PICO PARK

COMPTE RENDU C++

Jeux vidéo cocos 2d

- **Réalisé par :** Sabri Basma
  Fertat Oumaima

- **Encadrée par:** Ikram Ben Abdel Wahab

-Année scolaire :2022/2023

I. Processus de développement :

1.1- Installation des outils :

- Pour réaliser ce projet on a utilisé plusieurs outils :

1.1.1- Cocos2d-X :

- **Cocos2d** est un *framework* libre en Python, permettant de développer des applications ou des jeux vidéo 2.

Des jeux comme *FarmVille 3,* Geometry Dash ou *Angry Birds Fight!*[4] ont été développés avec Cocos2D

Création du nouveau projet Cocos2d :

- Pour faire de nouveaux projets, nous utiliserons la commande : “Cocos new 'ProJect name’ -l Cpp” dans le powershell.

1.1.2- 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.

Création Cmake dans notre fichier :

Pour ajouter Cmake a notre projet on utilise le command dans le fichier win32 : « Cmake .. -G « Visual Studio 16 2019 » -Awin32 ».

1.1.3- 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 . On va aller avec la communauté car c'est ce qui est supporté par Cocos2d-x.

• Télécharger et exécuter le programme d'installation de la communauté Visual Studio.
• Depuis l'onglet Disponible, installez Visual Studio 2017 (vous pouvez aussi opter pour 2019 !)
• Dans l'onglet Workloads, je recommande de sélectionner les unités « Python development », « Game development with C++ » et « Data storage and processing ». Vous en aurez probablement besoin plus tard.
• Appuyez sur Installer (ou Modifier) et suivez les instructions.

1.1.4- Langage C++ :

C++ est le langage de programmation le plus utilisé par les développeurs, notamment en ce qui concerne les applications.Il permet d'abord le développement sous plusieurs paradigmes : programmation génériques, procédurales et orienté objet.
Dans notre projet on a basé sur l'orienté objet en C++ qui est un paradigme informatique consistant à définir et à faire interagir des objets grâce à différentes technologies, notamment les langages de programmation (python, java, C++…).

1.1.5- Adobe Photoshop :

On a utilisé Adobe Photoshop pour créer les Sprite qu'on a ajouté à notre code.

1.2- Présentation du projet :

1.2.1- Logique de travail :

- On a commencé par la création de notre scène avec les physiques, et on a ajouté la gravité a notre scène, chaque <Level> a une différente map qu'on a créé en ajoutant des <Sprite> (une classe prédéfinie dans cocos qui sert a ajouté des photos) et par donner des <physical Body> pour chaque <Sprite> comme ça on peut créer un terrain où notre joueur peut déplacer. Puis en ajoutant le <Sprite> du joueur et lui donner un <physical Body> aussi maintenant on a un joueur sur un terrain qu'on a créé. Pour que notre code ne soit pas répétitif et on a décidé de séparer les différents objets dans des différents classes : par exemple une classe pour le joueur , On a créé le joueur à partir du <Sprite> et on lui donne des setters et getters pour qu'on peut l'accéder aux classes des Levels, en répétant ça pour tous les objets. Tout ce qu' on a besoin maintenant c'est

le mouvement du joueur et un puzzle pour chaque Levels et une fonction pour passer d'un level à un autre et fonction pour détecter les collisions.

1.2.2- Mouvement du joueur :

La déclaration :

```
void moveright(float dt);
void moveleft(float dt);
void movetop(float dt);
void movebot(float dt);
```

La définition :

```
void MyWorld::moveright(float dt) {
    Vec2 playerPos = player->getposition();
    player->setposition(playerPos.x + 80* dt, playerPos.y);
}
void MyWorld::moveleft(float dt) {
    Vec2 ballpos = player->getposition();
    player->setposition(ballpos.x - 80 * dt, ballpos.y);
}
void MyWorld::movebot(float dt) {
    Vec2 ballpos = player->getposition();
    player->setposition(ballpos.x, ballpos.y -80 * dt);
}
void MyWorld::movetop(float dt) {
    Vec2 ballpos = player->getposition();
    player->setposition(ballpos.x, ballpos.y + 160 * dt);
}
```

Pour déplacer le joueur dans notre scène à l'aide de notre « Keyboard » on a créé un « EventListener Keyboard » en utilisant une fonction « onkeypressed »
Pour arrêter le mouvement du joueur quand on arrête de toucher le bouton « Keyboard » on utilise une fonction « onkeyreleased »

1.2.3- Fonction update :

C'est la méthode qu'on va utiliser pour savoir dans chaque moment où est notre joueur, et modifier la scène avec ses mouvements.
Par exemple, le mouvement de la caméra avec notre joueur et la fonction de perdre et la fonction de gagner.

