

**Objectif :**

**L’objectif principal de ce projet est de maitriser la programmation orientée objet par la mise en place d ‘un jeu vidéo 2D, le jeu proposé s’appelle PICO PARK, un jeu de type Plat former Puzzle Game.**

**Outils :**

* **C++**
* **Cocos2D**
* **Paint3D**
* **GitHub**
* **Visual Studio 2022**

**Plan de travail :**

* **• Préciser les classes à utiliser**
* **• Définir les caractéristique et le rôle de chaque classe**
* **• Création de scènes et de layer**
* **• Définir les méthodes et les fonctions**
* **• Sélection des images et des arrière-plans**



 **La programmation orientée objet :**

La programmation orientée objet **(POO)** est un paradigme de programmation qui met l'accent sur l'utilisation de "classes" et "objets" pour concevoir et structurer un programme.

Dans la **POO**;

* une classe est un modèle qui définit les caractéristiques et le comportement d'un objet.
* Un objet est une instance d'une classe, créée à partir de cette classe. Les objets partagent les mêmes caractéristiques et comportements, qui sont définis dans leur classe respective.

La POO permet de créer des programmes qui sont plus flexibles et modulaires, en permettant de réutiliser du code et de mieux gérer les différentes parties d'un programme. Elle est utilisée dans de nombreux langages de programmation, tels que Java, C++ et Python.

**C++**

Le C++ est un langage de programmation de haut niveau qui a été créé en 1979 et qui est largement utilisé dans de nombreux domaines, notamment la programmation de jeux vidéo. En raison de sa performance, de sa flexibilité et de sa richesse en termes de bibliothèques et de fonctionnalités, le C++ est souvent le choix de langage privilégié pour la création de jeux vidéo à haute performance et à haute qualité graphique.

Le C++ est un langage de programmation orienté objet qui permet aux développeurs de créer des classes et des objets qui encapsulent des données et des fonctionnalités. Cela rend le code plus facile à maintenir et à réutiliser, ce qui est particulièrement utile pour les projets de jeux vidéo complexes qui nécessitent souvent une grande quantité de code réutilisable.







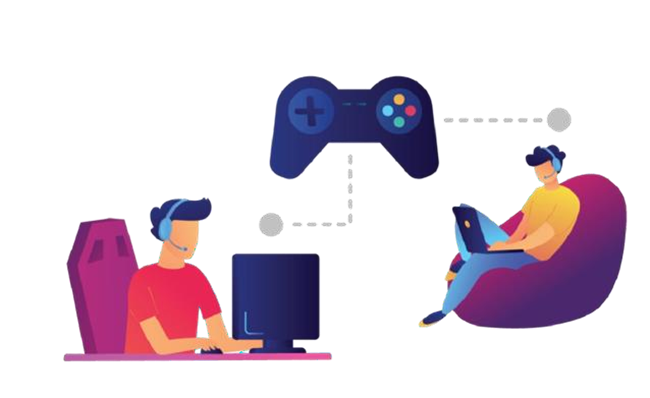
**Jeu Vidéo2D :**



Les jeux vidéo 2D sont des jeux vidéo qui sont présentés sur un écran en deux dimensions, généralement avec un affichage en 2D sur un plan horizontal ou vertical. Les jeux 2D ont été les premiers jeux vidéo créés et ont été très populaires pendant de nombreuses années avant l'arrivée des jeux 3D.

Il existe de nombreux genres de jeux 2D, notamment les jeux de plateforme, les jeux de combat, les jeux de puzzle et les jeux de stratégie. Les jeux 2D sont souvent plus simples à créer et à jouer que les jeux 3D et peuvent être joués sur une grande variété de plateformes, y compris les ordinateurs, les consoles de jeux, les téléphones portables et les tablettes.





