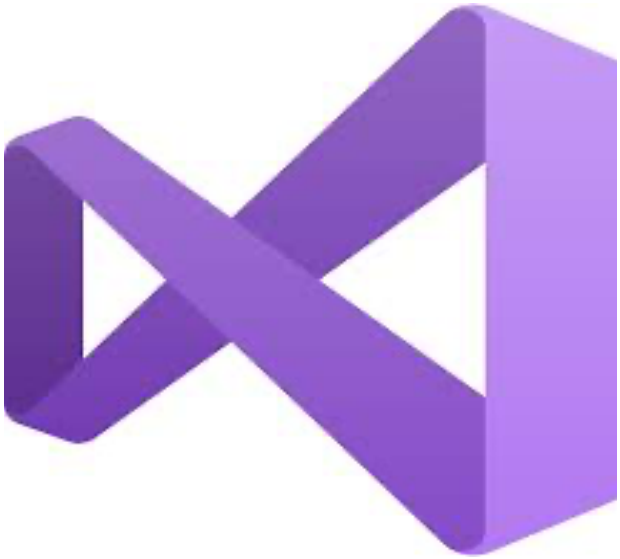
Jeux 2D C++



Projet POO en c++

Réaliser par :

- ESSETTI WIDAD
- AYA LAADAILI

ENCADRÉE PAR :

- Pr Lotfi EL AACHAK
- Pr Ikram BEN ABDEL OUAHAB

SOMMAIRE

INTRODUCTION

Définition de la POO

La P.O.O peut se définir comme l'art de décomposer une application en un certain nombre d'objets qui communiquent entre eux afin de réaliser une ou plusieurs tâches.

Autrement dit, si dans la programmation fonctionnelle, l'unité de travail est la fonction, alors, la programmation orientée se base sur les objets (qu'on va définir et expliquer dans la partie suivante du cours).

Le but de la POO :

Le but de la programmation orientée objet est de réduire la difficulté de la tâche à accomplir . En effet, et selon le principe "Diviser pour régner", on décompose le problème initial en un grand nombre de petits problèmes qui sont plus simples à comprendre et à résoudre.

LE LANGUAGE C++ :

Le langage C++ est typé statiquement, compilé, langage de programmation libre à usage général, sensible à la casse, qui prend en charge la programmation procédurale, orientée objet et générique.
C++ est considéré comme un langage de niveau moyen, car il combine des fonctionnalités de langage de haut niveau et de bas niveau.
C++ a été développé par Bjame Stroustrup à partir de 1979, au Bell Labs de Murray Hill, dans New Jersey, dans le but d'améliorer le langage C et s'appelait à l'origine C avec Classes, mais il a ensuite été renommé C++ en 1983.
C++ est un sur-ensemble de C et pratiquement tout programme C légal est un programme C++ légal.

COCOS 2D :

Cocos2d-x est l'une des bibliothèques logicielles de référence. En plus de permettre de créer des jeux en 2D pour les appareils mobiles Android, iOS et Windows Phone, elle compile sur Windows, Mac et Linux. La bibliothèque peut être utilisée pour le développement en C++, Javascript et Lua.

Contrairement à d'autres moteurs de développement de jeux vidéo multiplateformes qui sont basées sur JavaScript et HTML5, Cocos2d-x utilise l'API graphique OpenGL ES qui a pour avantage d'améliorer les performances des applications mobiles.

Les limites de Cocos2d-x :

Cocos2d-x… Le nom indique déjà sa plus grande limite. Bien que très complet et performant, il ne peut pas soutenir le développement de jeux 3D. Est-ce nécessaire de rappeler que même les grands classiques de jeux 2d, comme le célèbre Super Mario, ont dû passer en mode 3d pour s'adapter aux tendances actuelles?

