu_play->setSelectedImage(start_button_selected);
    menu_play->setCallback(CC_CALLBACK_1(HelloWorld::play, this));
    menu->addChild(menu_play, 4);
    menu_play->setPosition(Point(visibleSize.width / 2, (visibleSize.height / 2)));

    //////////////upload music

    cocos2d::AudioEngine::preload("music.mp3"); //upload de notre music mp3 en v4 de cocos
    auto a = cocos2d::AudioEngine::play2d("music.mp3", true);    //PLAY MUSIC //true whwnever
t7lat hello world
    return true;
}



///Fonction pour aller au niveau 1 quand on click sur PLAY

void HelloWorld::play(cocos2d::Ref* PSender) {
    auto scene1 = SCENE1::createScene();
    Director::getInstance()->pushScene(TransitionSplitCols::create(1, scene1)); //la scene qu
on veut y switch vers
    cocos2d::AudioEngine::stopAll(); //stoper la music quand on clique sur play
}
```

DANS CETTE SCENE ,en cliquant sur play, on switch vers le premier level .

La 2 eme scene (SCENE 1) LEVEL 1

- En creeant notre scene,(et tous les scenes du jeu), on ajoute la propriete de cocos2d : physics world,(car on utilise plusieurs objets et spectres),pour bien les organiser et qu ils soivent homogenes.

- On a cree notre sprite (nomme player, on lui a attribue ce qui appele en coocs 2d : physics body afin de pouvoir detecter les collisions, stabiliser le player sur la scene....)
- EXEMEPLE DES SPRITES ET OBSTACLES CRREE :

```
    auto player = Sprite::create("Player.png");
    player->setAnchorPoint(Vec2(0.5, 0.5));
    player->setPosition(Vec2(50, 60));
    player->setScale(0.3); //scale dyal player
    player->setName("player");
    this->addChild(player, 2);

    ////////////////////  creating physique for player

    auto physicsBody1 = PhysicsBody::createBox(player->getContentSize(),
PhysicsMaterial(1000.0f, 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() * 9.8f);
    physicsBody1->applyForce(force);


    ////////////////////create the background that iclude PICO PARK

    auto background = Sprite::create("background.png");
    background->setAnchorPoint(Vec2(0, 0));
    background->setPosition(Vec2(0, 0));
    background->setScale(0.125);
    this->addChild(background, 0);
    ////////////////////////////////// creating the floors and the obstacles

    auto* floor = Sprite::create("land-15.png");
    floor->setAnchorPoint(Vec2(0, 0));
    floor->setPosition(Vec2(0, 0));
    floor->setScale(0.125);
    this->addChild(floor, 1);

    auto physicsBody_floor = PhysicsBody::createBox(floor->getContentSize(),
PhysicsMaterial(1500.0f, 0.1f, 0.9f));
    physicsBody_floor->setDynamic(false);
    physicsBody_floor->setCollisionBitmask(1);
    physicsBody_floor->setCategoryBitmask(1);


    floor->setPhysicsBody(physicsBody_floor);
```

Partie : interaction entre le joueur du jeu et le clavier.

- On a utilise la propriete EventListenerKeyboard afin d'utiliser les boutons du clavier pour donner la propriete du mouvement a notre sprite.
- On keypressed: lorsqu on clique au contraire du onkyreleased.

```
// Create a keyboard event listener
    auto keyboardListener = EventListenerKeyboard::create();
    keyboardListener->onKeyPressed = CC_CALLBACK_2(SCENE1::onKeyPressed, this);
    keyboardListener->onKeyReleased = CC_CALLBACK_2(SCENE1::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);
            cocos2d::AudioEngine::preload("jump.mp3"); //upload de notre music mp3 en v4 de
cocos
            cocos2d::AudioEngine::play2d("jump.mp3");

        }
        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);

        }

    };


    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();
        }


          };
```

Partie : mouvement de la scene .

On a vraiment souffert pour trouver comment faire bouger l ecran au fur et a mesur de mouvement du player,on a arrive a cette solution :
Faire bouger tous les objets (floor, obstacles...)de la scene en leur donnant une meme vitesse.

