复习资料：

1.题型有四种，单项选择题（26分）、填空题（22分）、简答题（32分）和综合题（20分）；

1. 缓存（cache）机制的基本原理是什么？

纹理缓存、精灵缓存、动画缓存、hashmap存储、资源的key，先查找资源是否存在，存在直接返回，否则加载进内存。

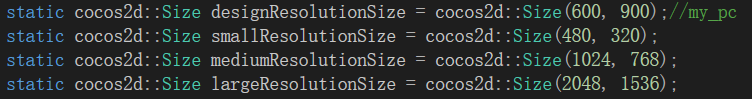
1. Node节点管理：

cocos2d-x使用节点组成一棵树，渲染的时候要遍历这棵树。Node是所有节点类的父类，他内部使用了一个Array对象管理他的所有子节点，当对象被添加为子节点时，实际上是被添加到Array对象中，同时会调用这个对象的retain方法。同理，从Array中移除时，也会调用release方法。

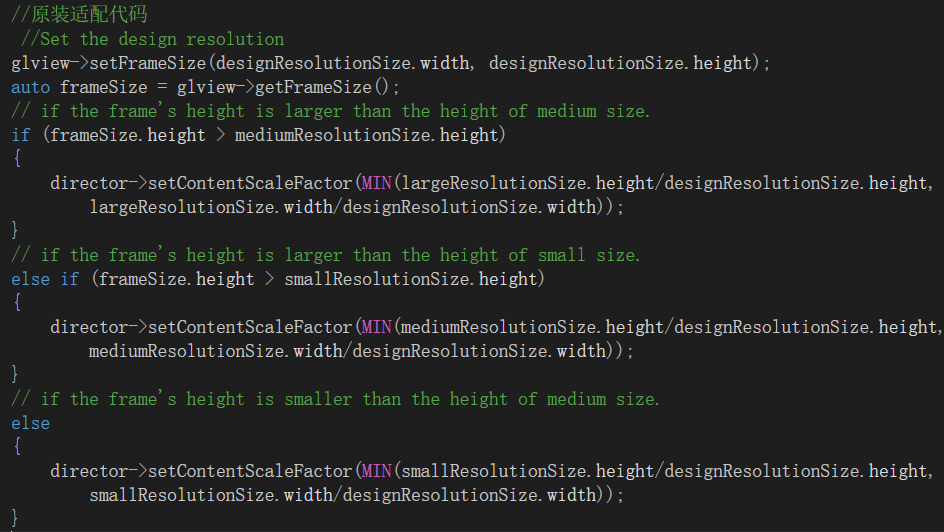
1. 关于数据存储：

关于Cocos2d-x游戏存储，对于简单的游戏可以使用UserDefalt，对于复杂游戏，可以考虑使用sqlite3。

1. cocos2d-x的屏幕适配的解决方案



设置缩放因子



1. 掌握创建工程的方法；

cocos new -l cpp -p com.meloor.test -d dir

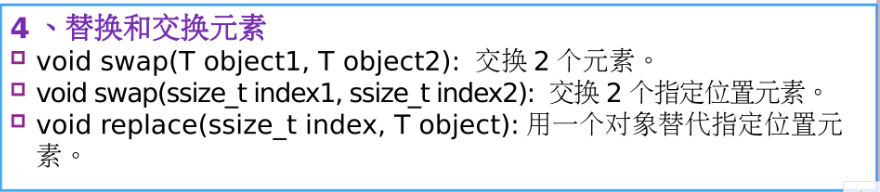
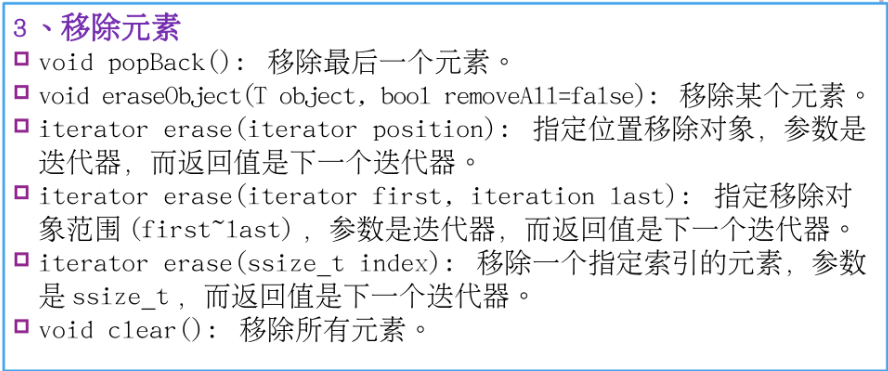
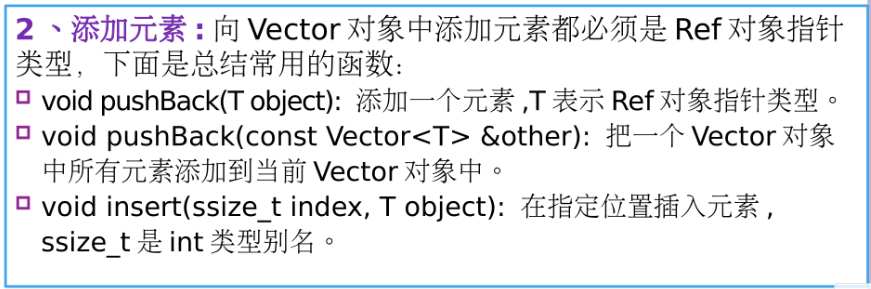
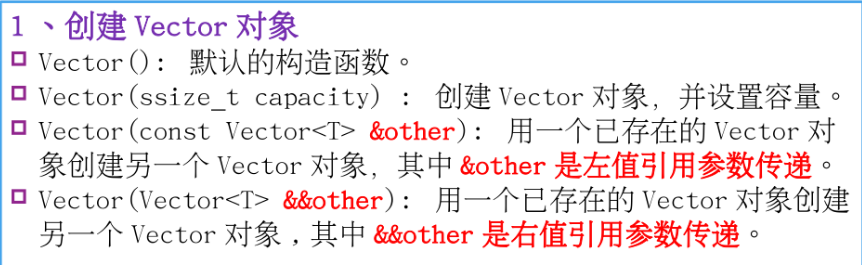
1. 明确在cocos2d-x程序出输出提示信息的方法；

