## 1.Action的引出

|  |
| --- |
|  |

## 2.Action类的继承树

|  |
| --- |
|  |

## 3.cocos2dx常见的Action

|  |
| --- |
|  |
|  |

## 4.Action的使用

|  |
| --- |
|  |

**注意moveTo动作是不能反转的。只有moveBy动作才可以**

## 5.Action的简单使用实例

|  |  |
| --- | --- |
| //ActionScene.h    #pragma once  #include "cocos2d.h"  class ActionScene : 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(ActionScene);  private:  //添加一个点击计数器  int m\_counter;  }; | //ActionScene.cpp  #include "ActionScene.h"  USING\_NS\_CC;  bool ActionScene::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(ActionScene::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);  //Action!!!  //创建一个动作实例  //相当于当前位置移动的位移  Action \* moveByAction = MoveBy::create(2.f, Vec2(100.f, 0));  //移动的终点坐标  //Action \* moveToAction = MoveTo::create(2.f, Vec2(100.f, position.y));  //moveTo实现moveBy的效果  Action \* moveToAction = MoveTo::create(2.f, Vec2(position.x + 100.f, position.y)); //在实际开发中尽量减少计算的次数，使用不用计算的方法可以提高效率  //创建用于移动的精灵  //auto polygon= AutoPolygon::generatePolygon("sprite/jl4cl.png");  auto polygon = AutoPolygon::generatePolygon("sprite/jl4cl.png",Rect(0,0,187,279));  auto polySprite = Sprite::create(polygon);  polySprite->setPosition(position);  this->addChild(polySprite);  //polySprite->runAction(moveByAction);  //将action翻转  polySprite->runAction(moveByAction->reverse());  //polySprite->runAction(moveToAction);  return true;  }  void ActionScene::menuCloseCallback(cocos2d::Ref \* pSender)  {  Director::getInstance()->end();  } |
| //AppDelegate.cpp  #include "AppDelegate.h"  //#include"SpriteScene2.h"  #include"HelloWorldScene.h"  //#include"ParallaxScene.h"  #include "ActionScene.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 = HelloWorld::createScene();  auto scene = ActionScene::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  } |  |

## 6.重复执行动作Repeat，需要FiniteTimeAction对象指针，也是用moveBy::create()方法来创建

实例：只需要修改一下ActionScene.cpp

|  |
| --- |
| // ActionScene.cpp  #include "ActionScene.h"  USING\_NS\_CC;  bool ActionScene::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(ActionScene::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);  //Action!!!  //创建一个动作实例  //相当于当前位置移动的位移  Action \* moveByAction = MoveBy::create(2.f, Vec2(100.f, 0));  //移动的终点坐标  //Action \* moveToAction = MoveTo::create(2.f, Vec2(100.f, position.y));  //moveTo实现moveBy的效果  Action \* moveToAction = MoveTo::create(2.f, Vec2(position.x + 100.f, position.y)); //在实际开发中尽量减少计算的次数，使用不用计算的方法可以提高效率  //创建用于移动的精灵  //auto polygon= AutoPolygon::generatePolygon("sprite/jl4cl.png");  auto polygon = AutoPolygon::generatePolygon("sprite/jl4cl.png",Rect(0,0,187,279));  auto polySprite = Sprite::create(polygon);  polySprite->setPosition(position);  this->addChild(polySprite);  //polySprite->runAction(moveByAction);  //将action翻转  //polySprite->runAction(moveByAction->reverse());  //polySprite->runAction(moveToAction);  //Repeat ,需要FiniteTimeAction对象指针或者ActionInterval类型的对象指针  //ActionInterval\* ftimeAction = MoveBy::create(2.f, Vec2(100.f, 0));  FiniteTimeAction\* ftimeAction = MoveBy::create(2.f, Vec2(100.f, 0));  //polySprite->runAction(Repeat::create(ftimeAction,1));  //RepeatForever，需要ActionInterval类型的对象指针,也可以从FiniteTimeAction对象指针转换而来  FiniteTimeAction\* foreverAction = RotateBy::create(0.02f, Vec3(0,5,0));  polySprite->runAction(RepeatForever::create(dynamic\_cast<ActionInterval\*>(foreverAction)));  return true;  }  void ActionScene::menuCloseCallback(cocos2d::Ref \* pSender)  {  Director::getInstance()->end();  } |

