# Cocos2d-x常见面试题

1、Cocos2d-x是怎样实现跨平台？

　　AppDelegate 作为跨平台程序入口，在这之上做了另一层的封装，封装了不同平台的不同实现。比如我们通常认为一个程序是由 main 函数开始运行，那我们就去找寻，我们看到了在 proj.linux 目录下存在 main.cpp 文件。在main.cpp 中 CCApplication::sharedApplication()–>run(); 这一句看起，这一句标志着， cocos2d-x 程序正式开始运行，现在定位到 sharedApplication() 方法的实现，在CCAplication类中我们可以看到从 sharedApplication() 方法，在调用 run() 方法，在这之前，我们需要调用到它的构造函数，否则不能运行，这就是为什么在 CCApplication::sharedApplication()–>run(); 之前，我们首先有语句 AppDelegate app; 而创建 AppDelegate 变量的原因是 AppDelegate 是 CCApplication 的子类，在创建子类对象的时候，调用其构造函数的同时，父类构造函数也会执行，然后就将 AppDelegate 的对象赋给了 CCApplication 的静态变量，而在 AppDelegate 之中我们实现了 applicationDidFinishLaunching方法，所以在 CCApplication 中 run 方法的开始处调用的就是 AppDelegate 之中的实现。而我们在此方法中我们初始化了一些变量，创建了第一个 CCScene 场景等，之后的控制权，便全权交给了CCDirector::sharedDirector()–>mainLoop(); 方法了。

在cocos2d-x的文件夹下，有一个platform文件夹，里面存放了跨平台的封装接口。当前目录下有 CCApplicationProtocol.h头文件，子目录有win32,Android,IOS三个文件夹，里面分别存放跨平台需要的函数，其中包 括CCApplication。而AppDelegate 类则是继承自CCApplication。CCApplication又继承自

CCApplicationProtocol。在CCApplicationProtocol中定义了applicationDidFinishLaunching虚方法,由CCApplication 继承， AppDelegate 实现的。以此实现了跨平台。

java输入→Jni→c++输入→c++处理（API实现）→c++输出→Jni→java输出

而在Android 平台启动 cocos2d-x程序。可以找到Android 平台与上面等价的入口点，proj.android/jni/hellocpp/main.cpp。在main.cpp文件里面并没有看到 main 函数，这是由于不同的平台封装所以有着不同的实现，在 Android 平台，默认是使用 Java 开发，可以使用 Java 通过 Jni 调用 C++ 程序，而这里也正式如此。我们暂且只需知道，由 Android 启动一个应用，通过各种峰回路转，最终执行到了 Java\_org\_cocos2dx\_lib\_Cocos2dxRenderer\_nativeInit 函数，由此，变开始了我们 cocos2d-x Android 平台的程序入口处。，其它平台程序的入口必然包含着其它平台的不同封装实现 。

参考：http://blog.leafsoar.com/archives/2013/05-05.html

2.cocos2d-x 程序的结束流程？

　　程序运行时期，由 mainLoop 方法维持运行着游戏之内的各个逻辑，当在弹出最后一个场景，或者直接调用 CCDirector::end(); 方法后，触发游戏的清理工作，执行 purgeDirector 方法，从而结束了 CCEGLView（不同平台不同封装，PC使用OpenGl封装，移动终端封装的为 OpenGl ES） 的运行，调用其 end() 方法，从而直接执行 exit(0); 退出程序进程，从而结束了整个程序的运行。（Android 平台的 end() 方法内部通过Jni 方法 terminateProcessJNI(); 调用 Java 实现的功能，其功能一样，直接结束了当前运行的进程）

3、cocos2d-x内存管理？

　　目前主要有两种实现智能管理内存的技术，一种是引用计数，一种是垃圾回收。Cocos2d-x采用的是引用计数机制。为此实现了自己的根类 CCObject，每个对象都包含了一个用来控制生命周期的引用计数器,就是CCObject的成员变量m\_uReference。