```
//////////////////////////////////action for the Groun objetcts

    auto* acc = MoveBy::create(0.07 * winSize.width, Point(-winSize.width * 4, 0));


    obstacle1->runAction(acc->clone());
    floor->runAction(acc->clone());
    floor2->runAction(acc->clone());
    floor3->runAction(acc->clone());
    floor4->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());
    door->runAction(acc->clone());
```

partie : detection des collision

On peut dire que cette partie était l une des plus difficiles taches dans le codage.
Mais on a pu comprendre que pour le faire, on devrait donner pour chaque objet qu on veut detetcter un entier dans la fct getcoollisionbitmask, et l utilisation de setcontactbitmask
Afin de les recuperer et leurs appliquer ce qu on veut :

```
auto contactListener = EventListenerPhysicsContact::create();
    contactListener->onContactBegin = [floor8](PhysicsContact& contact) {


        PhysicsBody* x = contact.getShapeA()->getBody();
        PhysicsBody* y = contact.getShapeB()->getBody();
        if (1 == x->getCollisionBitmask() && 3 == y->getCollisionBitmask() || 3 == x-
>getCollisionBitmask() && 1 == y->getCollisionBitmask()) {
            MoveBy* moveact = MoveBy::create(1.2, Vec2(0, 90));
            floor8->runAction(moveact);

        }

        if (1 == x->getCollisionBitmask() && 4 == y->getCollisionBitmask() || 4 == x-
>getCollisionBitmask() && 1 == y->getCollisionBitmask()) {

            auto scene = Gameover::createScene();
            Director::getInstance()->pushScene(TransitionFade::create(0.5, scene));
            cocos2d::AudioEngine::preload("gameover.mp3"); //upload de notre music mp3 en v4
de cocos
            cocos2d::AudioEngine::play2d("gameover.mp3");
        }

        if (1 == x->getCollisionBitmask() && 2 == y->getCollisionBitmask() || 2 == x-
>getCollisionBitmask() && 1 == y->getCollisionBitmask()) {
            auto scene2 = SCENE2::createScene();
            Director::getInstance()->pushScene(TransitionSplitCols::create(0.4, scene2));
            cocos2d::AudioEngine::preload("win.mp3"); //upload de notre music mp3 en v4 de
cocos
            cocos2d::AudioEngine::play2d("win.mp3");
        }
```

```
        return true;
    };

        this->getEventDispatcher()->addEventListenerWithSceneGraphPriority(contactListener,
      this);
```

partie : switcher entre scènes :

comme dans la scène précédente, et même pour toutes les scenes de ce jeu, Director controllait les scenes en toute facilitee.
Son utilisation comme se suit :

```
        auto scene = Gameover::createScene();

              Director::getInstance()->pushScene(TransitionFade::create(0.5, scene));
```

Ici ,lorsque le joueur tombe, director va nous ramener a la scene game over deeja creee.

SCENE2 (Level 2)

Ici, on ajoute 2 principaux proprietes :

La cle que doit prendre le player pour ganger et passer au level 3.

Le player vole et doit passer un level portant le principe de flappy bird.

```
  ////////keycntact
        int i = 0;
        if (2 == x->getCollisionBitmask() && 4 == y->getCollisionBitmask() || 4 == x-
>getCollisionBitmask() && 2 == y->getCollisionBitmask()) {
            key->setOpacity(0);
            i = 1;
        }
        //////door contact
        if (2 == x->getCollisionBitmask() && 5 == y->getCollisionBitmask() || 5 == x-
>getCollisionBitmask() && 2 == y->getCollisionBitmask()) {
            if (i == 1)
            {
                auto scene2 = SCENE3::createScene();
                Director::getInstance()->pushScene(TransitionSplitCols::create(0.4, scene2));
                cocos2d::AudioEngine::preload("win.mp3"); //upload de notre music mp3 en v4 de
cocos
                cocos2d::AudioEngine::play2d("win.mp3");
            }
            else
            {
                auto scene = Gameover::createScene();
                Director::getInstance()->pushScene(TransitionFade::create(0.5, scene));
                cocos2d::AudioEngine::preload("gameover.mp3"); //upload de notre music mp3 en
v4 de cocos
                cocos2d::AudioEngine::play2d("gameover.mp3");
            }
        }