Déclaration :

```
void update(float dt);
```

Définition :

```
void Lvl1::update(float dt) {

        Rect rect1 = player->getrect();
        Rect rect2 = key->getrect();
        Rect rect3 = door->getrect();
        Rect rect4 = lava->getrect();
        Rect rect5 = ground->getBoundingBox();

        auto v = Director::getInstance()->getWinSize();
        Layer::setAnchorPoint(Vec2(player->getposition().x / v.width, player->getposition().y /v.height));

        if (rect1.intersectsRect(rect2))
        {
            key->setposition(player->getposition().x-10, player->getposition().y + 10);
            if (!isKeyCollected)
            {
                cocos2d::AudioEngine::preload("audio/key.mp3");
                cocos2d::AudioEngine::play2d("audio/key.mp3", false, 0.3f);
                isKeyCollected = true;
            }
```

Par exemple ici on détecte si notre joueur est dans la même position que la clé si oui on donne a notre clé la position du joueur pour que le suivre.

Fonction pour gagner :

```
    }
    if (rect1.intersectsRect(rect3) && isDoorOpened == true && isUpPressed==true) {
        isUpPressed = false;
        auto scene = Lvl2::createScene();
        Director::getInstance()->replaceScene(TransitionFade::create(0.2, scene));
    }
    if (isApressed)
```

Et ici on détecte si le joueur et dans la même position que la porte et si la porte est ouverte et si on a cliqué sure flèche haut si oui on passe dans niveau prochain

1.2.4- Fonction OnContactBegin :

C'est la méthode qu'on va utiliser pour détecter les collisions dans notre jeu et faire quelque chose quand on a une collision.

Déclaration :

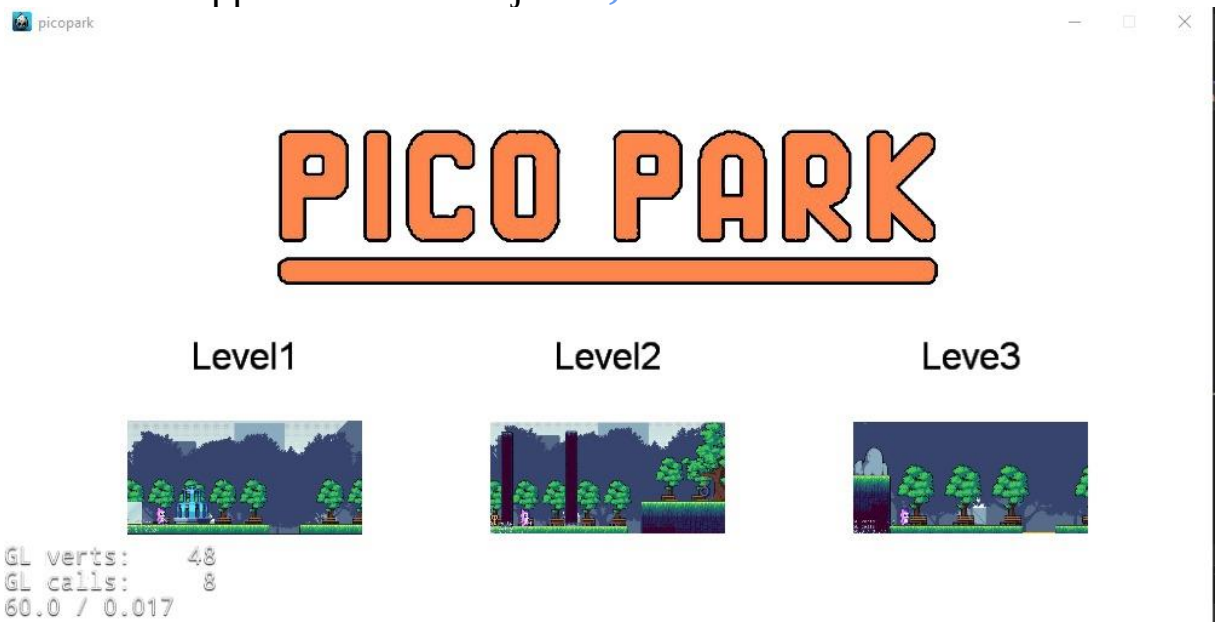
```
bool onContactBegin(cocos2d::PhysicsContact& contact);
```

