



RAPPORT DU JEU


ROLLER - SPLAT



Realise par les etudiants : ELMahjoub Jaber – Haji Achraf

Module POO en C++

FSTT 2021-2022



Le document que vous avez sous vos yeux représente le rapport du projet final pour le module du POO en C++ qui été sous forme d'un jeux vidéo nommé 'Roller Splat' créé en appliquant les connaissances requissent tout le long du module et en se basant sur le moteur de jeu 'Cocos2d-x'.

Avant de commencer, nous tenons à remercier notre profs Ben Abdelouahab Ikrame et Mr. El Aachak Lotfi qui nous ont encadré tout le long du projet, et on espère que ce simple jeu vous plait.

Table de contenu :

1- Preparation : -----

2- Creation des scenes : -----

➤ **Game scene 1**

- *Les fonctions + La syntax.*

➤ **Game scene 2**

- *Les fonctions + La syntax.*

➤ **Game scene 3**

- *Les fonctions + La syntax.*

3- Le deplacement du joueur : -----

➤ **Le concept du deplacement**

- Les fonctions+ La syntax

4- Le son : -----

- *Les fonctions + La syntax.*

Préparation :

Avant d'entamer la partie saisie du code, il fallait faire quelques modifications au niveau du code source donné par défaut lors de la création d'un nouveau projet sur 'Cocos2d-x'.

La 1^{ère} chose à modifier était la résolution de la fenêtre du jeu à travers le fichier AppDelegate.cpp a la ligne suivante :

