Laboratorio #21: Cocos2D-X

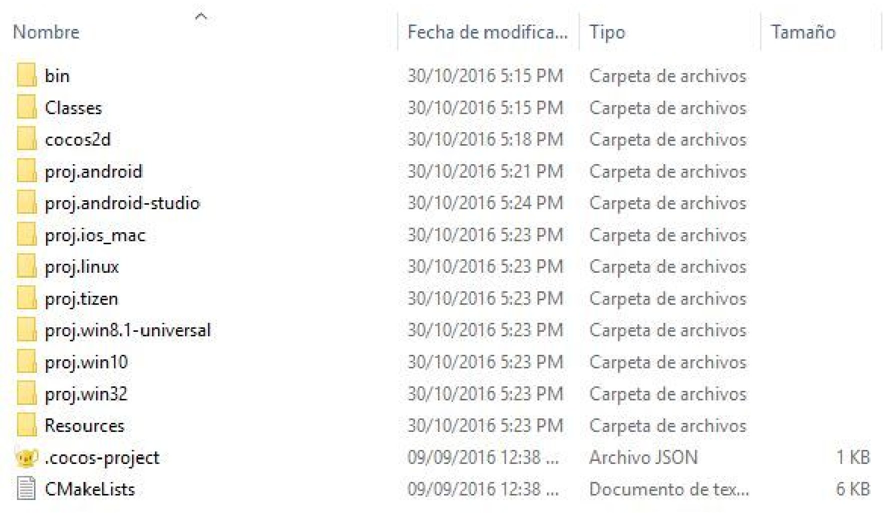
El siguiente laboratorio consiste en crear un pequeño juego de prueba que muestra la manera en la que el *framework* Cocos2D-X maneja recursos (imágenes, fuentes), utiliza su sistema de físicas, implementa un sistema de colisiones y una simple animación con efecto *Parallax.*

Links de interés:

- http://cocos2d-x.org/docs/installation/Android-terminal/index.html
- http://cocos2d-x.org/docs/installation/Android-Studio/index.html
- http://discuss.cocos2d-x.org/t/3-10-and-android-studio/27720/2
- http://www.gamefromscratch.com/page/Cocos2d-x-CPP-Game-Programming-Tutorial-Series.aspx
- http://cocos2d-x.org/docs/editors_and_tools/cocosCLTool/
- https://www.raywenderlich.com/33752/cocos2d-x-tutorial-for-ios-and-android-space-game

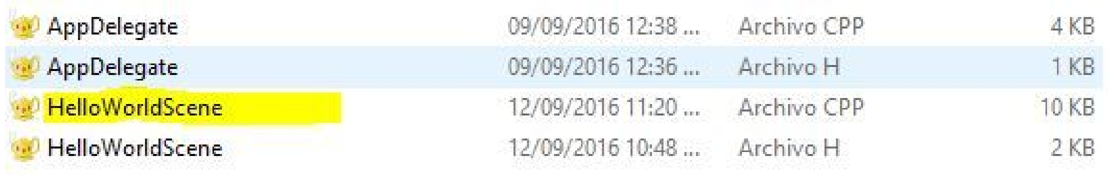
Implementar el código

En la carpeta MyGame, dentro de la carpeta MyCompany que se encuentra en el Escritorio, se encontrara un estructura como la siguiente:



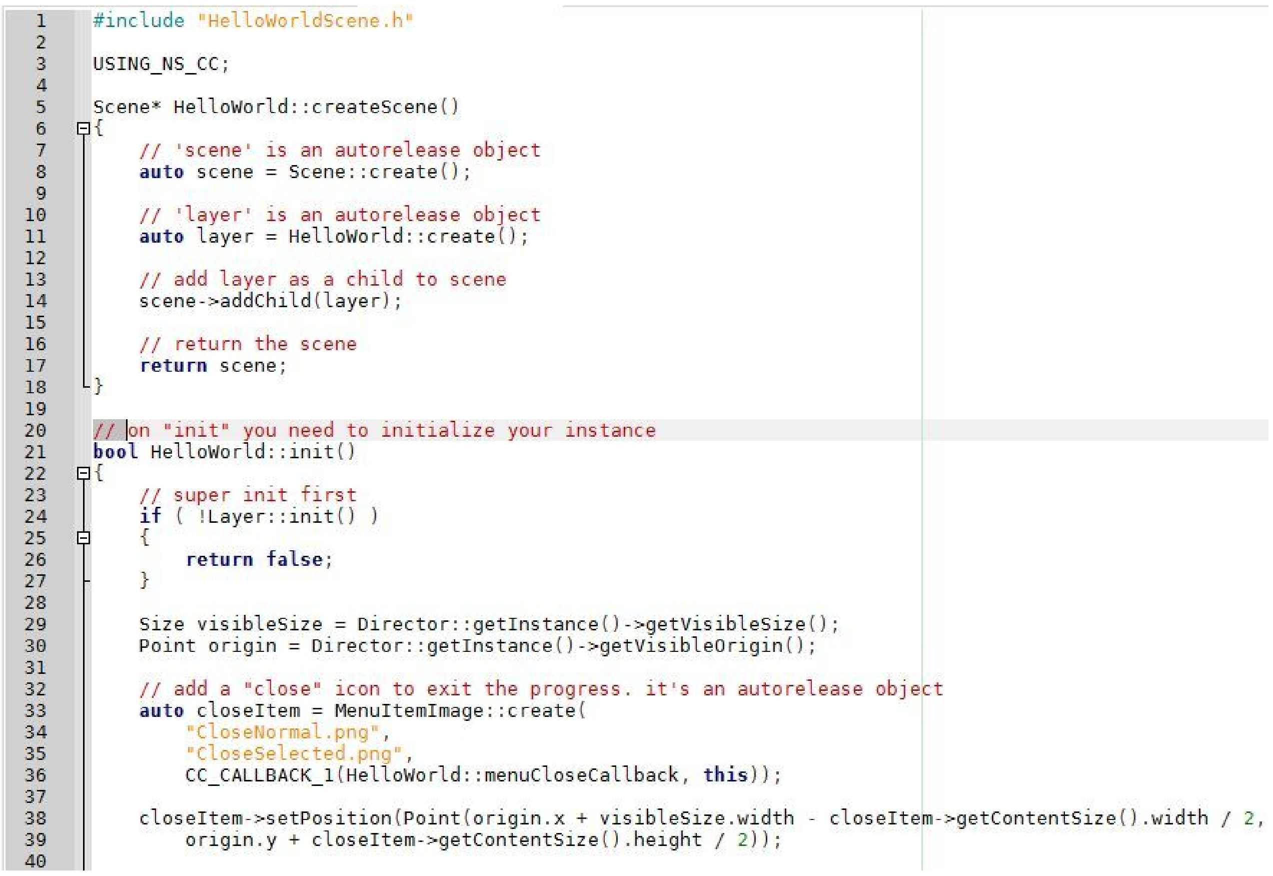
Nombre	Fecha de modifica...	Tipo	Tamaño
bin	30/10/2016 5:15 PM	Carpeta de archivos	
Classes	30/10/2016 5:15 PM	Carpeta de archivos	
cocos2d	30/10/2016 5:18 PM	Carpeta de archivos	
proj.android	30/10/2016 5:21 PM	Carpeta de archivos	
proj.android-studio	30/10/2016 5:24 PM	Carpeta de archivos	
proj.ios_mac	30/10/2016 5:23 PM	Carpeta de archivos	
proj.linux	30/10/2016 5:23 PM	Carpeta de archivos	
proj.tizen	30/10/2016 5:23 PM	Carpeta de archivos	
proj.win8.1-universal	30/10/2016 5:23 PM	Carpeta de archivos	
proj.win10	30/10/2016 5:23 PM	Carpeta de archivos	
proj.win32	30/10/2016 5:23 PM	Carpeta de archivos	
Resources	30/10/2016 5:23 PM	Carpeta de archivos	
.cocos-project	09/09/2016 12:38 ...	Archivo JSON	1 KB
CMakeLists	09/09/2016 12:38 ...	Documento de tex...	6 KB

