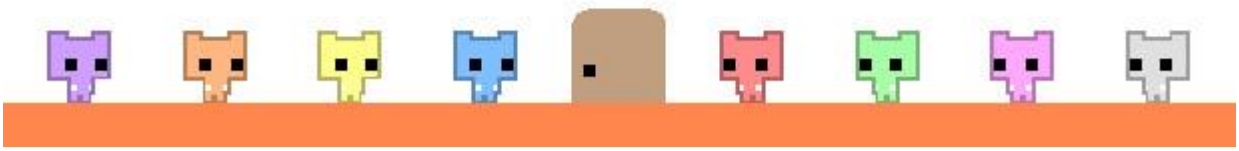
Royaume du Maroc
Université Abdelmalek Essaâdi
Faculté des Sciences et Techniques - Tanger

Licence Génie Informatique

# Jeu vidéo 2D avec Cocos2D-X




Réalisé PAR :

- Anas Bouteffah Touiki
- Zakariae Chelle

Année Universitaire : 2022/2023

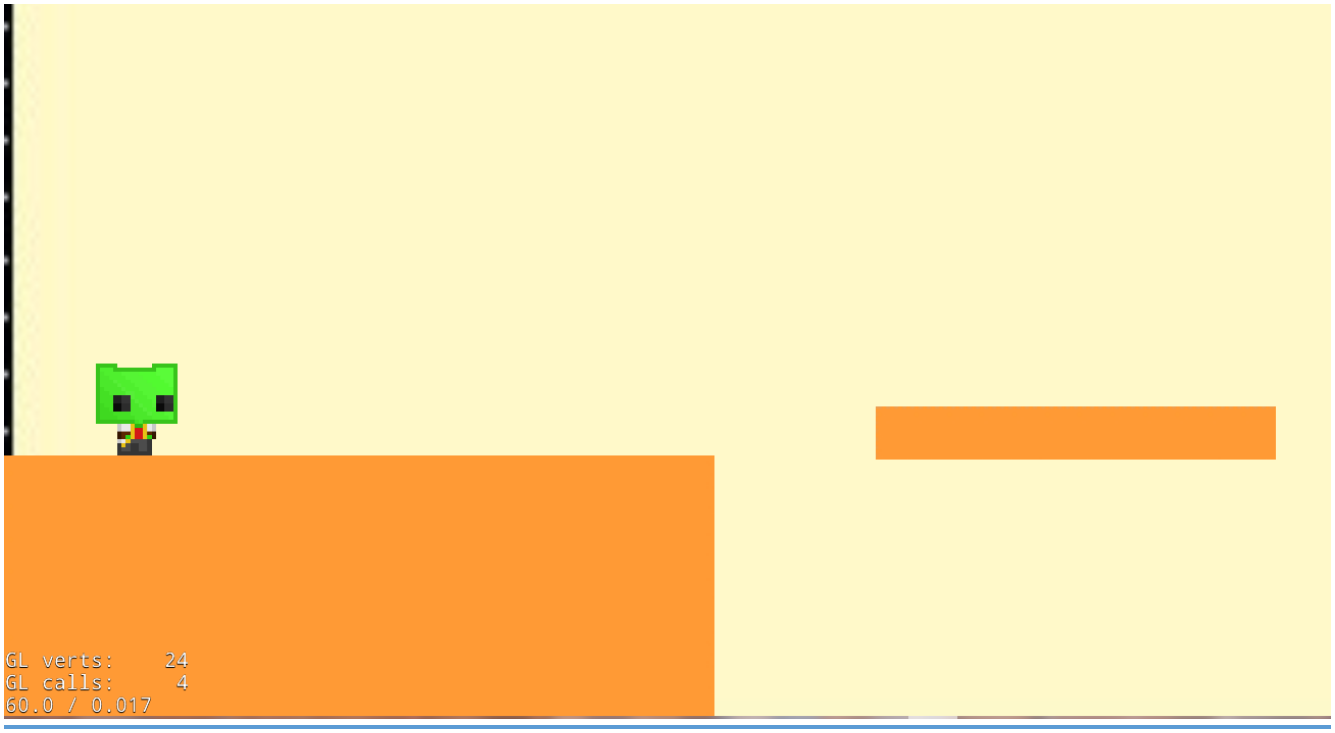
# Introduction :

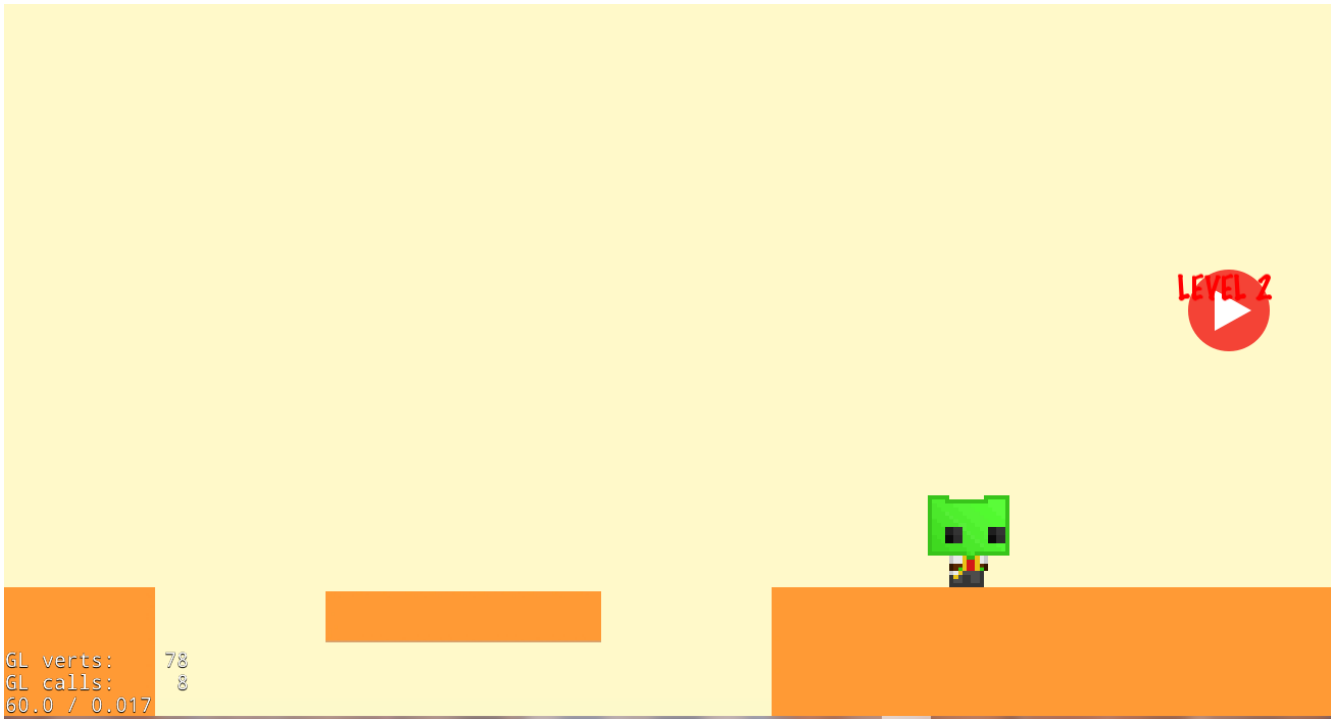
Ce rapport décrit le processus impliqué dans la création d'un jeu 2D Piko Park avec Cocos2D-X. Ce jeu est destiné aux appareils PC. Ce chapitre traite de la vue d'ensemble du jeu, y compris le synopsis, la description, puis se concentre sur la conception du jeu, décrivant comment le jeu est implémenté