**COCOS2D-X :**

Cocos2d-x est un moteur de jeu open-source pour la création de jeux 2D, de livres interactifs, de démos et d'autres applications graphiques. Il est écrit en C++ et conçu pour être multiplateforme, avec une prise en charge de iOS, Android, Windows, Mac et Linux. Le moteur fournit une gamme de fonctionnalités pour aider les développeurs à créer des jeux, notamment en matière de graphismes 2D, de physique, d'animation et d'audio. Il inclut également un éditeur de jeu simple appelé Cocos Creator, qui permet aux développeurs de créer et de tester rapidement des idées de jeux.

il y a beaucoup plus de jeux qui ont été développés en utilisant Cocos2d-x. Le moteur est particulièrement populaire pour la création de jeux 2D et a été utilisé pour créer une grande variété de jeux de différents genres, tels que l'action, les puzzles, la stratégie, etc.

Quelques exemples de jeu en utilisant le moteur Cocos2d-x :

**-PICO PARK :**

Pico Park est un jeu de puzzle d'action multijoueur local/en ligne coopératif pour 2 à 8 joueurs (2 à 10 joueurs dans la version classique) mais on a essayé de le développer en mode Monoplayer. Le but du jeu est d'obtenir une clé et de déverrouiller la porte verrouillée à la fin de la carte. Les joueurs doivent coopérer pour atteindre l'objectif, mais les joueurs ont également la possibilité d'empêcher d'atteindre cet objectif, par exemple en bloquant les mouvements des autres joueurs.

**-Fruit Ninja**

**-Plants vs Zombies 2**



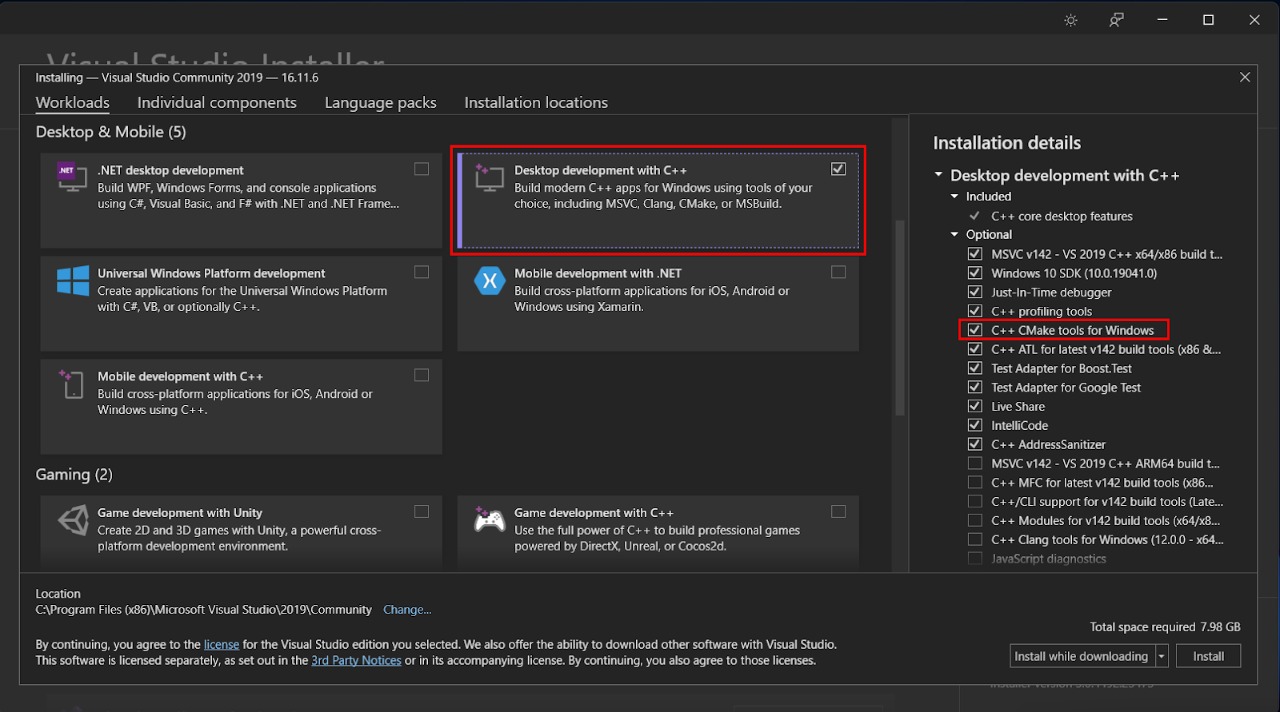


**Installation COCOS2D-X :**

[Installer Microsoft Visual Studio](https://rezghob.com/installing-cocos2d-x-windows/#install_visual_studio) :