En la carpeta *Classes,* abrir el archivo *HelloWorld.cpp.*

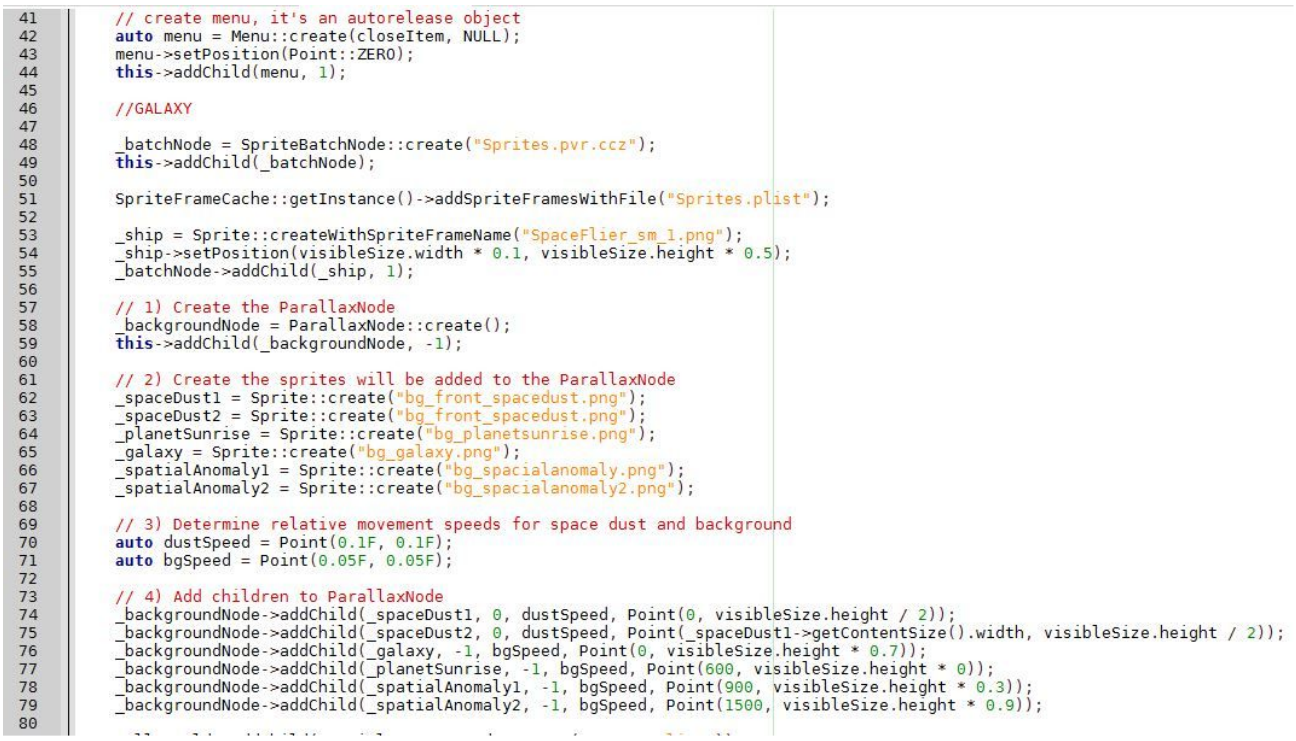


AppDelegate	09/09/2016 12:38 ...	Archivo CPP	4 KB
AppDelegate	09/09/2016 12:36 ...	Archivo H	1 KB
HelloWorldScene	12/09/2016 11:20 ...	Archivo CPP	10 KB
HelloWorldScene	12/09/2016 10:48 ...	Archivo H	2 KB

