

Cycle Licence : LST GI

POO C++

Pr . ELAACHAK LOTFI

Projet réalisé par :

Ayat SIDI EL KHIR

HACHRI Mouncef

PICO PARK



Objectif:

L'objectif principal de ce projet est de maîtriser 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.

Le 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.

Rapport détaillé sur les étapes suivies :

1. Comment installer Cocos2d-x 4.0 sur Windows ?

On a suivi les étapes indiquées ici :

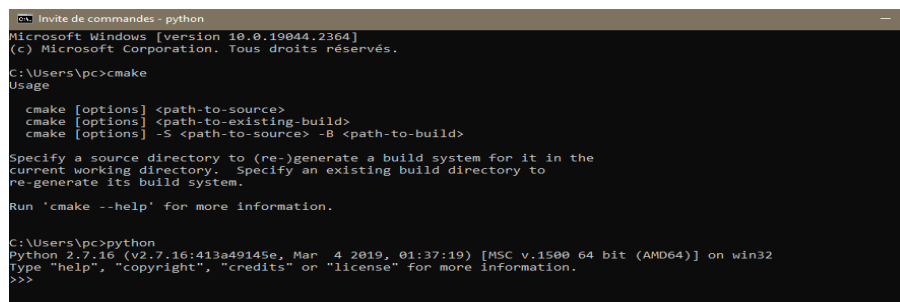
<https://www.youtube.com/watch?v=EPncmb5ujJo>

- Télécharger la dernière version 4.0 de cocos2d-x : <https://www.cocos.com>
- Télécharger python version 2.7.16 sur Windows x86-64 MSI installer :

<https://www.python.org/>

- Télécharger Cmake-3.17.1 –win64-x64msi : <https://cmake.org/>
- Installer visual studio : <https://visualstudio.microsoft.com/do...>

Pour vérifier si l'installation est bien faite, on ouvre la ligne de commandes :



```
Microsoft Windows [version 10.0.19044.2364]
(c) Microsoft Corporation. Tous droits réservés.

C:\Users\pc>cmake
Usage
  cmake [options] <path-to-source>
  cmake [options] <path-to-existing-build>
  cmake [options] -S <path-to-source> -B <path-to-build>

Specify a source directory to (re-)generate a build system for it in the
current working directory. Specify an existing build directory to
re-generate its build system.
Run 'cmake --help' for more information.

C:\Users\pc>python
Python 2.7.16 (v2.7.16:413a49145e, Mar  4 2019, 01:37:19) [MSC v.1500 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
```

2. Connaitre les bases de cocos2d :

Source vidéo :

https://www.youtube.com/watch?v=qXqgSNUf9Cc&list=PLRtjMdoYXLf4od_bOKN3WjAPr7s_nPXzoe

- **How to add a sprite :** A sprite can be created by specifying an image file to use.
`auto mySprite = Sprite::create("mysprite.png");`

- **Positioning :** There is a certain size and position of the sprite:

```
Vec2 point = sprite->getPosition();
float x = point.x;
float y = point.y;
```

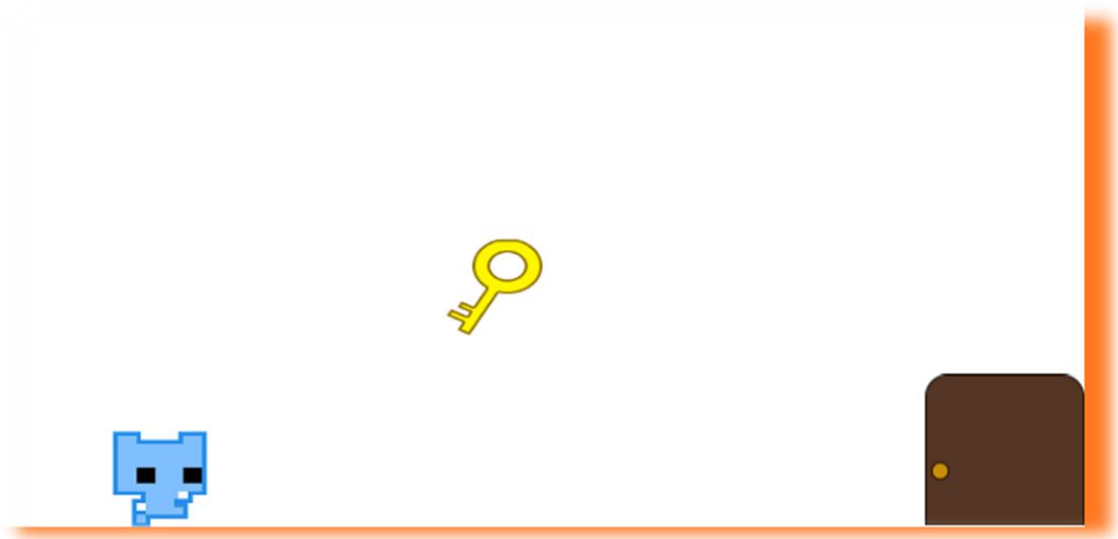
```
Size size = sprite->getContentSize();
float width = size.width;
float height = size.height;
```