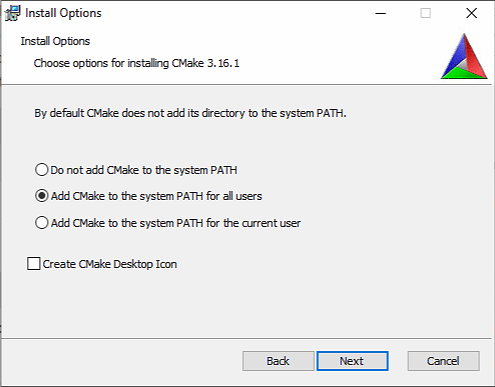
Microsoft a une version gratuite de son IDE appelée "Visual Studio Community". Il existe également une version open source gratuite appelée "Visual Studio Code". Je vais opter pour la Communauté car c'est ce qui est supporté par Cocos2d-x.

* Téléchargez et exécutez l'installateur de Visual Studio Community.
* Dans l'onglet Disponible, installez Visual Sudio 2022 v.04
* Dans l'onglet Charge de travail, on a sélectionner les unités "Développement de jeux en C++" et "Stockage et traitement des données".



[Installation CMake](https://rezghob.com/installing-cocos2d-x-windows/#install_cmake):

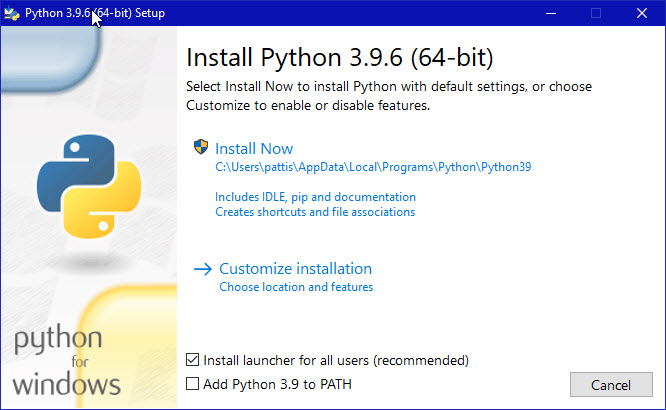
Cocos2d-x 4.0 utilise CMake pour créer des fichiers de projet pour différentes plates-formes. Pour cette raison, nous avons besoin de CMake installé sur notre machine.