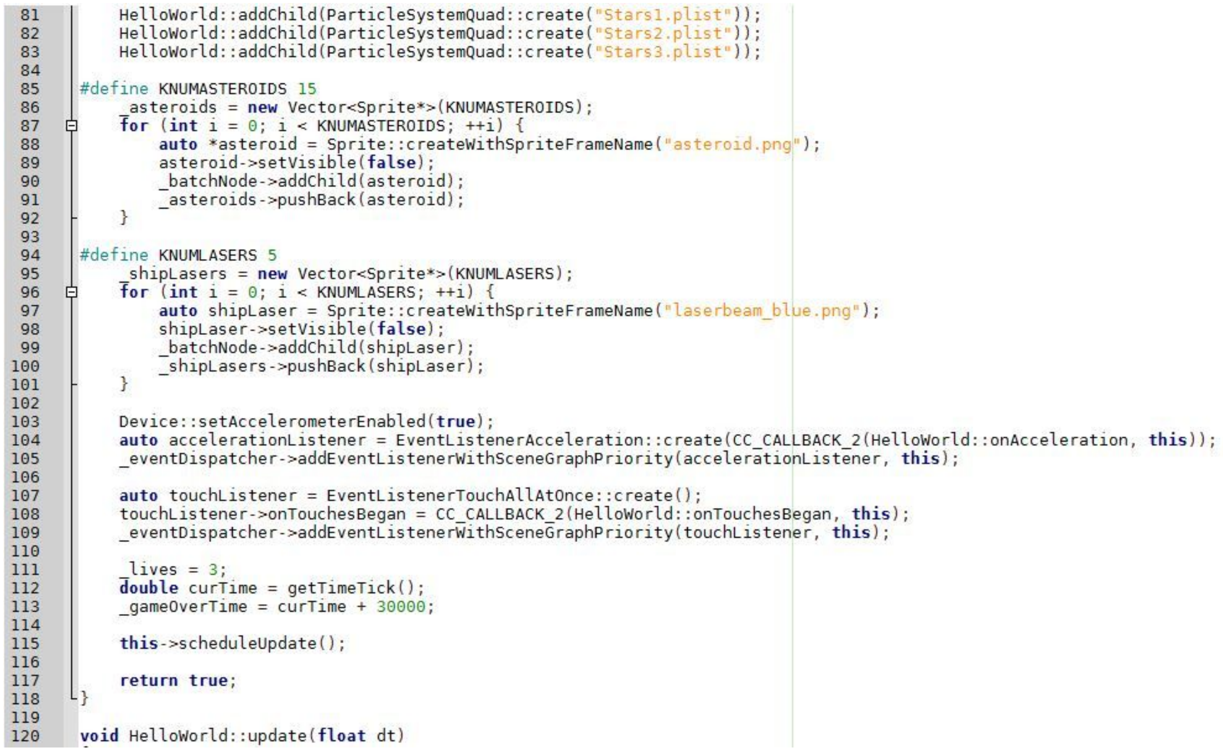
Una vez abierto, complete el código como se le muestra a continuación.



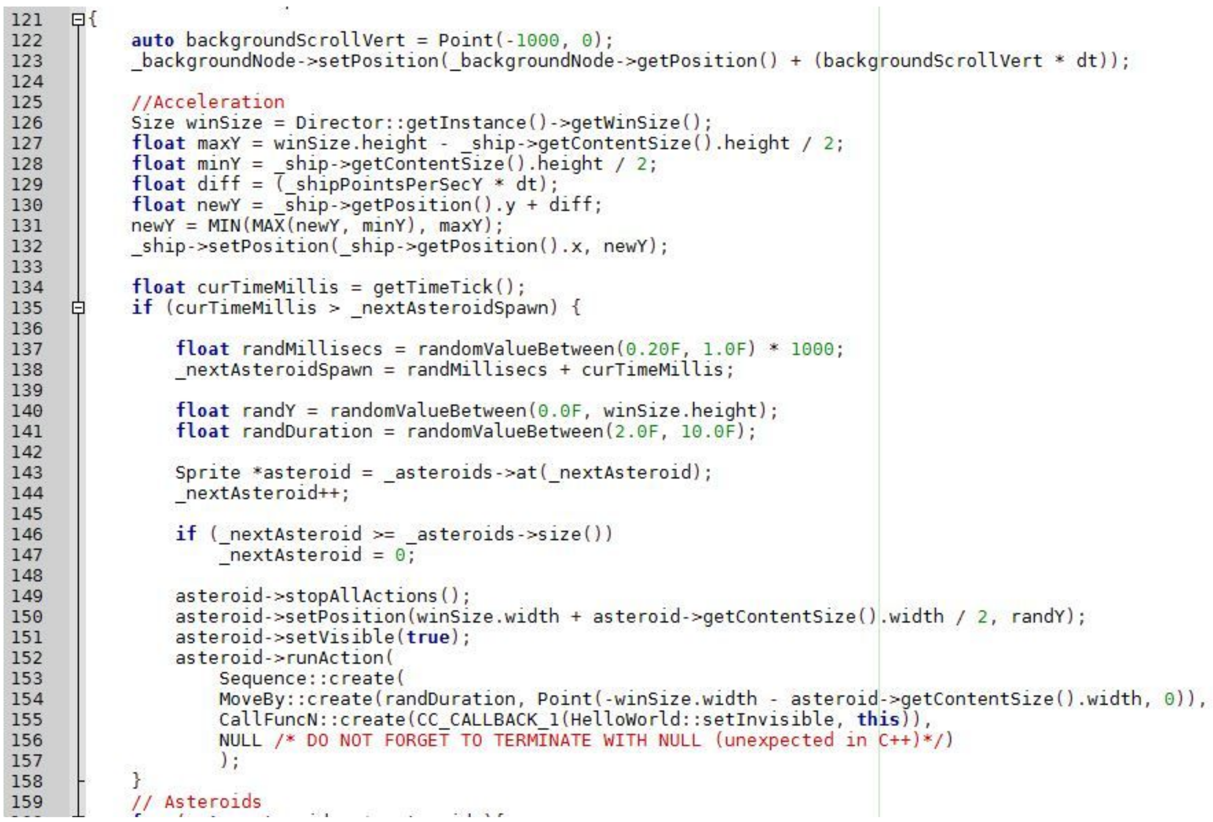
```cpp
#include "HelloWorldScene.h"

USING_NS_CC;

Scene* HelloWorld::createScene()
{
    // 'scene' is an autorelease object
    auto scene = Scene::create();

    // 'layer' is an autorelease object
    auto layer = HelloWorld::create();

    // add layer as a child to scene
    scene->addChild(layer);

    // return the scene
    return scene;
}

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

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

    // add a "close" icon to exit the progress. it's an autorelease object
    auto closeItem = MenuItemImage::create(
        "CloseNormal.png",
        "CloseSelected.png",
        CC_CALLBACK_1(HelloWorld::menuCloseCallback, this));

    closeItem->setPosition(Point(origin.x + visibleSize.width - closeItem->getContentSize().width / 2,
        origin.y + closeItem->getContentSize().height / 2));

