## 1.构建场景，有多种方式

|  |
| --- |
|  |

**注意：init方法只是在场景场景的时候调用，如果场景场景后需要改变场景，需要使用update方法**

## 2.场景的切换方法

|  |
| --- |
|  |

**注意，使用pushScene和popScene的方法是非常消耗内存的，因为所有的场景都在内存中，而replaceScene比较好，因为它加载了新场景后会把旧创建释放掉。**

利用**pushScene和popScene换场景的实例**

|  |  |
| --- | --- |
| //HelloWorldScene.h  #ifndef \_\_HELLOWORLD\_SCENE\_H\_\_  #define \_\_HELLOWORLD\_SCENE\_H\_\_  #include "cocos2d.h"  class HelloWorld : public cocos2d::Scene  {  public:  static cocos2d::Scene\* createScene();  virtual bool init();    // a selector callback  void menuCloseCallback(cocos2d::Ref\* pSender);    // implement the "static create()" method manually  CREATE\_FUNC(HelloWorld);  private:  //添加一个点击计数器  int m\_counter;  };  #endif // \_\_HELLOWORLD\_SCENE\_H\_\_ | //HelloWorldScene.cpp  #include "HelloWorldScene.h"  #include "SpriteScene2.h"  #include "SimpleAudioEngine.h"  USING\_NS\_CC;  Scene\* HelloWorld::createScene()  {  return HelloWorld::create();  }  // Print useful error message instead of segfaulting when files are not there.  static void problemLoading(const char\* filename)  {  printf("Error while loading: %s\n", filename);  printf("Depending on how you compiled you might have to add 'Resources/' in front of filenames in HelloWorldScene.cpp\n");  }  // on "init" you need to initialize your instance  bool HelloWorld::init()  {  //////////////////////////////  // 1. super init first  if ( !Scene::init() )  {  return false;  }  auto visibleSize = Director::getInstance()->getVisibleSize();  Vec2 origin = Director::getInstance()->getVisibleOrigin();  /////////////////////////////  // 2. add a menu item with "X" image, which is clicked to quit the program  // you may modify it.  // 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));  if (closeItem == nullptr ||  closeItem->getContentSize().width <= 0 ||  closeItem->getContentSize().height <= 0)  {  problemLoading("'CloseNormal.png' and 'CloseSelected.png'");  }  else  {  float x = origin.x + visibleSize.width - closeItem->getContentSize().width/2;  float y = origin.y + closeItem->getContentSize().height/2;  closeItem->setPosition(Vec2(x,y));  }  // create menu, it's an autorelease object  auto menu = Menu::create(closeItem, NULL);  menu->setPosition(Vec2::ZERO);  this->addChild(menu, 1);  /////////////////////////////  // 3. add your codes below...  // add a label shows "Hello World"  // create and initialize a label  auto label = Label::createWithTTF("Hello World", "fonts/Marker Felt.ttf", 24);  if (label == nullptr)  {  problemLoading("'fonts/Marker Felt.ttf'");  }  else  {  // position the label on the center of the screen  label->setPosition(Vec2(origin.x + visibleSize.width/2,  origin.y + visibleSize.height - label->getContentSize().height));  /\*label->setPosition(Vec2(origin.x + visibleSize.width\*5 / 12,  origin.y + visibleSize.height - label->getContentSize().height));\*/  // add the label as a child to this layer  this->addChild(label, 1);  }  // add "HelloWorld" splash screen"  auto sprite = Sprite::create("HelloWorld.png");  if (sprite == nullptr)  {  problemLoading("'HelloWorld.png'");  }  else  {  // position the sprite on the center of the screen  sprite->setPosition(Vec2(visibleSize.width/2 + origin.x, visibleSize.height/2 + origin.y));  // add the sprite as a child to this layer  this->addChild(sprite, 0);  }  //设置文本和图片左对齐  //写法1.  /\*label->setPosition(Vec2(origin.x + sprite->getContentSize().width,  origin.y + visibleSize.height - label->getContentSize().height));\*/  //写法2  label->setPositionX(label->getPosition().x - sprite->getContentSize().width / 2);  label->setAnchorPoint(Vec2::ANCHOR\_MIDDLE\_LEFT);  //label->setAnchorPoint(Vec2::ANCHOR\_BOTTOM\_LEFT);  //label->setAnchorPoint(Vec2::ANCHOR\_TOP\_LEFT);  //label->setAnchorPoint(Vec2::ANCHOR\_BOTTOM\_RIGHT);  //label->setAnchorPoint(Vec2::ANCHOR\_TOP\_RIGHT);  //label->setAnchorPoint(Vec2::ANCHOR\_MIDDLE);  //label->setAnchorPoint(Vec2::ANCHOR\_MIDDLE\_TOP);  //label->setAnchorPoint(Vec2::ANCHOR\_MIDDLE\_BOTTOM);  m\_counter = 0;  return true;  }  void HelloWorld::menuCloseCallback(Ref\* pSender)  {  m\_counter++;  //Close the cocos2d-x game scene and quit the application  //Director::getInstance()->end();  //修改为切换场景,方法1  //Director::getInstance()->replaceScene(SpriteScene::createScene());  //修改为切换场景,方法2  if (m\_counter % 2 == 1) {  Director::getInstance()->pushScene(SpriteScene::createScene());  }    #if (CC\_TARGET\_PLATFORM == CC\_PLATFORM\_IOS)  //exit(0);  #endif  /\*To navigate back to native iOS screen(if present) without quitting the application ,do not use Director::getInstance()->end() and exit(0) as given above,instead trigger a custom event created in RootViewController.mm as below\*/  //EventCustom customEndEvent("game\_scene\_close\_event");  //\_eventDispatcher->dispatchEvent(&customEndEvent);  } |
| //SpriteScene2.h  #pragma once  #include"cocos2d.h"  //#include"ui/CocosGUI.h"  USING\_NS\_CC;  //using namespace ui;  #include<string>  using namespace std;  class SpriteScene:public Scene  {  public:  static cocos2d::Scene\* createScene();  virtual bool init();  //menu callback  void menuCloseCallback(cocos2d::Ref\* sender);  CREATE\_FUNC(SpriteScene);  private:  Sprite\* m\_polySprite;  float m\_angle;  int m\_opacity;  float m\_scaleX;  float m\_minScaleX;  bool toLeft;  bool m\_rotate;  bool isMirrored;//是否已经镜像  }; | //SpriteScene2.cpp  #include "SpriteScene2.h"  cocos2d::Scene \* SpriteScene::createScene()  {  return SpriteScene::create();  }  bool SpriteScene::init()  {  if (!Scene::init()) return false;  //获取可视区域大小  auto visibleSize = Director::getInstance()->getVisibleSize();  auto origin = Director::getInstance()->getVisibleOrigin();  //获取原点坐标  //创建关闭按钮  auto closeItem = MenuItemImage::create("CloseNormal.png", "CloseSelected.png", CC\_CALLBACK\_1(SpriteScene::menuCloseCallback, this));  //设置位置  float x = origin.x + visibleSize.width - closeItem->getContentSize().width / 2;  float y = origin.y+ closeItem->getContentSize().height / 2;  closeItem->setPosition(Vec2(x, y));  auto menu = Menu::create(closeItem, NULL);  menu->setPosition(Vec2::ZERO);  this->addChild(menu, 1);  ////创建精灵1.使用图片，会失真  //auto sprite = Sprite::create("sprite/jlgirl.png");  ////缩放  //sprite->setScale(0.5f);  //sprite->setAnchorPoint(Vec2::ANCHOR\_MIDDLE);  //sprite->setPosition(Vec2(visibleSize / 2));  //this->addChild(sprite);  ////创建精灵2.使用图集，需要使用TexturePacker来生成。需要用到一个plist文件  ////a.创建背景  //auto bg = Sprite::create("spritesheet/background.png");  //bg->setAnchorPoint(Vec2::ZERO);  //this->addChild(bg, 0);  ////创建缓存对象,它是一个单例对象  //SpriteFrameCache\* frameCache = SpriteFrameCache::getInstance();  ////加载图集的plist文件  //frameCache->addSpriteFramesWithFile("spritesheet/SpriteSheet.plist");  ////通过一个mountain1精灵帧名称创建精灵  //auto mountain1 = Sprite::createWithSpriteFrameName("mountain1.png");  ////设置锚点  //mountain1->setAnchorPoint(Vec2::ZERO);  ////设置位置  //mountain1->setPosition(Vec2(-200,10));  //this->addChild(mountain1,0);  ////根据精灵帧创建精灵  //SpriteFrame\* heroFrm = frameCache->getSpriteFrameByName("hero1.png");  //auto hero = Sprite::createWithSpriteFrame(heroFrm);  //hero->setPosition(Vec2(visibleSize/2));  //this->addChild(hero,0);  ////创建精灵3.使用贴图，也就是纹理  ////创建纹理缓存对象  //auto textureCache = TextureCache::getInstance();  ////加载纹理  //auto cache = textureCache->addImage("sprite/gamebg1.png");  ////创建精灵  //auto cacheSprite = Sprite::createWithTexture(cache);  ////裁剪贴图,没有必要不要裁剪否则不好看  ////cacheSprite->setTextureRect(Rect(0,0,480,320));  //cacheSprite->setPosition(Vec2(visibleSize / 2));  //this->addChild(cacheSprite);  ////创建精灵4.