RAPPORT PICO PARK

Première scène :

```
auto scene = SplashScene::createScene();

    // run
    director->runWithScene(scene);

    return true;
```

La SplashScene est une classe qui représente une scène dans le jeu. Il est probable que la SplashScene soit la première scène affichée lorsque le jeu est lancé. La méthode createScene est une méthode statique qui crée une instance de la classe SplashScene et la renvoie sous forme d'objet Scene. La méthode runWithScene de l'objet Director est responsable de l'exécution de la scène, ce qui signifie qu'elle est affichée à l'écran.

Comment la SplashScene a été crée ?

```
1. auto scene = Scene::create();
2.     auto layer = SplashScene::create();
3.     scene->addChild(layer);
4.     return scene;
5.
```

```
1. auto visibleSize = Director::getInstance()->getVisibleSize();
2.     Vec2 origin = Director::getInstance()->getVisibleOrigin();
3.
4.     this->scheduleOnce(schedule_selector(SplashScene::GoToMainMenuScene),
DISPLAY_TIME_SPLASH_SCENE);
5.     //create background sprite
6.     auto backgroundSprite = Sprite::create("pico Splash.jpeg");
7.     backgroundSprite->setPosition(Point(visibleSize.width / 2 + origin.x, visibleSize.height /
2 + origin.y));
8.     this->addChild(backgroundSprite);
9.
```

Une méthode de la classe SplashScene pour aller à une autre scène :

```
void SplashScene::GoToMainMenuScene(float dt) {
    auto scene = MainMenuScene::createScene();
    Director::getInstance()->replaceScene(TransitionFade::create(TRANSITION_TIME, scene));
};
```

La classe SplashScene :

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

    virtual bool init();
```

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

    //creating a function go to main menu that will be called after an amoumt of time
private:
    void GoToMainMenuScene(float dt);
};
```

La méthode GoToMainMenuScene appartient à la classe SplashScene.

Elle prend un argument de type float nommé dt.

Elle crée une nouvelle instance de la classe MainMenuScene et la renvoie sous forme d'objet Scene.

Elle récupère l'instance unique de la classe Director et appelle sa méthode replaceScene en lui passant en argument une transition de fondu enchaîné vers la nouvelle scène Scene. La durée de la transition est définie par la constante TRANSITION_TIME.

En résumé, cette méthode permet de remplacer la scène actuelle par la scène du menu principal, en utilisant une transition de fondu enchaîné.

La deuxième scène :

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

    virtual bool init();



    // implement the "static create()" method manually
    CREATE_FUNC(MainMenuScene);
    //now we declare the function that will be called when the play button is pressed
private:
    void GoToGameScene(cocos2d::Ref* sender);

};
```

```
auto backgroundSprite = Sprite::create("pico menu.png");
    backgroundSprite->setPosition(Point(visibleSize.width / 2 + origin.x, visibleSize.height / 2
+ origin.y));
    this->addChild(backgroundSprite);
    //creating the start button
    auto playItem = MenuItemImage::create("unclicked button.png", "clicked button.png",
CC_CALLBACK_1(
        MainMenuScene::GoToGameScene, this));
    //position the button
    playItem->setPosition(Point(visibleSize.width / 2 + origin.x, visibleSize.height / 3 +
origin.y));