```



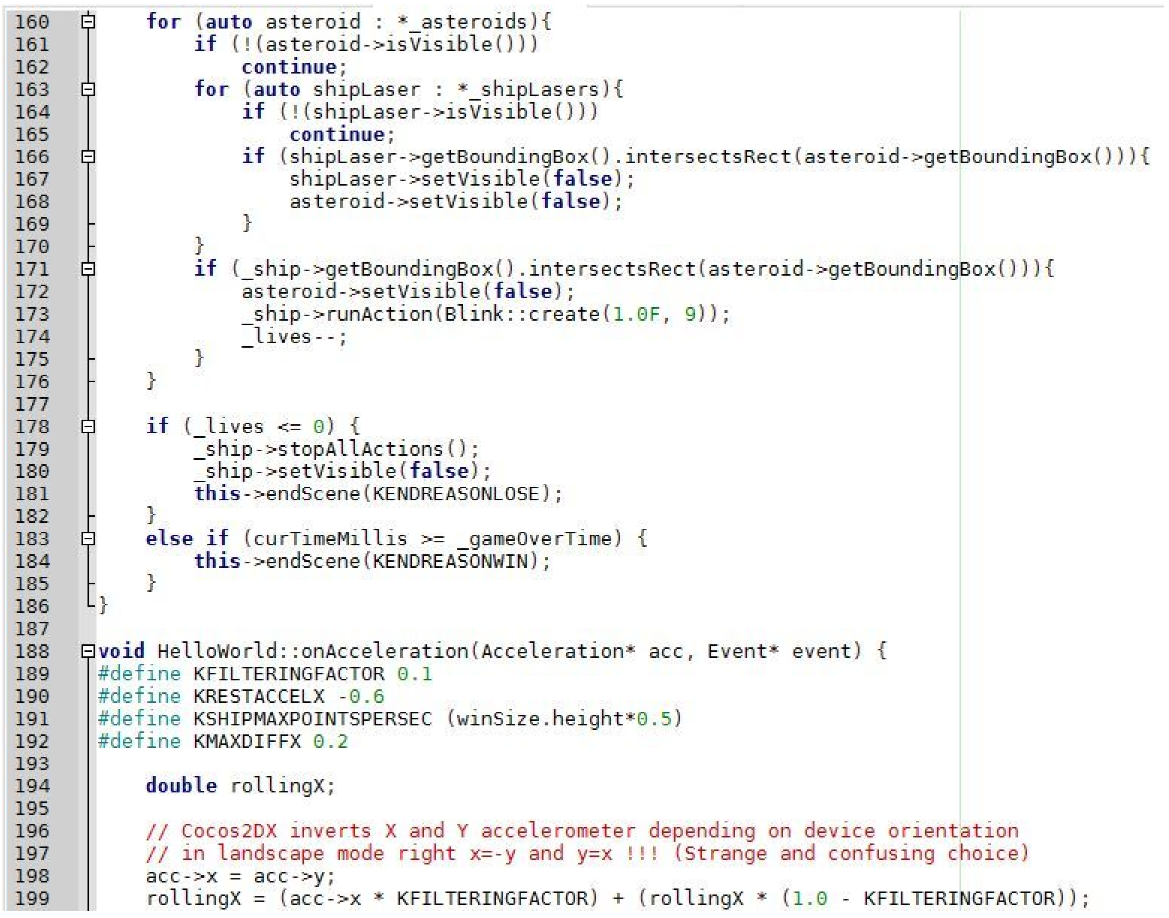
```cpp
    // create menu, it's an autorelease object
    auto menu = Menu::create(closeItem, NULL);
    menu->setPosition(Point::ZERO);
    this->addChild(menu, 1);

    //GALAXY

    _batchNode = SpriteBatchNode::create("Sprites.pvr.ccz");
    this->addChild(_batchNode);

    SpriteFrameCache::getInstance()->addSpriteFramesWithFile("Sprites.plist");

    _ship = Sprite::createWithSpriteFrameName("SpaceFlier_sm_1.png");
    _ship->setPosition(visibleSize.width * 0.1, visibleSize.height * 0.5);
    _batchNode->addChild(_ship, 1);

    // 1) Create the ParallaxNode
    _backgroundNode = ParallaxNode::create();
    this->addChild(_backgroundNode, -1);

    // 2) Create the sprites will be added to the ParallaxNode
    _spaceDust1 = Sprite::create("bg_front_spacedust.png");
    _spaceDust2 = Sprite::create("bg_front_spacedust.png");
    _planetSunrise = Sprite::create("bg_planetsunrise.png");
    _galaxy = Sprite::create("bg_galaxy.png");
    _spatialAnomaly1 = Sprite::create("bg_spacialanomaly.png");
    _spatialAnomaly2 = Sprite::create("bg_spacialanomaly2.png");

    // 3) Determine relative movement speeds for space dust and background
    auto dustSpeed = Point(0.1F, 0.1F);
    auto bgSpeed = Point(0.05F, 0.05F);

    // 4) Add children to ParallaxNode
    _backgroundNode->addChild(_spaceDust1, 0, dustSpeed, Point(0, visibleSize.height / 2));
    _backgroundNode->addChild(_spaceDust2, 0, dustSpeed, Point(_spaceDust1->getContentSize().width, visibleSize.height / 2));
    _backgroundNode->addChild(_galaxy, -1, bgSpeed, Point(0, visibleSize.height * 0.7));
    _backgroundNode->addChild(_planetSunrise, -1, bgSpeed, Point(600, visibleSize.height * 0));
    _backgroundNode->addChild(_spatialAnomaly1, -1, bgSpeed, Point(900, visibleSize.height * 0.3));
    _backgroundNode->addChild(_spatialAnomaly2, -1, bgSpeed, Point(1500, visibleSize.height * 0.9));