创建多边形精灵  auto plg = AutoPolygon::generatePolygon("sprite/zdj.png", Rect(0, 0, 256, 198));  m\_polySprite = Sprite::create(plg);  m\_polySprite->setPosition(Vec2(visibleSize / 2));  this->addChild(m\_polySprite);  //精灵控制  //旋转效果,需要使用调度器,还可以使用动作来实现  m\_angle = 0;//注意这里需要将角度定义为成员变量，否则只会旋转一次  //m\_polySprite->setRotation(m\_angle);  //this->schedule([&](float dlt) {  // m\_angle += 1.f;  // if (m\_angle > 360) {  // m\_angle -= 360;  // }  // m\_polySprite->setRotation(m\_angle);  //},0.02f,"schedule");  //实现精灵淡入效果    //m\_opacity = 0;  //m\_polySprite->setOpacity(m\_opacity); //先将透明度设置为0，这样子就不会有一闪的效果  //this->schedule([&](float dlt) {  // m\_opacity += 1;  // if (m\_opacity >=255) {  // //m\_opacity -= 250;//这样子会一直重复  // //如果只需要做一次，可以在这里注销调度器  // this->unschedule("schedule2");  // return;  // }  // m\_polySprite->setOpacity(m\_opacity);  //}, 0.02f, "schedule2");  ////实现精灵水平旋转的效果  //m\_scaleX = m\_polySprite->getScaleX();  ////float tmp = m\_scaleX;  //m\_minScaleX = - m\_polySprite->getScaleX();  //this->schedule([&](float dlt) {  // //翻转了就注销调度器  // /\*if (m\_scaleX < m\_minScaleX) {  // this->unschedule("schedule3");  // return;  // }\*/  // //重新设置缩放系数，这样子它就可以一直水平旋转下去  // if (m\_scaleX < m\_minScaleX) {  // m\_scaleX = -m\_minScaleX;  // }  // m\_scaleX -= 0.01f;  // m\_polySprite->setScaleX(m\_scaleX);  //},0.02f,"schedule3");    //实现精灵在屏幕在自动移动的效果  m\_polySprite->setScale(0.4f);  //m\_position = m\_polySprite->getPosition();  //Vec2 tmpPos = m\_position;  //m\_polySprite->setPosition(m\_position);    /\*this->schedule([&](float dlt) {    if (m\_polySprite-> getPositionX()+ m\_polySprite->getContentSize().width/2 \* m\_polySprite-> getScale()> Director::getInstance()->getVisibleSize().width  || m\_polySprite->getPositionX() - m\_polySprite->getContentSize().width / 2 \* m\_polySprite->getScale()<0)  {  this->unschedule("schedule4");  return;  }  m\_polySprite->setPositionX(m\_polySprite->getPositionX() - 1.f);  }, 0.02f, "schedule4");\*/    //先往右移动到边界就往左移动，移动到左边界就旋转360°然后淡出  m\_minScaleX = -m\_polySprite->getScaleX();  m\_opacity = m\_polySprite->getOpacity();  m\_rotate = false;  toLeft = false;  isMirrored = false;  this->schedule([&](float dlt) {    if (!toLeft ) {    if (m\_polySprite->getPositionX() + m\_polySprite->getContentSize().width / 2 \* m\_polySprite->getScale()> Director::getInstance()->getVisibleSize().width)  {  toLeft = true;    }  if (!m\_rotate) {  m\_polySprite->setPositionX(m\_polySprite->getPositionX() + 1.f);  }  else  {  if (m\_polySprite->getRotation() >= 360) {  if (m\_polySprite->getOpacity() <= 0) {  //因为在Hello场景在使用了pushScene，这里需要使用popScene切换回去  Director::getInstance()->popScene();  this->unschedule("schedule5");  return;  }  m\_polySprite->setOpacity(m\_polySprite->getOpacity() - 5.f);    }  m\_polySprite->setRotation(m\_polySprite->getRotation() + 1.f);  }  }  else if(toLeft) {    if (m\_polySprite->getPositionX() - m\_polySprite->getContentSize().width / 2 \* m\_polySprite->getScale() < 0) {  toLeft = false;  m\_rotate = true;  }  /\*if (!isMirrored) //这段代码有问题  {  if (m\_polySprite->getScaleX() < m\_minScaleX)  {  isMirrored = true;  m\_polySprite->setScaleX(-m\_minScaleX);  }  m\_polySprite->setScaleX(m\_polySprite->getScaleX() - 0.01f);  }\*/  m\_polySprite->setPositionX(m\_polySprite->getPositionX() - 1.f);  }    }, 0.02f, "schedule5");    return true;  }  void SpriteScene::menuCloseCallback(cocos2d::Ref \* sender)  {  Director::getInstance()->end();  } |
| //AppDelegate.cpp  #include "AppDelegate.h"  #include"SpriteScene2.h"  #include"HelloWorldScene.h"  // #define USE\_AUDIO\_ENGINE 1  // #define USE\_SIMPLE\_AUDIO\_ENGINE 1  #if USE\_AUDIO\_ENGINE && USE\_SIMPLE\_AUDIO\_ENGINE  #error "Don't use AudioEngine and SimpleAudioEngine at the same time. Please just select one in your game!"  #endif  #if USE\_AUDIO\_ENGINE  #include "audio/include/AudioEngine.h"  using namespace cocos2d::experimental;  #elif USE\_SIMPLE\_AUDIO\_ENGINE  #include "audio/include/SimpleAudioEngine.h"  using namespace CocosDenshion;  #endif  USING\_NS\_CC;  static cocos2d::Size designResolutionSize = cocos2d::Size(480, 320);  //static cocos2d::Size designResolutionSize = cocos2d::Size(600, 480);  static cocos2d::Size smallResolutionSize = cocos2d::Size(480, 320);  static cocos2d::Size mediumResolutionSize = cocos2d::Size(1024, 768);  static cocos2d::Size largeResolutionSize = cocos2d::Size(2048, 1536);  AppDelegate::AppDelegate()  {  }  AppDelegate::~AppDelegate()  {  #if USE\_AUDIO\_ENGINE  AudioEngine::end();  #elif USE\_SIMPLE\_AUDIO\_ENGINE  SimpleAudioEngine::end();  #endif  }  // if you want a different context, modify the value of glContextAttrs  // it will affect all platforms  void AppDelegate::initGLContextAttrs()  {  // set OpenGL context attributes: red,green,blue,alpha,depth,stencil  GLContextAttrs glContextAttrs = {8, 8, 8, 8, 24, 8};  GLView::setGLContextAttrs(glContextAttrs);  }  // if you want to use the package manager to install more packages,  // don't modify or remove this function  static int register\_all\_packages()  {  return 0; //flag for packages manager  }  bool AppDelegate::applicationDidFinishLaunching() {  // initialize director  auto director = Director::getInstance();  auto glview = director->getOpenGLView();  if(!glview) {  #if (CC\_TARGET\_PLATFORM == CC\_PLATFORM\_WIN32) || (CC\_TARGET\_PLATFORM == CC\_PLATFORM\_MAC) || (CC\_TARGET\_PLATFORM == CC\_PLATFORM\_LINUX)  glview = GLViewImpl::createWithRect("Hello", cocos2d::Rect(0, 0, designResolutionSize.width, designResolutionSize.height));  #else  glview = GLViewImpl::create("Hello");  #endif  director->setOpenGLView(glview);  }  // turn on display FPS  director->setDisplayStats(true);  // set FPS. the default value is 1.0/60 if you don't call this  director->setAnimationInterval(1.0f / 60);  // Set the design resolution  glview->setDesignResolutionSize(designResolutionSize.width, designResolutionSize.height, ResolutionPolicy::NO\_BORDER);  //glview->setDesignResolutionSize(designResolutionSize.width, designResolutionSize.height, ResolutionPolicy::SHOW\_ALL);  auto frameSize = glview->getFrameSize();  // if the frame's height is larger than the height of medium size.  if (frameSize.height > mediumResolutionSize.height)  {  director->setContentScaleFactor(MIN(largeResolutionSize.height/designResolutionSize.height, largeResolutionSize.width/designResolutionSize.width));  }  // if the frame's height is larger than the height of small size.  else if (frameSize.height > smallResolutionSize.height)  {  director->setContentScaleFactor(MIN(mediumResolutionSize.height/designResolutionSize.height, mediumResolutionSize.width/designResolutionSize.width));  }  // if the frame's height is smaller than the height of medium size.  else  {  director->setContentScaleFactor(MIN(smallResolutionSize.height/designResolutionSize.height, smallResolutionSize.width/designResolutionSize.width));  }  register\_all\_packages();  // create a scene. it's an autorelease object  auto scene = HelloWorld::createScene();  //auto scene = SpriteScene::createScene();  // run  director->runWithScene(scene);  return true;  }  // This function will be called when the app is inactive. Note, when receiving a phone call it is invoked.  void AppDelegate::applicationDidEnterBackground() {  Director::getInstance()->stopAnimation();  #if USE\_AUDIO\_ENGINE  AudioEngine::pauseAll();  #elif USE\_SIMPLE\_AUDIO\_ENGINE  SimpleAudioEngine::getInstance()->pauseBackgroundMusic();  SimpleAudioEngine::getInstance()->pauseAllEffects();  #endif  }  // this function will be called when the app is active again  void AppDelegate::applicationWillEnterForeground() {  Director::getInstance()->startAnimation();  #if USE\_AUDIO\_ENGINE  AudioEngine::resumeAll();  #elif USE\_SIMPLE\_AUDIO\_ENGINE  SimpleAudioEngine::getInstance()->resumeBackgroundMusic();  SimpleAudioEngine::getInstance()->resumeAllEffects();  #endif  } | 效果 |

## 3.场景切换的效果

|  |
| --- |
|  |

应用效果

|  |
| --- |
|  |

创建切换效果实例

|  |  |
| --- | --- |