## Vue d'ensemble du jeu :

L'objectif principal du joueur est d'essayer de ne pas tomber et de passer quelques obstacles sur son chemin.

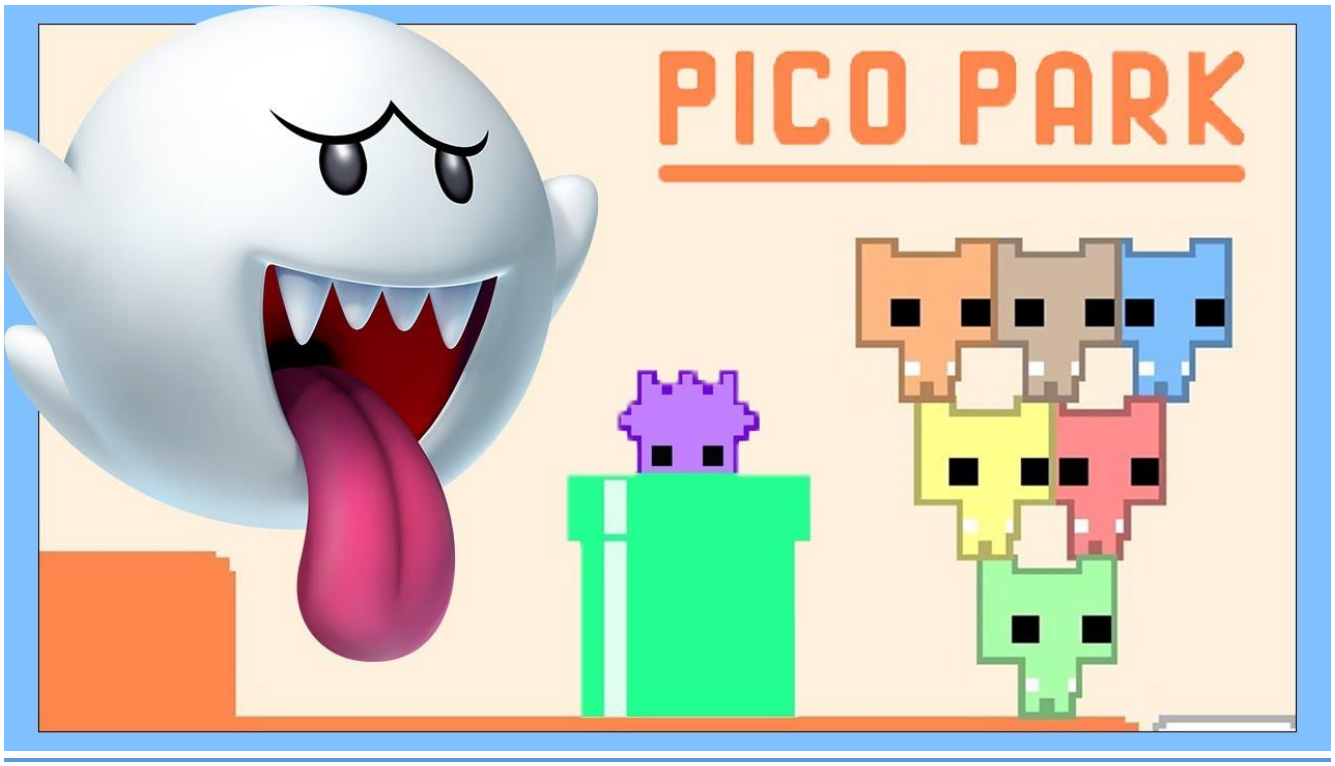
## Conception de niveau :



## Personnage principal :

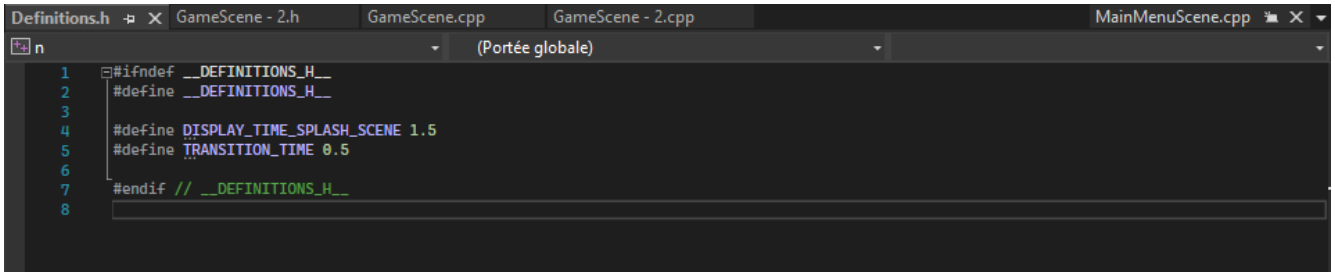
## Menu Scène :