        return true;
    };
```

```
    this->getEventDispatcher()->addEventListenerWithSceneGraphPriority(contactListener, this);
```

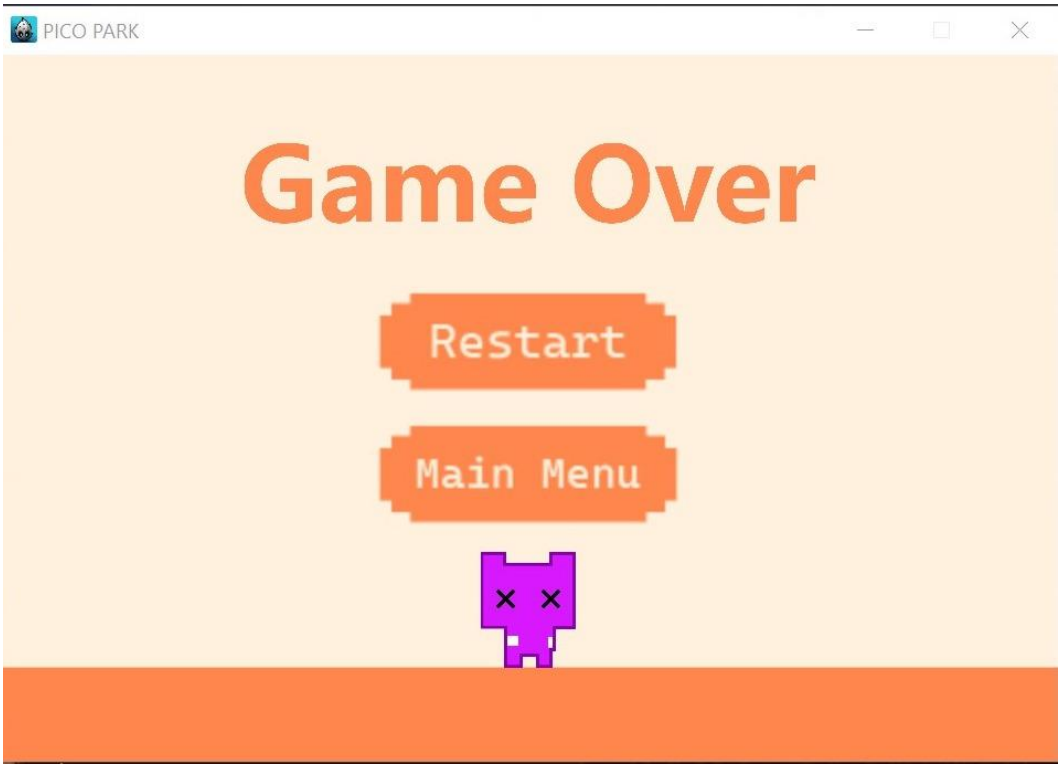
SCENE 3 : (Level 3)

Le level le plus difficile de notre jeu, ici on a ajoute 3 proprietes essentielles :
Des boutons qui ne doivent pas etre touches, sinon ,tu echouera.
Une partie de sol(cree avec des boxs) qui tombent apres contact qui depasse 4 secondes.
Le joueur pour arriver a la porte, il doit cliquer sur bouton pour qu il se minimalise et passe .
Le code des sols tombents

```
auto contactListener = EventListenerPhysicsContact::create();
    contactListener->onContactBegin = [player3,floor8,push](PhysicsContact& contact) {


        PhysicsBody* x = contact.getShapeA()->getBody();
        PhysicsBody* y = contact.getShapeB()->getBody();