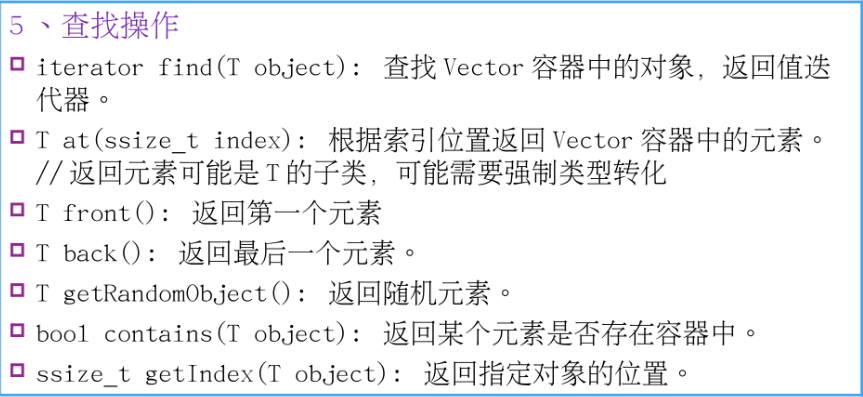
log(); //使用方法类似于printf

log 和 CClog 一个为3.0 一个为2.0

1. Vector的简单使用

Vector<T> vec;







1. 单点和多点触摸事件监听器如何创建。

//单点触摸监听器

listener = EventListenerTouchOneByOne::create();

listener->setSwallowTouches(true);

listener->onTouchBegan = CC\_CALLBACK\_2(HelloWorld::onTBegan, this);

listener->onTouchMoved = CC\_CALLBACK\_2(HelloWorld::onTMoved, this);

listener->onTouchEnded = CC\_CALLBACK\_2(HelloWorld::onTEnded,this);

//创建多点触摸监听器

auto listener = EventListenerTouchAllAtOnce::create();

//绑定事件处理函数

listener->onTouchesBegan=CC\_CALLBACK\_2(HelloWorld::onTouchesBegan,this);

listener->onTouchesMoved=CC\_CALLBACK\_2(HelloWorld::onTouchesMoved,this); listener->onTouchesEnded=CC\_CALLBACK\_2(HelloWorld::onTouchesEnded,this); listener->onTouchesCancelled=CC\_CALLBACK\_2(HelloWorld::onTouchesCancelled,this);

touch->getLocationInView();//获取UI坐标

单点<https://blog.csdn.net/Meloor/article/details/102775900>

多点https://blog.csdn.net/wandersm/article/details/51838697

1. 熟悉Array、Vector、Dictionary、Map<K，V>的基本特征(可能出简答题）

Array,继承于ref类，它所能容纳的是Ref类及其子类所创建的对象指针（**非Ref类不知道可不可以**）

Vector，模仿stl的vector设计的它所能容纳的是Ref类及其子类所创建的对象指针（**非Ref类不知道可不可以**），内存由编译器管理，不需要考虑内存释放的问题。

Dictionary,继承于ref类

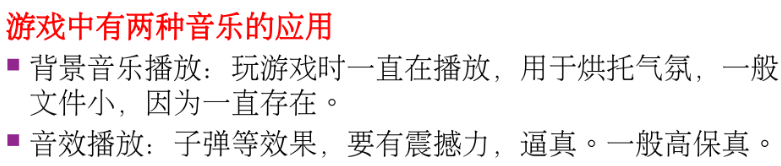
Map<K,V>，与Vector<T>内存管理相似

看实验报告

超星资料：

https://mooc1-2.chaoxing.com/mycourse/studentstudy?chapterId=174439915&courseId=204643543&clazzid=9171854&enc=37f3393a79314c0d2abee6e354d899ae

1. 了解音效和背景音乐的区别以及背景音乐播放和停止的时机



播放和停止时机看实验报告。

背一遍多场景的切换周期：

有过渡动画：

init

onExitTransitionDidStart

onEnter()

onExit()

onEnterTransitionDidFinish

Cleanup()//replace,push没有

没有过渡动画：

Init

onExitTransitionDidStart

onExit

Cleanup

onEnter

onEnterTransitionDidFinish

音乐的播放一般在onEnterTransitionDidFinish函数中

播放另一首音乐时，之前的音乐会自动停止播放，所以音乐一般不停止，要停止应该在cleanup函数中。

根据多场景切换的生命周期，并且经测试，**如果切换场景使用的是ReplaceScene函数，那么同样会出现下一个场景的音乐播放几秒就消失的情况**。

所以最终得出结论，**不要在场景切换时停止音乐**，可以设计音乐开关菜单来实现手动的音乐开关。

1. 导演类实例对场景的控制语句

AppDelegate.cpp中的applicationDidFinishLaunching函数中

director->runWithScene(scene);

//relpace,push,pop

Scene6::createScene();

auto reScene = TransitionFade::create(1.0f, scene);

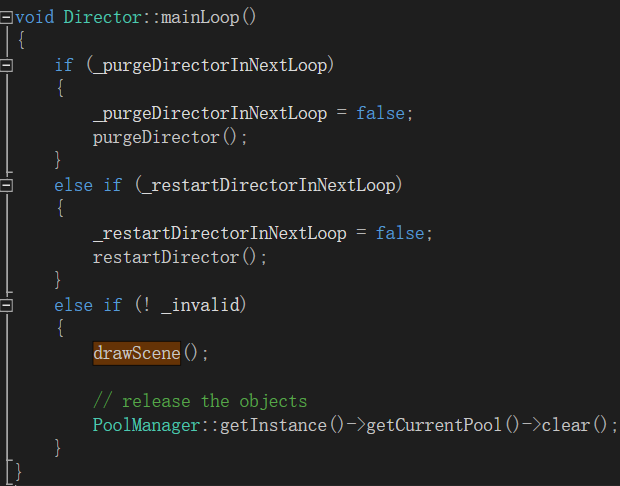
Director::getInstance()->pushScene(reScene);

//Director::getInstance()->->replaceScene(reScene);

//Director::getInstance()->popScene();

//Director::getInstance()->popToRootScene

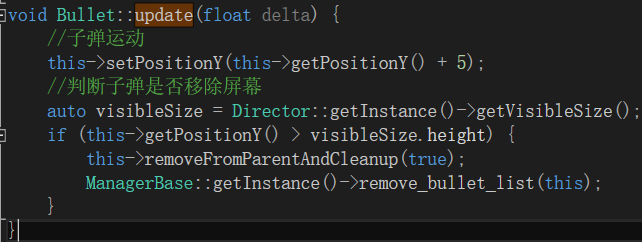
drawScene()