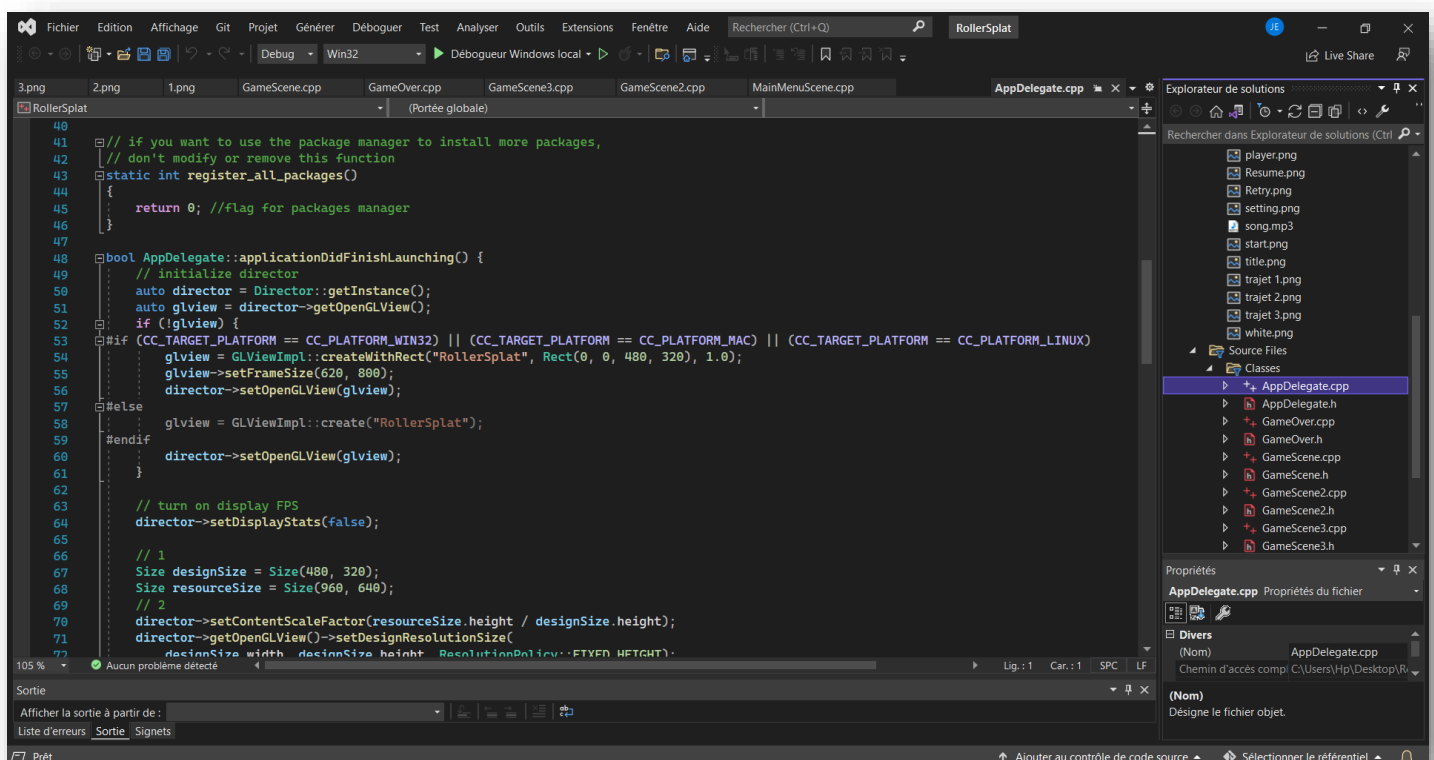
```
glview->setFrameSize(620, 800);
```

La résolution d'écran qui sera utilisé lors du jeu sera 6200 x 800 px.

Puis, désactiver l'affichage des statistiques du jeu (nombre des FPS ...) sur l'écran à travers la ligne suivante :

```
director->setDisplayStats(false);
```

Finalement effacer le contenu du fichier HelloWorldScene.cpp sauf la déclaration du fonction HelloWorld::init() puis renommer la nom du classe déclarée sur HelloWorldScene.h vers MainMenuScene puis tout le nom du fichier cpp et header.



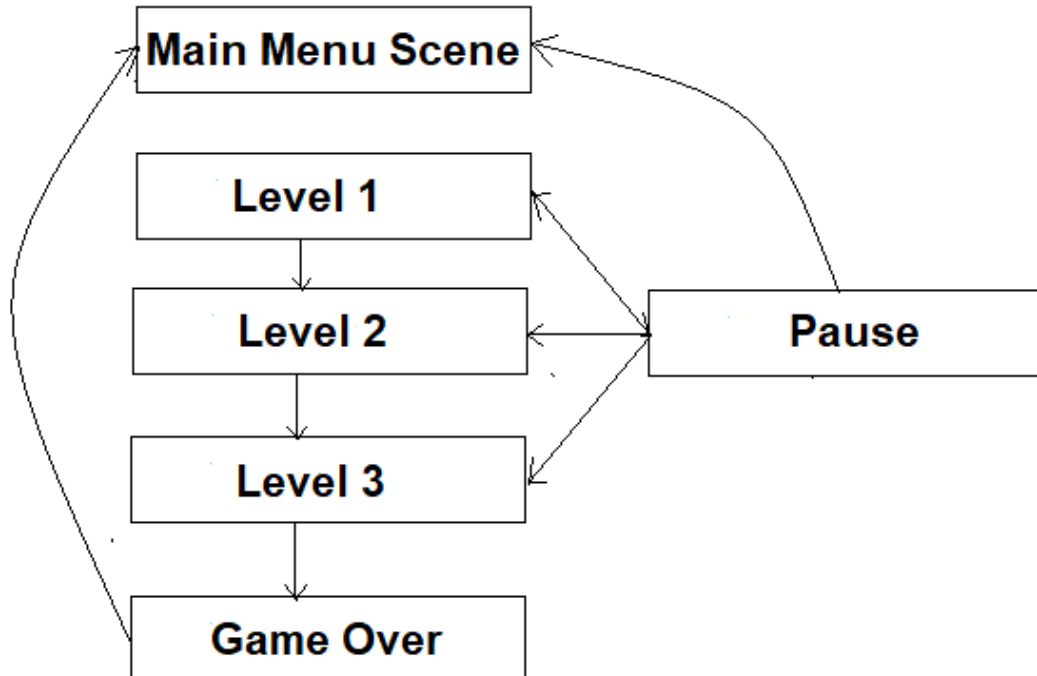
Création des scènes :

Le jeu sera composé d'une scène **MainMenuScene** lancée au démarrage du jeu. Cette Scène va permettre de lancer le 1ier niveau etc...

Chaque niveau aura son fichier différent des autres niveaux et sera considéré comme une scène indépendante.