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

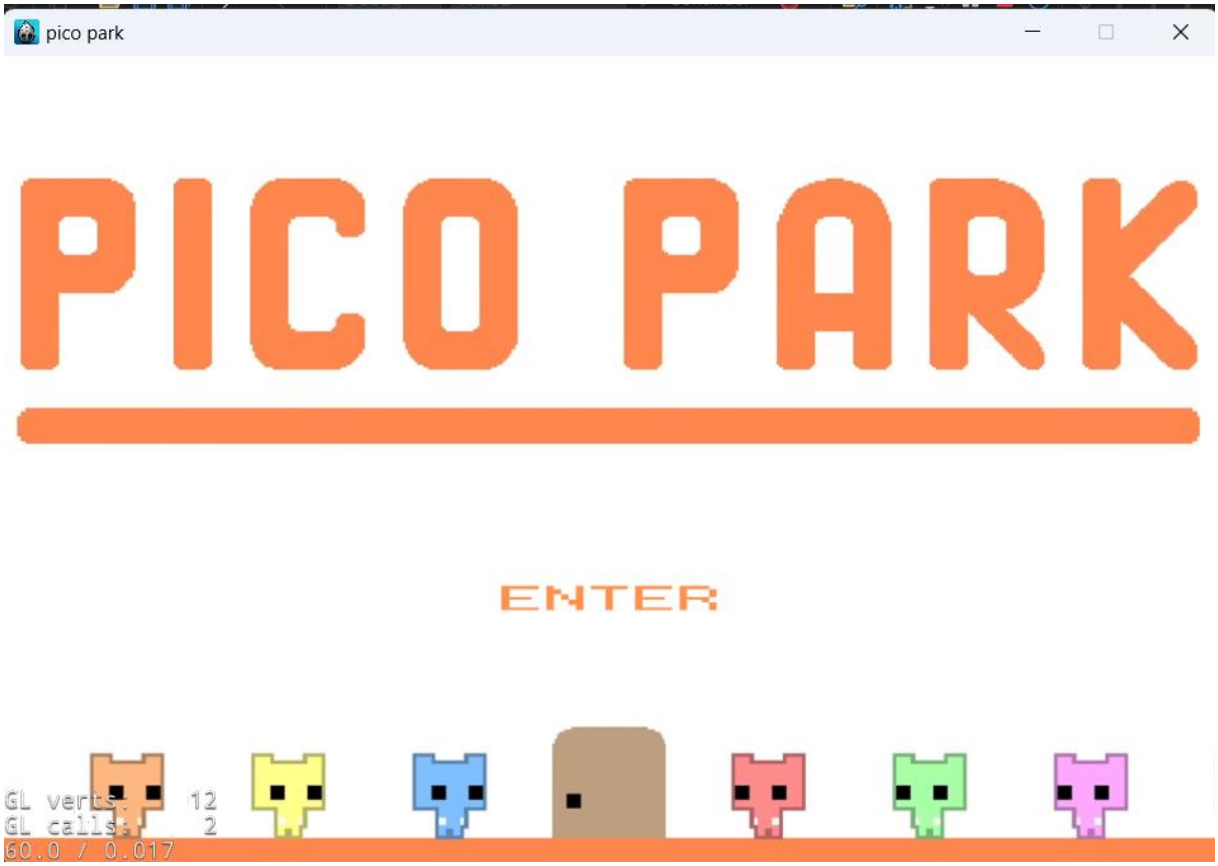
Ce code crée un nouvel objet Sprite à partir de l'image "pico menu.png" et le positionne au centre de l'écran.

Il crée un bouton de jeu en utilisant les images "unclicked button.png" et "clicked button.png" pour l'état normal et l'état sélectionné, respectivement. Le bouton appelle la méthode GoToGameScene lorsqu'il est cliqué.

Il positionne le bouton au tiers inférieur de l'écran.

Il crée un objet Menu contenant le bouton de jeu et le place à l'origine de la scène.

En résumé, ce code crée une scène de menu avec une image de fond et un bouton de jeu qui permet de lancer la partie lorsqu'il est cliqué.



Pour aller à la scène du jeu :

```
void MainMenuScene::GoToGameScene(cocos2d::Ref* sender) {
    auto scene = GameScene::createScene();
    Director::getInstance()->replaceScene(TransitionFade::create(TRANSITION_TIME, scene));

}
```

La scène du jeu :

La classe :

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

    virtual bool init();

    CREATE_FUNC(GameScene);

};
```

La création du fond de la scène :

```
auto backgroundSprite = Sprite::create("BG.png");
    backgroundSprite->setAnchorPoint(Vect::ZERO);
    this->addChild(backgroundSprite);
```

Ce code crée un sprite (une image) à partir de l'image "BG.png" et l'ajoute en tant qu'enfant au noeud actuel dans une scène. Le sprite sera ancré au point zéro (coin supérieur gauche) de la scène et sera affiché en arrière-plan.

Ajout d'obstacles :