```



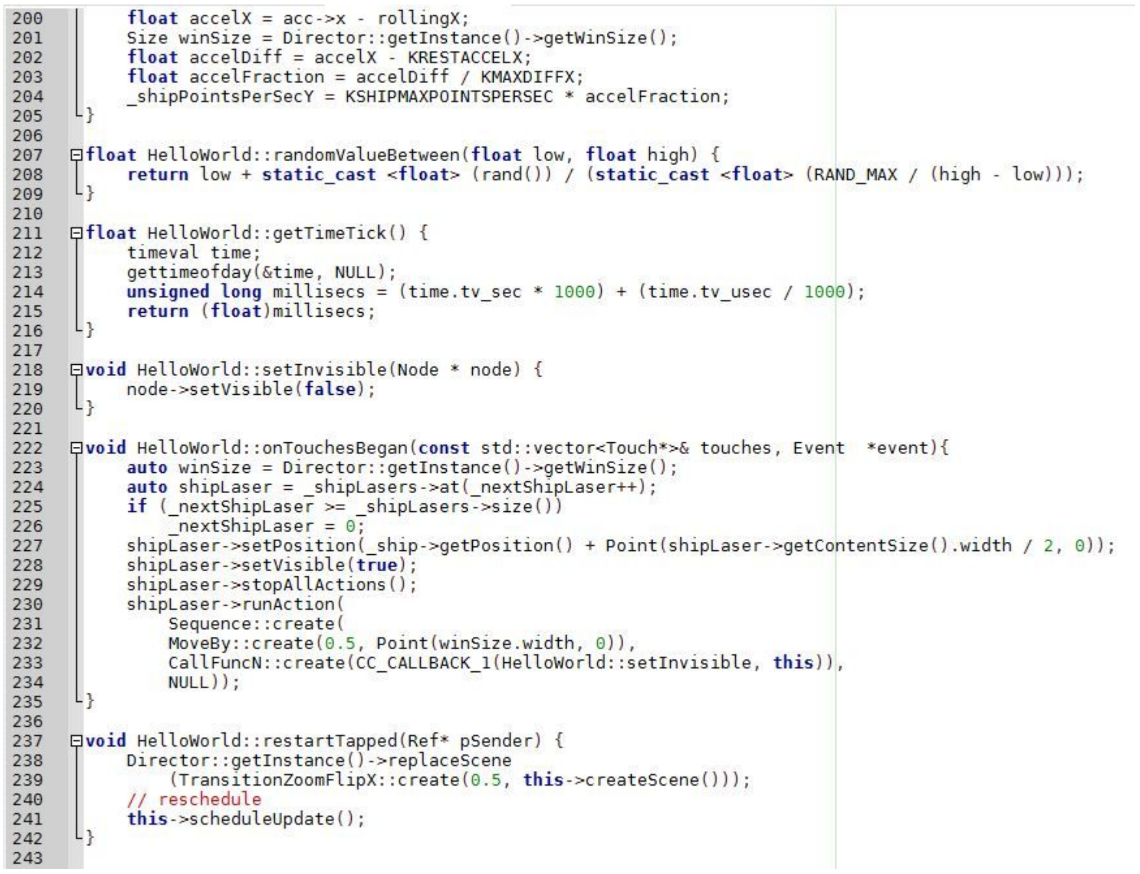
```cpp
    HelloWorld::addChild(ParticleSystemQuad::create("Stars1.plist"));
    HelloWorld::addChild(ParticleSystemQuad::create("Stars2.plist"));
    HelloWorld::addChild(ParticleSystemQuad::create("Stars3.plist"));

#define KNUMASTEROIDS 15
    _asteroids = new Vector<Sprite*>(KNUMASTEROIDS);
    for (int i = 0; i < KNUMASTEROIDS; ++i) {
        auto *asteroid = Sprite::createWithSpriteFrameName("asteroid.png");
        asteroid->setVisible(false);
        _batchNode->addChild(asteroid);
        _asteroids->pushBack(asteroid);
    }

#define KNUMLASERS 5
    _shipLasers = new Vector<Sprite*>(KNUMLASERS);
    for (int i = 0; i < KNUMLASERS; ++i) {
        auto shipLaser = Sprite::createWithSpriteFrameName("laserbeam_blue.png");
        shipLaser->setVisible(false);
        _batchNode->addChild(shipLaser);
        _shipLasers->pushBack(shipLaser);
    }

    Device::setAccelerometerEnabled(true);
    auto accelerationListener = EventListenerAcceleration::create(CC_CALLBACK_2(HelloWorld::onAcceleration, this));
    _eventDispatcher->addEventListenerWithSceneGraphPriority(accelerationListener, this);

    auto touchListener = EventListenerTouchAllAtOnce::create();
    touchListener->onTouchesBegan = CC_CALLBACK_2(HelloWorld::onTouchesBegan, this);
    _eventDispatcher->addEventListenerWithSceneGraphPriority(touchListener, this);

    _lives = 3;
    double curTime = getTimeTick();
    _gameOverTime = curTime + 30000;

    this->scheduleUpdate();

    return true;
}

void HelloWorld::update(float dt)
```