1. 掌握实时更新函数的开启和关闭,包括实时更新函数及其作用。（包括自定义的吗）

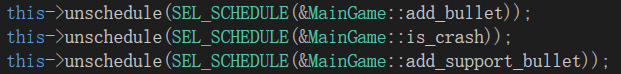
scheduleUpdate();//onEnter中开启

unscheduleUpdate();//onExit中关闭



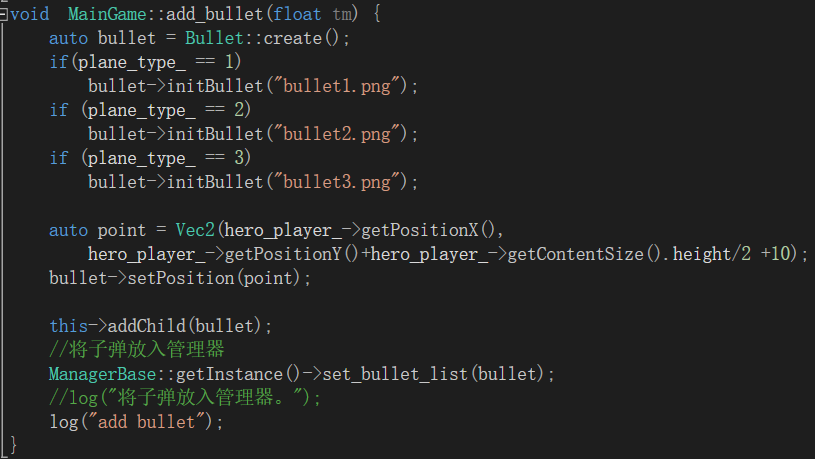
自定义的

开启



关闭





1. cocos2d的坐标系，相对和绝对，掌握父物体对子物体的影响

绝对坐标-世界坐标，节点相对于场景scene的坐标

相对坐标-模型坐标（本地坐标），节点相对于场景中另外一个节点（一般是其父节点）的坐标

1. 锚点和坐标的相对关系

**看实验报告**

关于Node的默认锚点

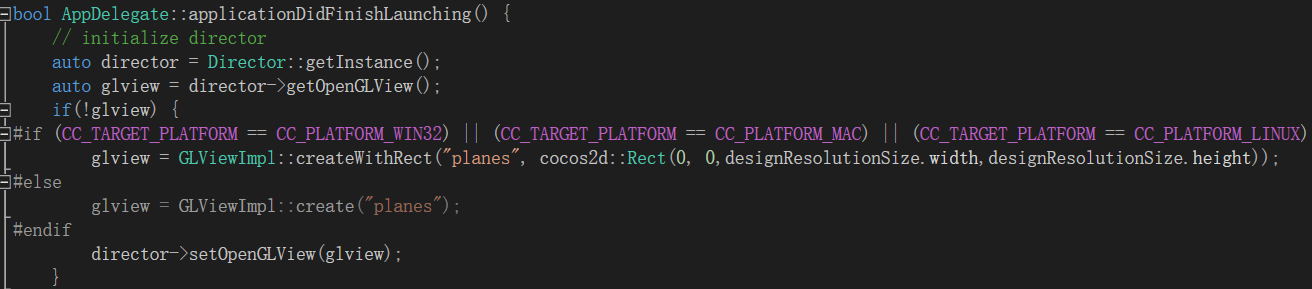
Node的默认锚点为Point(0.5,0.5),即中点

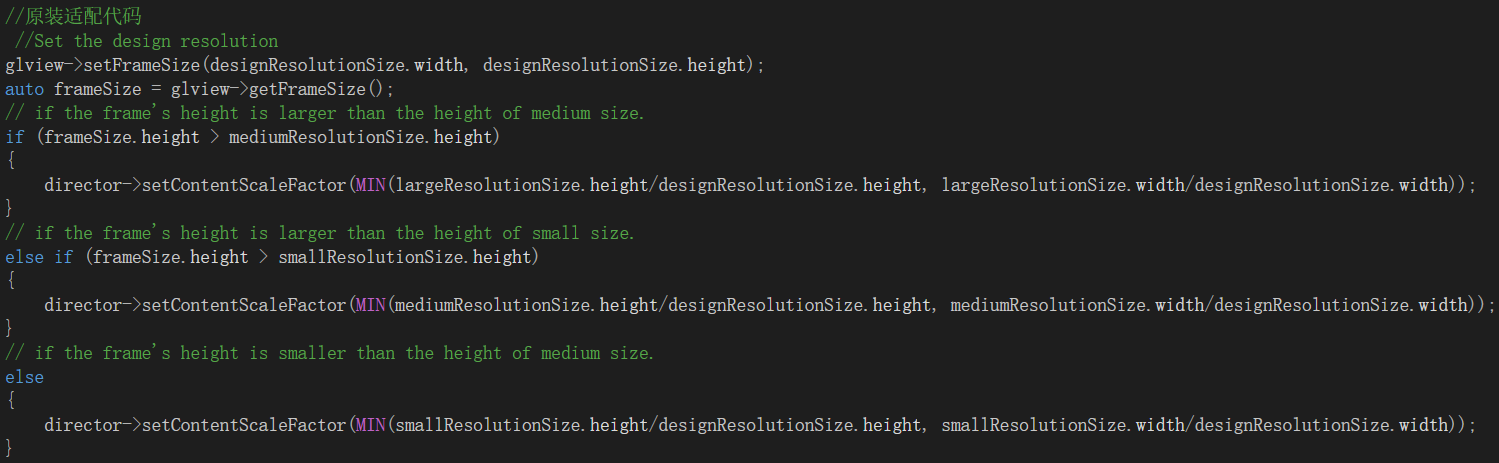
但是Layer和Scene例外 Layer和Scene的默认锚点是Point(0,0),即左下角，且不能改变