| //HelloWorldScene.h  #ifndef \_\_HELLOWORLD\_SCENE\_H\_\_  #define \_\_HELLOWORLD\_SCENE\_H\_\_  #include "cocos2d.h"  class HelloWorld : public cocos2d::Scene  {  public:  static cocos2d::Scene\* createScene();  virtual bool init();    // a selector callback  void menuCloseCallback(cocos2d::Ref\* pSender);    // implement the "static create()" method manually  CREATE\_FUNC(HelloWorld);  private:  //添加一个点击计数器  int m\_counter;  };  #endif // \_\_HELLOWORLD\_SCENE\_H\_\_ | //HelloWorldScene.cpp  #include "HelloWorldScene.h"  #include "SpriteScene2.h"  #include "SimpleAudioEngine.h"  USING\_NS\_CC;  Scene\* HelloWorld::createScene()  {  return HelloWorld::create();  }  // Print useful error message instead of segfaulting when files are not there.  static void problemLoading(const char\* filename)  {  printf("Error while loading: %s\n", filename);  printf("Depending on how you compiled you might have to add 'Resources/' in front of filenames in HelloWorldScene.cpp\n");  }  // on "init" you need to initialize your instance  bool HelloWorld::init()  {  //////////////////////////////  // 1. super init first  if ( !Scene::init() )  {  return false;  }  auto visibleSize = Director::getInstance()->getVisibleSize();  Vec2 origin = Director::getInstance()->getVisibleOrigin();  /////////////////////////////  // 2. add a menu item with "X" image, which is clicked to quit the program  // you may modify it.  // 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));  if (closeItem == nullptr ||  closeItem->getContentSize().width <= 0 ||  closeItem->getContentSize().height <= 0)  {  problemLoading("'CloseNormal.png' and 'CloseSelected.png'");  }  else  {  float x = origin.x + visibleSize.width - closeItem->getContentSize().width/2;  float y = origin.y + closeItem->getContentSize().height/2;  closeItem->setPosition(Vec2(x,y));  }  // create menu, it's an autorelease object  auto menu = Menu::create(closeItem, NULL);  menu->setPosition(Vec2::ZERO);  this->addChild(menu, 1);  /////////////////////////////  // 3. add your codes below...  // add a label shows "Hello World"  // create and initialize a label  auto label = Label::createWithTTF("Hello World", "fonts/Marker Felt.ttf", 24);  if (label == nullptr)  {  problemLoading("'fonts/Marker Felt.ttf'");  }  else  {  // position the label on the center of the screen  label->setPosition(Vec2(origin.x + visibleSize.width/2,  origin.y + visibleSize.height - label->getContentSize().height));  /\*label->setPosition(Vec2(origin.x + visibleSize.width\*5 / 12,  origin.y + visibleSize.height - label->getContentSize().height));\*/  // add the label as a child to this layer  this->addChild(label, 1);  }  // add "HelloWorld" splash screen"  auto sprite = Sprite::create("HelloWorld.png");  if (sprite == nullptr)  {  problemLoading("'HelloWorld.png'");  }  else  {  // position the sprite on the center of the screen  sprite->setPosition(Vec2(visibleSize.width/2 + origin.x, visibleSize.height/2 + origin.y));  // add the sprite as a child to this layer  this->addChild(sprite, 0);  }  //设置文本和图片左对齐  //写法1.  /\*label->setPosition(Vec2(origin.x + sprite->getContentSize().width,  origin.y + visibleSize.height - label->getContentSize().height));\*/  //写法2  label->setPositionX(label->getPosition().x - sprite->getContentSize().width / 2);  label->setAnchorPoint(Vec2::ANCHOR\_MIDDLE\_LEFT);  //label->setAnchorPoint(Vec2::ANCHOR\_BOTTOM\_LEFT);  //label->setAnchorPoint(Vec2::ANCHOR\_TOP\_LEFT);  //label->setAnchorPoint(Vec2::ANCHOR\_BOTTOM\_RIGHT);  //label->setAnchorPoint(Vec2::ANCHOR\_TOP\_RIGHT);  //label->setAnchorPoint(Vec2::ANCHOR\_MIDDLE);  //label->setAnchorPoint(Vec2::ANCHOR\_MIDDLE\_TOP);  //label->setAnchorPoint(Vec2::ANCHOR\_MIDDLE\_BOTTOM);  m\_counter = 0;  return true;  }  void HelloWorld::menuCloseCallback(Ref\* pSender)  {  m\_counter++;  //Close the cocos2d-x game scene and quit the application  //Director::getInstance()->end();  //修改为切换场景,方法1  //Director::getInstance()->replaceScene(SpriteScene::createScene());  //修改为切换场景,方法2  if (m\_counter % 2 == 1) {  //Director::getInstance()->pushScene(TransitionFade::create(2.f, SpriteScene::createScene(), Color3B::RED));//添加切换场景效果  //Director::getInstance()->pushScene(TransitionSlideInB::create(2.f, SpriteScene::createScene()));//添加切换场景效果,从下往上滑动  //Director::getInstance()->pushScene(TransitionSlideInT::create(2.f, SpriteScene::createScene()));//添加切换场景效果。