创建方向框需要5幅图片

1.简单使用

|  |  |
| --- | --- |
| //ChcekBoxDemo.h  #pragma once  #include"cocos2d.h"  #include"ui/CocosGUI.h"  USING\_NS\_CC;  using namespace ui;  class CheckBoxDemo:public Scene  {  public:  static cocos2d::Scene\* createScene();  virtual bool init();  //menu callback  void menuCloseCallback(cocos2d::Ref\* sender);  CREATE\_FUNC(CheckBoxDemo);  private:    }; | ////ChcekBoxDemo.cpp  #include "CheckBoxDemo.h"  using namespace ui;  cocos2d::Scene \* CheckBoxDemo::createScene()  {  return CheckBoxDemo::create();  }  bool CheckBoxDemo::init()  {  if (!Scene::init()) return false;  //获取可视区域的大小  auto visibleSize = Director::getInstance()->getVisibleSize();  //获取原点坐标  auto origin = Director::getInstance()->getVisibleOrigin();  //创建关闭按钮  auto closeItem = MenuItemImage::create("CloseNormal.png","CloseSelected.png",CC\_CALLBACK\_1(CheckBoxDemo::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);  //创建复选框  CheckBox\* check = CheckBox::create();  //加载CheckBox需要的图片资源  //1.正常情况下的背景  check->loadTextureBackGround("checkbox/cbx-normal.png");  //2.选中状态下的背景  check->loadTextureBackGroundSelected("checkbox/cbx-press.png");  //3.禁用状态下的背景  check->loadTextureBackGroundDisabled("checkbox/cbx-disable");  //4.勾选状态的图片  check->loadTextureFrontCross("checkbox/cbx-node-active.png");  //5.禁用状态的图片  check->loadTextureFrontCrossDisabled("checkbox/cbx-node-disable.png");  check->setAnchorPoint(Vec2::ANCHOR\_MIDDLE);  check->setPosition(Vec2(visibleSize/2));  this->addChild(check);  check->setSelectedState(true);//设置为选中状态  check->setEnabled(false);  return true;  }  void CheckBoxDemo::menuCloseCallback(cocos2d::Ref \* sender)  {  Director::getInstance()->end();  } |
| //AppDelegate.cpp  #include "AppDelegate.h"  #include"CheckBoxDemo.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 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);  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();  auto scene = CheckBoxDemo::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  } | //   |  |  |  | | --- | --- | --- | |  |  | | |  | | |

2.为CheckBox添加事件监听

|  |  |
| --- | --- |
| //ChcekBoxDemo.h  #pragma once  #include"cocos2d.h"  #include"ui/CocosGUI.h"  USING\_NS\_CC;  using namespace ui;  class CheckBoxDemo:public Scene  {  public:  static cocos2d::Scene\* createScene();  virtual bool init();  //menu callback  void menuCloseCallback(cocos2d::Ref\* sender);  CREATE\_FUNC(CheckBoxDemo);  private:    }; | //CheckBoxDemo.cpp  #include "CheckBoxDemo.h"  using namespace ui;  cocos2d::Scene \* CheckBoxDemo::createScene()  {  return CheckBoxDemo::create();  }  bool CheckBoxDemo::init()  {  if (!Scene::init()) return false;  //获取可视区域的大小  auto visibleSize = Director::getInstance()->getVisibleSize();  //获取原点坐标  auto origin = Director::getInstance()->getVisibleOrigin();  //创建关闭按钮  auto closeItem = MenuItemImage::create("CloseNormal.png", "CloseSelected.png", CC\_CALLBACK\_1(CheckBoxDemo::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);  //创建复选框  CheckBox\* check = CheckBox::create();  //加载CheckBox需要的图片资源  //1.正常情况下的背景  check->loadTextureBackGround("checkbox/cbx-normal.png");  //2.选中状态下的背景  check->loadTextureBackGroundSelected("checkbox/cbx-press.png");  //3.禁用状态下的背景  check->loadTextureBackGroundDisabled("checkbox/cbx-disable");  //4.勾选状态的图片  check->loadTextureFrontCross("checkbox/cbx-node-active.png");  //5.禁用状态的图片  check->loadTextureFrontCrossDisabled("checkbox/cbx-node-disable.png");  check->setAnchorPoint(Vec2::ANCHOR\_MIDDLE);  check->setPosition(Vec2(visibleSize/2));  this->addChild(check);  //check->setSelectedState(true);//设置为选中状态  //check->setEnabled(false);  //添加事件处理  check->addEventListener([&](Ref\* sender,CheckBox::EventType type) {  if (type == CheckBox::EventType::SELECTED) {  CCLOG("Silence....");  }  else if (type == CheckBox::EventType::UNSELECTED)  {  CCLOG("Playing...");  }  });  return true;  }  void CheckBoxDemo::menuCloseCallback(cocos2d::Ref \* sender)  {  Director::getInstance()->end();  } |

