深入浅出安卓，如何从零学好移动开发

原创

jzj1993 于 2014-10-12 22:49:36 发布 2943 收藏 1
分类专栏： 学习方法 安卓开发 文章标签： 安卓 移动开发 编程 软件 工程

本文链接：https://blog.csdn.net/jzj1993/article/details/40025209


学习方法 同时被 2 个专栏收录
44 篇文章 3 订阅
订阅专栏


安卓开发
16 篇文章 0 订阅
订阅专栏

由于近几年来互联网的飞速发展，安卓和iOS平台的大量普及推广，移动开发在当前是非常热门的一个方向。

有不少同学问我如何学习安卓，要学些什么，难不难学。之前一直没有想好应该怎么回答这个问题，只是简单的说安卓自身门槛不高，并不难学。因为我觉得准确回答一个类似这样的问题往往需要灵感。现在根据我的学习体验，做个大概的总结。

1、我为什么学安卓

我从刚开始接触安卓开发到现在也有两三年的时间了，原本我是打算做硬件方向的，对安卓只是感兴趣，完全没有考虑过工作的问题。

后来慢慢感受到硬件难度偏大，成就感比较低，也不太想做那种技术性很强的东西。纯技术思维，基本没必要和大众用户交流，因为嵌入式、硬件方面一般都是比较底层的，直接用户还是技术人员。个人感觉长期这样容易降低情商，作为一个情商本来就偏低的技术宅，与非技术人员交流会更加困难（导致找对象都比较困难）。

于是后来慢慢改成了安卓开发，安卓属于大众型产品，很多时候需要从大众用户的角度去考虑问题，技术性相对弱化了。前段时间面试，顺利的进了美团的安卓研发岗位。

我之所以学安卓，也是因为碰巧大一时学校有一门安卓选修课，就去学了下。有打算转做iOS，主要是iOS开发条件比较高，最好要苹果笔记本和苹果设备，还有每年100美元的开发者账号（虽然可以用所谓的黑苹果，但是据说开发起来容易出现问题）；加上之前一直没时间，所以也还没开始去学。

2、安卓要学些什么，难度如何（重要）

在我看来安卓开发有两层意思，第一层意思就是安卓自身的开发知识，而第二层意思是安卓、移动应用乃至各种软件开发的编程思想。这两者的关系，就像文字和写作的关系一样。

小学的时候我们就在学习识字写字，要说这件事难不难，显然只要肯花时间就不那么难，毕竟我们小时候都是这么过来的，你能看懂我写的这个，说明你也是认识字的。我们识字的目的是干什么呢？目的是看懂别人写的东西，以及自己写东西表达想法传达给别人。会识字而没有太多思想的人，写不出来好文章；而有思想的人，不需要学会很多汉字就能写出不错的文章，还可以随时查字典。

同样，我觉得安卓学习也是如此。

2.1 安卓自身开发知识

第一层含义，安卓自身开发知识，只要肯花时间，理解力稍微好一些的人都能学会。

安卓开发首先最好有Java基础，没学过可以先简单的去学习一下。推荐两本书《Head First Java》和《Java编程思想》（英文名《Thinking in Java》）。Head First系列的书，特点是简单好懂，适合入门（最好有其他语言编程的基础），但讲解不是很深入，对于理解力够强的人，看起来比较费时间。而后者是Java语言公认的权威经典书籍，如果想全面的学习Java，可以看后面这本书，但是难度偏大，尤其是对于没有学习过面向对象编程语言的人来说。

安卓开发本身的技术知识，无非就是各种封装好的API接口函数（API=Application Programming Interface 应用程序编程接口），你只要按照它的规则去调用就行了。安卓的接口有官方给出的完整说明文档，安装了安卓开发包后，也有自带docs文件夹，里面就是说明文档。对于英文水平欠缺的人可能稍微有点难度，另外，由于谷歌访问不了，网页版的说明文档有些可能会打不开。如果你不想自己看官方英文文档，你可以直接买本安卓开发的书，或者在网上找安卓学习资料。这些资料也是别人根据官方文档和自己学习经验总结出来的。有时候如果需要用到一些别人很少用到的东西，或者网上说的比较含糊，这时官方文档是最权威、准确的参考资料。