Enfin, Cocos2d-x manque d'un véritable service de support technique. Ce souci, bien sûr, ne concerne pas seulement Cocos2d-x, mais de manière générale, tous les outils en open source. Contrairement aux moteurs de jeux payants, comme Unity, il n'existe pas d'équipe d'experts « obligés » de répondre aux incidents et sollicitations divers. La gestion se fait entièrement par la communauté. Aussi passionnée qu'elle soit, elle ne peut pas garantir l'efficacité des solutions qu'elle apporte. Apparemment, les développeurs ne semblent pas gênés par ce petit inconvénient de Cocos2d-x qui est aujourd'hui l'outil le plus utilisé pour le développement de jeux en 2D.

Processus de devellopement

GIMP :

Pour le sprite du jeux on a pris un designe de google est par gimp ou an enlever l'arriere plan , pour le back ground de meme on a juste au lien d'enlever le background on a ajouter de l'ecriture dessus

Et pour l'obstacle on a rogner une photo prise sur internet par gimp

PROGRAMME

On a créer 5 fichier .cpp et 6.h

Premier fichier adding.h :

contient les definitions utiliser pour les differentes fonctionalité du jeux

```
#ifndef __ADDING_H__
#define __ADDING_H__
#define DISPLAY_TIME_PAUSE_SCENE 2
#define TRANSITION_TIME 0.5
#define PLAYER_COLLISION_BITMASK 1
#define OBSTACLE_COLLISION_BITMASK 1
#define TRANSITION_TIME 0.5
```

#endif // __ADDING_H__

APP DELEGATE .cpp:

```
#include "AppDelegate.h"
#include "MainMenuScene.h"
#include "PauseScene.h"
#include "GameScene.h"
#include "../proj.win32/PauseScene.h"
USING_NS_CC;
static cocos2d::Size designResolutionSize = cocos2d::Size(480, 320);
static cocos2d::Size smallResolutionSize = cocos2d::Size(480, 320);
static cocos2d::Size mediumResolutionSize = cocos2d::Size(1024, 768);
static cocos2d::Size largeResolutionSize = cocos2d::Size(2048, 1536);
AppDelegate::AppDelegate()
{
}
AppDelegate::~AppDelegate()
{
}
// if you want a different context, modify the value of glContextAttrs
// it will affect all platforms
void AppDelegate::initGLContextAttrs()
{
    // set OpenGL context attributes: red,green,blue,alpha,depth,stencil,multisamplesCount
    GLContextAttrs glContextAttrs = { 8, 8, 8, 8, 24, 8, 0 };

    GLView::setGLContextAttrs(glContextAttrs);
}

// if you want to use the package manager to install more packages,
// don't modify or remove this function
static int register_all_packages()
{
    return 0; //flag for packages manager
}
bool AppDelegate::applicationDidFinishLaunching() {
    // initialize director
    auto director = Director::getInstance();
    auto glview = director->getOpenGLView();
    if (!glview) {
```

```
#if (CC_TARGET_PLATFORM == CC_PLATFORM_WIN32) || (CC_TARGET_PLATFORM == CC_PLATFORM_MAC) ||
(CC_TARGET_PLATFORM == CC_PLATFORM_LINUX)
    glview = GLViewImpl::createWithRect("kid", cocos2d::Rect(0, 0, designResolutionSize.width,
designResolutionSize.height));
#else
    glview = GLViewImpl::create("kid");
#endif
    director->setOpenGLView(glview);
  }

  // turn on display FPS
  director->setDisplayStats(true);

  // set FPS. the default value is 1.0/60 if you don't call this
  director->setAnimationInterval(1.0f / 60);

  // Set the design resolution

  glview->setFrameSize(1440, 900);
  glview->setDesignResolutionSize(designResolutionSize.width, designResolutionSize.height,
ResolutionPolicy::NO_BORDER);
  auto frameSize = glview->getFrameSize();
  // if the frame's height is larger than the height of medium size.
  if (frameSize.height > mediumResolutionSize.height)
  {
    director->setContentScaleFactor(MIN(largeResolutionSize.height / designResolutionSize.height,
largeResolutionSize.width / designResolutionSize.width));
  }
  // if the frame's height is larger than the height of small size.
  else if (frameSize.height > smallResolutionSize.height)
  {
    director->setContentScaleFactor(MIN(mediumResolutionSize.height / designResolutionSize.height,
mediumResolutionSize.width / designResolutionSize.width));
  }
  // if the frame's height is smaller than the height of medium size.
  else
  {
    director->setContentScaleFactor(MIN(smallResolutionSize.height / designResolutionSize.height,
smallResolutionSize.width / designResolutionSize.width));
  }

  register_all_packages();
```

```
    // create a scene. it's an autorelease object
    auto scene = Pause::createScene();

    // run
    director->runWithScene(scene);



    return true;
}
```