        if (1== x->getCollisionBitmask() && 5== y->getCollisionBitmask() || (5 == x-
>getCollisionBitmask() && 1 == y->getCollisionBitmask())) {
            if (x->getNode()->getScale() == 0.125 || x->getNode()->getScale() == 0.130)
            {
                MoveBy* moveact0 = MoveBy::create(4.0, Vec2(0, -150));
                x->getNode()->runAction(moveact0);
            }
            if (y->getNode()->getScale() == 0.125|| y->getNode()->getScale() == 0.130)
            {
                MoveBy* moveact0 = MoveBy::create(4.0, Vec2(0, -150));
                y->getNode()->runAction(moveact0);
            }
```

La scene : Game over

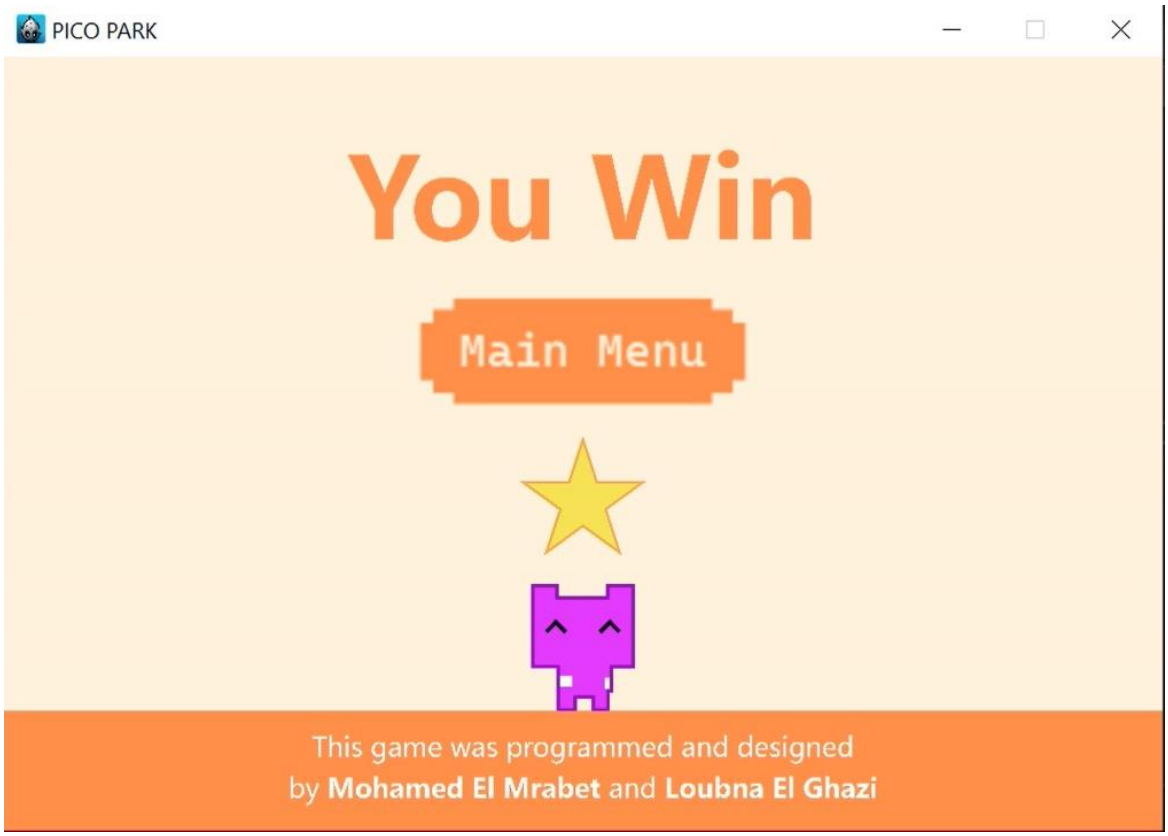
Scène simple, appelée lorsque le player echoue ou tombe dans le jeu.



```cpp
#include "HelloWorldScene.h"
#include "cocos2d.h"
#include "AudioEngine.h"
#include "SCENE1.h"
#include "Gameover.h"

USING_NS_CC;

Scene* Gameover::createScene()
{
    auto scene = Scene::create();
    auto layer = Gameover::create();

    scene->addChild(layer);

    return scene;
}

bool Gameover::init()
{
    if (!Layer::init()) {
        return false;
    }
    auto  origin = Director::getInstance()->getVisibleOrigin();
    auto visibleSize = Director::getInstance()->getVisibleSize();


    auto* gameover = Sprite::create("gameover.png");
    gameover->setPosition(Point((visibleSize.width / 2), (visibleSize.height / 2)));
    gameover->setScale(0.125);
    this->addChild(gameover);

    auto* menu2 = Menu::create();
    menu2->setPosition(Point(0, 0));
    this->addChild(menu2);

cocos2d::AudioEngine::preload("gameover.mp3");
cocos2d::AudioEngine::play2d("gameover.mp3");

    ///ajout des elements:

    auto* restart = Sprite::create("restart.png");
    auto* restart_bu = Sprite::create("restartS.png");
    MenuItemImage* menu_restart = MenuItemImage::create();
    menu_restart->setNormalImage(restart);
    menu_restart->setSelectedImage(restart_bu);
    menu_restart->setCallback(CC_CALLBACK_1(Gameover::reload, this)); //reload du game on
click
    menu_restart->setPosition(Point((visibleSize.width / 2), (visibleSize.height / 2) + 30));
    menu2->addChild(menu_restart, 3);


