## 1.声音引擎的引入

|  |
| --- |
|  |

## 2.SimpleAudioEngine支持哪些音频格式？

|  |
| --- |
|  |

**注意，SimpleAudioEngine在windows平台上不支持mp3格式，AudioEngine才支持。而且不能同时使用这两个引擎**

## 3.声音的控制-暂停

|  |
| --- |
|  |

## 4.声音的控制-停止

|  |
| --- |
|  |

## 5.声音控制-恢复播放

|  |
| --- |
|  |

## 实例，音频引擎的简单使用

声音的控制

|  |
| --- |
|  |

|  |  |
| --- | --- |
| //MusicDemoScene.h  #pragma once  #include"ui/CocosGUI.h"  #include "cocos2d.h"  USING\_NS\_CC;  using namespace ui;  class MusicDemoScene : 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(MusicDemoScene);  private:    }; | //MusicDemoScene.cpp  #include "MusicDemoScene.h"  USING\_NS\_CC;  #include "audio/include/SimpleAudioEngine.h"  using namespace CocosDenshion;  bool MusicDemoScene::init()  {  if(!Scene::init())return false;  //获取屏幕大小  auto visibleSize = Director::getInstance()->getVisibleSize();  //获取坐标原点  auto origin = Director::getInstance()->getVisibleOrigin();  //创建关闭按钮  auto closeItem = MenuItemImage::create("CloseNormal.png", "CloseSelected.png", CC\_CALLBACK\_1(MusicDemoScene::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);  //定义一个基准点，位于屏幕的正中心  Vec2 base = Vec2(visibleSize / 2);  //创建4个按钮    //播放  auto playBtn = Button::create();  playBtn->setTitleText("Play");  playBtn->setTitleFontSize(36);  playBtn->setPositionY(base.y + playBtn->getContentSize().height \* 1.5 + 15);  playBtn->setPositionX(base.x);  //添加点击事件  playBtn->addClickEventListener([](Ref\* btn) {  //播放背景音乐  auto audio = SimpleAudioEngine::getInstance();  audio->playBackgroundMusic("sounds/wav/LuckyDay.wav", true);  });  this->addChild(playBtn);  //暂停  auto pauseBtn = Button::create();  pauseBtn->setTitleText("Pause");  pauseBtn->setPositionY(base.y + playBtn->getContentSize().height \* 0.5 + 5);  pauseBtn->setPositionX(base.x);  pauseBtn->setTitleFontSize(36);  pauseBtn->addClickEventListener([](Ref\* btn) {  //暂停播放背景音乐  auto audio = SimpleAudioEngine::getInstance();  audio->pauseBackgroundMusic();  //设置音量  audio->setBackgroundMusicVolume(audio->getBackgroundMusicVolume() \* 1.2f);  });  this->addChild(pauseBtn);  //继续播放  auto resumeBtn = Button::create();  resumeBtn->setTitleText("Resume");  resumeBtn->setPositionY(base.y - playBtn->getContentSize().height \* 0.5 - 5);  resumeBtn->setPositionX(base.x);  resumeBtn->addClickEventListener([](Ref\* btn) {  //继续播放背景音乐  auto audio = SimpleAudioEngine::getInstance();  audio->resumeBackgroundMusic();  });    resumeBtn->setTitleFontSize(36);  this->addChild(resumeBtn);  //停止  auto stopBtn = Button::create();  stopBtn->setTitleText("Stop");  stopBtn->setPositionY(base.y - playBtn->getContentSize().height \* 1.5 - 15);  stopBtn->setPositionX(base.x);  stopBtn->setTitleFontSize(36);  stopBtn->addClickEventListener([](Ref\* btn) {  //停止播放背景音乐  auto audio = SimpleAudioEngine::getInstance();  audio->stopBackgroundMusic();  });  this->addChild(stopBtn);  return true;  }  void MusicDemoScene::menuCloseCallback(cocos2d::Ref \* pSender)  {  Director::getInstance()->end();  } |
| //AppDelegate.cpp  #include "AppDelegate.h"  #include "MusicDemoScene.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, 400);  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 = MusicDemoScene::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  } |  |