Et header:

```
#ifndef  APP_DELEGATE_H
#define  APP_DELEGATE_H
#include "cocos2d.h"
class  AppDelegate : private cocos2d::Application
{
public:
    AppDelegate();
    virtual ~AppDelegate();

    virtual void initGLContextAttrs();

    virtual bool applicationDidFinishLaunching();



    virtual void applicationDidEnterBackground();



    virtual void applicationWillEnterForeground();
};

#endif // APP_DELEGATE_H
```

Header et le code de Pause scene :

Contient la premiere scene qui est un background avec ecrit dessus WELCOME

CPP :

```
#include "PauseScene.h"
#include "MainMenuScene.h"
#include "Adding.h"

USING_NS_CC;

Scene* Pause::createScene()
{
   return Pause::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 Pause::init()
{
   ////////////////////////////////
   // 1. super init first
   if (!Scene::init())
   {
      return false;
   }

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

   this->scheduleOnce(CC_SCHEDULE_SELECTOR(Pause::GoToMainMenuScene), DISPLAY_TIME_PAUSE_SCENE);

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

   return true;
}
```

```
void Pause::GoToMainMenuScene(float dt)
{
    auto scene = MainMenu::createScene();

    Director::getInstance()->replaceScene(TransitionFade::create(TRANSITION_TIME, scene));
}
```

HEADER:

```
#ifndef _PAUSE_SCENE_H_
#define _PAUSE_SCENE_H_

#include "cocos2d.h"

class Pause : public cocos2d::Scene
{
public:
    static cocos2d::Scene* createScene();

    virtual bool init();

    // implement the "static create()" method manually
    CREATE_FUNC(Pause);

private:
    void GoToMainMenuScene(float dt);

};

#endif // _PAUSE_SCENE_H_
```

Header et code de MAINMENU SCENE :

Boutton d'entree et boutton de sortie et le back ground

CPP :

```
#include "MainMenuScene.h"
#include "GameScene.h"
#include "GameOverScene.h"
#include "Adding.h"
```

```
#include "../proj.win32/GameScene.h"
#include "../proj.win32/Adding.h"

USING_NS_CC;

Scene* MainMenu::createScene()
{
   return MainMenu::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 MainMenu::init()
{
   //////////////////////////////
   // 1. super init first
   if (!Scene::init())
   {
      return false;
   }

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

   auto backgroundSprite = Sprite::create("welcome.jpeg");
   backgroundSprite->setPosition(Point(visibleSize.width / 2 + origin.x, visibleSize.height / 2 + origin.y));
   backgroundSprite->setScale(1.5);
   this->addChild(backgroundSprite);

   auto playItem = MenuItemImage::create("p.png", "p.png", CC_CALLBACK_1(MainMenu::GoToLevel, this));
   playItem->setPosition(Point(visibleSize.width / 2 + origin.x, visibleSize.height / 2 + origin.y - 30));
   playItem->setScale(1);
   auto playItem1 = MenuItemImage::create("pause(1).png", "pause(1).png",
CC_CALLBACK_1(MainMenu::menuCloseCallback, this));
   playItem1->setPosition(Point(visibleSize.width / 2 + origin.x, visibleSize.height / 2 + origin.y - 60));
```

```
    playItem1->setScale(1);

    auto menu = Menu::create(playItem, playItem1, NULL);
    menu->setPosition(Point::ZERO);
    this->addChild(menu);

    return true;
}

void MainMenu::GoToLevel(cocos2d::Ref* pSender) {

    auto scene = Level::createScene();
    Director::getInstance()->pushScene(TransitionFade::create(TRANSITION_TIME, scene));

}


void MainMenu::GoToGameOver(cocos2d::Ref* pSender) {

    auto scene = GameOver::createScene();
    Director::getInstance()->pushScene(TransitionFade::create(TRANSITION_TIME, scene));

}
void MainMenu::menuCloseCallback(Ref* pSender)
{
    //Close the cocos2d-x game scene and quit the application
    Director::getInstance()->end();

}
```