从上往下滑动  //Director::getInstance()->pushScene(TransitionSlideInL::create(2.f, SpriteScene::createScene()));//添加切换场景效果,从左往右  Director::getInstance()->pushScene(TransitionSlideInR::create(2.f, SpriteScene::createScene()));//添加切换场景效果，从右往左  //Director::getInstance()->pushScene(TransitionFlipX::create(2.f, SpriteScene::createScene()));//添加切换场景效果,翻转x轴  //Director::getInstance()->pushScene(TransitionFlipY::create(2.f, SpriteScene::createScene()));//添加切换场景效果，翻转y轴 }    #if (CC\_TARGET\_PLATFORM == CC\_PLATFORM\_IOS)  //exit(0);  #endif  /\*To navigate back to native iOS screen(if present) without quitting the application ,do not use Director::getInstance()->end() and exit(0) as given above,instead trigger a custom event created in RootViewController.mm as below\*/  //EventCustom customEndEvent("game\_scene\_close\_event");  //\_eventDispatcher->dispatchEvent(&customEndEvent);  } |
| //SpriteScene2.h  #pragma once  #include"cocos2d.h"  //#include"ui/CocosGUI.h"  USING\_NS\_CC;  //using namespace ui;  #include<string>  using namespace std;  class SpriteScene:public Scene  {  public:  static cocos2d::Scene\* createScene();  virtual bool init();  //menu callback  void menuCloseCallback(cocos2d::Ref\* sender);  CREATE\_FUNC(SpriteScene);  private:  Sprite\* m\_polySprite;  float m\_angle;  int m\_opacity;  float m\_scaleX;  float m\_minScaleX;  bool toLeft;  bool m\_rotate;  bool isMirrored;//是否已经镜像  }; | //SpriteScene2.cpp  #include "SpriteScene2.h"  cocos2d::Scene \* SpriteScene::createScene()  {  return SpriteScene::create();  }  bool SpriteScene::init()  {  if (!Scene::init()) return false;  //获取可视区域大小  auto visibleSize = Director::getInstance()->getVisibleSize();  auto origin = Director::getInstance()->getVisibleOrigin();  //获取原点坐标  //创建关闭按钮  auto closeItem = MenuItemImage::create("CloseNormal.png", "CloseSelected.png", CC\_CALLBACK\_1(SpriteScene::menuCloseCallback, this));  //设置位置  float x = origin.x + visibleSize.width - closeItem->getContentSize().width / 2;  float y = origin.y+ closeItem->getContentSize().height / 2;  closeItem->setPosition(Vec2(x, y));  auto menu = Menu::create(closeItem, NULL);  menu->setPosition(Vec2::ZERO);  this->addChild(menu, 1);  ////创建精灵1.使用图片，会失真  //auto sprite = Sprite::create("sprite/jlgirl.png");  ////缩放  //sprite->setScale(0.5f);  //sprite->setAnchorPoint(Vec2::ANCHOR\_MIDDLE);  //sprite->setPosition(Vec2(visibleSize / 2));  //this->addChild(sprite);  ////创建精灵2.使用图集，需要使用TexturePacker来生成。需要用到一个plist文件  ////a.创建背景  //auto bg = Sprite::create("spritesheet/background.png");  //bg->setAnchorPoint(Vec2::ZERO);  //this->addChild(bg, 0);  ////创建缓存对象,它是一个单例对象  //SpriteFrameCache\* frameCache = SpriteFrameCache::getInstance();  ////加载图集的plist文件  //frameCache->addSpriteFramesWithFile("spritesheet/SpriteSheet.plist");  ////通过一个mountain1精灵帧名称创建精灵  //auto mountain1 = Sprite::createWithSpriteFrameName("mountain1.png");  ////设置锚点  //mountain1->setAnchorPoint(Vec2::ZERO);  ////设置位置  //mountain1->setPosition(Vec2(-200,10));  //this->addChild(mountain1,0);  ////根据精灵帧创建精灵  //SpriteFrame\* heroFrm = frameCache->getSpriteFrameByName("hero1.png");  //auto hero = Sprite::createWithSpriteFrame(heroFrm);  //hero->setPosition(Vec2(visibleSize/2));  //this->addChild(hero,0);  ////创建精灵3.使用贴图，也就是纹理  ////创建纹理缓存对象  //auto textureCache = TextureCache::getInstance();  ////加载纹理  //auto cache = textureCache->addImage("sprite/gamebg1.png");  ////创建精灵  //auto cacheSprite = Sprite::createWithTexture(cache);  ////裁剪贴图,没有必要不要裁剪否则不好看  ////cacheSprite->setTextureRect(Rect(0,0,480,320));  //cacheSprite->setPosition(Vec2(visibleSize / 2));  //this->addChild(cacheSprite);  ////创建精灵4.