## 1.Animate类

|  |
| --- |
|  |

## 帧动画的简单使用实例

|  |  |
| --- | --- |
| //AnimateScene.h  #pragma once  #include "cocos2d.h"  class AnimateScene : 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(AnimateScene);  private:    }; | #include "AnimateScene.h"  USING\_NS\_CC;  bool AnimateScene::init()  {  if (!Scene::init())return false;  //获取屏幕大小  auto visibleSize = Director::getInstance()->getVisibleSize();  //获取原点坐标  auto origin = Director::getInstance()->getVisibleOrigin();  //设置起始坐标  Vec2 position = Vec2(visibleSize / 2);  //创建关闭按钮  auto closeItem = MenuItemImage::create("CloseNormal.png", "CloseSelected.png", CC\_CALLBACK\_1(AnimateScene::menuCloseCallback, this));  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);  //创建帧动画  //1.加载图集资源  auto cache = SpriteFrameCache::getInstance();  cache->addSpriteFramesWithFile("spritesheet/animation\_knight.plist");  //利用图集资源的第一帧创建精灵  Sprite\* knightSprite = Sprite::createWithSpriteFrameName("1");  //将精灵添加到场景  this->addChild(knightSprite);  //设置精灵位置  knightSprite->setPosition(Vec2(visibleSize / 2));  //创建帧动画  Vector<SpriteFrame\*> images;  for (int i = 1; i <= 6; i++) {  images.pushBack(cache->getSpriteFrameByName(Value(i).asString()));  }  Animation\* anim = Animation::createWithSpriteFrames(images, 1.f / 6);  knightSprite->runAction(RepeatForever::create(Animate::create(anim)));  return true;  }  void AnimateScene::menuCloseCallback(cocos2d::Ref \* pSender)  {  Director::getInstance()->end();  } |
| //AppDelegate.cpp  #include "AppDelegate.h"  #include"AnimateScene.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(960, 540);  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 = AnimateScene::create();  // 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  } |  |

## 2.回调函数包装器

|  |
| --- |
|  |

## 回调函数包装器实例，这里只需要修改AnimateScene.cpp文件

|  |
| --- |
| //AnimateScene.cpp  #include "AnimateScene.h"  USING\_NS\_CC;  bool AnimateScene::init()  {  if (!Scene::init())return false;  //获取屏幕大小  auto visibleSize = Director::getInstance()->getVisibleSize();  //获取原点坐标  auto origin = Director::getInstance()->getVisibleOrigin();  //设置起始坐标  Vec2 position = Vec2(visibleSize / 2);  //创建关闭按钮  auto closeItem = MenuItemImage::create("CloseNormal.png", "CloseSelected.png", CC\_CALLBACK\_1(AnimateScene::menuCloseCallback, this));  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);  //创建帧动画  //1.加载图集资源  auto cache = SpriteFrameCache::getInstance();  cache->addSpriteFramesWithFile("spritesheet/animation\_knight.plist");  //利用图集资源的第一帧创建精灵  Sprite\* knightSprite = Sprite::createWithSpriteFrameName("1");  //将精灵添加到场景  this->addChild(knightSprite);  //设置精灵位置  knightSprite->setPosition(Vec2(visibleSize / 2));  //创建帧动画  Vector<SpriteFrame\*> images;  for (int i = 1; i <= 6; i++) {  images.pushBack(cache->getSpriteFrameByName(Value(i).asString()));  }  Animation\* anim = Animation::createWithSpriteFrames(images, 1.f / 6);  Animate\* animate = Animate::create(anim);  //knightSprite->runAction(RepeatForever::create(animate));  //回调函数包装器，特点是以动作的方式调用,实现在动画播放3次后移除精灵节点  knightSprite->runAction(  Sequence::createWithTwoActions(  Repeat::create(animate,3),  CallFunc::create([knightSprite]() {  knightSprite->removeFromParent();  })  ));  return true;  }  void AnimateScene::menuCloseCallback(cocos2d::Ref \* pSender)  {  Director::getInstance()->end();  } |