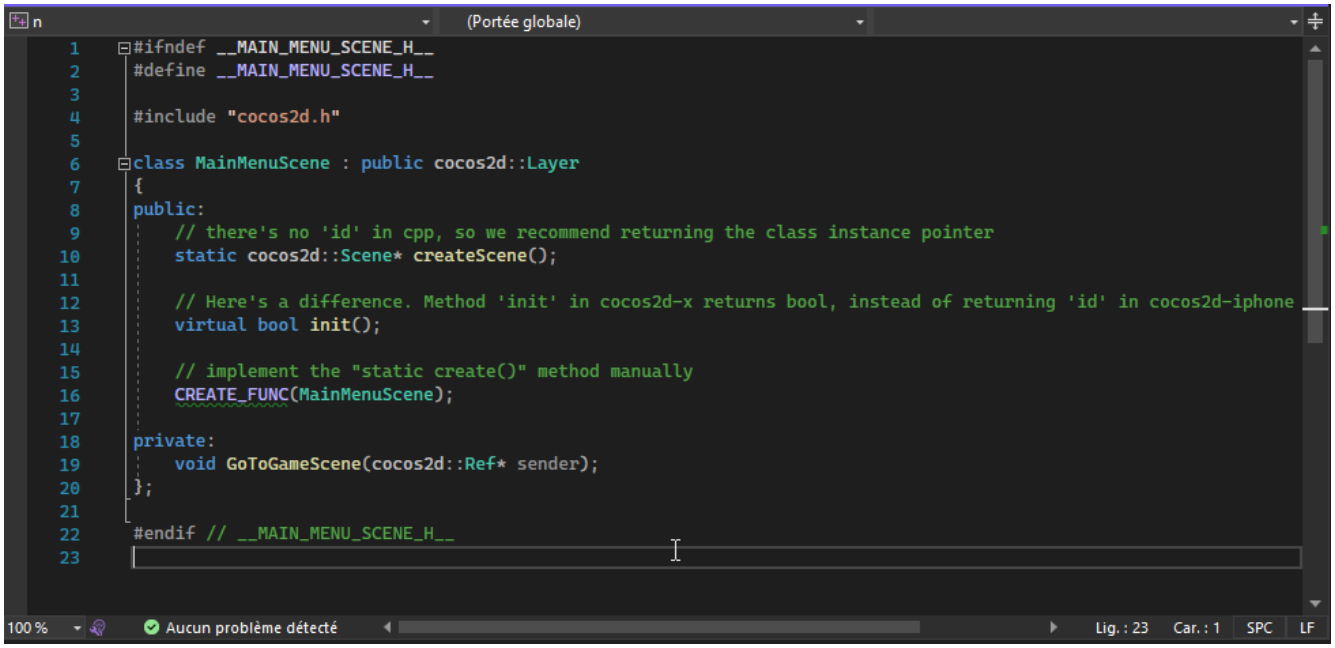

## 1-Main Menu :


## Definitions.h

ici nous avons défini les variables globales




## Code source : MainMenu.h




## MainMenu.cpp

Nous avons utiliser La méthode

### 1-initialiser :

Toutes les sous-classes Ref sur cocos2d-x ont une méthode statique pour créer, initialiser et renvoyer une instance autoreleased. Bien que ces méthodes soient très pratiques pour créer de nouvelles instances, elles répètent généralement le même morceau de code sur chaque sous-classe Ref.

### 2-Sprites :

Cocos2d-x fournit Sprite ce sont des images pour presque tout ce dont votre jeu a besoin.
Pour créer un Sprite :

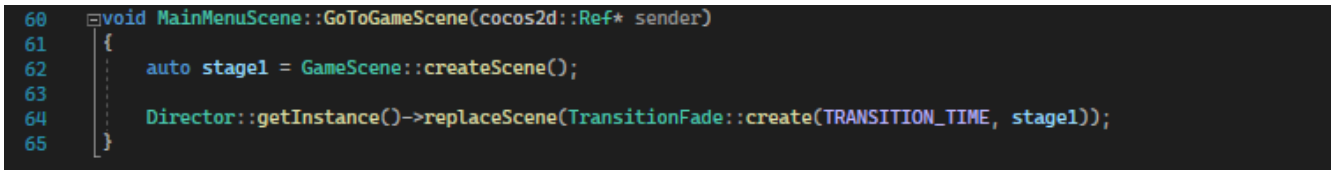
```
auto mySprite = Sprite::create("mysprite.png");
```

## 3- Remplacer les scènes:

```
Director::getInstance()->replaceScene(myScene);
```

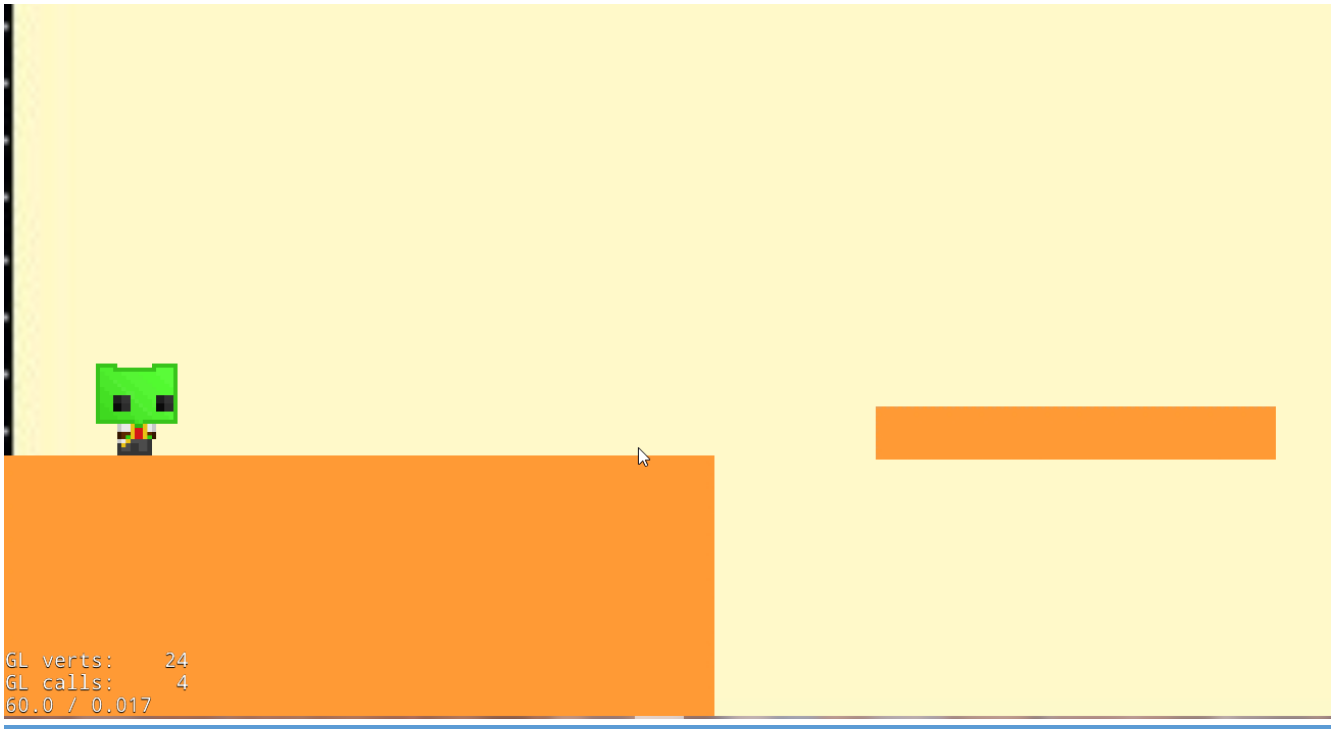
# Voici le coude source :