Le jeu aura aussi une scène Pause qui va permettre au joueur de :

- **Continuer de jouer**
- **Réessayer le niveau**
- **Revenir au menu principal**



I- MainMenu scène :

Le menu principal du jeu sera composé d'un arrière plan et d'un seul bouton 'Play' qui va permettre de commencer le premier Niveau du jeu.

L'arrière-plan est de type Sprite.

Cela est possible à l'aide d'une fonction `GoToGameScene` qui effectue le passage par référence entre la scène du menu et celle du 1er niveau :

```
void MainMenu::GoToGameScene(Ref* pSender)
{
    auto scene = GameScreen::createScene();
    Director::getInstance()->replaceScene(TransitionFade::create(1.0, scene));
}
```

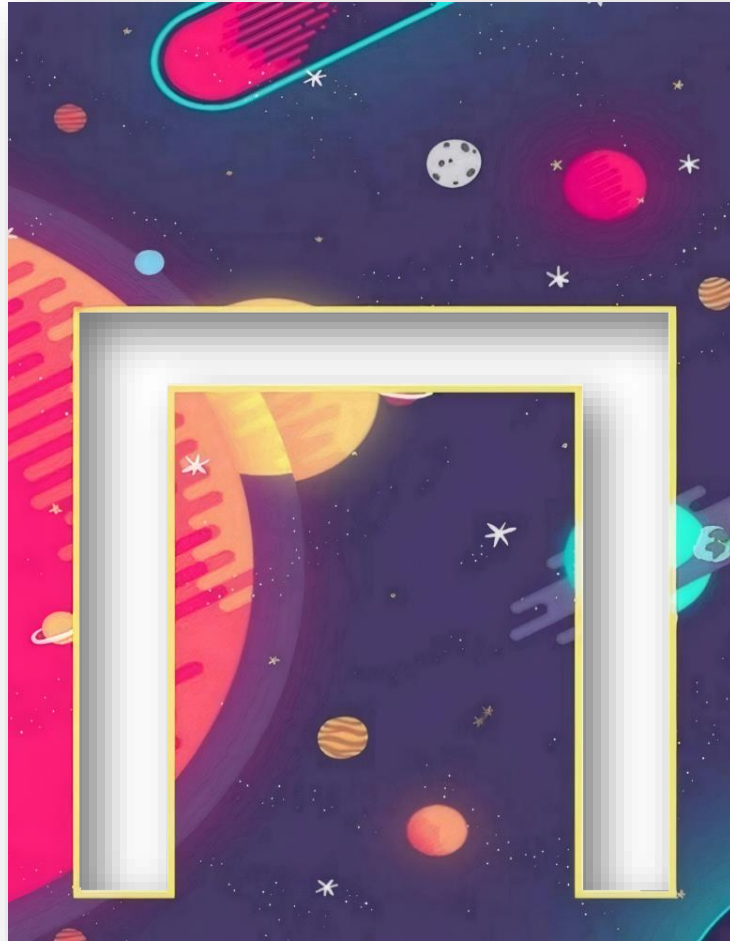
Le passage est effectué avec une animation de transition Fade :

`TransitionFade::create`

Lors de l'appui sur le bouton, la fonction précédente sera invoquée et permettra l'accès au niveau à l'aide de la commande :

```
CC_CALLBACK_1(MainMenu::GoToGameScene, this));
```


Cette scène représente le premier niveau du jeu. La structure du niveau est la suivante :



Dans cette scène, et au niveau du fichier GameScene.cpp, un Sprite a été en utilisant la ligne suivante :

```
auto sprite = Sprite::create("player.png");  
sprite->setPosition(Vec2(155, 28));  
this->addChild(sprite, 1);
```

Cette Sprite va prendre la position (155,28) qui représente sa position initiale puis ajoutée à la scène.

On a aussi un bouton qui va permettre de mettre le jeu en pause. Cela a été implémenté avec :

```
auto pauseItem =  
    MenuItemImage::create("Pause2.png",  
        "Pause2.png",  
        CC_CALLBACK_1(GameScreen::GoToPauseScene, this));
```

La fonction **CC_CALLBACK** permet d'appeler la fonction **GoToPauseScene** qui change la scène vers la scène **PauseScene**.