- **MoveTo:** Déplacer un objet Node à la position x,y.

- **JumpBy:** Déplace un objet Node simulant un mouvement de saut parabolique en modifiant son attribut de position
- **Setting up a Menu:** Un objet Menu est un type spécial d'objet Node. Vous pouvez créer un objet Menu vide comme espace réservé pour vos éléments de menu
- **Adding a Menu font item:** Classe d'assistance qui crée une classe MenuItemLabel avec un Label.
- **Adding a Menu image item:** accepte les images comme éléments affichés. Les éléments affichés ont 3 états différents : **Normal-selected-disabled**
- **Creating a new scene**
- **Popping a scene**
- **Replace a scene**

3. Concept du jeu :

L'idée: Sauter pour prendre la clé, puis la porte s'ouvre et le joueur peut passer.



Code correspondant :

En utilisant la physique prédéfinie par cocos2d, on a eu la possibilité d'ajouter un sens de gravité au monde, comme suit :

Pour se déplacer on a fait comme suit :

```
Scene* LevelOne::createScene()
{
    auto scene = Scene::createWithPhysics();
    scene->getPhysicsWorld()->setGravity(Vec2(0, 0));
    auto layer = LevelOne::create();
    layer->SetPhysicsWorld(scene->getPhysicsWorld());
    scene->addChild(layer);

    return scene;
}
```

```

auto keyboardListener = EventListenerKeyboard::create();

keyboardListener->onKeyPressed = [playerX, playerY, player, visibleSize](EventKeyboard::KeyCode keyCode, Event* event) mutable
{
    auto jump = JumpBy::create(0.6, Point(0, 0), 80, 1);

    float currentPositionY = player->getPositionY();

    switch (keyCode)
    {
        case EventKeyboard::KeyCode::KEY_UP_ARROW:

            player->runAction(jump);
            break;

        case EventKeyboard::KeyCode::KEY_LEFT_ARROW:

            player->runAction(move(-50, 0));
            break;

        case EventKeyboard::KeyCode::KEY_RIGHT_ARROW:

            player->runAction(move(50, 0));
            break;
    }
};

```

Pour prendre la clé :

```

_eventDispatcher->addEventListenerWithSceneGraphPriority(keyboardListener, this);

auto contactListener = EventListenerPhysicsContact::create();
contactListener->onContactBegin = CC_CALLBACK_1(LevelOne::onContactBegin, this);

this->getEventDispatcher()->addEventListenerWithSceneGraphPriority(contactListener, this);

return true;
}

bool LevelOne::onContactBegin(cocos2d::PhysicsContact& contact) {
    PhysicsBody* a = contact.getShapeA()->getBody();
    PhysicsBody* b = contact.getShapeB()->getBody();

    if (1 == a->getCollisionBitmask() && 2 == b->getCollisionBitmask()) {
        this->SetKeyObtained();
        b->getNode()->removeFromParent();
    }
}

```

Pour passer au niveau suivant :

```

if (1 == a->getCollisionBitmask() && 3 == b->getCollisionBitmask()) {
    if (this->GetKeyObtained()) Director::getInstance()->replaceScene(TransitionFade::create(2, LevelTwo::createScene()));
}

return true;
}