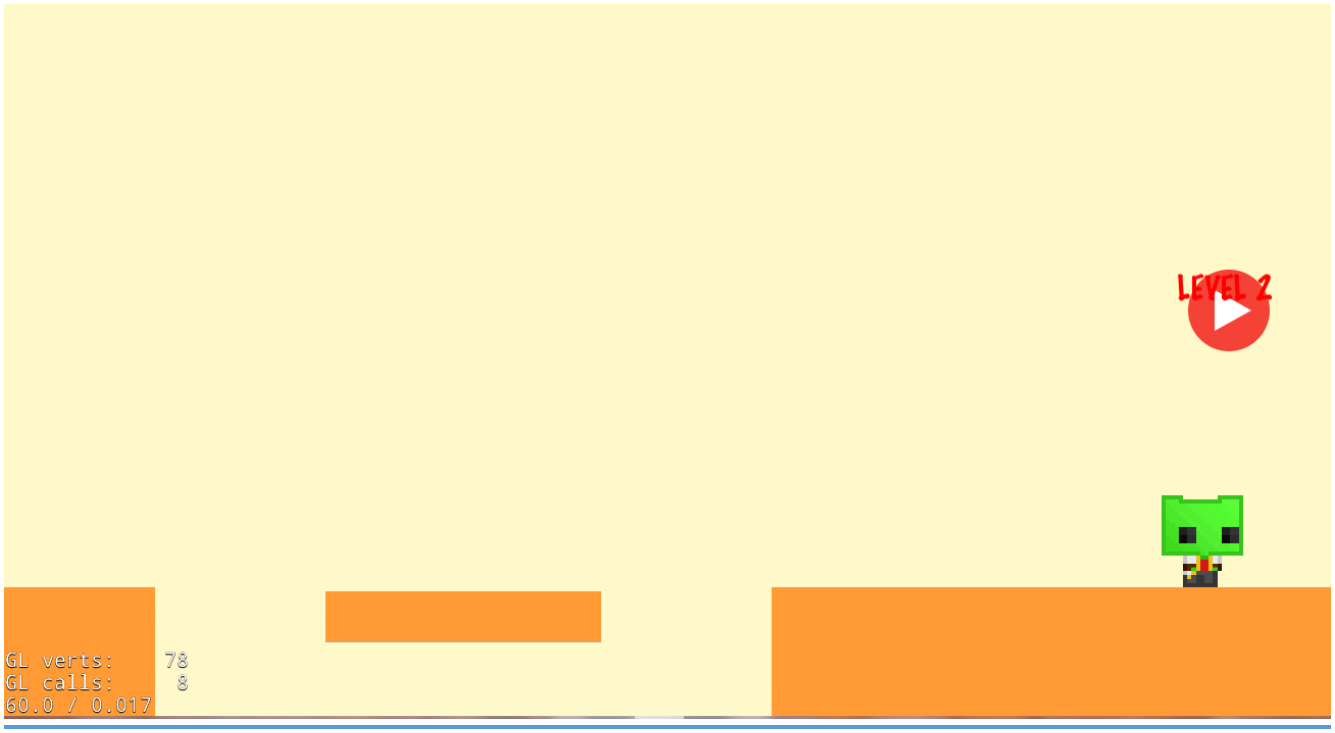
```
// on "init" you need to initialize your instance
bool MainMenuScene::init()
{
    //////////////////////////////
    // 1. super init first
    if ( !Layer::init() )
    {
        return false;
    }

    Size visibleSize = Director::getInstance()->getVisibleSize();
    Vec2 origin = Director::getInstance()->getVisibleOrigin();

    auto backgroundSprite = Sprite::create("cool-background (1).png");
    backgroundSprite->setPosition(Point(visibleSize.width / 2 + origin.x, visibleSize.height / 2 + origin.y));

    this->addChild(backgroundSprite);

    auto titleSprite = Sprite::create("LOGO.png");
    titleSprite->setPosition(Point(visibleSize.width / 2 + origin.x, visibleSize.height - titleSprite->getContentSize().height));

    this->addChild(titleSprite);

    auto playItem = MenuItemImage::create("p.png", "Play Button Clicked.png", CC_CALLBACK_1(MainMenuScene::GoToGameScene, this));
    playItem->setPosition(Point(visibleSize.width / 2 + origin.x, visibleSize.height / 2 + origin.y));

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

    this->addChild(menu);
```



```
void MainMenuScene::GoToGameScene(cocos2d::Ref* sender)
{
    auto stage1 = GameScene::createScene();

    Director::getInstance()->replaceScene(TransitionFade::create(TRANSITION_TIME, stage1));
}
```

## 2-Niveau 1 :




Création de la arrière-plan et la positione au début de Scène par la fonction « setPosition » et donner son taille

```
auto background = Sprite::create("background.jpg");
background->setAnchorPoint(Vec2::ZERO);
background->setPosition(0, 0);
background->setContentSize(CCSize(5000, 2000));
```

Création de jouer avec la taille, position et mentionner comme un solide avec la fonction « Bitmask » .

```
auto player = Sprite::create("player.png");
player->setPosition((Vec2(visibleSize.width * 0.1, visibleSize.height * 0.5)));
player->setContentSize(CCSize(80, 90));
player->setName("player");
auto spriteBody = PhysicsBody::createBox(player->getContentSize() / 1, PhysicsMaterial(0.2f, 0.2f, 0.2f));
spriteBody->setGravityEnable(true);
spriteBody->setDynamic(true);
spriteBody->setContactTestBitmask(1);
spriteBody->setCollisionBitmask(1);
spriteBody->setCategoryBitmask(1);
player->setPhysicsBody(spriteBody);
```