2.2 安卓项目实战

第一层意思很容易掌握，也就是安卓开发的基础知识，相当于学习怎么识字和写字。而第二层含义，安卓项目的实际开发，难度就要大一些了。

有些人一辈子就那么过去了，从来很少思考复杂的问题，也没有什么思想。而写作是需要灵感和思想的，只是会写字，写不出来好的文章。同样的道理，有些人学安卓，只是掌握了基本的API，却怎么也写不出好的程序来。然后他们不知道应该怎么做，但是又不甘落后，索性把iOS开发、Windows开发，各种程序API都学一遍，而始终只能做出来一些很简单的东西。

现在问题来了，编程到底需要学习哪些思想呢？应该怎么学呢？这里的编程思想，并不是专门针对安卓而言，而是针对所有软件开发而言。找工作的时候，你会发现，有些公司的软件研发岗位命名招聘的是iOS，但是并不一定要求掌握iOS，有安卓、Windows或其他上层软件开发经验也可以。这就是因为，只要从一种编程语言、一种开发环境学到了软件编程思想，再去学其他环境和语言下的编程就容易得多。

2.3 理论和工程

软件分为两部分，理论和工程。理论方面，国内的发展不是很好。比如人脸识别的程序，需要用到一些数学理论模型，并以此做出算法来解决问题。在一些有实力的公司，会有这样的理论研究部门。最典型的像苹果、谷歌、微软这种巨头，研究自然语言处理、图像处理、大数据分析、人工智能等等诸多问题，苹果的Siri，谷歌的安卓内置的语音识别引擎，微软小冰等等。而在国内，百度的搜索引擎对自然语言的处理、科大讯飞的中英文语音处理方案等，也都是需要大量的理论知识。

因为理论研究很大的一个特点是不确定性，很可能研究了很久也没有成果，而实力不足的公司很难有这样的资源进行理论研究，所以在国内主要是有实力的大公司，以及国家提供经费的研究所、一些大学的实验室，才能有条件进行这方面的深入研究了（顺便提一句，也正是因为这种不确定性，国内学术腐败比较严重，各种抄袭，另外即使研究不出来成果也有经费）。

通常如果一个公司软件的研发需要用到深厚的理论，会专门给这个设置一些岗位，比如算法工程师、图像算法工程师等。而安卓研发、iOS开发这类，则更偏向于工程应用。当然有时候，也会涉及到一些简单的算法问题，那些更像是小学奥数题，关键看解决问题的思路是否灵活，往往不需要很强的理论知识。例如我在网上看到一道历年的百度笔试题：百度地图当中，每个地点的左侧或右侧需要放置地名，地名的文字是矩形区域；设计一个算法，使得尽可能多的显示地名，同时要考虑地图的缩放。这个问题没有什么标准答案，出题者或许也想知道最好的答案；而实际实现时，就要看有没有足够聪明的程序员，能给出一个尽可能好的方法了。

工程的特点就是把理论应用到实际上来，并且要考虑到开发成本、时间、安全性等实际问题。专门研究这些工程的东西组成了《软件工程》的学科，但是只有软件相关学院才会去上这种课程，而且这种课程太抽象了，理论性太强，往往忽视了实践的环节。

2.4 软件工程的重要思想：模块化和代码复用

软件工程思想有很多，模块化、代码复用是其中很基本、很重要的编程思想。所谓模块化，就是把一个完整的东西拆分成很多个小的模块，每个模块完成一定的功能，分工协作，然后按照合适的规则则合成一个完整的系统。拿整个人类社会来说，每个人都需要衣食住行，但是实际上，有的人专门从事服装制造，有的人专门做食品，有的人负责建筑……最后整个人类社会分工协作，效率大大提高，构成一个整体。拿计算机来说，我们的电脑由主板、内存、硬盘、屏幕、各种外设组成，每个模块被独立设计制造出来，并且只要接口吻合，可以随意进行组合。买电脑的时候我不一定需要内置蓝牙模块，但是在需要用的时候，我可以很轻松的买一个USB蓝牙模块装在电脑上。嫌内存太小，我也可以自己给电脑换内存，而不需要更换整个电脑。