HEADER :

```
#ifndef _MAIN_MENU_SCENE_H_
#define _MAIN_MENU_SCENE_H_

#include "cocos2d.h"

class MainMenu : public cocos2d::Scene
{
public:
    static cocos2d::Scene* createScene();
```

```
    void menuCloseCallback(cocos2d::Ref* pSender);
    virtual bool init();

    // implement the "static create()" method manually
    CREATE_FUNC(MainMenu);
private:
    void GoToLevel(cocos2d::Ref* sender);
    void GoToGameOver(cocos2d::Ref* sender);
};

#endif // _MAIN_MENU_SCENE_H_
```

GAMESCENE header et cpp : contient le seul niveau qu'on a (fonctionalités de contact…)

CPP :

```
#include "GameScene.h"
#include "GameOverScene.h"
#include "Adding.h"
using namespace cocos2d;

USING_NS_CC;

Scene* Level::createScene()
{
    // 'scene' is an autorelease object
    auto scene = Scene::createWithPhysics();
    PhysicsWorld* world = scene->getPhysicsWorld();
    // 'layer' is an autorelease object
    auto layer = Level::create();

    // add layer as a child to scene
    scene->addChild(layer);

    // return the scene
    return scene;
}

// on "init" you need to initialize your instance
bool Level::init()
```

```
{


    if (!Scene::initWithPhysics())
    {
        return false;
    }

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

    auto backgroundSprite = Sprite::create("BGFV.jpeg");
    backgroundSprite->setPosition(Point(visibleSize.width / 2 + origin.x, visibleSize.height / 2 + origin.y));
    backgroundSprite->setScale(2.2);
    this->addChild(backgroundSprite);

    auto foregroundSprite = Sprite::create("7.png");
    foregroundSprite->setPosition(Point(2, 80));
    foregroundSprite->setAnchorPoint(Vect::ZERO);
    foregroundSprite->setScale(1);
    foregroundSprite->setName("foregroundSprite");
    auto physicsBody2 = PhysicsBody::createBox(foregroundSprite->getContentSize() / 1.5, PhysicsMaterial(1.0f, 1.0f,
1.0f));
    physicsBody2->setGravityEnable(false);
    physicsBody2->setDynamic(false);
    physicsBody2->setContactTestBitmask(1);
    physicsBody2->setCollisionBitmask(1);
    physicsBody2->setCategoryBitmask(1);
    foregroundSprite->setRotation(0.0f);
    foregroundSprite->setPhysicsBody(physicsBody2);
    this->addChild(foregroundSprite);

    auto second = Sprite::create("7.png");
    second->setPosition(Point(100, 79));

    second->setAnchorPoint(Vect::ZERO);
    second->setScale(1);
    second->setName("second");
    auto physicsBody3 = PhysicsBody::createBox(second->getContentSize() / 1.5, PhysicsMaterial(1.0f, 1.0f, 1.0f));
    physicsBody3->setGravityEnable(false);
    physicsBody3->setDynamic(false);
    physicsBody3->setContactTestBitmask(1);
```

```
physicsBody3->setCollisionBitmask(1);
physicsBody3->setCategoryBitmask(1);
second->setRotation(0.of);
second->setPhysicsBody(physicsBody3);
this->addChild(second);

auto third = Sprite::create("7.png");
third->setPosition(Point(210, 5));

third->setAnchorPoint(Vect::ZERO);
third->setScale(1);
third->setName("third");
auto physicsBody4 = PhysicsBody::createBox(third->getContentSize() / 1.5, PhysicsMaterial(1.of, 1.of, 1.of));
physicsBody4->setGravityEnable(false);
physicsBody4->setDynamic(false);
physicsBody4->setContactTestBitmask(4);
physicsBody4->setCollisionBitmask(1);
physicsBody4->setCategoryBitmask(1);
third->setRotation(0.of);
third->setPhysicsBody(physicsBody4);
this->addChild(third);
auto third316 = Sprite::create("7.png");
third316->setPosition(Point(375, 86));
third316->setAnchorPoint(Vect::ZERO);
third316->setScale(1.3);
third316->setName("third316");
auto physicsBody17 = PhysicsBody::createBox(third316->getContentSize(), PhysicsMaterial(1.of, 0.of, 1.of));