创建多边形精灵  auto plg = AutoPolygon::generatePolygon("sprite/zdj.png", Rect(0, 0, 256, 198));  m\_polySprite = Sprite::create(plg);  m\_polySprite->setPosition(Vec2(visibleSize / 2));  this->addChild(m\_polySprite);  //精灵控制  //旋转效果,需要使用调度器,还可以使用动作来实现  m\_angle = 0;//注意这里需要将角度定义为成员变量，否则只会旋转一次  //m\_polySprite->setRotation(m\_angle);  //this->schedule([&](float dlt) {  // m\_angle += 1.f;  // if (m\_angle > 360) {  // m\_angle -= 360;  // }  // m\_polySprite->setRotation(m\_angle);  //},0.02f,"schedule");  //实现精灵淡入效果    //m\_opacity = 0;  //m\_polySprite->setOpacity(m\_opacity); //先将透明度设置为0，这样子就不会有一闪的效果  //this->schedule([&](float dlt) {  // m\_opacity += 1;  // if (m\_opacity >=255) {  // //m\_opacity -= 250;//这样子会一直重复  // //如果只需要做一次，可以在这里注销调度器  // this->unschedule("schedule2");  // return;  // }  // m\_polySprite->setOpacity(m\_opacity);  //}, 0.02f, "schedule2");  ////实现精灵水平旋转的效果  //m\_scaleX = m\_polySprite->getScaleX();  ////float tmp = m\_scaleX;  //m\_minScaleX = - m\_polySprite->getScaleX();  //this->schedule([&](float dlt) {  // //翻转了就注销调度器  // /\*if (m\_scaleX < m\_minScaleX) {  // this->unschedule("schedule3");  // return;  // }\*/  // //重新设置缩放系数，这样子它就可以一直水平旋转下去  // if (m\_scaleX < m\_minScaleX) {  // m\_scaleX = -m\_minScaleX;  // }  // m\_scaleX -= 0.01f;  // m\_polySprite->setScaleX(m\_scaleX);  //},0.02f,"schedule3");    //实现精灵在屏幕在自动移动的效果  m\_polySprite->setScale(0.4f);  //m\_position = m\_polySprite->getPosition();  //Vec2 tmpPos = m\_position;  //m\_polySprite->setPosition(m\_position);    /\*this->schedule([&](float dlt) {    if (m\_polySprite-> getPositionX()+ m\_polySprite->getContentSize().width/2 \* m\_polySprite-> getScale()> Director::getInstance()->getVisibleSize().width  || m\_polySprite->getPositionX() - m\_polySprite->getContentSize().width / 2 \* m\_polySprite->getScale()<0)  {  this->unschedule("schedule4");  return;  }  m\_polySprite->setPositionX(m\_polySprite->getPositionX() - 1.f);  }, 0.02f, "schedule4");\*/    //先往右移动到边界就往左移动，移动到左边界就旋转360°然后淡出  m\_minScaleX = -m\_polySprite->getScaleX();  m\_opacity = m\_polySprite->getOpacity();  m\_rotate = false;  toLeft = false;  isMirrored = false;  this->schedule([&](float dlt) {    if (!toLeft ) {    if (m\_polySprite->getPositionX() + m\_polySprite->getContentSize().width / 2 \* m\_polySprite->getScale()> Director::getInstance()->getVisibleSize().width)  {  toLeft = true;    }  if (!m\_rotate) {  m\_polySprite->setPositionX(m\_polySprite->getPositionX() + 1.f);  }  else  {  if (m\_polySprite->getRotation() >= 360) {  if (m\_polySprite->getOpacity() <= 0) {  //因为在Hello场景在使用了pushScene，这里需要使用popScene切换回去  Director::getInstance()->popScene();  this->unschedule("schedule5");  return;  }  m\_polySprite->setOpacity(m\_polySprite->getOpacity() - 5.f);    }  m\_polySprite->setRotation(m\_polySprite->getRotation() + 1.f);  }  }  else if(toLeft) {    if (m\_polySprite->getPositionX() - m\_polySprite->getContentSize().width / 2 \* m\_polySprite->getScale() < 0) {  toLeft = false;  m\_rotate = true;  }  /\*if (!isMirrored) //这段代码有问题  {  if (m\_polySprite->getScaleX() < m\_minScaleX)  {  isMirrored = true;  m\_polySprite->setScaleX(-m\_minScaleX);  }  m\_polySprite->setScaleX(m\_polySprite->getScaleX() - 0.01f);  }\*/  m\_polySprite->setPositionX(m\_polySprite->getPositionX() - 1.f);  }    }, 0.02f, "schedule5");    return true;  }  void SpriteScene::menuCloseCallback(cocos2d::Ref \* sender)  {  Director::getInstance()->end();  } |