　　对于m\_uReference，构造函数创建时对该引用计数器赋值为1（自引用，并没有实际的使用），当需要引用对象时调用 retain()方法增1，当引用结束的时候调用release()方法减一。而autorelease()方法(create工厂方法迫切需要)会将对 象放入自动回收池（CCAutoReleasePool)实现灵活的垃圾回收。当每一帧结束的时候，自动回收池中的对象都会被执行一次release()

autorelease()方法里面代码将该对象添加到自动释放池中：CCPoolManager::sharedPoolManager()->addObject(this);

 create方法将对象加入内存池后,对象的所有权已经属于内存池了, 我们返回的指针其实是没有所有权的.  主循环mainloop干了件非常重要的事情, 那就是pop最上层的autorelease pool, 此时是在release全部仅仅由此内存池所有的对象. 就是依靠这样的原理, 我们可以放心的将对象放在autorelease pool中, 知道在需要的时候, 这个对象就能正确的释放, 同时只要有上层的父节点通过addChild对游戏对象有了所有权以后, 又能正确的保证该对象不会被删除.

如果在一帧之内生成了大量的autorelease对象，将会导致回收池性能下降。因此，在生成autorelease对象密集的区域（通常循环中）的前后，我们最好可以手动创建并释放一个回收池。CCAut0m\_pReleasePoolStack

 CCPoolManager::sharePoolManage()->push()

 for(){}

   CCPoolManager::sharePoolManage()->pop()

CCAutoreleasePool

4、Cocos2d-x中如何处理内存泄露，处理内存泄露有哪些检测工具，如何针对crash后的游戏声称报告发送回服务器

　VS下内存泄露检测工具Visual Leak Detector：教程http://blog.csdn.net/onerain88/article/details/8574938

xcode ios下内存泄露使用Instruments来查找程序中的内存泄露：http://blog.csdn.net/totogo2010/article/details/8233565

3、Cocos2d-x 3版本的特性？

3.0版 1、将有一个新的渲染系统。

　　　 2、支持多线程，并且易于支持新的GPU平台。

    3、更快，更高效也更易于维护的Label文本绘制。

　　  4、拥有一个新的，统一的事件派发器。

　　  5、减少对objectc的兼容考虑，更多考虑对C++开发者更友好。用C++最佳实践，替换掉了objectc模式。移除匈牙利命名法。

　　   6、menu和action可以接受Lambda表达式作为输入

5、C++ 11的特性？

　　　　参考文章：C++11 语法记录    Cocos2dx3.2学习准备（一）：C++11新特性  Cocos2d-x 3.1.1 学习日志6--30分钟了解C++11新特性

6、阐述cocos2d-x 中CCScene CCLayer CCSprite CCNode

  CCNode是CCScene,CCLayer,CCSprite的基类，是一个抽象类，没有可视化的表现形式。是为了方便构造渲染树而定 义的一个类。CCScence是场景类，里面可以放CCLayer和CCSprite。一个app里面可以放多个scence,但是同一时刻只有一个 scence被激活。CCLayer是层类，里面可以放CCSprite。CCSprite是最小的精灵单元。

7、说一下CCAction和CCActionMessager

CCAction是动作的基类，所有的动作都派生自这个类，它创建的一个对象代表一个动作，动作作用于CCNode。主要使用 CCFiniteTimeAction有限次动作执行，就是按时间顺序执行一系列动作，执行完后动作结束；CCFiniteTimeAction 继承自CCAction。CCFiniteTimeAction又分为CCActionInstanse（瞬时动作的基类）和 CCActionInterval（延时动作的基类）。

CCActionInstanse：没什么特别，跟CCActionInterval主要区别是没有执行过程，动作瞬间就执行完成了；CCActionInterval：执行需要一定的时间（或者说一个过程）。我们用的最多的就是延时动作，下面对它进行单独介绍。

     CCActionMessage是管理所有Action的单例，一般情况下并不直接使用这个单例，而是使用CCNode的接口，（CCNiode\*)->runaction()

而在runaction函数里，通过CCActionMessage动作管理类将新的CCAction和对应的目标节点添加到其管理的动作表中。

 m\_pActionManager->addAction(action, this, !m\_bIsRunning); 再调用 action->startWithTarget(pTarget);

CCActionMessage的初始化在CCDirector的初始化里执行。在里面通过CCSchedule定时调度器为CCActionMessage注册了一个定期更新任务。所以每一帧刷新更新时都会触发CCActionMessage的update方法，系统都会遍历动作表中每一个动作，并调用该动作的setup方法。