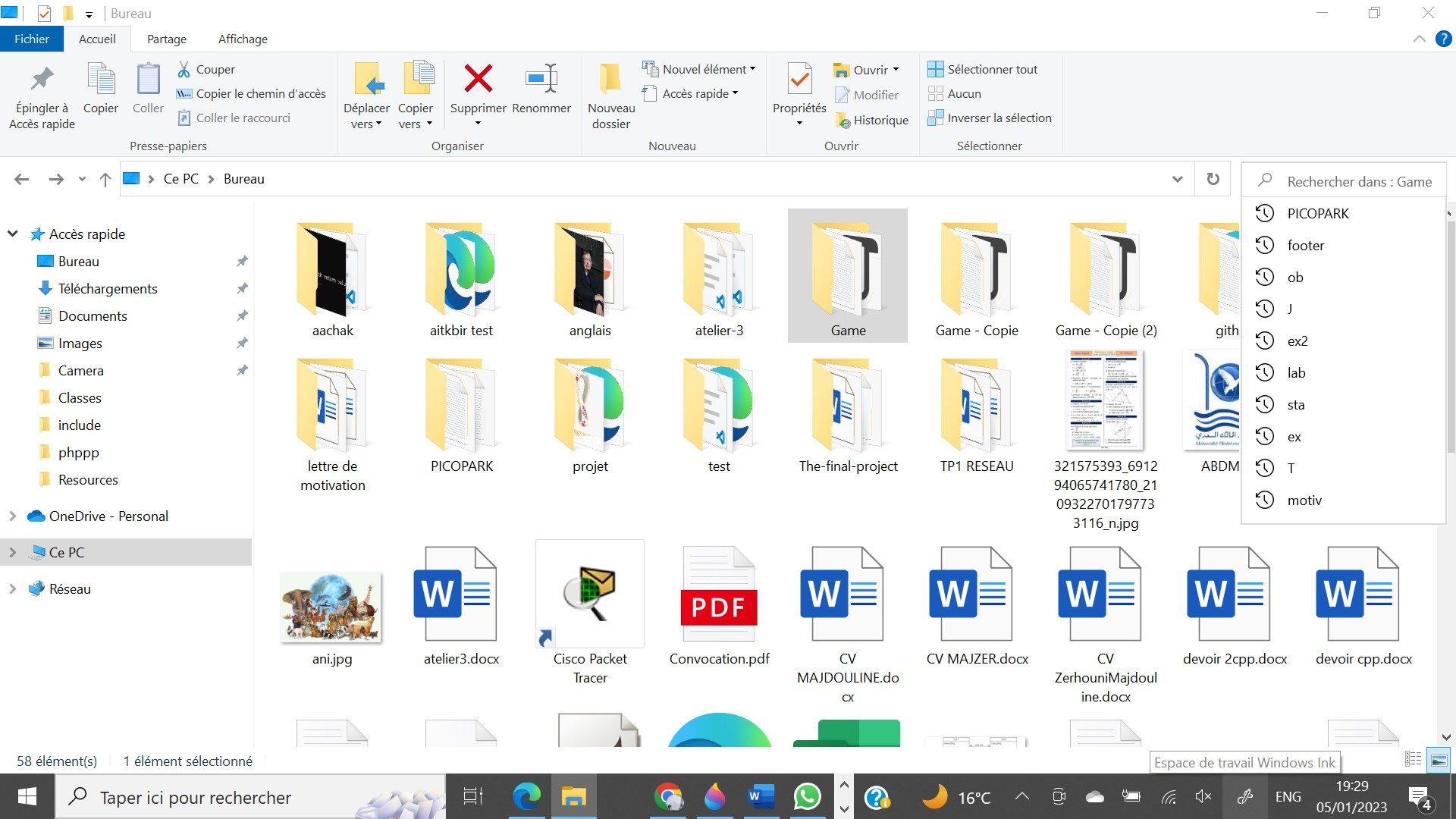
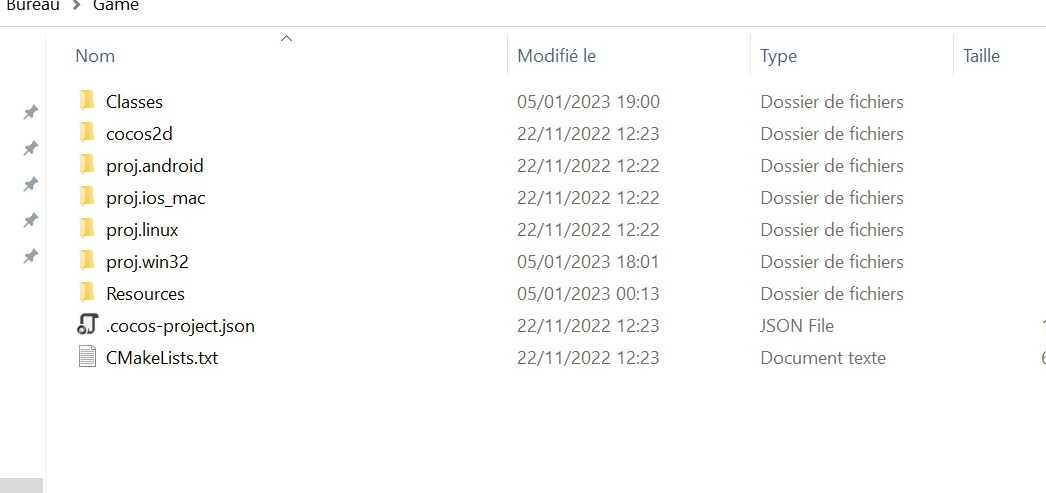