## 7.多个动作的执行—动作系列

|  |
| --- |
| **这个DelayTime可以实现一个修改的延时展现。在实际开发在非常有用！！！** |

## Sequence动作实例，这里只需要修改ActionScene.cpp文件

|  |
| --- |
| //ActionScene.cpp  #include "ActionScene.h"  USING\_NS\_CC;  bool ActionScene::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(ActionScene::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);  //Action!!!  //创建一个动作实例  //相当于当前位置移动的位移  Action \* moveByAction = MoveBy::create(2.f, Vec2(100.f, 0));  //移动的终点坐标  //Action \* moveToAction = MoveTo::create(2.f, Vec2(100.f, position.y));  //moveTo实现moveBy的效果  Action \* moveToAction = MoveTo::create(2.f, Vec2(position.x + 100.f, position.y)); //在实际开发中尽量减少计算的次数，使用不用计算的方法可以提高效率  //创建用于移动的精灵  //auto polygon= AutoPolygon::generatePolygon("sprite/jl4cl.png");  auto polygon = AutoPolygon::generatePolygon("sprite/jl4cl.png",Rect(0,0,187,279));  auto polySprite = Sprite::create(polygon);  polySprite->setPosition(position);  this->addChild(polySprite);  //polySprite->runAction(moveByAction);  //将action翻转  //polySprite->runAction(moveByAction->reverse());  //polySprite->runAction(moveToAction);  //Repeat ,需要FiniteTimeAction对象指针或者ActionInterval类型的对象指针  //ActionInterval\* ftimeAction = MoveBy::create(2.f, Vec2(100.f, 0));  FiniteTimeAction\* ftimeAction = MoveBy::create(2.f, Vec2(100.f, 0));  //polySprite->runAction(Repeat::create(ftimeAction,1));  //RepeatForever，需要ActionInterval类型的对象指针,也可以从FiniteTimeAction对象指针转换而来  FiniteTimeAction\* foreverAction = RotateBy::create(0.02f, Vec3(0,5,0));  //polySprite->runAction(RepeatForever::create(dynamic\_cast<ActionInterval\*>(foreverAction)));  //系列动作,可以用来作为RepeatForever的参数来构建永久动作  ActionInterval\* fadeInAction = FadeIn::create(1.f);  ActionInterval\* fadeOutAction = FadeOut::create(1.f);  //闪烁效果  //polySprite->runAction(RepeatForever::create(Sequence::create(fadeOutAction, fadeInAction,NULL)));  //旋转效果  //polySprite->runAction(RepeatForever::create(Sequence::create(RotateBy::create(0.2f, Vec3(90, 0, 0)), RotateBy::create(0.2f, Vec3(0, 90, 0)), NULL)));  //移动和弹回效果  //polySprite->runAction(RepeatForever::create(Sequence::create(MoveBy::create(2.f, Vec2(150.f, 0)), MoveBy::create(2.f, Vec2(-150.f, 0)), nullptr)));  //不断缩小放大的效果  //polySprite->runAction(RepeatForever::create(Sequence::create(ScaleBy::create(2.f,0.1f), ScaleBy::create(2.f, 10.f), nullptr)));  polySprite->runAction(RepeatForever::create(Sequence::createWithTwoActions(ScaleBy::create(2.f, 0.1f), ScaleBy::create(2.f, 10.f))));  return true;  }  void ActionScene::menuCloseCallback(cocos2d::Ref \* pSender)  {  Director::getInstance()->end();  } |