　但是假如你想操作的目标不是CCNode的子类或者你想暂停/恢复行动就要使用到CCActionMessager

7、你常用的cocos2d-x工具有哪些？

　　TiledMap （地图编辑器）ParticleEditor（粒子编辑器）cocosBuilder（可视化编辑）Texture Packer（图片组合工具） plistEditor工具等

8、简单介绍一下CocoStudio

CocoStudio是一套基于Cocos2D-X引擎的工具集，包括UI编辑器、动画编辑器、场景编辑器和数据编辑器。UI编辑器和动画编辑 器主要面向美术，而场景编辑器和数据编辑器面则面向游戏策划，这四个工具合在一起构成了一套完整的游戏开发体系，帮助开发者进一步降低开发难度、提高开发 效率、减少开发成本。

UI编辑器：支持目前Cocos2D-X的所有控件，同时支持多分辨率适配、碎图合并以及自定义UI。

动画编辑器：支持骨骼关键帧动画、序列帧动画，同时支持碎图合并、Flash动画直接导入等。

数据编辑器：把策划用的Excel数值表分解，然后转化成Cocos2D-X可以识别的格式。

场景编辑器：可以整合我们之前CocoStudio其他编辑器的资源，编辑当前的游戏场景，实时演示，所见即所得。同时也支持第三方的工具模式。

9、简述CCSpriteframeCache   CCSpriteBatchNode，并说出CCNode，CCSprite是如何实现绘制的？

CCSpriteframeCache:精灵框帧缓存，缓存了精灵框帧，主要是为了处理多张碎图合并出来的纹理图片。由于一张大图包含了多张小图所以不适合使用CCTextureCache。

CCSpriteBatchNode 中的所有CCSprite只会被渲染1次，因此可以提高游戏的FPS，但是加入到 CCSpriteBatchNode 中的CCSprite必须使用同一张纹理图。

10、cocos2d-x的屏幕适配解决方案？

11、cocos2d-x游戏储存 CCUserDefault和SQLite

CCUserDefault 是Cocos2d-x引擎提供的持久化方案，存储所有游戏通用的用户配置信息。例如音乐和音效配置。本质是一个XML文件。使用只需一行代码：

CCUserDefault::sharedUserDefault()->setIntegerForKey("coin",coin-1);

缺点：每次设置和读取都会遍历整棵XML树，效率不高，且值类型具有局限性，只支持int和float等基本类型。

 http://blog.163.com/zhoulong19880518@126/blog/static/607097022012985417804/

SQList是一个嵌入式数据库，具有开源，轻量等特点,用于高速且安全地在本地存储数据。核心接口函数只有一个：

12、autorelease和release的区别：

release 会立马对对象进行引用计数减一操作，如果当前对象的引用计数小于0，则会进行释放。

autorelease 则会将该对象放入到自动释放池中，当一帧结束的时候会执行release操作进行引用计数减一操作，如果当前对象的引用计数小于0，则会进行释放。

详细由第三个问题cocos2d-x内存管理可以详细解答。

13、cache机制原理是什么？

　　把新加进内存的资源做一个hashmap存储，每一个资源加一个key。每次加载资源的时候，先查找资源是否存在，存在直接返回，否则加载进内存。

14、减少内存开销的方法有哪些，图片压缩方法有哪些 及时释放，减少泄露，重用资源，延迟加载，分部加载等

　　1）不使用JPG,因为JPG纹理在加载的时候，会实时地转化为PNG格式的纹理。

　　2）预先加载所有的纹理。

　　3）在后台加载纹理CCTextureCache类还支持异步加载资源的功能，利用addImageAsync方法。你可以很方面地给addImageAsync方法添加一个回调方法，这样，当纹理异步加载结束的时候，可以得到通知。

4）如果游戏有很多场景，在切换场景的时候可以把前一个场景的内存全部释放，防止总内存过高. 一般在切换场景的时候释放资源，如果从A场景切换到B场景，调用的函数顺序为 B::init()---->A::exit()---->B::onEnter() 。可如果使用了切换效果，比如CTransitionJumpZoom::transitionWithDuration这样的函数，则函数的调用顺序变 为B::init()---->B::onEnter()---->A::exit() 。