无论父控件移动到哪里，子控件的getPosition和setPosition都是指相对于父控件**锚点**的位置。

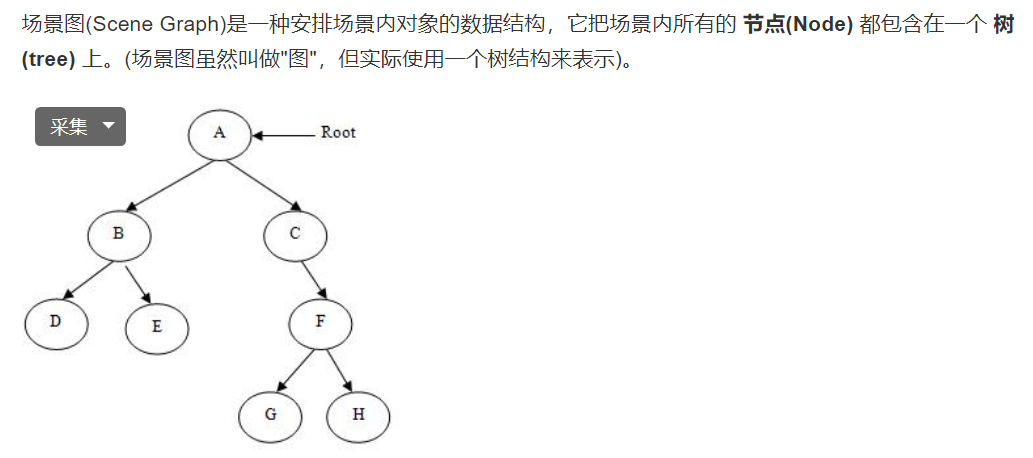
1. 获取窗口大小的方法及对窗口大小的控制

auto visibleSize = Director::getInstance()->getVisibleSize();





1. 掌握精灵的显示层次和坐标系的关系

****z-order**** 

另一点要考虑的是，****z-order**** 为负的元素，z-order 为负的节点会被放置在左子树，非负的节点会被放在右子树。实际开发的过程中，你可以按照任意顺序添加对象，他们会按照你指定的 z-order 自动排序。

渲染时 z-order 值大的节点对象会后绘制，值小的节点对象先绘制。如果两个节点对象的绘制范围有重叠，z-order 值大的可能会覆盖 z-order 值小的。

Z-order和setPosition的关系

即PositionZ的值即为opengl的z值VertexZ。同样节点的PositionZ也是决定了该节点的渲染顺序，值越大，但是与zOrder不同的区别在于，PositionZ是全局渲染顺序即在根节点上的渲染顺序，而zOrder则是局部渲染顺序，即该节点在其父节点上的渲染顺序，与Node的层级有关。  
  
我的理解:  
我们开发中真正用到的也就是PositionZ，而PositionZ等效于VertexZ。PositionZ是什么呢，其实PositionZ就是三维坐标中的Z轴，而cocos2dx的坐标系和OpenGl、笛卡尔坐标系是一致的，简而言之前两种都是笛卡尔坐标系。也就是说我们以屏幕左下角为原点。  
向右为X轴正向，向上为Y轴正向。所以Z轴就是向我们对着屏幕的这边，也就是所我们设置Z轴，其实可以想象成立体的空间，在这个空间内，节点的是有顺序的，Z值越高，我们看过去越能先看到，因为我们所看的屏幕相当于俯视图。对把屏幕想成一张俯视图。  
所以Z轴值越大，就越先看到，越能盖住它下面的东西，因为我们看的是俯视图。有趣吧!.  
那么OrderZ又是什么呢，OrderZ只是设置，在父亲节点下，他们的绘制顺序。但是PositionZ是决定最终呈现在屏幕面前的视觉顺序。

（要清楚getPosition和setPosition读取修改的是节点在其父坐标系）

根据岳丽莎的观点，

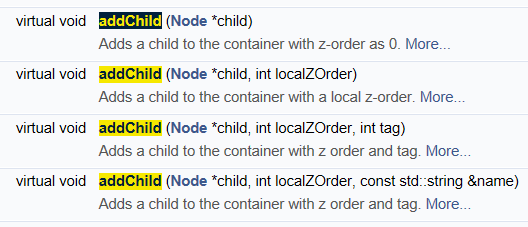
其z-order和positionZ是不一样的，显示时positionZ起决定性因素，相同才比较z-order。positionZ有点类似于opengl中的深度缓存，即使在后面绘制，其深度更深的话一样会被深度浅的遮挡。深度相同时，会出现覆盖的情况，这时要比较z-order，z-order值小的先绘制，后面绘制的会覆盖前面绘制的。

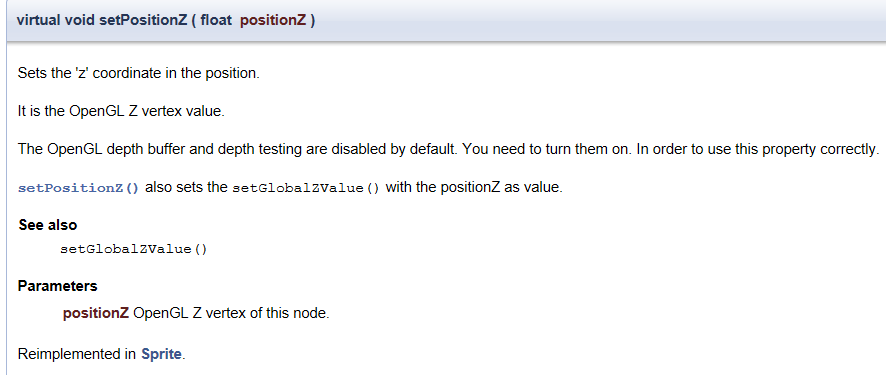
牛人博客http://www.cppblog.com/sixleaves/archive/2015/04/06/210254.aspx

官方文档：

https://docs.cocos.com/cocos2d-x/manual/zh/basic\_concepts/scene.html

1. 熟悉节点加入到节点树的几种重载函数及其参数的含义，并能与setPositionZ的作用进行区分。



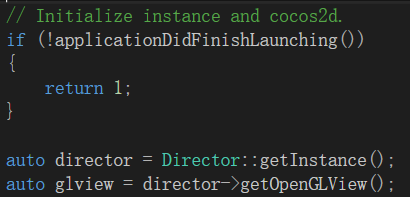


1. 游戏入口

首先在main函数中调用Application类的run函数

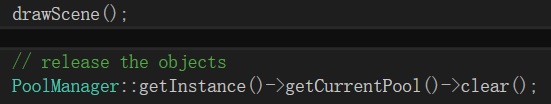


Application类的run函数中先调用AppDelegate类的applicationDidFinishLaunching函数后面循环调用了导演的mainLoop函数