程序也是这样的，我可以写一个模块专门用于网络连接的相关控制。以后不管做什么应用，只要用到网络，直接把这个模块放进去调用。积累的模块多了，后面就能像搭积木一样搭建不少的代码，大大减轻了程序开发的负担，提高了效率，节省了成本。而模块化开发也有利于分工合作，一个庞大的程序一个人不一定能做完，比如我们用的Windows操作系统，代码量可能达到几千万行甚至更多，这时候就需要很多人共同完成。每个人或一个小团队完成一个小的模块，并且不同的模块之间规定好接口，然后同时进行开发。模块化编程实现了代码复用、提高了开发效率、有利于分工协作，等等优点，是软件开发的核心思想之一。

为了实现模块化，不同的模块之间要尽可能减小耦合度。也就是说，一个模块对于外部相当于一个黑盒子，我们只能看到对外的接口，而模块内部的具体实现，与其他模块之间的关联应该尽可能小。这样在修改一个模块的时候，只要保持接口不变，对于整个软件来说就没有影响。

2.5 软件研发相关的学科知识

软件开发需要的一些公共的知识，也是面试经常会问的学科知识有《数据结构和基本算法》《数据库》《操作系统》《计算机网络》《设计模式》等。

首先《数据结构和基本算法》几乎是所有软件相关技术岗位必会的。数据结构可以简单理解成数据是如何进行组织并保存在电脑的内存中的，而基本算法则是研究如何高效的对这些数据进行读取和处理，比如查找、排序，比较考验智商。数据结构和算法原本是两种知识，但是由于他们之间的关系非常密切，所以这两者常会作为一个学科，一起学习。通常如果你从事软件研发，要求掌握基本算法就可以了，也就是数据结构课程中介绍的算法。如果你的算法更强，可以考虑专门从事算法研究，那也很不错（如果算法学的很好，可以去谷歌总部，顺便就出国了）。数据结构推荐书籍《大话数据结构》。

《操作系统》看上去似乎和应用软件没有密切联系，但是有很多时候，软件设计需要用到多线程等知识，这个时候，对操作系统的原理有所了解，会做的更好。毕竟应用软件是运行在操作系统之上的。《计算机网络》在应用软件中使用很广泛，我们用的大多数应用都需要用到网络，所以这门课必然是很重要的。推荐书籍《现代操作系统》、《计算机网络》。

所谓《数据库》，就是最常用的一种数据的保存手段。我们用QQ给被人发送消息，一条一条的消息并不是简单的用文本文件保存在手机里的，而是通过数据库进行保存的。对于应用软件开发来说，我们所要学习的是数据库的使用，一般不需要深入了解数据库的实现原理，所以学起来不会太难。数据库最常用的是SQL和SQLite，两者语法很接近。SQL语言号称是第四代编程语言，而C语言这种是第三代，越是上层的语言越接近自然语言，所以SQL语法也很好理解，有些时候用到一些不太好理解的语句，主要是因为语句包含的

逻辑比较难理解，倒不是SQL自身的问题。

举个例子，在一个表格mytable里保存了全班学生的信息，有number和name两列分别表示学号和名字。这时我想知道小明同学的学号，我只需要用下面的语句选择他的学号就可以了，几乎和英语一样：

SELECT number FROM mytable WHERE name='小明';

编程有很多优化思想，除了提高开发效率、分工协作，还会考虑到安全问题等。这些编程思想的大量研究，人们积累了很多技巧，《设计模式》这一课程就是对一些使用频繁、经过了很多人考验、并且很有借鉴价值的程序设计思想进行的总结。而设计模式的精髓并不只是照搬那些模式，更多的是以前人的经验积累作为灵感和素材，根据实际需求，创造出更多好的编程技巧和思想。推荐书籍：《设计模式》（机械工业出版社），《Head First Design Patterns》（中文名《深入浅出设计模式》），《大话设计模式》。

2.6 移动开发独有的特点

除了上面这些以外，移动应用开发与传统桌面应用开发相比，还有一些特别的东西。移动开发，也就是针对移动平台进行的应用开发，手机、平板等产品。受限于有限的屏幕、CPU速度、内存、电源供应、可以随便移动、网络费用可能比较高等特点，移动开发就有一些比较值得注意的东西了。移动应用的界面应该简洁、方便，按钮文字等设置的大一些，方便操作，充分利用手势进行操作，还有针对安卓和iOS等不同平台进行优化，符合用户使用习惯（例如安卓有返回键，但是iOS没有）。然后在程序的数据处理方面，要充分考虑移动设备自身性能，进行各种调整。这类的问题有很多，如果需要深入了解，可以看一些相关的书籍。