而且第二种方式会有一瞬间将两个场景的资源叠加在一起，如果不采取过度，很可能会因为内存吃紧而崩溃。

5）尽量去拼接图片，使图片边长尽可能的保持2的N次方并且装的很满。但要注意，有逻辑关系的图片尽量打包在一张大图里，另外一点就是打包的时 候要考虑到层的分布。因为为了渲染效率可能会用到CCSpriteBatchNode;同一个BatchNode里的图片都是位于一个层级的，因此必须根 据各个图片的层级关系，打包到不同的plist里。有时内存和效率不可以兼得，只能尽量平衡了。

6）避免一个接一个地加载PNG和JPG纹理（他们之间至少等待一帧），因为等待一帧，引用计数会把临时的UIImage对象释放掉，减少内存压力。

7）使用pvr格式的纹理时，只使用pvr.ccz格式，不要使用其它格式！因为它加载速度超快，而且加载的时候使用更少的内存！

7）最快速地减少纹理内存占用的办法就是把它们作为16位颜色深度的纹理来加载。cocos2d默认的纹理像素格式是32位颜色深度。如果把颜色深度减半，那么内存消耗也就可以减少一半。并且这还会带来渲染效率的提升。

 15、说说Lua

17、Cocos2d-x中使用的各种设计模式

　　观察者模式：NotificationCenter

　　 装饰者模式：Action和它的继承类之间的关系

　　 单例模式：Director有一个sharedDirector，NotificationCenter也有sharedNotificationCenter

      工厂方法：Create方法

19、Unity和Cocos2d-x的区别

Unity3D的亮点是可视化[编程](http://www.2cto.com/kf" \t "_blank)， 资源轻松导入，一键部署各个平台，拥有众多第三方插件，轻松处理音频/视频的兼容。Cocos2d-x是拥 有发达的开发者社区，能够方便的找到各种问题的解决方案。spine、TexturePacker、Cocostudio等工具让引擎方便的处理各种资 源，UI问题。而且开源方式也使开发者很方便的研发出适合自己项目的编辑器。

使用方式：Unity3D任何功能都可以拆分成单个组件来实现;Cocos2d-x开源的优势在于可以根据自身的需求进行自定义修改。

可视化编辑：

Unity3D对可视化编辑的支持更好，脚本编译时间很快，可以快速出原型出Demo Cocos2d-x更多是代码层面的编写，为了补充可视化编辑方面的问题推出了Cocos studio 。

形象生动的比较：http://unity3d.9tech.cn/news/2013/1223/39327.html

20、游戏中的状态机的设计，动画曲线的设计，相机的移动，游戏中如何使用MVC之类

21、如何对手机游戏进行优化

  一般分为内存优化帧数优化，内存优化和运存优化。

 帧数优化可以考虑对一个message loop中的逻辑运算进行优化，比如可以考虑A\*的剪枝。或者进行time slice

体积和运行内存优化有以下几点

使用TexturePacker等工具把多张资源合成一张图片。

采用png压缩工具，在打包图片之前对每张图片进行压缩，比如将32bit颜色深度改为16bit颜色深度以降低图片质量。

 针对不同的平台使用特定的压缩格式的图片

 如果项目中帧序列占的比较多，那么可以采用降帧的方式来优化。

缩放图片，将原来图片缩小为原来的70% ~ %80，再对图像进行放大 采用编辑器，

将大图转化为拼接，那么就可以利用地图编辑器、动作编辑器等从而减少体积，降低内存的使用。

22、如何在对游戏的“手感”进行改进？

　　游戏手感一般指的是打击感，那么我就在打击到一个游戏对象时，游戏对象要产生击退的效果，产生该对象被打击的感觉。 时间控制要恰当，要让某个对象（比如拳头）打击到另一个游戏对象的时候，才产生击退效果，这就需要进行使用消息机制和回调来解决。

23、如何在[数据库](http://www.2cto.com/database/" \t "_blank)中存储一个人的所有装备

建立一个人物ID和装备ID的关系表。

将人物的所有装备的id序列化为一个JSON字符串存储为人物的一个字段。 这两个最大的区别是在修改装备时，第一个只会影响一条记录，当时第二个会影响所有装备，一旦出现bug还让玩家损失所有装备。两者各有利弊，根据使用场景自己权衡。。

FILE\_LIST := hellocpp/main.cpp

FILE\_LIST += $(wildcard $(LOCAL\_PATH)/../../Classes/\*.cpp)

LOCAL\_SRC\_FILES := $(FILE\_LIST:$(LOCAL\_PATH)/%=%)