Compte rendu

Jeu Vidéo 2D C++ COCOS2D-x

Pico Park

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 Platformer Puzzle Game.

Travail demandé :

• Développer le jeu 2D via le moteur de jeu Cocos2D « C++ » avec le respect du paradigme POO.

• Il faut au moins développer 3 niveaux de jeu.

• Le jeu doit être en mode mono Player.

Outils : C++, Cocos2D, Photoshop/Gimp.

Encadré par : Pr. Lotfi ELAACHAK

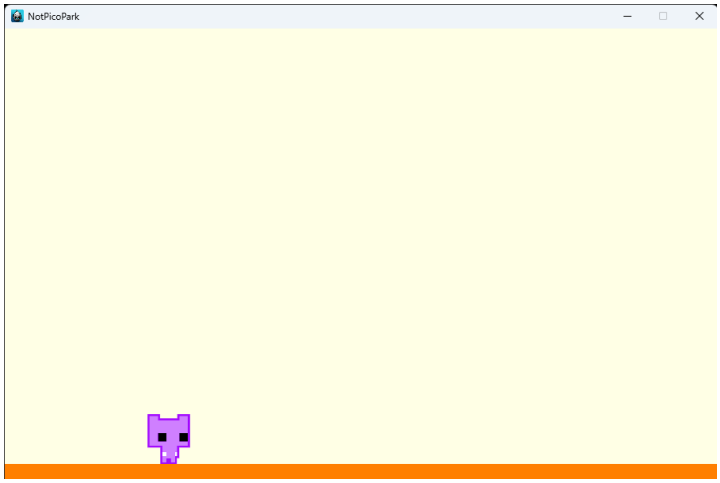
Réalisé par : Abdelali IBN TABET

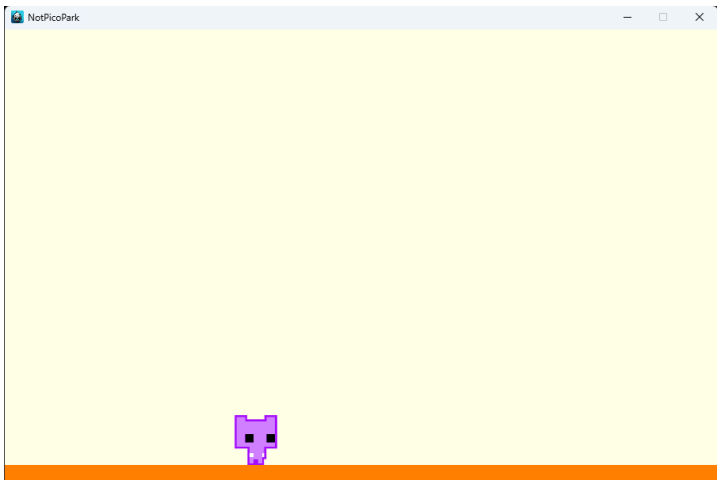
Ahmed HOURRI

Note : Notre projet est incomplet

Game view :

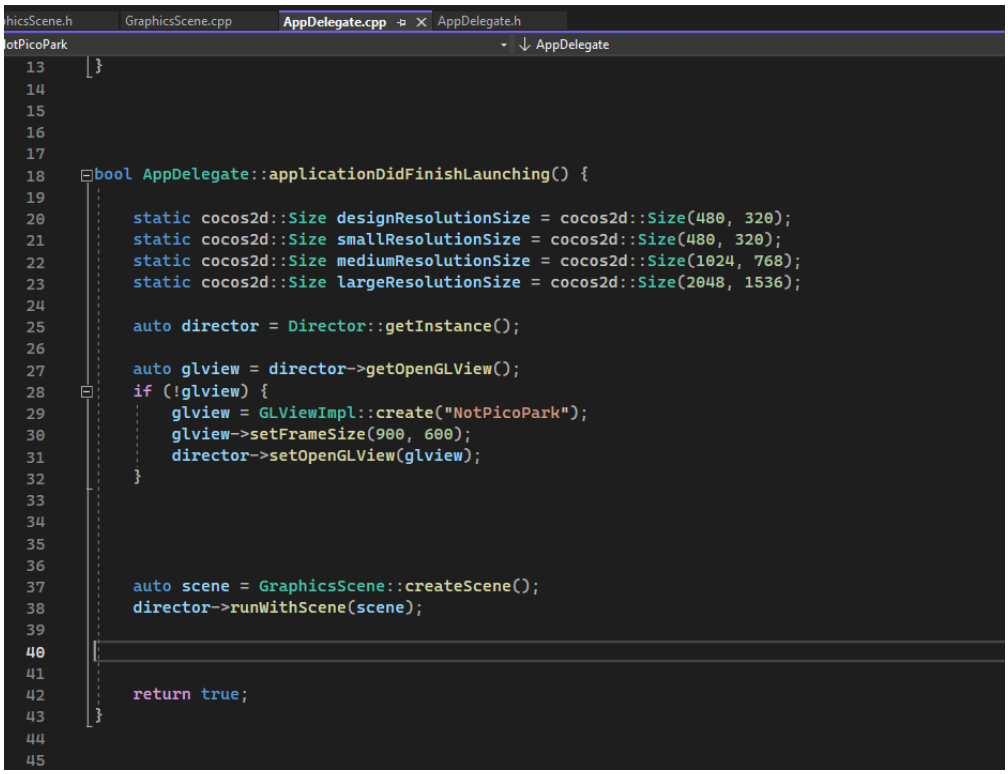
Le joueur peut se déplacer a droite et a gauche et peut sauter.