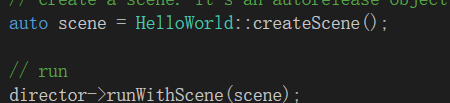




MainLoop函数中调用了drawScene函数



AppDelegate类调用了导演的runWithScene函数，进入到初始场景

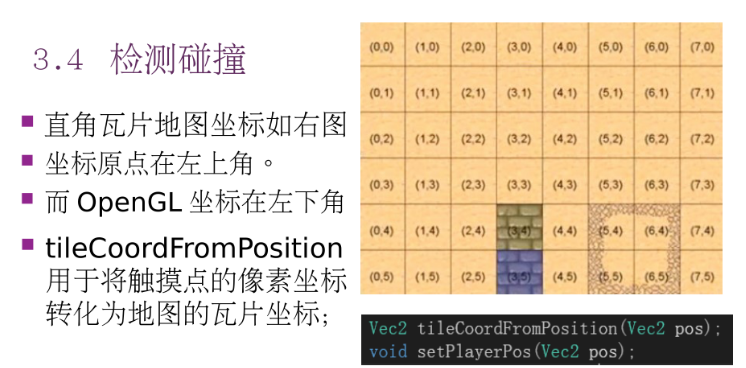


1. 瓦片地图的编辑工具、瓦片地图的坐标系的特点以及简单使用瓦片地图的方法(认识核心函数，知道怎么用编辑好的地图)；

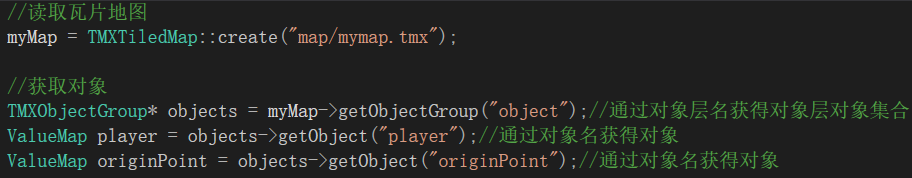
看实验报告九

瓦片地图的坐标系的特点

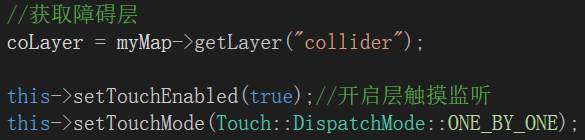
原点在左上角，并且是以格子为单位。



瓦片地图的使用





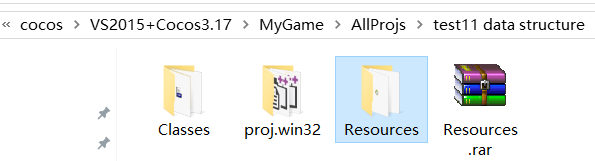


1. 新建文件和资源文件的存放位置

新建文件（讲的可能也是Resources）

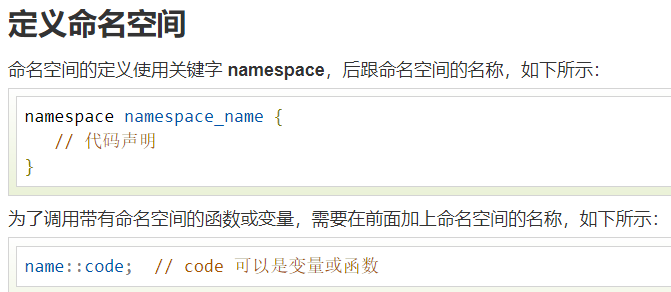
新建什么文件，源码文件一般放在Classes目录下，位置也是可以自己设置的，这个有唯一答案吗？？？

资源文件



1. 了解命名空间的含义及用法

它可作为附加信息来区分不同库中相同名称的函数、类、变量等。使用了命名空间即定义了上下文。本质上，命名空间就是定义了一个范围。





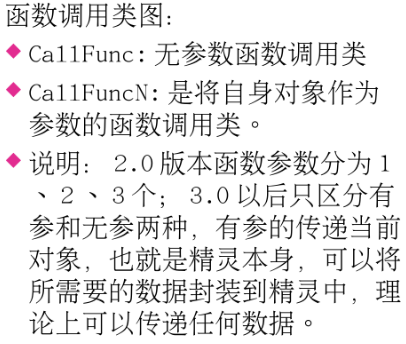
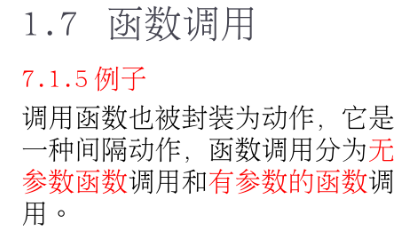
菜鸟教程https://www.runoob.com/cplusplus/cpp-namespaces.html

1. 掌握对精灵位移、缩放、旋转的控制方法

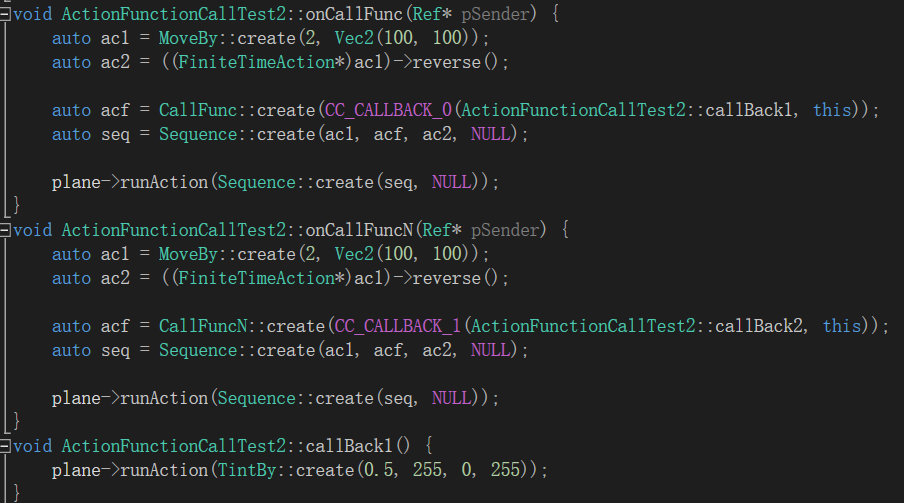
Sprite->SetPositon,setScale,setRotate

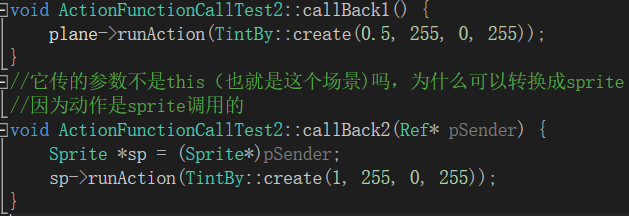
还可以使用动作MoveTo, MoveBy, ScaleTo, ScaleBy, RotateTo, RotateBy 