[Installation Cocos2d-x 4.0](https://rezghob.com/installing-cocos2d-x-windows/#install_cocos2dx) :

Pour installer Cocos2d-x, on a besoin d’installer Python 2.7x :



Téléchargement de fichier d'installation de Cocos2d-x à partir du site Web officiel de Cocos2d-x :

L’execution de Helloworld.h et Helloworld.cpp:

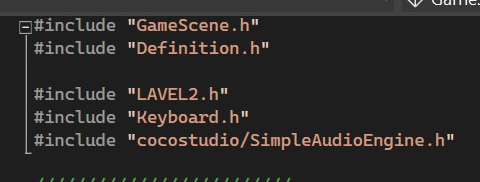






**Implémentation :**

Les classes utilisées : Dans cette bibliothèque cocos2d-x pour la majorité de classe on trouve la déclaration de deux fichiers : le premier est de type « .h » qui sert à déclaration du classe, les méthodes … et le deuxième est de type « .cpp » qui sert à la définition et les implémentations. Dans chaque classe, avant de commencer, on fait l’appel à des bibliothèques sous cette forme, #include “NomClass.h” car aura besoin des fonctions et des variables qui sont définies dans ces classes, c’est un principe fondamental pour la POO.

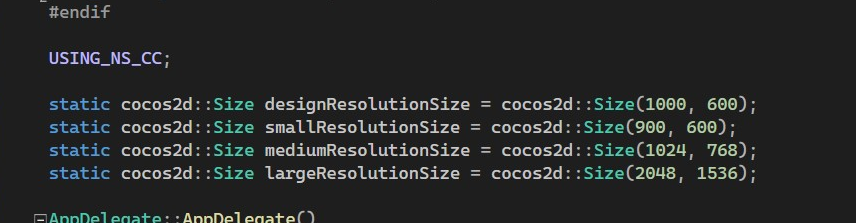
****

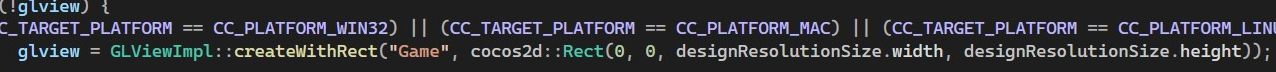
1. On a créer plusieurs fichiers sur le dossier classes pour manipuler le jeu ,chaque fichier a son role



**2-AppDelegate.h / AppDelegate.cpp:**

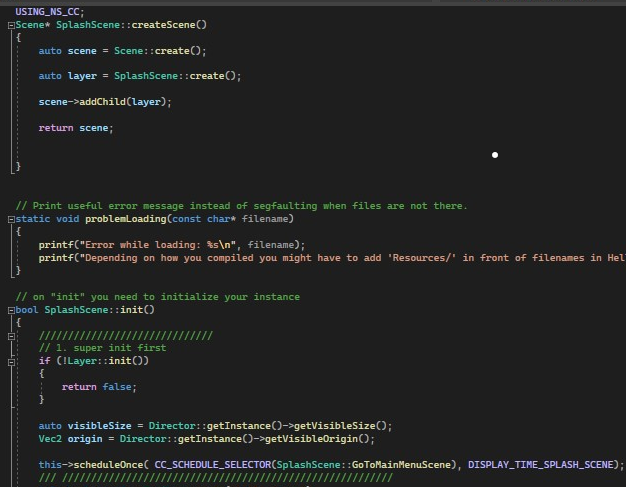
Cette classe est générée automatiquement par cocos2d , elle sert à : changer le nom de notre jeux , redimensionner la taille de notre fenêtre , création de notre scènes et l’appel de la scène suivante (**SplashScene**)

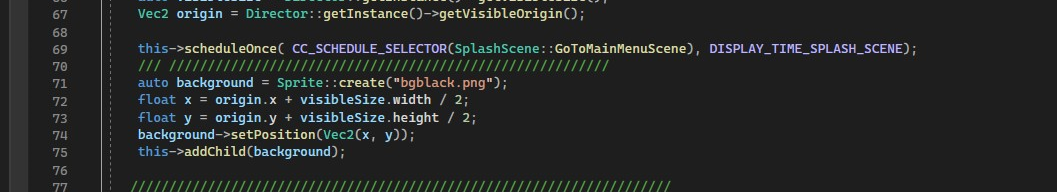




**3-SplashScene.h / SplachScene.cpp:**

La scène de démarrage, également connue sous le nom de scène de splash, est une scène qui s'affiche lorsque vous démarrez un jeu donc nous avons créé notre scène initiale (**SplashScene**) Ainsi nous avons initialisé notre variable local (**Director**):