## 8.多个动作同时执行Spawn

|  |
| --- |
|  |

## Spawn动作实例，这里只需要修改ActionScene.cpp文件

|  |
| --- |
| #include "ActionScene.h"  USING\_NS\_CC;  bool ActionScene::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(ActionScene::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);  //Action!!!  //创建一个动作实例  //相当于当前位置移动的位移  Action \* moveByAction = MoveBy::create(2.f, Vec2(100.f, 0));  //移动的终点坐标  //Action \* moveToAction = MoveTo::create(2.f, Vec2(100.f, position.y));  //moveTo实现moveBy的效果  Action \* moveToAction = MoveTo::create(2.f, Vec2(position.x + 100.f, position.y)); //在实际开发中尽量减少计算的次数，使用不用计算的方法可以提高效率  //创建用于移动的精灵  //auto polygon= AutoPolygon::generatePolygon("sprite/jl4cl.png");  auto polygon = AutoPolygon::generatePolygon("sprite/jl4cl.png",Rect(0,0,187,279));  auto polySprite = Sprite::create(polygon);  polySprite->setPosition(position);  this->addChild(polySprite);  //polySprite->runAction(moveByAction);  //将action翻转  //polySprite->runAction(moveByAction->reverse());  //polySprite->runAction(moveToAction);  //Repeat ,需要FiniteTimeAction对象指针或者ActionInterval类型的对象指针  //ActionInterval\* ftimeAction = MoveBy::create(2.f, Vec2(100.f, 0));  FiniteTimeAction\* ftimeAction = MoveBy::create(2.f, Vec2(100.f, 0));  //polySprite->runAction(Repeat::create(ftimeAction,1));  //RepeatForever，需要ActionInterval类型的对象指针,也可以从FiniteTimeAction对象指针转换而来  FiniteTimeAction\* foreverAction = RotateBy::create(0.02f, Vec3(0,5,0));  //polySprite->runAction(RepeatForever::create(dynamic\_cast<ActionInterval\*>(foreverAction)));  //系列动作,可以用来作为RepeatForever的参数来构建永久动作  ActionInterval\* fadeInAction = FadeIn::create(1.f);  ActionInterval\* fadeOutAction = FadeOut::create(1.f);  //闪烁效果  //polySprite->runAction(RepeatForever::create(Sequence::create(fadeOutAction, fadeInAction,NULL)));  //旋转效果  //polySprite->runAction(RepeatForever::create(Sequence::create(RotateBy::create(0.2f, Vec3(90, 0, 0)), RotateBy::create(0.2f, Vec3(0, 90, 0)), NULL)));  //移动和弹回效果  //polySprite->runAction(RepeatForever::create(Sequence::create(MoveBy::create(2.f, Vec2(150.f, 0)), MoveBy::create(2.f, Vec2(-150.f, 0)), nullptr)));  //不断缩小放大的效果  //polySprite->runAction(RepeatForever::create(Sequence::create(ScaleBy::create(2.f,0.1f), ScaleBy::create(2.f, 10.f), nullptr)));  //polySprite->runAction(RepeatForever::create(Sequence::createWithTwoActions(ScaleBy::create(2.f, 0.1f), ScaleBy::create(2.f, 10.f))));    //多个动作同时执行Spawn  //polySprite->runAction(Spawn::createWithTwoActions(ScaleBy::create(2.f, 0.1f), MoveBy::create(2.f, Vec2(150.f, 0))));    //可以利用Sequence把两个或者两个以上的Spawn动作组合起来，然后再把Sequence创建的结果传递给RepeatForever，可以生成非常复杂的动作  polySprite->runAction(RepeatForever::create(  Sequence::createWithTwoActions(  Spawn::createWithTwoActions(ScaleBy::create(2.f, 0.1f), MoveBy::create(2.f, Vec2(150.f, 0))),  Spawn::createWithTwoActions(ScaleBy::create(2.f, 10.f), MoveBy::create(2.f, Vec2(-150.f, 0))))  ));    return true;  }  void ActionScene::menuCloseCallback(cocos2d::Ref \* pSender)  {  Director::getInstance()->end();  } |