Code source :

AppDelegate.cpp




```cpp
}



bool AppDelegate::applicationDidFinishLaunching() {

    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);

    auto director = Director::getInstance();

    auto glview = director->getOpenGLView();
    if (!glview) {
        glview = GLViewImpl::create("NotPicoPark");
        glview->setFrameSize(900, 600);
        director->setOpenGLView(glview);
    }




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


    return true;
}
```

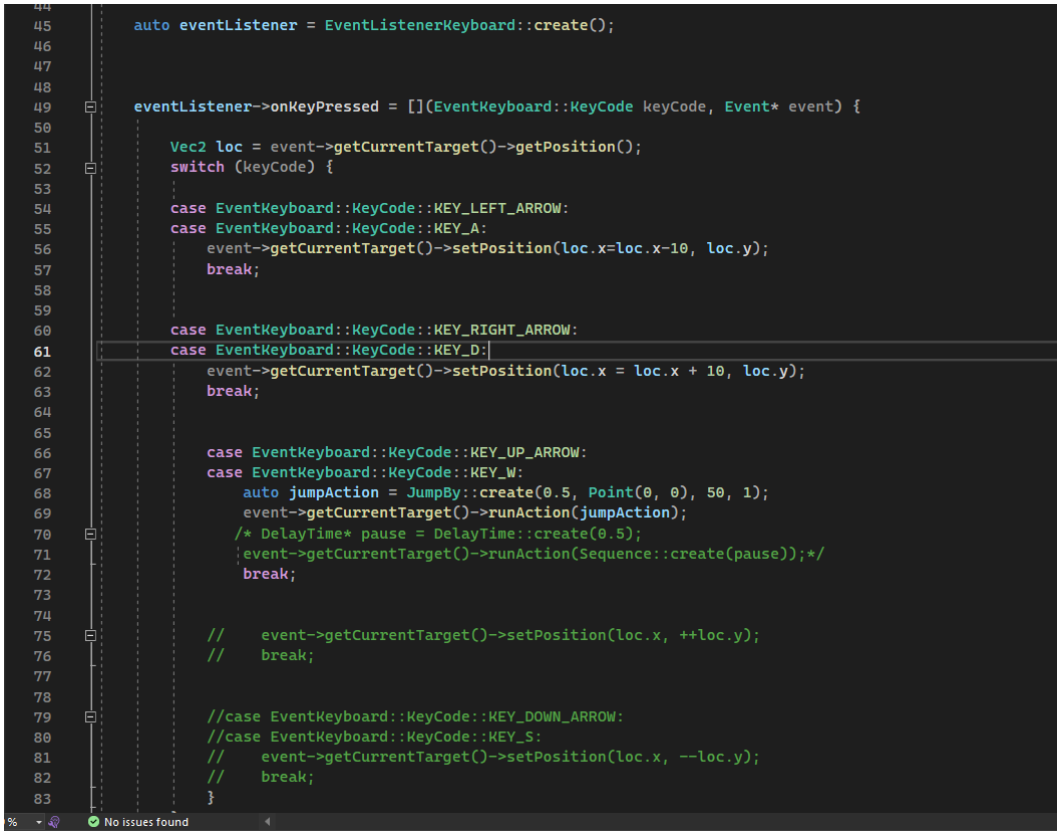
On règle quelques paramètres de résolution, on met la résolution de la fenêtre du jeu a 900x600 et on la nomme « NotPicoPark ». On crée la scène GraphicsScene.

GraphicsScene.cpp

```cpp

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


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

    auto background = DrawNode::create();
    background->drawSolidRect(origin, winSize, Color4F(1, 1, 0.9, 1.0));
    this->addChild(background);

    auto player = Sprite::create("player1.png");
    player->setScale(0.4);
    player->setAnchorPoint(Vec2(0, 0));
    player->setPosition(Vec2(200, 50));

    auto ground = Sprite::create("ground.png");
    ground->setScale(1.0);
    ground->setAnchorPoint(Vec2(0, 0));
    ground->setPosition(Vec2(0, 0));



    this->addChild(player, 0);
    this->addChild(ground, 0);


    auto eventListener = EventListenerKeyboard::create();



    eventListener->onKeyPressed = [](EventKeyboard::KeyCode keyCode, Event* event) {

        Vec2 loc = event->getCurrentTarget()->getPosition();
        switch (keyCode) {

```

On crée un background qui va être une couleur qui s’affiche en fond arrière du jeu,

Et on crée des sprites pour le joueur et ground, et puis on les donne des positions dans la fenêtre du jeu.



```
auto eventListener = EventListenerKeyboard::create();



eventListener->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()->setPosition(loc.x=loc.x-10, loc.y);
        break;


    case EventKeyboard::KeyCode::KEY_RIGHT_ARROW:
    case EventKeyboard::KeyCode::KEY_D:
        event->getCurrentTarget()->setPosition(loc.x = loc.x + 10, loc.y);
        break;


        case EventKeyboard::KeyCode::KEY_UP_ARROW:
        case EventKeyboard::KeyCode::KEY_W:
            auto jumpAction = JumpBy::create(0.5, Point(0, 0), 50, 1);
            event->getCurrentTarget()->runAction(jumpAction);
           /* DelayTime* pause = DelayTime::create(0.5);
            event->getCurrentTarget()->runAction(Sequence::create(pause));*/
            break;


        //    event->getCurrentTarget()->setPosition(loc.x, ++loc.y);
        //    break;


        //case EventKeyboard::KeyCode::KEY_DOWN_ARROW:
        //case EventKeyboard::KeyCode::KEY_S:
        //    event->getCurrentTarget()->setPosition(loc.x, --loc.y);
        //    break;
        }
```

No issues found

On ajoute un eventListener aux touches du clavier qu’on a besoin pour contrôler le joueur.