1. CallFunc和CallFuncN的区别。







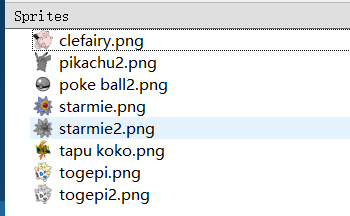


1. 场景的图片组合工具。以及如何使用plist文件来创建帧动画进行动画播放。

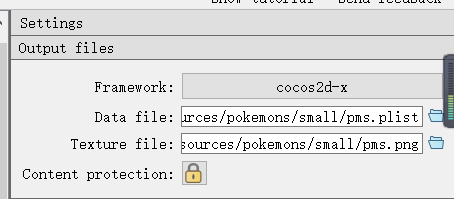
Texture packager

用一下就会了

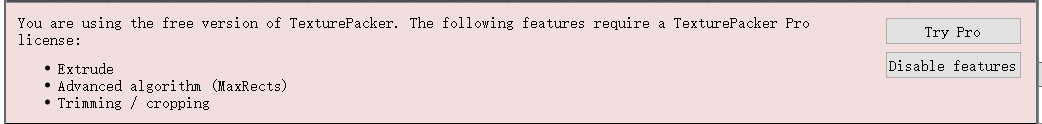
拖入图片到这

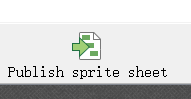


设置输出格式

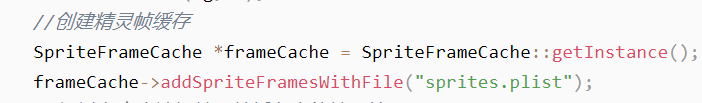


弹出这个框选择disable features

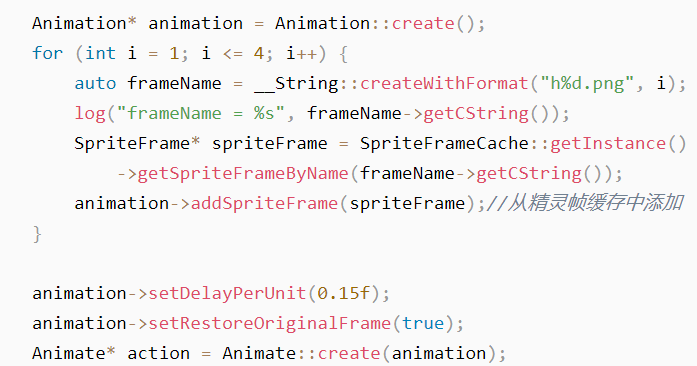
导出



创建精灵帧缓存

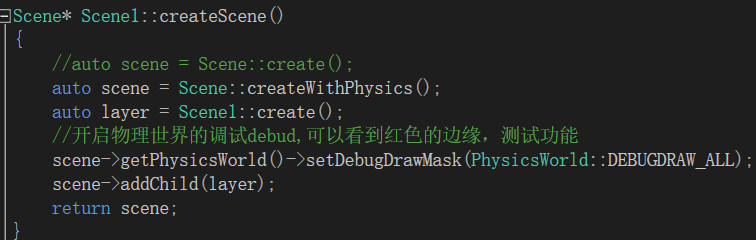


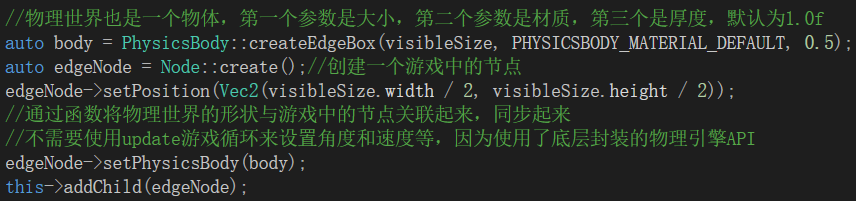
用精灵帧缓存创建帧动画



https://blog.csdn.net/Meloor/article/details/103117176

1. 明确chipmunk中创建物理世界的方法

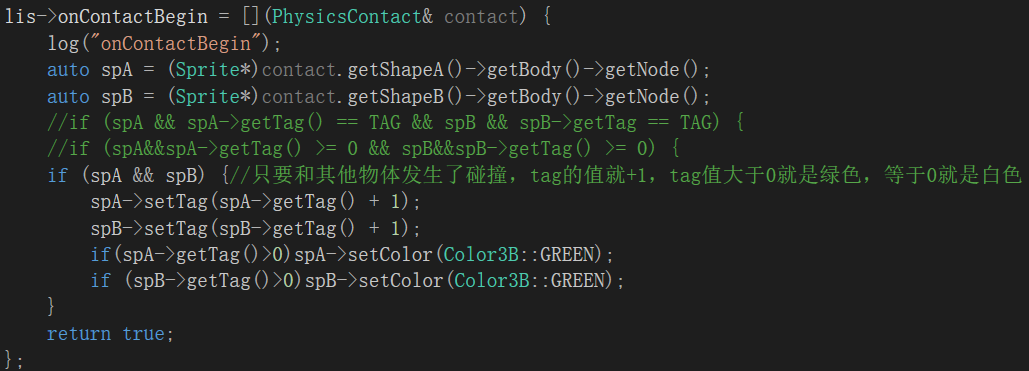