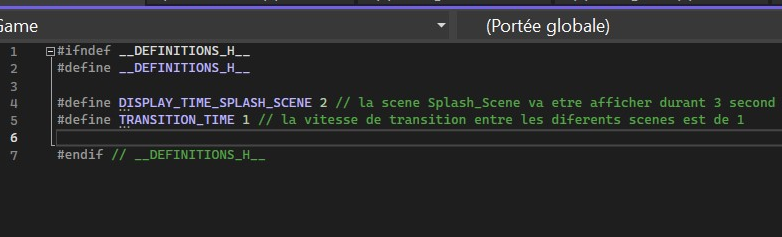


* Dans cette class tout d’abord on fait la création de notre scène puis on fait la création d’un variable local(layer) et on l’ajout par la fonction « addChild »
* Grace à la fonction scheduleOnce et DISPLAY\_TIME notre scène va disparaitre après qlq second puis on passe à la scène suivante (MaineMenue).la position du background de notre scène est déterminé avec la fonction : setPosition ,Et bien sur la fonction addchild permet l’ajout de ce background, par contre le deuxième addchild permet d’ajouter un autre petit background sur le premier qui est aussi un sprite créé  
  
* Et finalement nous avons créé une fct de type void permettant de passer à la scène suivante (MainMenue), et bien sûr il faut créer cette dernière avec la fonction createScene



**4-Definitions.h :**

Ce fichier sert à la définition de chaque variable qu’on va utiliser après ainsi la vitesse de transition entre les scènes





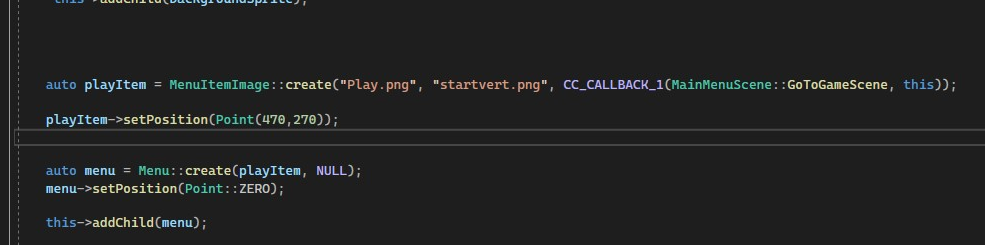




**5 – MainMenu.h / MainMenu.cpp**

Cette classe sert à insérer un menu pour note jeu ; ce menu contient un background et un bouton (Play) qu’on a inséré à l’aide des fonctions suivantes :

Sprite sert à ajouter notre photo à la scène, float x et float y précise la position exacte de cette photo et finalement comme nous avons besoin d’un bouton cliquable affin de commencer notre jeu nous avons utilisé la fonction playItem qui sert de rendre une image cliquable. Et finalement nous avons remplacé cette scène avec la suivante (GameScène) avec la fonction déjà expliquer en dessus.