Ce bouton prend la position haute à gauche avec :

```
pauseItem->setPosition(Point(pauseItem->getContentSize().width -  
    (pauseItem->getContentSize().width / 4) + origin.x,  
    visibleSize.height - pauseItem->getContentSize().height +  
    (pauseItem->getContentSize().width / 4) + origin.y));
```

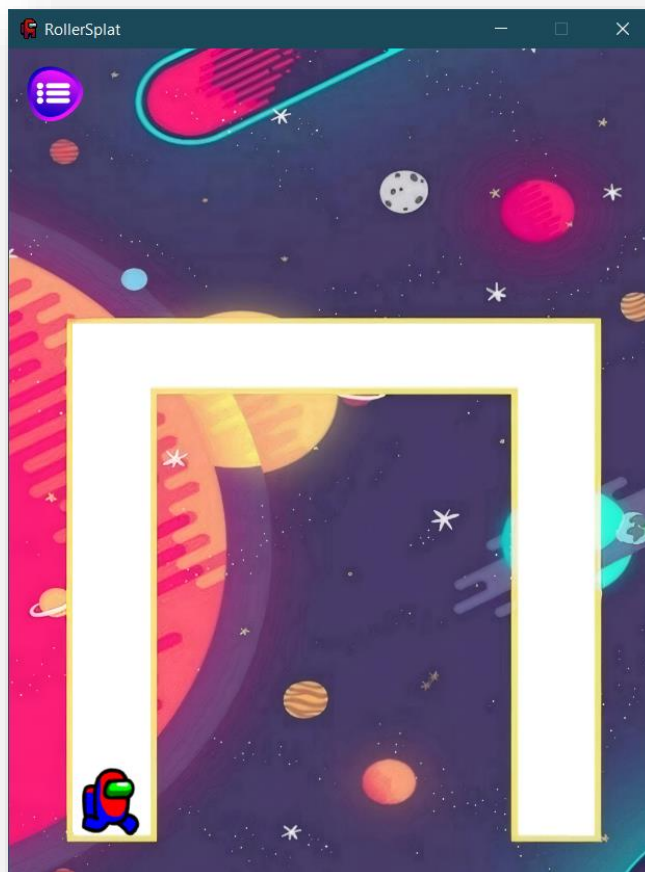
De même pour le joueur, l'arrière-plan a été ajoutée prendra la "couche" (layer)-1 de l'écran avec la ligne suivante :

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

Ensuite un autre deuxième arrière-plan blanc qui servira lors du mouvement du joueur (on verra son utilisation par la suite dans les chapitres qui viennent après)

```
auto back = Sprite::create("white.png");  
back->setPosition(Point(visibleSize.width / 2 + origin.x,  
    visibleSize.height / 2 + origin.y -50));  
this->addChild(back, -2);
```

Voici la scène complète :



Game scène 2 :

Cette scène représente le 2eme niveau du jeu. La structure du niveau est la suivante :



Dans cette scène, et au niveau du fichier `GameScene2.cpp`, un Sprite a été en utilisant la ligne suivante :

```
auto sprite = Sprite::create("player.png");  
sprite->setPosition(Vec2(149, 24));  
this->addChild(sprite, 1);
```

Cette Sprite va prendre la position (149,24) qui représente sa position initiale puis ajoutée à la scène.

On a aussi un bouton qui va permettre de mettre le jeu en pause. Cela a été implémenté avec :

```
auto pauseItem =  
    MenuItemImage::create("Pause2.png",  
        "Pause2.png",  
        CC_CALLBACK_1(GameScreen::GoToPauseScene, this));
```

La fonction `CC_CALLBACK` permet d'appeler la fonction `GoToPauseScene` qui change la scène vers la scène `PauseScene`.