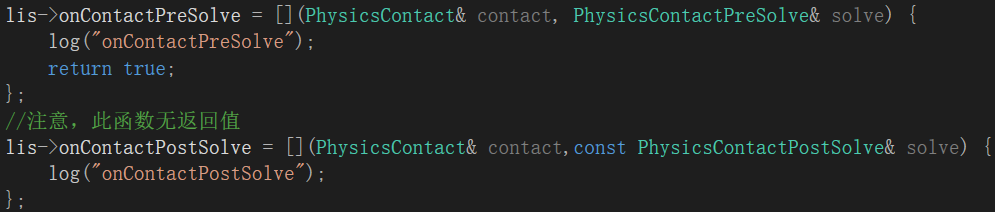


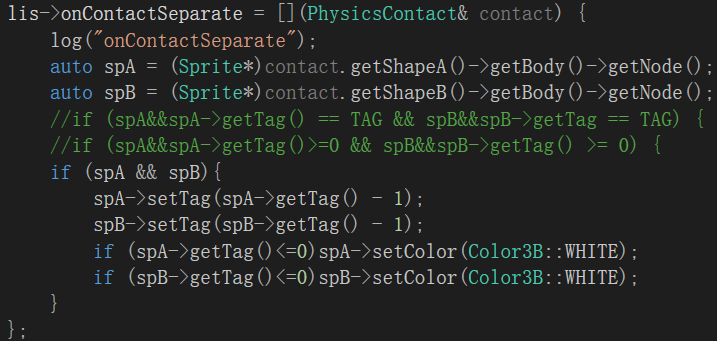
1. 了解chipmunk物理引擎碰撞监听器的创建





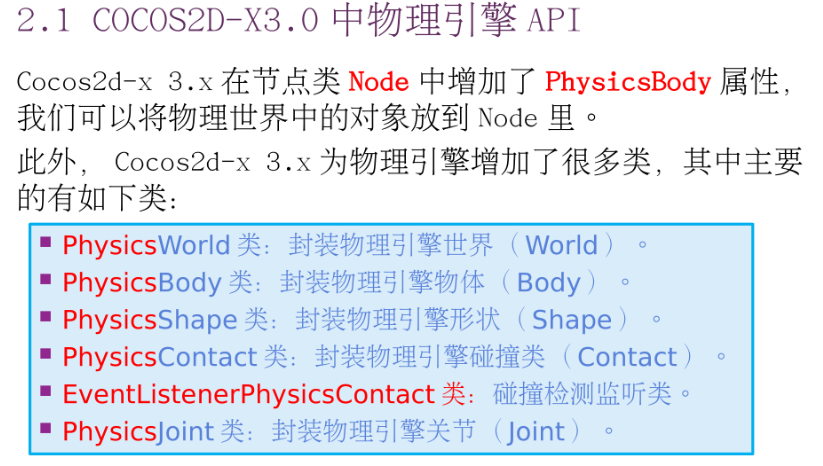




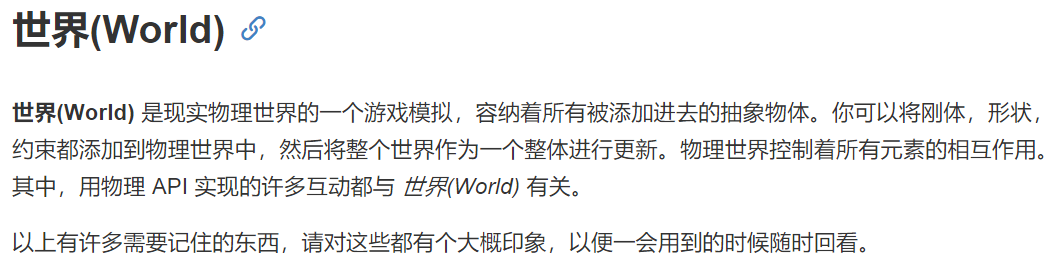




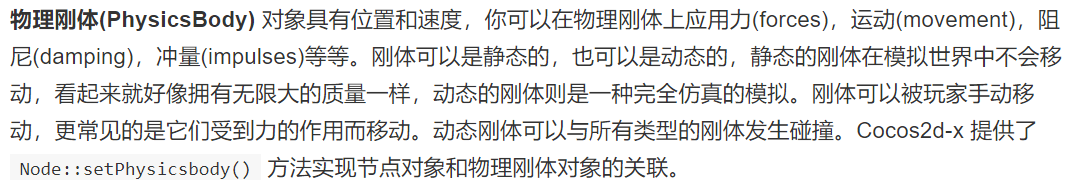
1. 物理引擎中的世界、物体、形状、接触点、关节的含义及其相互作用是什么？



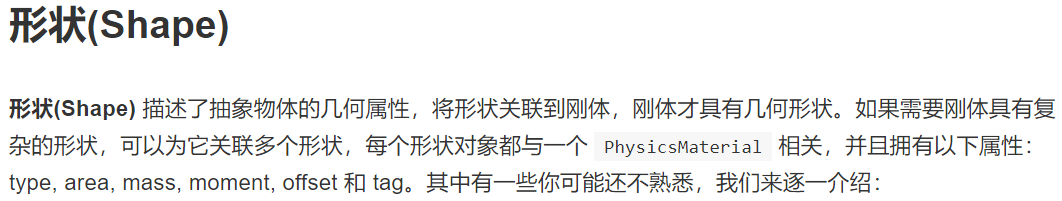
世界



物体

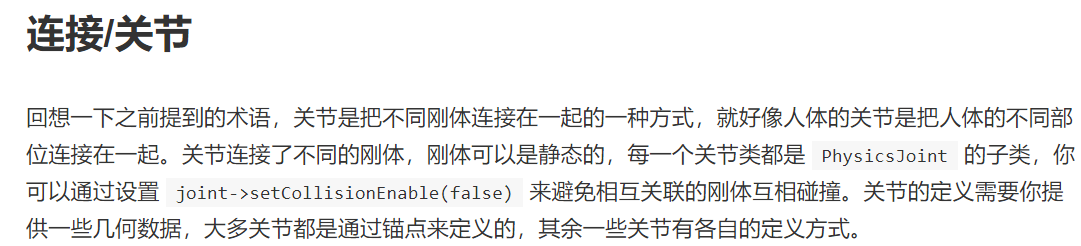


形状



接触点(指的是碰撞吧)

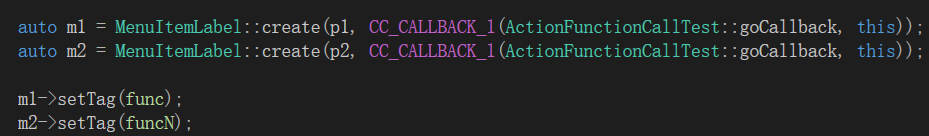
关节及其相互作用

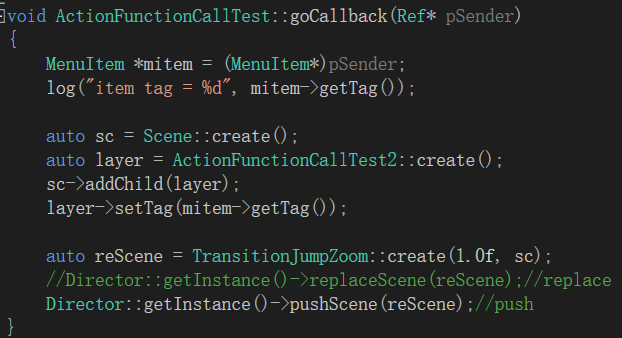


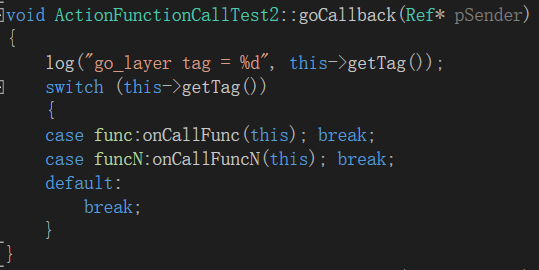
官方文档https://docs.cocos.com/cocos2d-x/manual/zh/physics/concepts.html