physicsBody17->setGravityEnable(false);
physicsBody17->setDynamic(false);
physicsBody17->setContactTestBitmask(5);
physicsBody17->setCollisionBitmask(1);
physicsBody17->setCategoryBitmask(5);
third316->setRotation(0.of);
third316->setPhysicsBody(physicsBody17);
this->addChild(third316);

auto gameovertest = Sprite::create("gameovertest.png");
gameovertest->setPosition(Point(0, -20));
gameovertest->setAnchorPoint(Vect::ZERO);
gameovertest->setScale(1);
gameovertest->setName("gameovertest");
```

```
    auto physicsBody18 = PhysicsBody::createBox(gameovertest->getContentSize() / 1.5, PhysicsMaterial(1.of, 1.of,
1.of));
    physicsBody18->setGravityEnable(false);
    physicsBody18->setDynamic(false);
    physicsBody18->setContactTestBitmask(3);
    physicsBody18->setCategoryBitmask(3);
    gameovertest->setRotation(0.of);
    gameovertest->setPhysicsBody(physicsBody18);
    this->addChild(gameovertest);

    auto player = Sprite::create("sprie.png");
    player->setScale(0.2);
    player->setPosition(Vec2(50, 120));

    player->setName("player");
    auto physicsBody1 = PhysicsBody::createBox(player->getContentSize() / 1.5, PhysicsMaterial(1.of, 1.of, 1.of));
    physicsBody1->setGravityEnable(true);
    physicsBody1->setDynamic(true);
    physicsBody1->setContactTestBitmask(1);
    physicsBody1->setCollisionBitmask(1);
    physicsBody1->setCategoryBitmask(1);
    player->setRotation(0.of);
    player->setPhysicsBody(physicsBody1);
    this->addChild(player, 0, 1);



    auto listener = EventListenerKeyboard::create();
    auto eventListner = EventListenerKeyboard::create();
    eventListner->onKeyPressed = [](EventKeyboard::KeyCode KeyCode, Event* event) {
      Vec2 loc = event->getCurrentTarget()->getPosition();
      switch (KeyCode)
      {
      case EventKeyboard::KeyCode::KEY_LEFT_ARROW:
      case EventKeyboard::KeyCode::KEY_A:
        event->getCurrentTarget()->runAction(MoveBy::create(0.01, Vec2(-10, 0)));
        break;
      case cocos2d::EventKeyboard::KeyCode::KEY_RIGHT_ARROW:
      case cocos2d::EventKeyboard::KeyCode::KEY_D:
        event->getCurrentTarget()->runAction(MoveBy::create(0.01, Vec2(10, 0)));
        break;

      case cocos2d::EventKeyboard::KeyCode::KEY_SPACE:
```

```
            event->getCurrentTarget()->runAction(JumpBy::create(0.5, Vec2(60, 0), 50, 1));
            break;

        case cocos2d::EventKeyboard::KeyCode::KEY_DOWN_ARROW:
        case cocos2d::EventKeyboard::KeyCode::KEY_S:
            event->getCurrentTarget()->runAction(JumpBy::create(0.5, Vec2(-60, 0), 50, 1));
            break;
        }
    };
    this->_eventDispatcher->addEventListenerWithSceneGraphPriority(eventListner, player);

    auto contactListener = EventListenerPhysicsContact::create();
    contactListener->onContactBegin = CC_CALLBACK_1(Level::onContactBegin1, this);
    _eventDispatcher->addEventListenerWithSceneGraphPriority(contactListener, this);

    auto contactListener2 = EventListenerPhysicsContact::create();
    contactListener2->onContactBegin = CC_CALLBACK_1(Level::onContactBegin2, this);
    _eventDispatcher->addEventListenerWithSceneGraphPriority(contactListener2, this);

    return true;
}

bool Level::onContactBegin1(cocos2d::PhysicsContact& contact)
{
    auto bodyA = contact.getShapeA()->getBody();
    auto bodyB = contact.getShapeB()->getBody();
    if (bodyA->getContactTestBitmask() == 1 && bodyB->getContactTestBitmask() == 3)
    {
        auto newScene = GameOver::create();
        Director::getInstance()->pushScene(newScene);
    }
    return true;
}
```