```

Pour revenir au menu principal :

```

void LevelOne::GoBackToMenu(Ref* pSender) {

    auto scene = SplashScreen::createScene();

    Director::getInstance()->replaceScene(TransitionFade::create(2, scene));
}

```

Afin de créer le niveau, on a positionné chaque élément visible sur des positions spécifiques :

```
auto level = Label::createWithTTF("LEVEL 1", "fonts/unispace bd.ttf", 24);
level->setPosition(Vec2(visibleSize.width - 200, visibleSize.height - 52));
level->setColor(Color3B(255, 121, 33));
this->addChild(level, 1);

auto menuButton = MenuItemImage::create("menu_button.png", "menu_button_pressed.png", CC_CALLBACK_1(LevelOne::GoBackToMenu, this));
menuButton->setPosition(Vec2(visibleSize.width - 90, visibleSize.height - 50));
auto menu = Menu::create(menuButton, NULL);
menu->setPosition(Point::ZERO);
this->addChild(menu, 1);

auto edgeBody = PhysicsBody::createEdgeBox(visibleSize, PHYSICSBODY_MATERIAL_DEFAULT, 30);
auto edgeNode = Node::create();
edgeNode->setPosition(Point(visibleSize.width / 2, visibleSize.height / 2));
edgeNode->setPhysicsBody(edgeBody);
this->addChild(edgeNode);

auto ground = DrawNode::create();
ground->drawSolidRect(origin, Size(visibleSize.width, 30), Color4F(1, 0.4745, 0.1294, 1));
ground->setPosition(Point(0, 0));
this->addChild(ground, 1);

auto beginWall = DrawNode::create();
beginWall->drawSolidRect(origin, Size(30, visibleSize.height), Color4F(1, 0.4745, 0.1294, 1));
beginWall->setPosition(Vec2(0, 0));
this->addChild(beginWall, 1);

auto endWall = DrawNode::create();
endWall->drawSolidRect(origin, Size(30, visibleSize.height), Color4F(1, 0.4745, 0.1294, 1));
endWall->setPosition(Vec2(visibleSize.width - 30, 0));
this->addChild(endWall, 1);
```

```
auto beginWall = DrawNode::create();
beginWall->drawSolidRect(origin, Size(30, visibleSize.height), Color4F(1, 0.4745, 0.1294, 1));
beginWall->setPosition(Vec2(0, 0));
this->addChild(beginWall, 1);

auto endWall = DrawNode::create();
endWall->drawSolidRect(origin, Size(30, visibleSize.height), Color4F(1, 0.4745, 0.1294, 1));
endWall->setPosition(Vec2(visibleSize.width - 30, 0));
this->addChild(endWall, 1);

auto door = Sprite::create("door.png");
door->setPosition(Vec2(visibleSize.width - 80, 78));

auto doorBody = PhysicsBody::createBox(door->getContentSize(), PhysicsMaterial(1, 0, 1));
doorBody->setDynamic(false);
doorBody->setCollisionBitmask(3);
doorBody->setContactTestBitmask(true);
door->setPhysicsBody(doorBody);

this->addChild(door, 1);

auto key = Sprite::create("key.png");
key->setContentSize(cocos2d::Size(60, 60));
key->setPosition(Vec2(visibleSize.width - 400, 180));

auto keyBody = PhysicsBody::createBox(key->getContentSize(), PhysicsMaterial(0, 0, 1));
keyBody->setDynamic(false);
keyBody->setCollisionBitmask(2);
keyBody->setContactTestBitmask(true);
key->setPhysicsBody(keyBody);
```

```
this->addChild(key, 1);

float playerX = 60;
float playerY = 60;

auto player = Sprite::create("player.png");
player->setContentSize(cocos2d::Size(60, 60));
player->setPosition(Point(playerX, playerY));
```

Pour les autres niveaux on a utilisé la même logique, malgré la possibilité d'avoir inclus l'intégralité du code dans un seul fichier .cpp,

chose qui aurait été plus efficace.