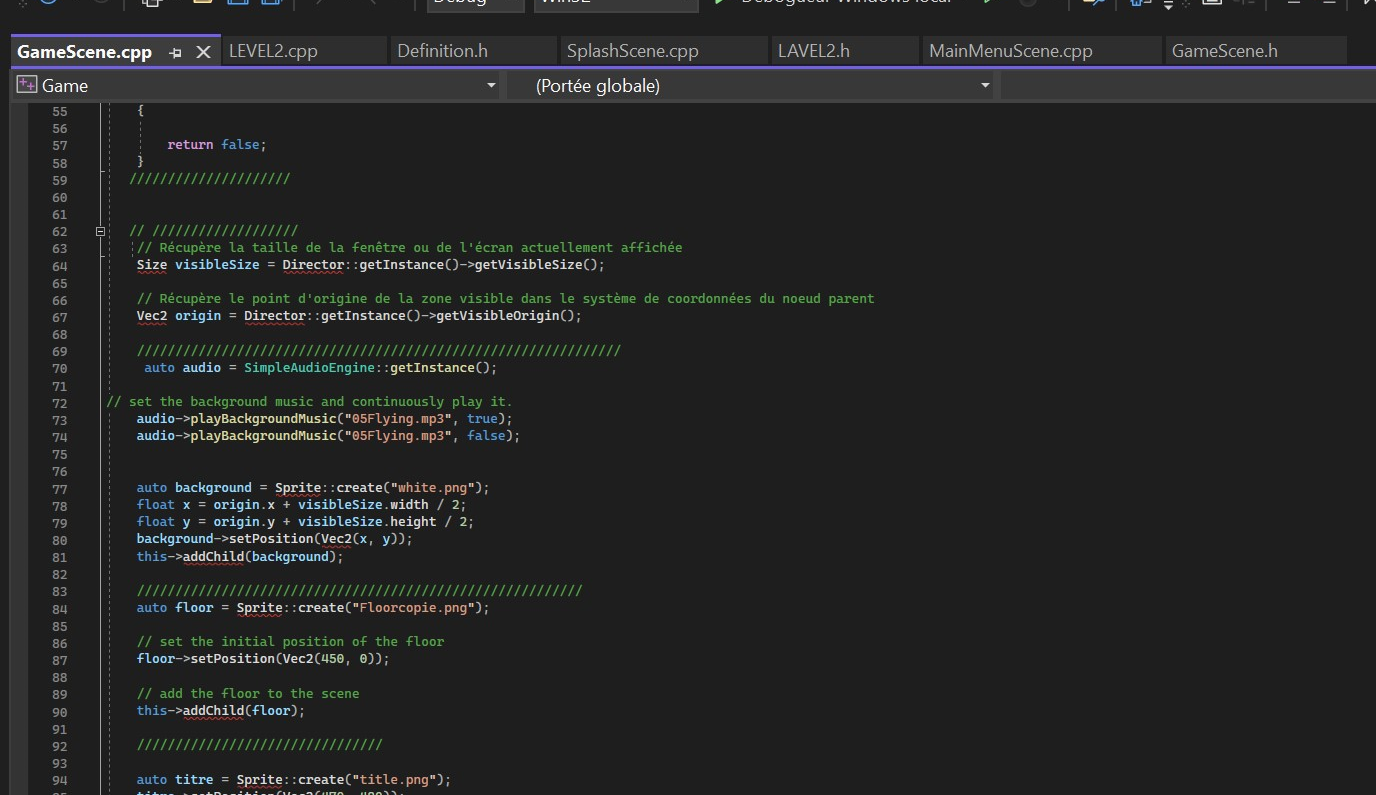


**4-GameScene.h / GameScene.cpp :**

C’est Le fichier code source qui contient le code de la scène principale du jeu il représente la création de la scène, Layer et d’ajouter Layer en tant qu’enfant à la Scène et aussi l’ajout de notre arrière-plan en précisant ses coordonnées et sa taille d’affichage sur la fenêtre de l'écran

on a aussi determiner l’image de notre monoplayer et en même temps on a précisé les coordonnées de l’image avec la fonction setPosition





notre monoplayer representer par un sprite ="Cat" qu'est une image qui affiche à l'écran et qui fait animer . et on sait déjà que Les spirites sont souvent utilisés dans les jeux et les applications d'animation pour représenter des personnages, des objets ou des éléments de l'environnement.

on a aussi déterminer l’image de notre monoplayer et en même temps on a précisé les coordonnées de l’image avec la fonction setPosition.

on a utiliser la fonction qui va nous per permetter de faire déplacer le sprite et on a associe une images "chat PICO PARK" à lui, on ajouter aussi les propriétés physiques, comme la vitesse, et de matter ce sprite en collision avec d'autres sprite pour creer un passage a level2







on a aussi déterminer l’image de notre monoplayer et en même temps on a précisé les coordonnées de l’image avec la fonction setPosition