| //AppDelegate.cpp  #include "AppDelegate.h"  #include"SpriteScene2.h"  #include"HelloWorldScene.h"  // #define USE\_AUDIO\_ENGINE 1  // #define USE\_SIMPLE\_AUDIO\_ENGINE 1  #if USE\_AUDIO\_ENGINE && USE\_SIMPLE\_AUDIO\_ENGINE  #error "Don't use AudioEngine and SimpleAudioEngine at the same time. Please just select one in your game!"  #endif  #if USE\_AUDIO\_ENGINE  #include "audio/include/AudioEngine.h"  using namespace cocos2d::experimental;  #elif USE\_SIMPLE\_AUDIO\_ENGINE  #include "audio/include/SimpleAudioEngine.h"  using namespace CocosDenshion;  #endif  USING\_NS\_CC;  static cocos2d::Size designResolutionSize = cocos2d::Size(480, 320);  //static cocos2d::Size designResolutionSize = cocos2d::Size(600, 480);  static cocos2d::Size smallResolutionSize = cocos2d::Size(480, 320);  static cocos2d::Size mediumResolutionSize = cocos2d::Size(1024, 768);  static cocos2d::Size largeResolutionSize = cocos2d::Size(2048, 1536);  AppDelegate::AppDelegate()  {  }  AppDelegate::~AppDelegate()  {  #if USE\_AUDIO\_ENGINE  AudioEngine::end();  #elif USE\_SIMPLE\_AUDIO\_ENGINE  SimpleAudioEngine::end();  #endif  }  // if you want a different context, modify the value of glContextAttrs  // it will affect all platforms  void AppDelegate::initGLContextAttrs()  {  // set OpenGL context attributes: red,green,blue,alpha,depth,stencil  GLContextAttrs glContextAttrs = {8, 8, 8, 8, 24, 8};  GLView::setGLContextAttrs(glContextAttrs);  }  // if you want to use the package manager to install more packages,  // don't modify or remove this function  static int register\_all\_packages()  {  return 0; //flag for packages manager  }  bool AppDelegate::applicationDidFinishLaunching() {  // initialize director  auto director = Director::getInstance();  auto glview = director->getOpenGLView();  if(!glview) {  #if (CC\_TARGET\_PLATFORM == CC\_PLATFORM\_WIN32) || (CC\_TARGET\_PLATFORM == CC\_PLATFORM\_MAC) || (CC\_TARGET\_PLATFORM == CC\_PLATFORM\_LINUX)  glview = GLViewImpl::createWithRect("Hello", cocos2d::Rect(0, 0, designResolutionSize.width, designResolutionSize.height));  #else  glview = GLViewImpl::create("Hello");  #endif  director->setOpenGLView(glview);  }  // turn on display FPS  director->setDisplayStats(true);  // set FPS. the default value is 1.0/60 if you don't call this  director->setAnimationInterval(1.0f / 60);  // Set the design resolution  glview->setDesignResolutionSize(designResolutionSize.width, designResolutionSize.height, ResolutionPolicy::NO\_BORDER);  //glview->setDesignResolutionSize(designResolutionSize.width, designResolutionSize.height, ResolutionPolicy::SHOW\_ALL);  auto frameSize = glview->getFrameSize();  // if the frame's height is larger than the height of medium size.  if (frameSize.height > mediumResolutionSize.height)  {  director->setContentScaleFactor(MIN(largeResolutionSize.height/designResolutionSize.height, largeResolutionSize.width/designResolutionSize.width));  }  // if the frame's height is larger than the height of small size.  else if (frameSize.height > smallResolutionSize.height)  {  director->setContentScaleFactor(MIN(mediumResolutionSize.height/designResolutionSize.height, mediumResolutionSize.width/designResolutionSize.width));  }  // if the frame's height is smaller than the height of medium size.  else  {  director->setContentScaleFactor(MIN(smallResolutionSize.height/designResolutionSize.height, smallResolutionSize.width/designResolutionSize.width));  }  register\_all\_packages();  // create a scene. it's an autorelease object  auto scene = HelloWorld::createScene();  //auto scene = SpriteScene::createScene();  // run  director->runWithScene(scene);  return true;  }  // This function will be called when the app is inactive. Note, when receiving a phone call it is invoked.  void AppDelegate::applicationDidEnterBackground() {  Director::getInstance()->stopAnimation();  #if USE\_AUDIO\_ENGINE  AudioEngine::pauseAll();  #elif USE\_SIMPLE\_AUDIO\_ENGINE  SimpleAudioEngine::getInstance()->pauseBackgroundMusic();  SimpleAudioEngine::getInstance()->pauseAllEffects();  #endif  }  // this function will be called when the app is active again  void AppDelegate::applicationWillEnterForeground() {  Director::getInstance()->startAnimation();  #if USE\_AUDIO\_ENGINE  AudioEngine::resumeAll();  #elif USE\_SIMPLE\_AUDIO\_ENGINE  SimpleAudioEngine::getInstance()->resumeBackgroundMusic();  SimpleAudioEngine::getInstance()->resumeAllEffects();  #endif  } | 效果 |