Définition :

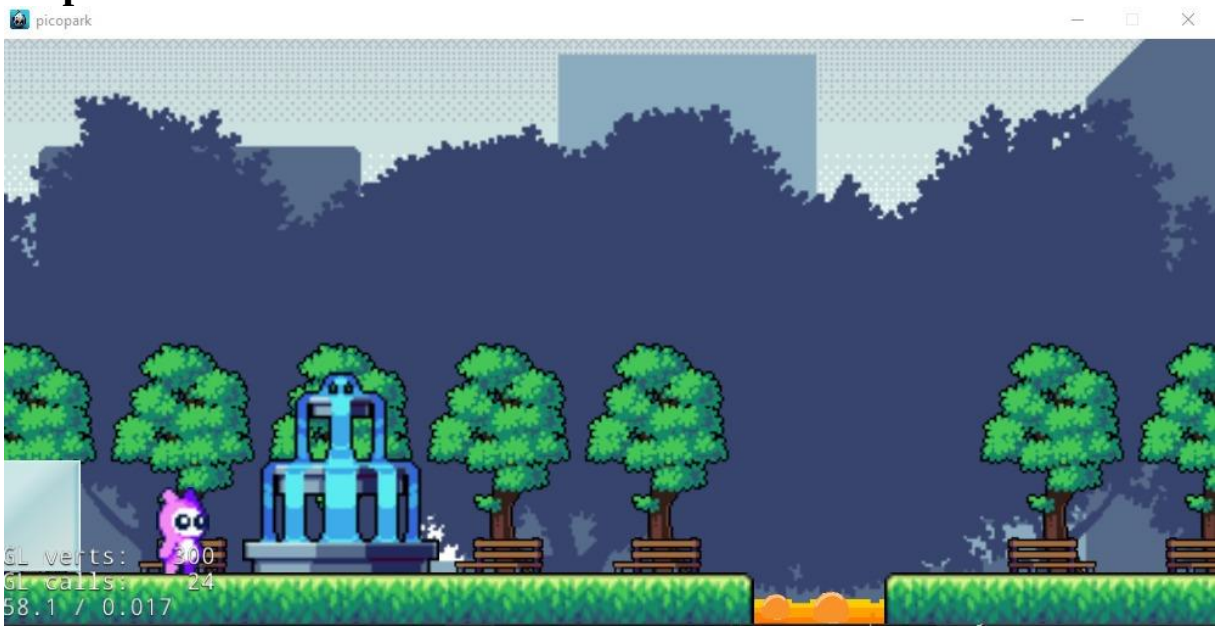
```
if ((1 == a->getCollisionBitmask() && 2 == b->getCollisionBitmask()) || (2 == a->getCollisionBitmask() && 1 == b->getCollisionBitmask()))
{
    ifSpacePressed = false;
    isOnGround = true;
}
if ((1 == a->getCollisionBitmask() && 3 == b->getCollisionBitmask()) || (3 == a->getCollisionBitmask() && 1 == b->getCollisionBitmask())) {
    if (ifDpressed)
    {
        this->unschedule(SEL_SCHEDULE(&MyWorld::moveright));
        player->setposition(player->getposition().x - 10, player->getposition().y);
    }
    else if (ifApressed)
    {
        this->unschedule(SEL_SCHEDULE(&MyWorld::moveleft));
        player->setposition(player->getposition().x + 10, player->getposition().y);
    }
}
if ((1 == a->getCollisionBitmask() && 4 == b->getCollisionBitmask()) || (4 == a->getCollisionBitmask() && 1 == b->getCollisionBitmask()))
{
    ispushing = true;
    ifSpacePressed = false;
    if (ifDpressed) {
        box->setposition(box->getposition().x + 4, box->getposition().y);

    }
    else if (ifApressed)
    {
        box->setposition(box->getposition().x - 4, box->getposition().y);
    }

}

if ((3 == a->getCollisionBitmask() && 4 == b->getCollisionBitmask()) || (4 == a->getCollisionBitmask() && 3 == b->getCollisionBitmask()))
{
    box->fall();
    box->removeFromParent();
```