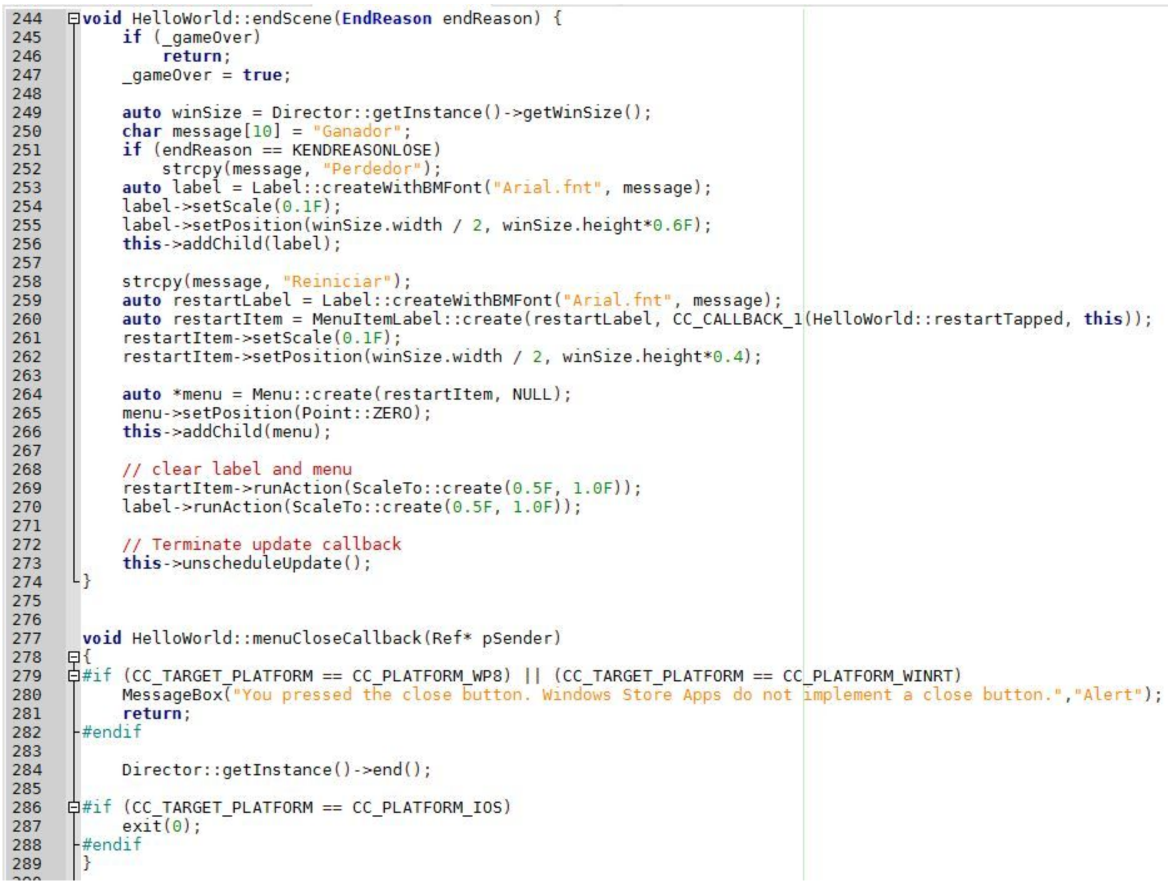
```cpp
{
    auto backgroundScrollVert = Point(-1000, 0);
    _backgroundNode->setPosition(_backgroundNode->getPosition() + (backgroundScrollVert * dt));

    //Acceleration
    Size winSize = Director::getInstance()->getWinSize();
    float maxY = winSize.height - _ship->getContentSize().height / 2;
    float minY = _ship->getContentSize().height / 2;
    float diff = (_shipPointsPerSecY * dt);
    float newY = _ship->getPosition().y + diff;
    newY = MIN(MAX(newY, minY), maxY);
    _ship->setPosition(_ship->getPosition().x, newY);

    float curTimeMillis = getTimeTick();
    if (curTimeMillis > _nextAsteroidSpawn) {

        float randMillisecs = randomValueBetween(0.20F, 1.0F) * 1000;
        _nextAsteroidSpawn = randMillisecs + curTimeMillis;

        float randY = randomValueBetween(0.0F, winSize.height);
        float randDuration = randomValueBetween(2.0F, 10.0F);

        Sprite *asteroid = _asteroids->at(_nextAsteroid);
        _nextAsteroid++;

        if (_nextAsteroid >= _asteroids->size())
            _nextAsteroid = 0;

        asteroid->stopAllActions();
        asteroid->setPosition(winSize.width + asteroid->getContentSize().width / 2, randY);
        asteroid->setVisible(true);
        asteroid->runAction(
            Sequence::create(
            MoveBy::create(randDuration, Point(-winSize.width - asteroid->getContentSize().width, 0)),
            CallFuncN::create(CC_CALLBACK_1(HelloWorld::setInvisible, this)),
            NULL /* DO NOT FORGET TO TERMINATE WITH NULL (unexpected in C++)*/)
            );
    }
    // Asteroids
```



```cpp
    for (auto asteroid : *_asteroids){
        if (!(asteroid->isVisible()))
            continue;
        for (auto shipLaser : *_shipLasers){
            if (!(shipLaser->isVisible()))
                continue;
            if (shipLaser->getBoundingBox().intersectsRect(asteroid->getBoundingBox())){
                shipLaser->setVisible(false);
                asteroid->setVisible(false);
            }
        }
        if (_ship->getBoundingBox().intersectsRect(asteroid->getBoundingBox())){
            asteroid->setVisible(false);
            _ship->runAction(Blink::create(1.0F, 9));
            _lives--;
        }
    }

    if (_lives <= 0) {
        _ship->stopAllActions();
        _ship->setVisible(false);
        this->endScene(KENDREASONLOSE);
    }
    else if (curTimeMillis >= _gameOverTime) {
        this->endScene(KENDREASONWIN);
    }
}

void HelloWorld::onAcceleration(Acceleration* acc, Event* event) {
#define KFILTERINGFACTOR 0.1
#define KRESTACCELX -0.6
#define KSHIPMAXPOINTSPERSEC (winSize.height*0.5)
#define KMAXDIFFX 0.2

    double rollingX;

    // Cocos2DX inverts X and Y accelerometer depending on device orientation
    // in landscape mode right x=-y and y=x !!! (Strange and confusing choice)
    acc->x = acc->y;
    rollingX = (acc->x * KFILTERINGFACTOR) + (rollingX * (1.0 - KFILTERINGFACTOR));
```



```cpp
    float accelX = acc->x - rollingX;
    Size winSize = Director::getInstance()->getWinSize();
    float accelDiff = accelX - KRESTACCELX;
    float accelFraction = accelDiff / KMAXDIFFX;
    _shipPointsPerSecY = KSHIPMAXPOINTSPERSEC * accelFraction;
}

float HelloWorld::randomValueBetween(float low, float high) {
    return low + static_cast <float> (rand()) / (static_cast <float> (RAND_MAX / (high - low)));
}

float HelloWorld::getTimeTick() {
    timeval time;
    gettimeofday(&time, NULL);
    unsigned long millisecs = (time.tv_sec * 1000) + (time.tv_usec / 1000);
    return (float)millisecs;
}

void HelloWorld::setInvisible(Node * node) {
    node->setVisible(false);
}

void HelloWorld::onTouchesBegan(const std::vector<Touch*>& touches, Event  *event){
    auto winSize = Director::getInstance()->getWinSize();
    auto shipLaser = _shipLasers->at(_nextShipLaser++);
    if (_nextShipLaser >= _shipLasers->size())
        _nextShipLaser = 0;
    shipLaser->setPosition(_ship->getPosition() + Point(shipLaser->getContentSize().width / 2, 0));
    shipLaser->setVisible(true);
    shipLaser->stopAllActions();
    shipLaser->runAction(
        Sequence::create(
        MoveBy::create(0.5, Point(winSize.width, 0)),
        CallFuncN::create(CC_CALLBACK_1(HelloWorld::setInvisible, this)),
        NULL));
}

void HelloWorld::restartTapped(Ref* pSender) {
    Director::getInstance()->replaceScene
        (TransitionZoomFlipX::create(0.5, this->createScene()));
    // reschedule
    this->scheduleUpdate();
}