Création de premier terre de début de stage 1 avec un obstacle qui est la première terre et après on a huit terres pour atteindre la fin de stage 1 en rencontre le bouton pour passer au stage 2

```
ground1->setPosition(0, 0);
ground1->setScale(2);
ground1->setName("ground1");
auto spriteBody0 = PhysicsBody::createBox(ground1->getContentSize() / 1, PhysicsMaterial(1.0f, 1.0f, 1.0f));
spriteBody0->setGravityEnable(false);
spriteBody0->setDynamic(false);
spriteBody0->setContactTestBitmask(1);
spriteBody0->setCollisionBitmask(1);
spriteBody0->setCategoryBitmask(1);
ground1->setRotation(0.0f);
ground1->setPhysicsBody(spriteBody0);
//obstacle 1
auto obs1 = Sprite::create("box2.png");
obs1->setPosition(1050, 280);
obs1->setScale(2);
auto spriteBody1 = PhysicsBody::createBox(obs1->getContentSize() / 1, PhysicsMaterial(1.0f, 1.0f, 1.0f));
spriteBody1->setGravityEnable(false);
spriteBody1->setDynamic(false);
spriteBody1->setContactTestBitmask(1);
spriteBody1->setCollisionBitmask(1);
spriteBody1->setCategoryBitmask(1);
obs1->setRotation(0.0f);
obs1->setPhysicsBody(spriteBody1);
```

Keyboard mouvements:

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


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

    Vec2 loc = player->getCurrentTarget()->getPosition();
    auto jump = JumpTo::create(1, Point(loc.x + 70, loc.y), 70, 1);

    switch (keyCode) {
    case EventKeyboard::KeyCode::KEY_LEFT_ARROW:
    case EventKeyboard::KeyCode::KEY_A:
        player->getCurrentTarget()->setPosition(loc.x - 50, loc.y);
        break;
    case EventKeyboard::KeyCode::KEY_RIGHT_ARROW:
    case EventKeyboard::KeyCode::KEY_D:
        player->getCurrentTarget()->setPosition(loc.x + 50, loc.y);
        break;
    case EventKeyboard::KeyCode::KEY_SPACE:
    case EventKeyboard::KeyCode::KEY_W:
        player->getCurrentTarget()->runAction(jump);
        break;
    }
};

this->_eventDispatcher->addEventListenerWithSceneGraphPriority(eventListener, player);
```

Si on clique sur les buttons on a des actions différentes :

« A ou gauche flèche » mouvement à gauche

« A ou droit flèche » mouvement à gauche

« W ou espace » mouvement à gauche

En fin on donne au fonction la cible et l'évent qui est le bouton

Camera:

```
CCRect frame = CCRect(0, 0, 4000, 720);
auto followPlayer = Follow::create(player, frame);
followPlayer->setTarget(player);
this->runAction(followPlayer);
```

CCRECt frame permet de donner la taille de la scène

Passage de stage :

Bouton pour passer du niveau a un autre quand attend

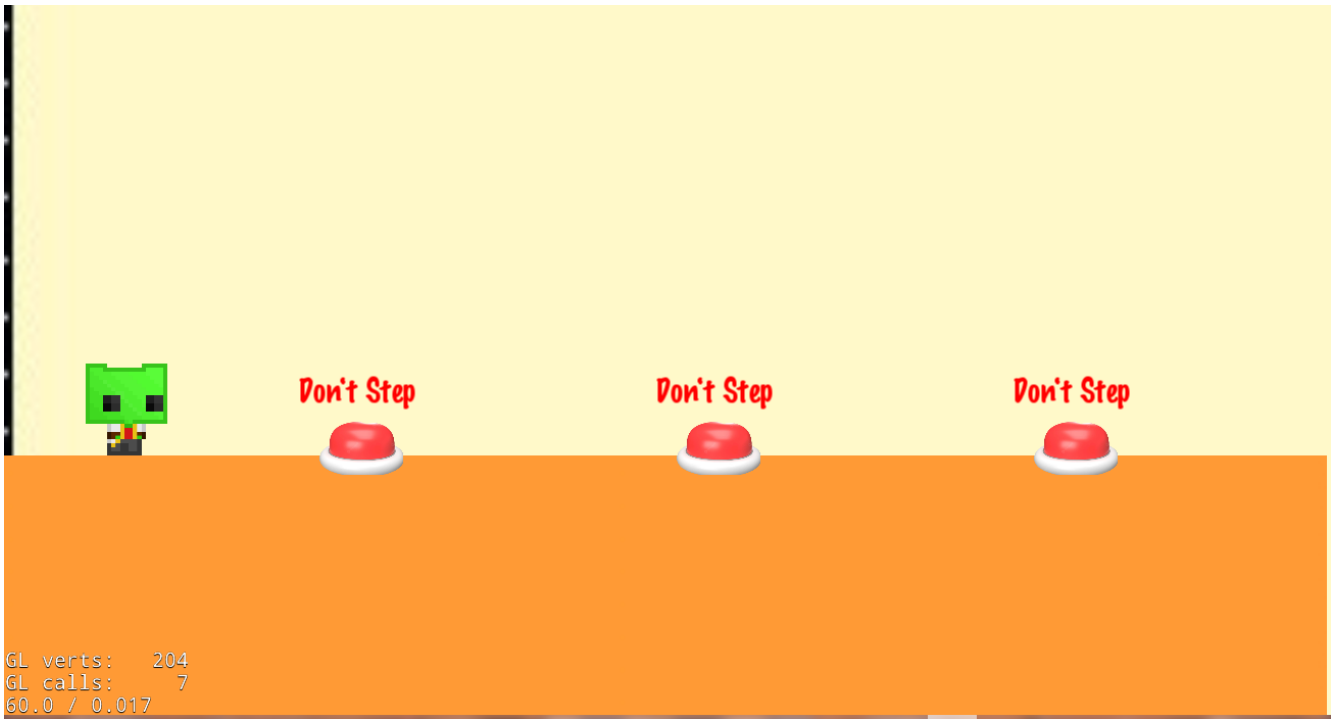
La dernier Terre

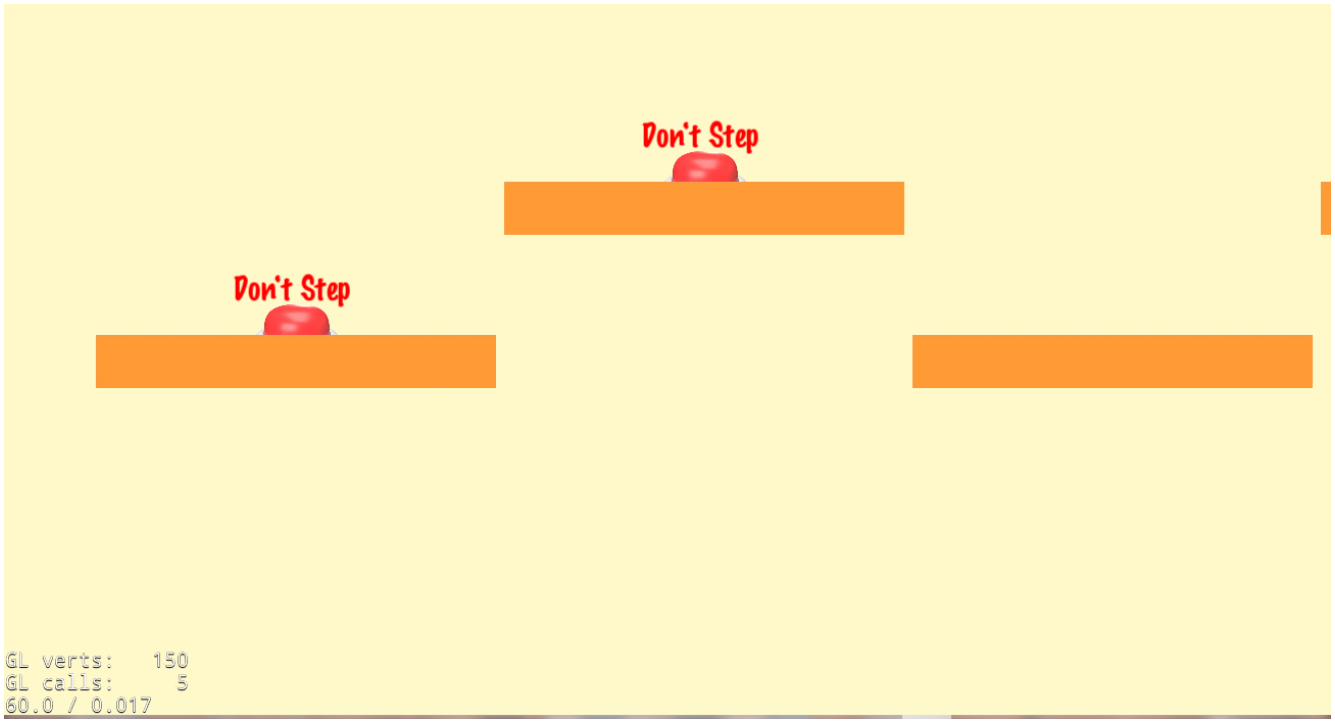
```
ui::Button* btn1 = ui::Button::create("p.png");
btn1->setPosition(Vec2(3900,420));
btn1->addTouchEventListener(CC_CALLBACK_1(GameScene::GoToGameScene2, this));
this->addChild(btn1, 3);
```

## 3-Niveau 2 :

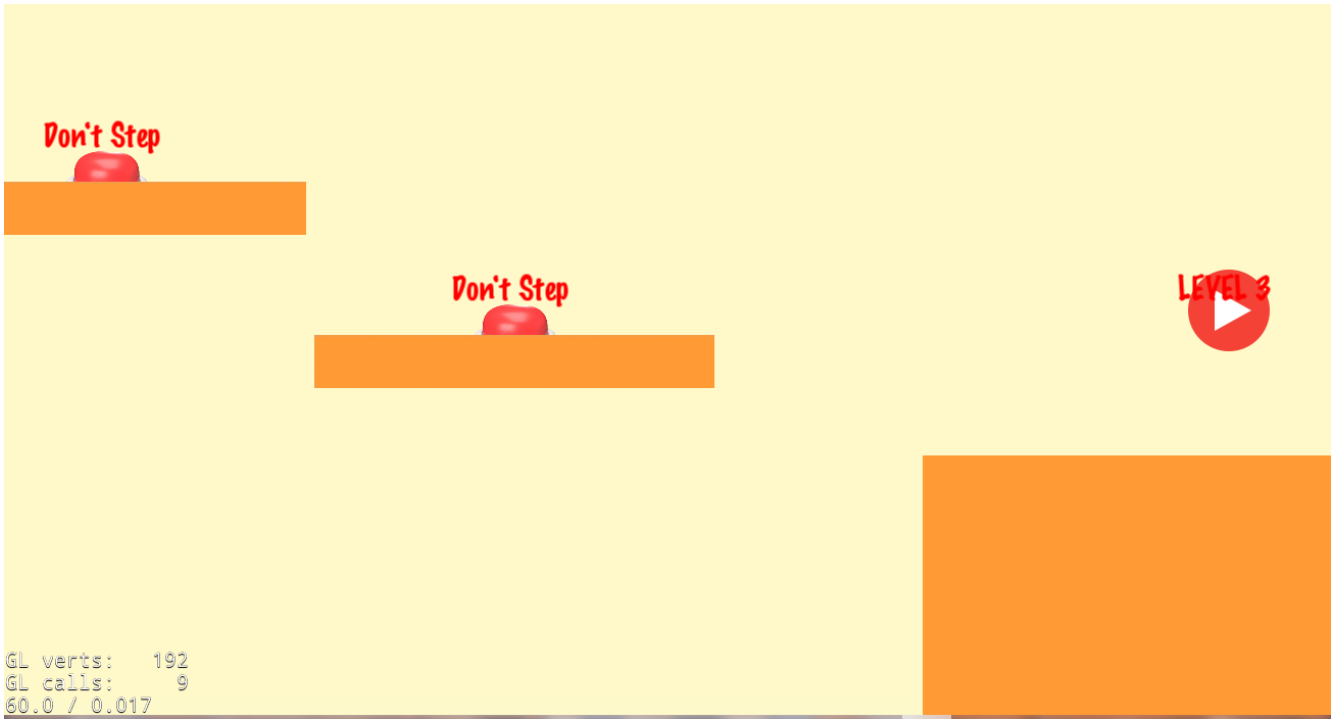
L'objectif principal du joueur dans ce niveau est d’évitez de franchir l'obstacle pour éviter la perte :

Début de Level :






La fin de level:




Voici le code source :