3.实例，性别选择

|  |
| --- |
|  |

用到的资源

|  |
| --- |
|  |

|  |  |
| --- | --- |
| //CheckBoxDemo.h  #pragma once  #include"cocos2d.h"  #include"ui/CocosGUI.h"  USING\_NS\_CC;  using namespace ui;  #include<string>  using namespace std;  class CheckBoxDemo:public Scene  {  public:  static cocos2d::Scene\* createScene();  virtual bool init();  //menu callback  void menuCloseCallback(cocos2d::Ref\* sender);  CREATE\_FUNC(CheckBoxDemo);  private:  CheckBox\* createCheckBox(string pic1, string pic2, string pic3, string pic4, string pic5);  CheckBox\* checkMan,\*checkWoman;  }; | //CheckBoxDemo.cpp  #include "SliderDemo.h"  using namespace ui;  cocos2d::Scene \* SliderDemo::createScene()  {  return SliderDemo::create();  }  bool SliderDemo::init()  {  //调用父类的init  if (!Scene::init()) return false;  //获取显示大小和原点位置  auto visibleSize = Director::getInstance()->getVisibleSize();  auto origin = Director::getInstance()->getVisibleOrigin();  //创建关闭按钮  auto closeItem = MenuItemImage::create("CloseNormal.png","CloseSelected.png",CC\_CALLBACK\_1(SliderDemo::menuCloseCallback,this));  float x = +visibleSize.width - closeItem->getContentSize().width / 2;  float y = origin.y + closeItem->getContentSize().height / 2;  closeItem->setPosition(x, y);    auto menu = Menu::create(closeItem, NULL);  menu->setPosition(Vec2::ZERO); //注意菜单是比较特殊的需要设置在ZERO的位置  this->addChild(menu, 1);  //创建label  currPos = 0;  maxPos = 100;  String\* str = String::createWithFormat("%d/%d", currPos, maxPos);  p\_lbl = Label::createWithSystemFont(str->getCString(), "Arial", 40);  p\_lbl->setAnchorPoint(Vec2::ANCHOR\_MIDDLE);  this->addChild(p\_lbl);  //创建Slider  //1.创建空滑动条  auto slider = Slider::create();  //加载滑动条的背景  slider->loadBarTexture("slider/slider-bg.png");  //加载滑块图片资源  slider->loadSlidBallTextures("slider/sb-normal.png", "slider/sb-pressed.png", "slider/sb-disabled.png");  //加载滑动条图片资源  slider->loadProgressBarTexture("slider/slider-press.png");  //设置滑动条的起始值  if(maxPos>0)  slider->setPercent(currPos\*100/maxPos);  //添加事件监听  slider->addEventListener([&](Ref\* sender, Slider::EventType type) {  Slider\* pslider = static\_cast<Slider\*>(sender);  if (type == Slider::EventType::ON\_PERCENTAGE\_CHANGED) {  int pc = pslider->getPercent();  currPos = pc \* maxPos / 100;  p\_lbl->setString(StringUtils::format("%d/%d",currPos,maxPos));  }  });    this->addChild(slider);  slider->setAnchorPoint(Vec2::ANCHOR\_MIDDLE);  slider->setPosition(Vec2(visibleSize / 2));  //设置Lable的位置  p\_lbl->setPosition(Vec2(slider->getPosition())+Vec2(0,40));  return true;  }  void SliderDemo::menuCloseCallback(cocos2d::Ref \* sender)  {  Director::getInstance()->end();  } |
|  |  |