```



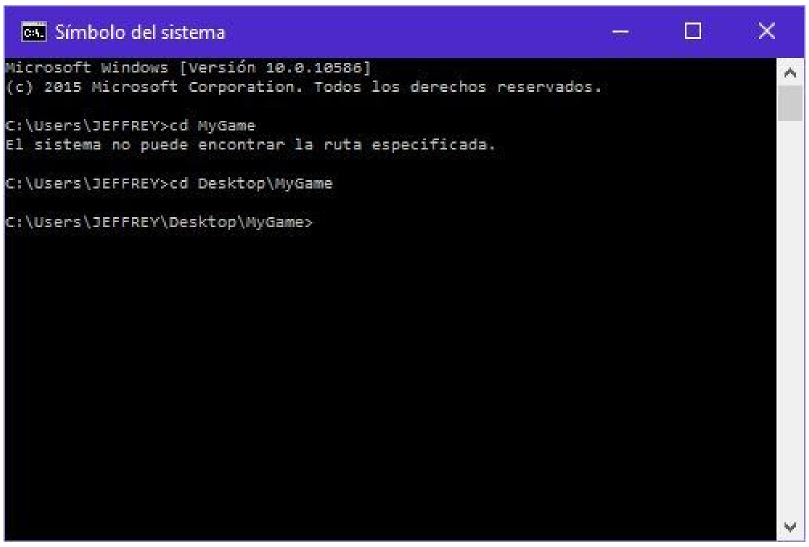
```cpp
void HelloWorld::endScene(EndReason endReason) {
    if (_gameOver)
        return;
    _gameOver = true;

    auto winSize = Director::getInstance()->getWinSize();
    char message[10] = "Ganador";
    if (endReason == KENDREASONLOSE)
        strcpy(message, "Perdedor");
    auto label = Label::createWithBMFont("Arial.fnt", message);
    label->setScale(0.1F);
    label->setPosition(winSize.width / 2, winSize.height*0.6F);
    this->addChild(label);

    strcpy(message, "Reiniciar");
    auto restartLabel = Label::createWithBMFont("Arial.fnt", message);
    auto restartItem = MenuItemLabel::create(restartLabel, CC_CALLBACK_1(HelloWorld::restartTapped, this));
    restartItem->setScale(0.1F);
    restartItem->setPosition(winSize.width / 2, winSize.height*0.4);

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

    // clear label and menu
    restartItem->runAction(ScaleTo::create(0.5F, 1.0F));
    label->runAction(ScaleTo::create(0.5F, 1.0F));

    // Terminate update callback
    this->unscheduleUpdate();
}


void HelloWorld::menuCloseCallback(Ref* pSender)
{
#if (CC_TARGET_PLATFORM == CC_PLATFORM_WP8) || (CC_TARGET_PLATFORM == CC_PLATFORM_WINRT)
    MessageBox("You pressed the close button. Windows Store Apps do not implement a close button.","Alert");
    return;
#endif

    Director::getInstance()->end();

#if (CC_TARGET_PLATFORM == CC_PLATFORM_IOS)
    exit(0);
#endif
}
```

Compilación

Una vez transcrito el código anterior, abra una consola (cmd) y acceda a la carpeta *MyGame* del proyecto para proseguir con la compilación.

Para entrar la carpeta del proyecto:

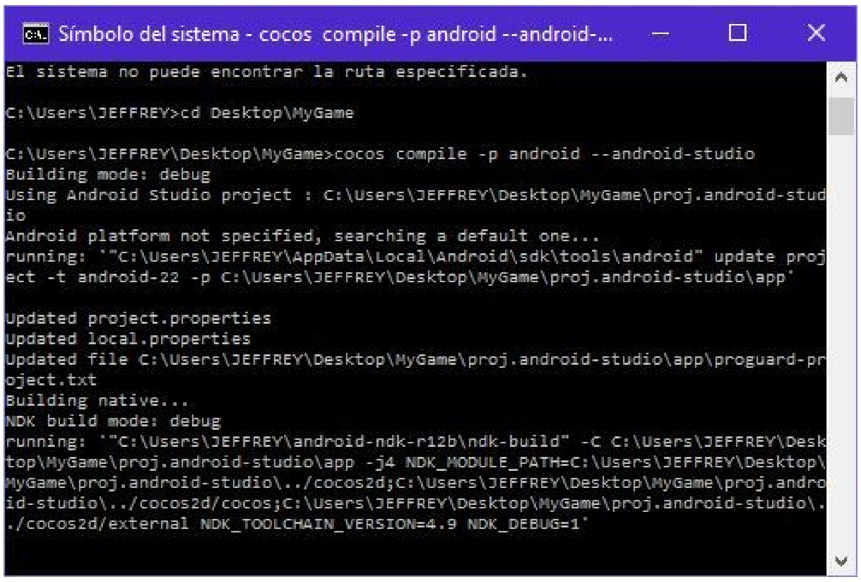
- `cd Desktop\MyCompany\MyGame`



```
Microsoft Windows [Versión 10.0.10586]
(c) 2015 Microsoft Corporation. Todos los derechos reservados.

C:\Users\JEFFREY>cd MyGame
El sistema no puede encontrar la ruta especificada.

C:\Users\JEFFREY>cd Desktop\MyGame

C:\Users\JEFFREY\Desktop\MyGame>
```