    auto main_menu = Sprite::create("mainmenu.png");
    auto main_menu_bu = Sprite::create("mainmenuS.png");
    MenuItemImage* menu_main = MenuItemImage::create();
    menu_main->setNormalImage(main_menu);
    menu_main->setSelectedImage(main_menu_bu);
    menu_main->setPosition(Point((visibleSize.width / 2), (visibleSize.height / 2) - 30));
    menu_main->setCallback(CC_CALLBACK_1(Gameover::reload2, this)); // du game on click
    menu2->addChild(menu_main, 4);

    return true;
}
void Gameover::reload(cocos2d::Ref* pSender) {
    auto scene = SCENE1::createScene();
    Director::getInstance()->replaceScene(TransitionFade::create(0.4, scene)); //game over et
refaire la scene
}
```

```
void Gameover::reload2(cocos2d::Ref* pSender) {
    auto scene2 = HelloWorld::createScene();
    Director::getInstance()->replaceScene(TransitionFade::create(0.4, scene2)); //game over et
refaire la scene

}
```

Scène WIN :



Scene appelee lorsque le joueure depasse tous les levels et gagne .

```
#include "HelloWorldScene.h"
#include "cocos2d.h"
#include "SCENE3 .h"
#include "win.h"
#include "AudioEngine.h"
USING_NS_CC;

Scene* win::createScene()
{

    auto scene4 = Scene::create();
    auto layer = win::create();

    scene4->addChild(layer);

    return scene4;
}

bool win::init()
{
    if (!Layer::init()) {
        return false;
    }
    auto  origin6 = Director::getInstance()->getVisibleOrigin();
    auto visibleSize6 = Director::getInstance()->getVisibleSize();

  ////////////////////////////////background
    auto win1 = Sprite::create("youwin.png");
```

```
    win1->setPosition(Point((visibleSize6.width / 2) + origin6.x, (visibleSize6.height / 2) +
origin6.y));
    win1->setScale(0.125);
    this->addChild(win1);
//////////////////////////////////////////
    auto* menu5 = Menu::create();
    menu5->setPosition(Point(0, 0));
    this->addChild(menu5);
    ////////////////////////


    ///ajout des elements:

    auto main_button = Sprite::create("mainmenu.png");
    auto  main_button_selected = Sprite::create("mainmenus.png");
    MenuItemImage* main_button1 = MenuItemImage::create();
    main_button1->setNormalImage(main_button);
    main_button1->setSelectedImage(main_button_selected);
    main_button1->setPosition(Point((visibleSize6.width / 2), (visibleSize6.height / 2) +38));
    main_button1->setCallback(CC_CALLBACK_1(win::winer, this));

    ////////////
    menu5->addChild (main_button1, 4);
    cocos2d::AudioEngine::preload("youwin.mp3"); //upload de notre music mp3 en v4 de cocos
    cocos2d::AudioEngine::play2d("youwin.mp3");    //PLAY MUSIC //

    return true;
}



//implemetation des fcts crees :
void  win::winer (cocos2d::Ref* PSender) {

    auto scene1 = HelloWorld::createScene();
    Director::getInstance()->pushScene(TransitionSplitCols::create(1, scene1)); //la scene qu
on veut y switch vers
}
```

- les difficultés rencontrées :

  1-Manque de ressources, on a vraiment lis beaucoup de livres et documentations, Regarder des vidéos tutos afin de juste comprendre les fonctionnalités de moteur de jeu et comment utiliser les différentes fonctions et propriétés.

  2-on rencontrait certaines erreurs d'exécution qui étaient vraiment complexes a régler.

  3-C'était la première fois pour nous d'utiliser les Game engine avec le langage c++(première expérience ce qui était difficile d étudier les notions de base et appliquer en même temps).

  MAIS on a pu récupérer et régler tous grâce a nous efforts doubles afin de construire un jeu complet .

- ## Répartition du travail en groupe

  Pour le level 1 (EL MRABET MOHAMMED ET EL GHAZI LOUBNA) : c' était le level de base, par lequel on a pu assimiler les différentes fonctions et leurs utilisations, comment créer le player, comment changer le background, détecter les collisions…

  Après on a décomposé le travail comme ceci :

  -Mohamed El Mrabet : tout ce qui concerne cote Graphic Design du jeu (tous les éléments du jeu sont crées avec ADOBE ILLUSTRATOR : player ,floor……) & Level 2.

  -El ghazi Loubna :Création des scènes : MENU,GAMEOVER,WIN ET LEVEL 3.