Ce bouton prend la position haute à gauche avec :

```
pauseItem->setPosition(Point(pauseItem->getContentSize().width -  
    (pauseItem->getContentSize().width / 4) + origin.x,  
    visibleSize.height - pauseItem->getContentSize().height +  
    (pauseItem->getContentSize().width / 4) + origin.y));
```

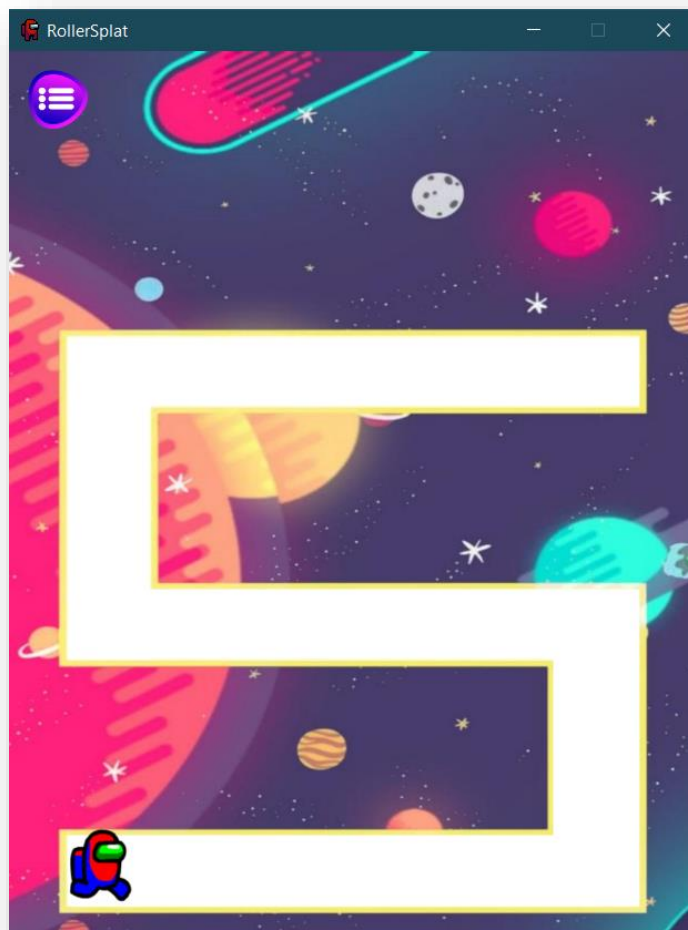
De même pour le joueur, l'arrière-plan a été ajoutée et prendra la "couche" (layer) -1 de l'écran avec la ligne suivante :

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

Ensuite l'arrière plan blanche.

```
auto back = Sprite::create("white.png");  
back->setPosition(Point(visibleSize.width / 2 + origin.x,  
    visibleSize.height / 2 + origin.y - 50));  
this->addChild(back, -2);
```

Voici la scene complete:



Game scène3 :

Cette scène représente le 3eme niveau du jeu. La structure du niveau est la suivante :



Dans cette scène, et au niveau du fichier `GameScene3.cpp`, une Sprite a été en utilisant la ligne suivante :

```
auto sprite = Sprite::create("player.png");  
sprite->setPosition(Vec2(144, 27));  
this->addChild(sprite, 1);
```

Cette Sprite va prendre la position (144,27) qui représente sa position initiale puis ajoutée à la scène.

On a aussi un bouton qui va permettre de mettre le jeu en pause. Cela a été implémenté avec :

```
auto pauseItem =  
MenuItemImage::create("Pause2.png",  
"Pause2.png",  
CC_CALLBACK_1(GameScreen::GoToPauseScene, this));
```

La fonction `CC_CALLBACK` permet d'appeler la fonction `GoToPauseScene` qui change la scène vers la scène `PauseScene`.

Ce bouton prend la position haute à gauche avec :

```
pauseItem->setPosition(Point(pauseItem->getContentSize().width -  
    (pauseItem->getContentSize().width / 4) + origin.x,  
    visibleSize.height - pauseItem->getContentSize().height +  
    (pauseItem->getContentSize().width / 4) + origin.y));
```

De même pour le joueur, l'arrière-plan a été ajoutée et prendra la "couche" (layer) -1 de l'écran avec la ligne suivante :

```
auto obs = Sprite::create("trajet 3.png");  
obs->setPosition(Point(visibleSize.width / 2 + origin.x,  
    visibleSize.height / 2 + origin.y));  
this->addChild(obs, -1);
```