```
auto foregroundSprite = Sprite::create("lvl1 1.png");
    foregroundSprite->setPosition(Point(2, 80));

    foregroundSprite->setAnchorPoint(Vect::ZERO);
    foregroundSprite->setScale(0.7); //scale dyal test
    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);
```

Ce code crée un sprite (une image) à partir de l'image "lvl1 1.png" et l'ajoute en tant qu'enfant au noeud actuel dans une scène. Le sprite sera positionné à (2, 80) dans la scène et sera ancré au point zéro (coin supérieur gauche). Le sprite sera redimensionné à 70% de sa taille originale et aura le nom "foregroundSprite".

Ensuite, un corps physique (physics body) est créé pour le sprite en utilisant une boîte (box shape) qui a une taille légèrement inférieure à celle du sprite (divisée par 1,5). Le corps physique n'est pas affecté par la gravité, n'est pas dynamique (c'est-à-dire qu'il ne bouge pas sous l'influence de forces extérieures) et est configuré pour détecter les collisions et être détecté par d'autres corps physiques. Le sprite est finalement associé à ce corps physique et est ajouté à la scène.

La 2eme :

```
auto second = Sprite::create("lvl1 2.png");
    second->setPosition(Point(100, 80));

    second->setAnchorPoint(Vect::ZERO);
    second->setScale(0.7); //scale dyal test
    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(0.0f);
    second->setPhysicsBody(physicsBody3);
    this->addChild(second);
```

La 3eme :

```
auto third = Sprite::create("lvl1 4.png");
```

```
    third->setPosition(Point(250, 0));

    third->setAnchorPoint(Vect::ZERO);
    third->setScale(0.7); //scale dyal test
    third->setName("third");
    auto physicsBody4= PhysicsBody::createBox(third->getContentSize() / 1.5,
PhysicsMaterial(1.0f, 1.0f, 1.0f));
    physicsBody4->setGravityEnable(false);
    physicsBody4->setDynamic(false);
    physicsBody4->setContactTestBitmask(1);
    physicsBody4->setCollisionBitmask(1);
    physicsBody4->setCategoryBitmask(1);
    third->setRotation(0.0f);
    third->setPhysicsBody(physicsBody4);
    this->addChild(third);
```

Creation du joueur:

```
auto player = Sprite::create("girl.png");
    player->setScale(0.15);
    player->setPosition(Vec2(50, 120));

    player->setName("player");
    auto physicsBody1 = PhysicsBody::createBox(player->getContentSize() / 1.5,
PhysicsMaterial(1.0f, 1.0f, 1.0f));
    physicsBody1->setGravityEnable(true);


    physicsBody1->setDynamic(true);
    physicsBody1->setContactTestBitmask(1);
    physicsBody1->setCollisionBitmask(1);
    physicsBody1->setCategoryBitmask(1);

    player->setPhysicsBody(physicsBody1);
    this->addChild(player, 0, 1); //character on top of background
```

Ce code crée un sprite (une image animée) avec l'image "girl.png", le met à l'échelle 0,15 et le place en position (50, 120). Le sprite est donné le nom "player".

Ensuite, le code crée un corps physique en forme de boîte pour le sprite et lui applique une matière physique (avec des propriétés de densité, de rebondissement et de frottement). Le corps physique est activé pour la gravité et défini comme dynamique (pouvant bouger et être affecté par d'autres forces).

Le corps physique est configuré pour être détecté en cas de contact (grâce au masque de test de contact) et pour entrer en collision avec d'autres corps physiques (grâce au masque de collision). Il est également classé dans une catégorie spécifique (grâce au masque de catégorie).

Enfin, le sprite est associé au corps physique et ajouté à la scène avec une profondeur de 0 et un identifiant 1.

Déplacer le joueur avec le clavier :

```
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_UP_ARROW:
        case cocos2d::EventKeyboard::KeyCode::KEY_W:
            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);
```

Ce code crée un écouteur d'événements clavier (EventListenerKeyboard) et lui affecte une fonction à exécuter lorsqu'une touche est enfoncée. La fonction déplace le sprite (que l'on

suppose être l'objet cible de l'événement) dans une direction spécifique en fonction de la touche enfoncée.

Les touches fléchées de gauche et A déplacent le sprite vers la gauche, tandis que les touches fléchées de droite et D déplacent le sprite vers la droite. Les touches fléchées du haut et W font sauter le sprite vers le haut, tandis que les touches fléchées du bas et S font sauter le sprite vers le bas.