```
n                                   GameScene2                          init()
25  // on "init" you need to initialize your instance
26  bool GameScene2::init()
27  {
28      //////////////////////////
29      // 1. super init first
30      if ( !Layer::init() )
31      {
32          return false;
33      }
34
35      Size visibleSize = Director::getInstance()->getVisibleSize();
36      Vec2 origin = Director::getInstance()->getVisibleOrigin();
37
38      auto background = Sprite::create("background.jpg");
39      background->setAnchorPoint(Vec2::ZERO);
40      background->setPosition(0, 0);
41      background->setContentSize(CCSize(5000, 2000));
42
43
44      auto player = Sprite::create("player.png");
45      player->setPosition((Vec2(visibleSize.width * 0.1, visibleSize.height * 0.5)));
46      player->setContentSize(CCSize(80, 90));
47      player->setName("player");
48      auto spriteBody = PhysicsBody::createBox(player->getContentSize() / 1, PhysicsMaterial(0.2f, 0.2f, 0.2f));
49      spriteBody->setGravityEnable(true);
50      spriteBody->setDynamic(true);
51      spriteBody->setContactTestBitmask(1);
52      spriteBody->setCollisionBitmask(1);
53      spriteBody->setCategoryBitmask(1);
54      player->setPhysicsBody(spriteBody);
```

```
59      auto ground1 = Sprite::create("box1.png");
60      ground1->setAnchorPoint(Vec2::ZERO);
61      ground1->setPosition(0, 0);
62      ground1->setPosition(0, 0);
63      ground1->setScale(2);
64      ground1->setName("ground1");
65
66
67      auto spriteBody0 = PhysicsBody::createBox(ground1->getContentSize() / 1, PhysicsMaterial(1.0f, 1.0f, 1.0f));
68      spriteBody0->setGravityEnable(false);
69      spriteBody0->setDynamic(false);
70      spriteBody0->setContactTestBitmask(1);
71      spriteBody0->setCollisionBitmask(1);
72      spriteBody0->setCategoryBitmask(1);
73      ground1->setRotation(0.0f);
74      ground1->setPhysicsBody(spriteBody0);
75      //obstacle 1
76      auto ground2 = Sprite::create("box1.png");
77      ground2->setAnchorPoint(Vec2::ZERO);
78      ground2->setPosition(600, 0);
79      ground2->setPosition(600, 0);
80      ground2->setScale(2);
81      ground2->setName("ground2");
82
```

```
82
83      auto Push = Label::createWithTTF("Don't Step ", "fonts/Marker Felt.ttf", 30);
84      Push->setTextColor(Color4B::RED);
85      Push->setPosition(Vec2(350, 320));
86      auto panic = Sprite::create("redd.png");
87      panic->setScale(0.3);
88      panic->setPosition(Vec2(350, 265));
89
90      auto Push1 = Label::createWithTTF("Don't Step ", "fonts/Marker Felt.ttf", 30);
91      Push1->setPosition(Vec2(700, 320));
92      Push1->setTextColor(Color4B::RED);
93      auto panic2 = Sprite::create("redd.png");
94      panic2->setScale(0.3);
95      panic2->setPosition(Vec2(700, 265));
96
97      auto Push2 = Label::createWithTTF("Don't Step ", "fonts/Marker Felt.ttf", 30);
98      Push2->setPosition(Vec2(1050, 320));
99      Push2->setTextColor(Color4B::RED);
100     auto panic3 = Sprite::create("redd.png");
101     panic3->setPosition(Vec2(1050, 265));
102     panic3->setScale(0.3);
103
104
105
106
107
```

```cpp
//obsatcle7
auto ground3 = Sprite::create("box1.png");
ground3->setAnchorPoint(Vec2::ZERO);
ground3->setPosition(3600, 0);
ground3->setPosition(3600, 0);
ground3->setScale(2);
ground3->setName("ground3");

auto spriteBody7 = PhysicsBody::createBox(ground3->getContentSize() / 1, PhysicsMaterial(1.0f, 1.0f, 1.0f));
spriteBody7->setGravityEnable(false);
spriteBody7->setDynamic(false);
spriteBody7->setContactTestBitmask(1);
spriteBody7->setCollisionBitmask(1);
spriteBody7->setCategoryBitmask(1);
ground3->setRotation(0.0f);
ground3->setPhysicsBody(spriteBody7);
```

# Keyboard Events :

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


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

    Vec2 loc = player->getCurrentTarget()->getPosition();
    auto jump = JumpTo::create(1, Point(loc.x + 70, loc.y), 70, 1);

    switch (keyCode) {
    case EventKeyboard::KeyCode::KEY_LEFT_ARROW:
    case EventKeyboard::KeyCode::KEY_A:
        player->getCurrentTarget()->setPosition(loc.x - 50, loc.y);
        break;
    case EventKeyboard::KeyCode::KEY_RIGHT_ARROW:
    case EventKeyboard::KeyCode::KEY_D:
        player->getCurrentTarget()->setPosition(loc.x + 50, loc.y);
        break;
    case EventKeyboard::KeyCode::KEY_SPACE:
    case EventKeyboard::KeyCode::KEY_W:
        player->getCurrentTarget()->runAction(jump);
        break;
    }
};

this->_eventDispatcher->addEventListenerWithSceneGraphPriority(eventListener, player);
```

# Adding scenes :

```cpp
followPlayer->setTarget(player);
this->runAction(followPlayer);
this->addChild(ground1, 2);
this->addChild(ground2, 2);
this->addChild(ground3, 2);

this->addChild(player, 2);
this->addChild(background,1);

this->addChild(Push, 2);
this->addChild(Push1, 2);
this->addChild(Push2, 2);
this->addChild(Push3, 2);
this->addChild(Push4, 2);
this->addChild(Push5, 2);
this->addChild(Push6, 2);
this->addChild(panic, 2);
this->addChild(panic2, 2);
this->addChild(panic3, 2);
this->addChild(panic4, 2);
this->addChild(panic5, 2);
this->addChild(panic6, 2);
this->addChild(panic7, 2);
this->addChild(obs2, 2);
this->addChild(obs3, 2);
this->addChild(obs4, 2);
this->addChild(obs5, 2);
this->addChild(obs6, 2);
```

## 4-Niveau 3 :

Création de arrière-plan de stage 3 et le jouer et la fonction de camera qui permet de suivre le jouer