Développements de jeux :

on a développé 3 niveaux de jeux ;

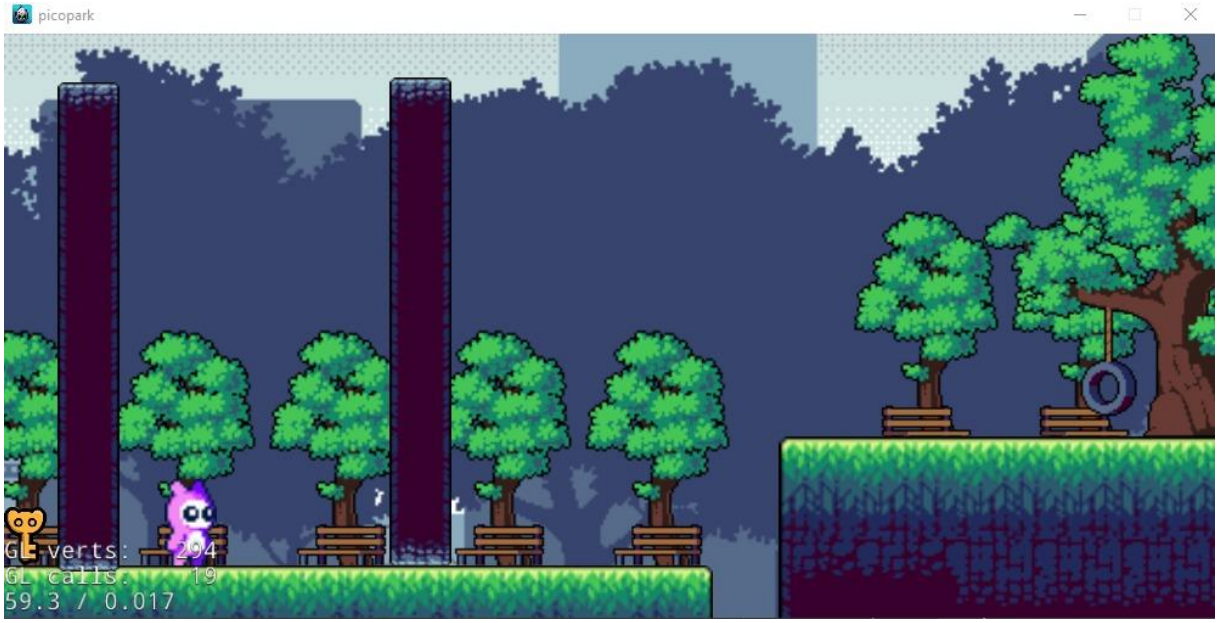


Le premier niveau :



C’est la scène du premier niveau ,le joueur doit dépasser les balles et prendre la clé pour qu’il puissent ouvrir la porte après avoir dépassé l’obstacle

Le deuxième niveau :



La scène du deuxième niveau ,le joueur doit pousser les barrières pour qu’il puisse prendre la clé et termine vers la porte en passant par les obstacles .

Le Troisième niveaux :

C’est la scène du troisième niveau le joueur doit passer les obstacles pour arriver vers la porte et terminer le niveau .

2. Vidéo démonstrative

https://www.youtube.com/watch?v=ZXBqbHnlp9A

3 .Difficulté rencontrée :

On a trouvé des difficultés pour trouver des documentations et des tutoriaux pour cocos, car la documentation n’a pas des exemples et pas très bien définis.

Et la communauté de cocos est très limitée, Quand on est coincé dans un problème il n’y a pas beaucoup de références qu’on peut voir.
Et on a trouvé une difficulté aussi dans la création de niveau avec diffèrent concept et puzzle pour chaque niveau car dans <pico Park> la plupart des niveaux sont des niveaux multi-joueurs qu’on ne peut pas l’implémenter dans un jeu à un seul Joueur.

Conclusion

Après avoir programmé ce jeu on a appris des nouvelles choses comme c’est quoi cocos2d-x ses différentes fonctions qui sont déjà pris définit la logique des jeux et pour terminer c’était amusant de programmer ce jeux .