1. 掌握两个场景间参数传递的方法；

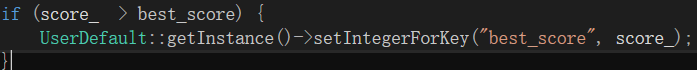
看实验五,为层设置Tag







Userdefault





1. 掌握cocos2d中内存管理机制中引用计数机制的核心内容（retain和release、autorelease、create、 cache图片缓存、addchild）

看实验报告

1. autorelease和release的区别

release是立即将Ref对象的引用计数减一

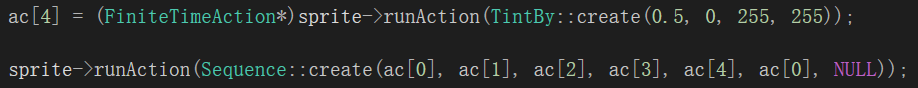
Autorelease是延迟减一，首先会将该Ref对象加入到autoReleasePool,autorReleasePool在消息循环（每一帧）结束时销毁，会对其里面的对象进行release。

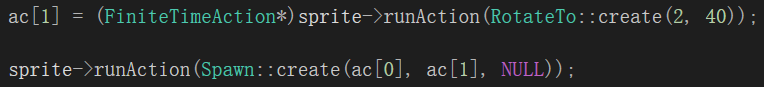
1. 掌握触摸事件（单点和多点）响应函数及事件与对象的绑定，包括触摸点UI坐标的获取和转换。

看第8条

1. 掌握动作序列的简单使用，掌握基本动作语句及其区别

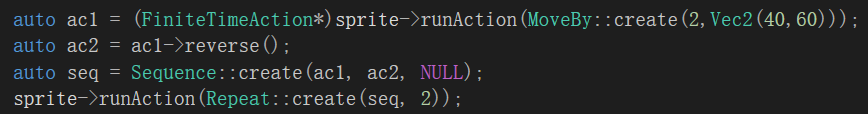
实验五





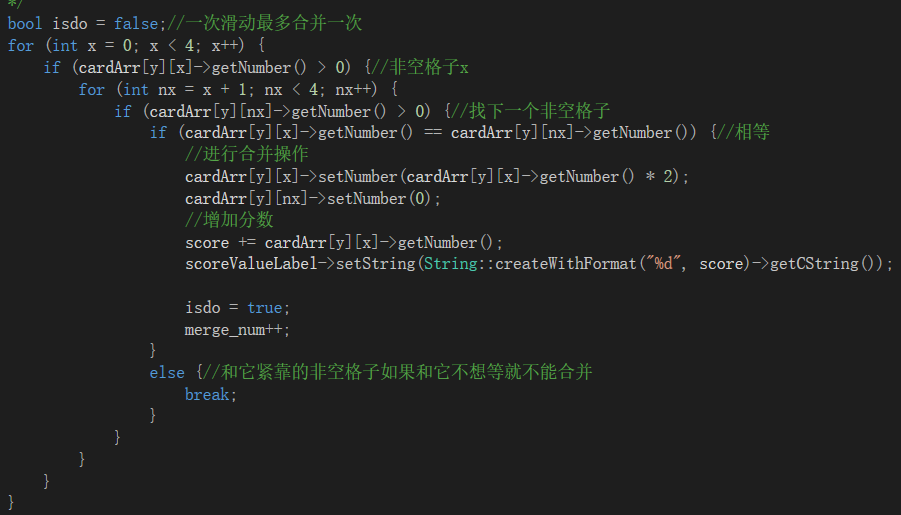


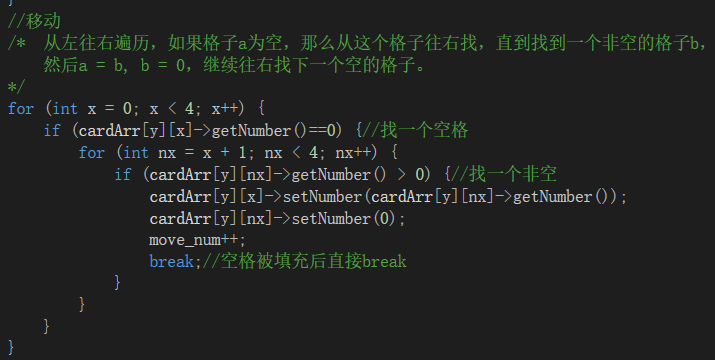




1. 掌握“2048”的核心逻辑及实现（包括代码逻辑），如移动、边界判断、相同数字处理，终止条件等。

先合并再移动





1. 场景切换的生命周期。

https://blog.csdn.net/Meloor/article/details/102468470

1. 掌握创建新类的核心代码及其意义（如何从头创建一个新类，并**纳入到cocos2d-x内存管理机制**中。宏CREATE\_FUN?）

这个宏创建静态create方法，意义就是创建后调用autoRealease，实现引擎管理其内存。

class MyScene: public cocos2d::Scene

{

private:

Sprite\* plane;

public:

CREATE\_FUNC(MyScene);

};

不过我觉得她可能要讲的是如何创建一个继承与Scene,layer,Sprite的类，然后其中要重载的相关函数

1. HelloWorld::createScene()的代码。

auto scene = HelloWorld::create();

auto layer = Layer::create();

scene->addChild(layer);

return scene;

//return HelloWorld::create();

1. 掌握标签按钮的绑定回调函数语句



https://blog.csdn.net/Meloor/article/details/102765820

1. 掌握cocos2dx的基本概念(导演、场景、层等)及关系
2. 掌握精灵的创建

auto sp = Sprite：：create(“sprite.png”);

1. 掌握animation和animate及其使用

看实验五



https://blog.csdn.net/Meloor/article/details/103117176