Ensuite l'arrière-plan blanche.

```
auto back = Sprite::create("white.png");  
back->setPosition(Point(visibleSize.width / 2 + origin.x,  
    visibleSize.height / 2 + origin.y -15 ));  
this->addChild(back, -2);
```

Voici la scene complete:



Pause scène :

Cette Scène est sous forme d'un menu avec une arrière-plan. Elle peut être accessible depuis la scène du jeu avec la fonction suivante :

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

L'arriéré plan est sous forme d'une Sprite (image) :

```
auto obs =
Sprite::create("ddd.jpg");
obs->
>setPosition(Point(visibleSize.width
/ 2 + origin.x, visibleSize.height /
2 + origin.y - 20));
this->addChild(obs, -1);
```

Cette scène est composée d'un menu qui permet de retourner (continuer le niveau), réessayer depuis le ier niveau ou bien revenir au menu d'accueil

Retourner au niveau :

```
auto resumeItem =
MenuItemImage::create("Resume.png",
"Resume.png",
CC_CALLBACK_1(PauseMenu::Resume, this));
void PauseMenu::Resume(cocos2d::Ref* pSender)
{
    Director::getInstance()->popScene();
}
```



La fonction précédente permet de revenir vers la scène précédente en supprimant celle actuelle

Réessayer :

```
auto retryItem =
    MenuItemImage::create("Retry.png",
        "Retry.png",
        CC_CALLBACK_1(PauseMenu::Retry, this));
void PauseMenu::Retry(cocos2d::Ref* pSender)
{
    auto scene = GameScreen::createScene();
    Director::getInstance()->popScene();
    Director::getInstance()->replaceScene(scene); }
```

La fonction précédente permet de revenir relancer le niveau 1 en remplaçant la scène actuelle avec celle du niveau 1

Revenir au menu d'accueil :

```
auto mainMenuItem =
    MenuItemImage::create("MainMenu.png",
        "MainMenu.png",
        CC_CALLBACK_1(PauseMenu::GoToMainMenuScene, this));
void PauseMenu::GoToMainMenuScene(cocos2d::Ref* pSender)
{
    auto scene = MainMenu::createScene();
    Director::getInstance()->popScene();
    Director::getInstance()->replaceScene(scene);
}
```

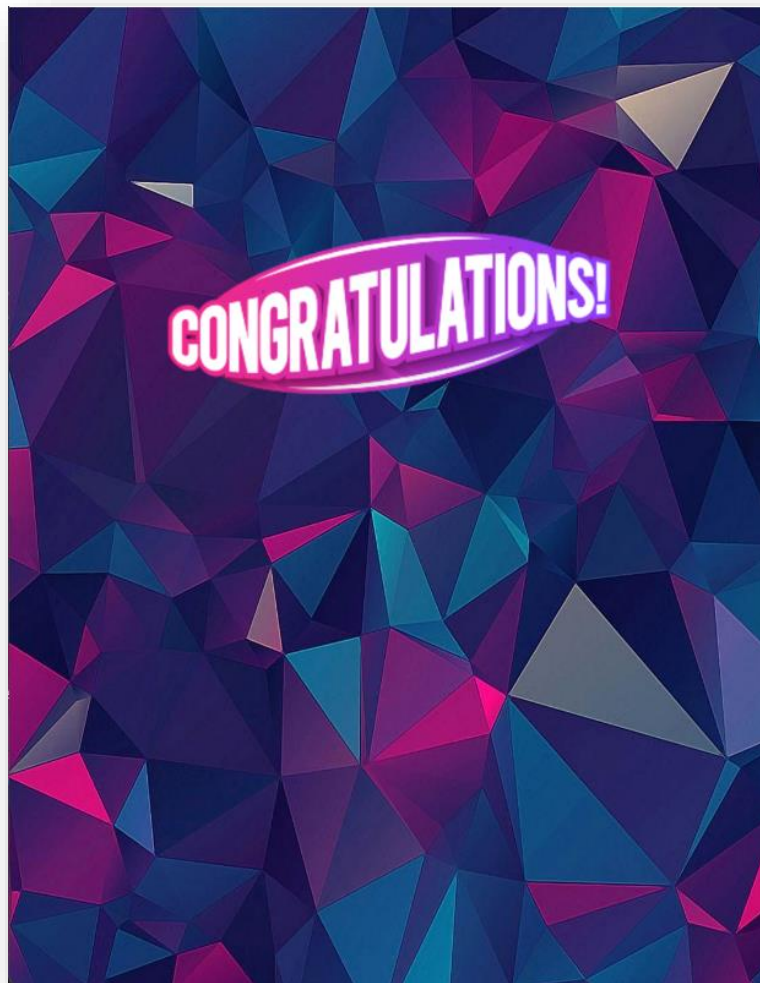
Tous ces éléments de menu ont été organisés avec :

```
menu->alignItemsVerticallyWithPadding(visibleSize.height / 16);
this->addChild(menu);
```