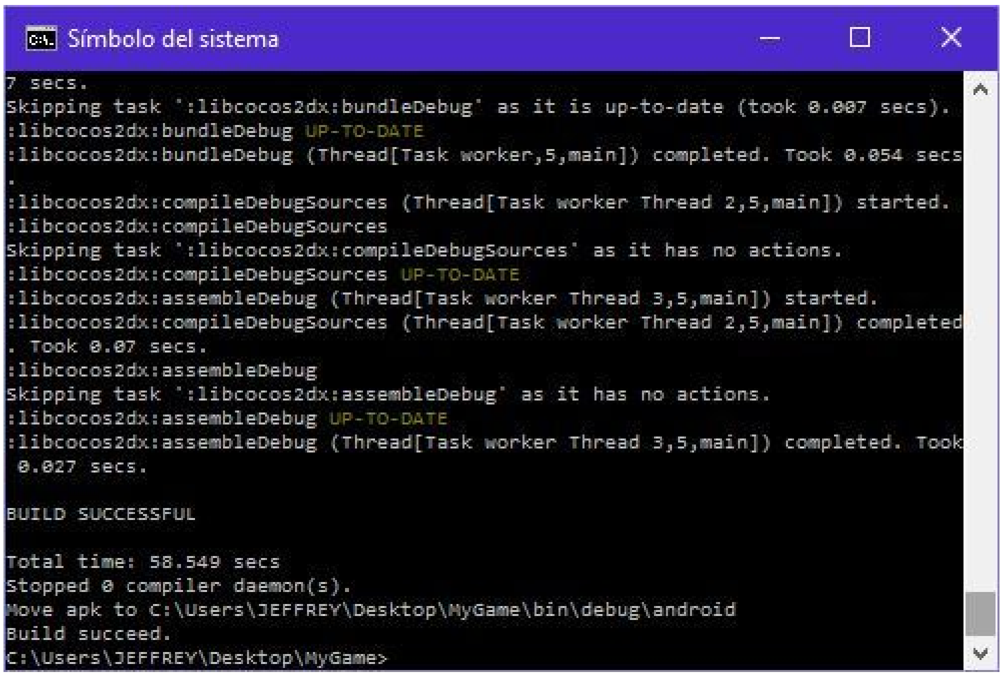
Para compilar el proyecto:

- `cocos compile -p android --android-studio`


Símbolo del sistema - cocos compile -p android --android-...

```
El sistema no puede encontrar la ruta especificada.

C:\Users\JEFFREY>cd Desktop\MyGame

C:\Users\JEFFREY\Desktop\MyGame>cocos compile -p android --android-studio
Building mode: debug
Using Android Studio project : C:\Users\JEFFREY\Desktop\MyGame\proj.android-stud
io
Android platform not specified, searching a default one...
running: '"C:\Users\JEFFREY\AppData\Local\Android\sdk\tools\android" update proj
ect -t android-22 -p C:\Users\JEFFREY\Desktop\MyGame\proj.android-studio\app'

Updated project.properties
Updated local.properties
Updated file C:\Users\JEFFREY\Desktop\MyGame\proj.android-studio\app\proguard-pr
oject.txt
Building native...
NDK build mode: debug
running: '"C:\Users\JEFFREY\android-ndk-r12b\ndk-build" -C C:\Users\JEFFREY\Desk
top\MyGame\proj.android-studio\app -j4 NDK_MODULE_PATH=C:\Users\JEFFREY\Desktop\
MyGame\proj.android-studio\../cocos2d;C:\Users\JEFFREY\Desktop\MyGame\proj.andro
id-studio\../cocos2d/cocos;C:\Users\JEFFREY\Desktop\MyGame\proj.android-studio\.
./cocos2d/external NDK_TOOLCHAIN_VERSION=4.9 NDK_DEBUG=1'
```

Una vez finalizada la compilación debe aparecer un mensaje como el siguiente:



```
7 secs.
Skipping task ':libcocos2dx:bundleDebug' as it is up-to-date (took 0.007 secs).
:libcocos2dx:bundleDebug UP-TO-DATE
:libcocos2dx:bundleDebug (Thread[Task worker,5,main]) completed. Took 0.054 secs
.
:libcocos2dx:compileDebugSources (Thread[Task worker Thread 2,5,main]) started.
:libcocos2dx:compileDebugSources
Skipping task ':libcocos2dx:compileDebugSources' as it has no actions.
:libcocos2dx:compileDebugSources UP-TO-DATE
:libcocos2dx:assembleDebug (Thread[Task worker Thread 3,5,main]) started.
:libcocos2dx:compileDebugSources (Thread[Task worker Thread 2,5,main]) completed
. Took 0.07 secs.
:libcocos2dx:assembleDebug
Skipping task ':libcocos2dx:assembleDebug' as it has no actions.
:libcocos2dx:assembleDebug UP-TO-DATE
:libcocos2dx:assembleDebug (Thread[Task worker Thread 3,5,main]) completed. Took
 0.027 secs.

BUILD SUCCESSFUL

Total time: 58.549 secs
Stopped 0 compiler daemon(s).
Move apk to C:\Users\JEFFREY\Desktop\MyGame\bin\debug\android
Build succeed.
C:\Users\JEFFREY\Desktop\MyGame>
```

El archivo *.apk* de la aplicación se encuentra en la ruta:

- `Desktop\MyCompany\MyGame\bin\debug\android.`