## 游戏中常见的对精灵的控制

|  |
| --- |
|  |

实例，实现对精灵的控制

|  |
| --- |
|  |

需要资源

|  |
| --- |
|  |

|  |  |
| --- | --- |
| //SpriteScene.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;  }; | //SpriteScene.cpp  #include "SpriteScene.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(0.2f);  //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，每到边界都朝着相反方向移动    this->schedule([&](float dlt) {    if (!toLeft) {  m\_polySprite->setPositionX(m\_polySprite->getPositionX() + 1.f);  if (m\_polySprite->getPositionX() + m\_polySprite->getContentSize().width / 2 \* m\_polySprite->getScale()> Director::getInstance()->getVisibleSize().width)  {  toLeft = true;  }  }  else {  m\_polySprite->setPositionX(m\_polySprite->getPositionX() - 1.f);  if (m\_polySprite->getPositionX() - m\_polySprite->getContentSize().width / 2 \* m\_polySprite->getScale() < 0) {  toLeft = false;  }  }    }, 0.02f, "schedule5");    return true;  }  void SpriteScene::menuCloseCallback(cocos2d::Ref \* sender)  {  Director::getInstance()->end();  } |
| //AppDelegate.cpp  #include "AppDelegate.h"  #include"SpriteScene.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(755, 544);  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 = 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  } |  |