HEADER :

```
#ifndef _GAME_SCENE_H_
#define _GAME_SCENE_H_

#include "cocos2d.h"

using namespace cocos2d;
```

```
class Level : public cocos2d::Scene
{

public:
   static cocos2d::Scene* createScene();

   virtual bool onContactBegin1(PhysicsContact& contact);
   virtual bool onContactBegin2(PhysicsContact& contact);
   virtual bool init();

   CREATE_FUNC(Level);

};

#endif // _GAME_SCENE_H_
```

Game Over header et CPP :

definit les fonctions pour réessayer ou quitter

CPP :

```
#include "GameOverScene.h"
#include "GameScene.h"
#include "Adding.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()
{
```

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

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

    auto backgroundSprite = Sprite::create("Gameover.jpeg");
    backgroundSprite->setPosition(Point(visibleSize.width / 2 + origin.x, visibleSize.height / 2 + origin.y));
    backgroundSprite->setScale(1.5);
    this->addChild(backgroundSprite);

    auto playItem = MenuItemImage::create("TAButt.png", "TAButt.png", CC_CALLBACK_1(GameOver::GoToLevel,
this));
    playItem->setPosition(Point(visibleSize.width / 2 + origin.x - 100, visibleSize.height / 2 + origin.y - 45));
    playItem->setScale(1);
    auto playItem1 = MenuItemImage::create("QButt DS.png", "QButt DS.png",
CC_CALLBACK_1(GameOver::menuCloseCallback, this));
    playItem1->setPosition(Point(visibleSize.width / 2 + origin.x + 100, visibleSize.height / 2 + origin.y - 45));
    playItem1->setScale(1);

    auto menu = Menu::create(playItem, playItem1, NULL);
    menu->setPosition(Point::ZERO);
    this->addChild(menu);



    return true;
}
void GameOver::GoToLevel(cocos2d::Ref* pSender) {

    auto scene = Level::createScene();
    Director::getInstance()->pushScene(TransitionFade::create(TRANSITION_TIME, scene));

}
void GameOver::menuCloseCallback(Ref* pSender)
{
    //Close the cocos2d-x game scene and quit the application
    Director::getInstance()->end();
```

```
}
```

HEADER :

```
#ifndef _GAME_OVER_SCENE_H_
#define _GAME_OVER_SCENE_H_

#include "cocos2d.h"

class GameOver : public cocos2d::Scene
{
public:
  static cocos2d::Scene* createScene();

  void menuCloseCallback(cocos2d::Ref* pSender);
  virtual bool init();

  // implement the "static create()" method manually
  CREATE_FUNC(GameOver);
private:
  void GoToLevel(cocos2d::Ref* sender);
};

#endif // _GAME_OVER_SCENE_H_
```

Difficulté rencontrer

On a rencontrer des difficultés pour d'abord trouver des tutos qui expliquent bien cocos2d

Partage de tache

On a dédie 2hr par jour pendant 15 jours ou on se rencontrer pour travailler ensemble et avancée sur le projet