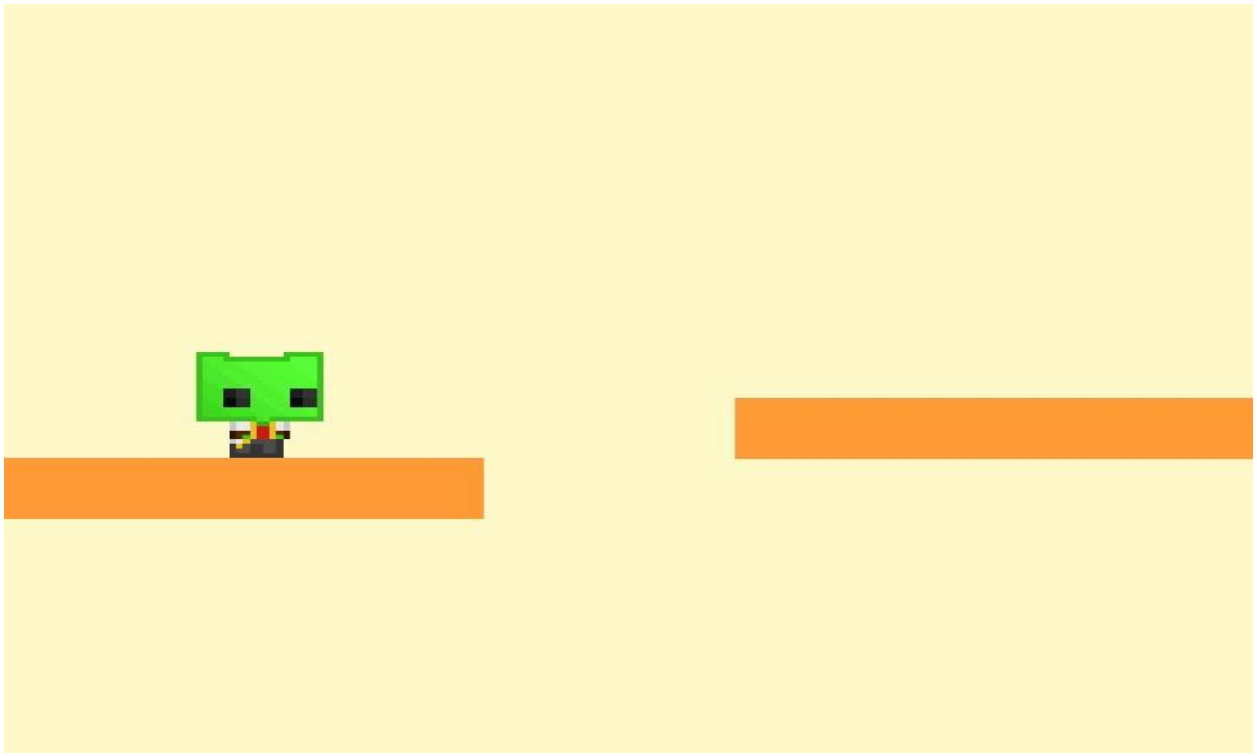
```
auto background = Sprite::create("background.jpg");
auto player = Sprite::create("player.png");
player->setPosition((Vec2(visibleSize.width * 0.1, visibleSize.height * 0.5)));
player->setContentSize(CCSize(80, 90));
player->setName("player");
auto spriteBody = PhysicsBody::createBox(player->getContentSize() / 1, PhysicsMaterial(0.2f, 0.2f, 0.2f));
spriteBody->setGravityEnable(true);
spriteBody->setDynamic(true);
spriteBody->setContactTestBitmask(1);
spriteBody->setCollisionBitmask(1);
spriteBody->setCategoryBitmask(1);
player->setPhysicsBody(spriteBody);
this->addChild(player, 3);
background->setAnchorPoint(Vec2::ZERO);
background->setPosition(0, 0);
background->setContentSize(CCSize(5000, 2000));
this->addChild(background, 2);
CCRect frame = CCRect(0, 0, 5000, 720);
auto followplayer = Follow::create(player, frame);
followplayer->setTarget(player);
this->runAction(followplayer);
```

Création les terres de plan on a 9 obstacle dans tous la map :

```
auto frstground = Sprite::create("box1.png");
frstground->setPosition(0, 0);
frstground->setScale(2);
auto spriteBody0 = PhysicsBody::createBox(frstground->getContentSize() / 1, PhysicsMaterial(1.0f, 1.0f, 1.0f));
spriteBody0->setGravityEnable(false);
spriteBody0->setDynamic(false);
spriteBody0->setContactTestBitmask(1);
spriteBody0->setCollisionBitmask(1);
spriteBody0->setCategoryBitmask(1);
frstground->setRotation(0.0f);
frstground->setPhysicsBody(spriteBody0);
this->addChild(frstground, 2);
//o1
auto o1 = Sprite::create("small.png");
o1->setPosition(600, 180);
o1->setScale(0.7);
auto spriteBody1 = PhysicsBody::createBox(o1->getContentSize() / 1, PhysicsMaterial(1.0f, 1.0f, 1.0f));
spriteBody1->setGravityEnable(false);
spriteBody1->setDynamic(false);
spriteBody1->setContactTestBitmask(1);
spriteBody1->setCollisionBitmask(1);
spriteBody1->setCategoryBitmask(1);
o1->setRotation(0.0f);
o1->setPhysicsBody(spriteBody1);
this->addChild(o1, 2);
```

Deux boxes qui remplacer du bas vers le haut

```
//o9
auto o9 = Sprite::create("box2.png");
o9->setScale(2);
auto spriteBody9 = PhysicsBody::createBox(o9->getContentSize() / 1, PhysicsMaterial(1.0f, 1.0f, 1.0f));
spriteBody9->setGravityEnable(false);
spriteBody9->setDynamic(false);
spriteBody9->setContactTestBitmask(1);
spriteBody9->setCollisionBitmask(1);
spriteBody9->setCategoryBitmask(1);
o9->setRotation(0.0f);
o9->setPhysicsBody(spriteBody9);
this->addChild(o9, 2);
o9->setPosition(Vec2(3600, 0));
auto moveAction1 = MoveTo::create(100, Vec2(3600, 800));
auto repeatAction1 = RepeatForever::create(moveAction1);
o9->runAction(repeatAction1);
//11
```



APRES LE PASSAGE DU DEUX MOUVENT BAR IL ATTIEND LA FIN DU MAP