Cette scène est accessible seulement après avoir terminé le jeu.

Elle se compose d'un arrière-plan qui félicite et remercie le joueur d'avoir joué à ce jeu. L'arrière-plan est créé et positionnée avec :

```
auto back = Sprite::create("ddd.jpg");  
    back->setPosition(Point(visibleSize.width / 2 + origin.x,  
visibleSize.height / 2 + origin.y ));  
    this->addChild(back, -1);  
Le petit message de felicitation :  
auto cong = Sprite::create("cong.png");  
    cong->setPosition(Point(visibleSize.width / 2 + origin.x,  
visibleSize.height / 2 + origin.y + 60));  
    this->addChild(cong, 1);
```



Le déplacement du joueur :

Pour permettre de déplacer le joueur avec le clavier, un écouteur (listener) a été créé afin de trouver le bouton cliqué du clavier puis effectuer un déplacement selon la position du joueur (afin d'éviter l'utilisation des collisions). C'est-à-dire on limite le déplacement du joueur à seulement les déplacements possibles.

L'écouteur :

```
auto eventListener = EventListenerKeyboard::create();
eventListener->onKeyPressed = [&](EventKeyboard::KeyCode
keyCode, Event* event) {
    int offsetX = 0, offsetY = 0;
    float x, y;
```

Les variables offsetX et offsetY déterminent le déplacement du joueur selon le 2 axes X et Y, ainsi que les variables x et y vont servir à stocker la position du joueur. Exemple du déplacement à droite du niveau 1 :

```
case EventKeyboard::KeyCode::KEY_RIGHT_ARROW:
    case EventKeyboard::KeyCode::KEY_D:
        x = event->getCurrentTarget()->getPositionX();
        y = event->getCurrentTarget()->getPositionY();
        if ((x == 155 && y == 202))
        {
            CCLOG("success");
            offsetX = 172;
            drawNode = DrawNode::create();
            drawNode->drawSegment(Vec2(x, y), Vec2(x + 172, y),
19.0f, Color4F::GREEN);
            this->addChild(drawNode, -2); }
        else;
```

Dans le code ci-dessus, on va recevoir la position actuelle du joueur puis vérifier s'il est possible de se déplacer à droite. Si oui, la variable de l'axe des abscisses va recevoir la valeur 172 (valeur change selon le niveau) puis dessiner un segment à partir de la position initiale du joueur avant mouvement jusqu'à sa position après. L'arrière-plan blanche initialisée précédemment va couvrir la partie dans laquelle se déplace le joueur et le segment dessiné sera dans la couche entre l'arrière-plan de niveau et le joueur puisque cette partie de l'arrière-plan est vide.

```
auto moveTo = MoveTo::create(0.2, Vec2(event->
getCurrentTarget()->getPositionX() + offsetX, event->
getCurrentTarget()->getPositionY() + offsetY));
    event->getCurrentTarget()->runAction(moveTo);
this->_eventDispatcher->
addEventListenerWithSceneGraphPriority(eventListener, sprite);
```

Le code ci-dessus va exécuter le mouvement avec la fonction **MoveTo** et affecter ce déplacement à la Sprite (le joueur)

Le son :

Lors du démarrage du jeu, un fichier audio est lancé avec la commande suivante :

```
int id = AudioEngine::play2d("song.mp3");
```

Cette commande appartient à la bibliothèque des audios incluse dans cocos2d est déclarée au début du fichier **MainMenuScene.cpp**

```
#include "AudioEngine.h"
```