2.7 安卓开发可能用到的知识，或研究方向

在Facebook等一些公司，流行一种概念，叫做全栈工程师。所谓全栈工程师，说的直白一点，就是一个人独立完成整个项目，包括客户端开发、前台网页设计、后台服务器搭建等。这对于工程师的要求非常高，不仅要知识面广，而且各个方面的学习都得有一定的深度。

这里我不打算讨论什么全栈工程师，我自己对服务器那些也没什么概念。我只是根据自己的经验和了解，总结下我所认为的、安卓开发还有可能要涉及的知识。

首先是平面设计、交互设计、用户体验。移动应用作为大众型产品，用户体验相当重要。如果有过个同类产品，功能接近，用户肯定更喜欢用户体验更好的产品。用户界面需要用到设计方面的知识。当然在大部分公司，一般会有专门的交互设计相关职位，所以对于应用开发者来说，设计方面不需要掌握的太深入，做一点简单了解当然是没错的。

然后上面已经说了很多软件工程方面的问题，数据结构、设计模式、操作系统、计算机网络等等，不再重复。

安卓NDK环境和JNI开发。安卓基于Linux操作系统，主要由Java编程，但是有些时候需要用到C++，例如核心代码需要保密，而Java保密性有所欠缺；有些程序只有C++环境才能实现；要用到一些高性能的算法等的支持，而Java执行效率偏低。这是我们可以使用Java的JNI，调用C++开发的程序库完成功能。C++的开发基本上就和在Linux上编程差不多，区别在于安卓系统中有一定的权限限制。而安卓NDK就是官方给出的、用于快速开发安卓JNI程序的开发环境。

安卓系统的实现、系统级开发。安卓系统有个很大的特点是开源免费，因此我们很容易就能获取安卓系统的源码进行学习，了解安卓系统的设计。了解安卓系统设计，后来我们就可以从事偏底层的安卓开发，系统订制，乃至安卓驱动开发、操作系统开发方面的工作。安卓是个优秀的操作系统（例如小米手机系统就进行了深度订制）。

游戏开发。移动游戏目前是很火的行业，很多公司从游戏产品中获得了大量的收入，游戏开发自然是一个不错的选择。大型3D游戏往往会使用各种游戏引擎来进行开发。由于我目前几乎没有做过安卓游戏，所以也没有太多的了解，不做过多讨论了。

另外还有服务器方面的研发。我们手机上必备的软件有QQ、微信、支付宝等，这些软件很重要的特点，不是在于软件自身，而是因为强大的后台网络服务支持。于是服务器方面的研发也不错，不过这已经不直接属于安卓应用开发的范畴了。另外还有推广运营、管理等，和安卓开发有直接关联，我并不太了解，也不做过多讨论。

总体来说，学会安卓门槛很低；但是学好安卓绝非易事。

3、安卓学习方法和技巧

3.1 总体学习思想

回到一开始文字和文章的类比上来。我们大部分人两三岁的时候就能学会用汉语说话，小学的时候就能认识很多汉字。但是为什么学英语似乎比汉语难很多呢？至少从初中就开始学英语，各种语法、单词，一直到大学，考四六级，直到大学毕业，很多人的英语还是远不如小学毕业时的汉语水平。我也是其中之一，大学不仅没进步，反而以前学的单词都快忘记了。

原因很简单，缺乏实践。我们学习语言的目的是为了应用，但是在大部分人的生活环境中，根本没有太多需要用英语来交流的地方，然后自然连单词也慢慢忘了。如果我们想写文章，没有好的想法，可以多去看看别人的文章和书籍学习学习。而有一些个人的想法，只需掌握基本的文字，就可以写了。遇到不会写的字，翻翻字典就好了，没必要把各种生僻字都记住，一样能写出好文章。而背单词，死记硬背记不牢；多练习多实践，不熟悉的东西也慢慢熟悉，自然记住了。另外实践多了，我们还能积累很多好的句子，比如各种诗词之类，写作时就可以充分利用。

同样的道理，安卓的学习，首先是应该知道最基本的一些东西。我们可以大致的看一本安卓入门的书籍，按照书上说明，搭建开发环境，把常见的基本接口简单的实践一下，有个整体了解（认识常用字）。然后我们就可以实际运用了，也就是做项目（写作文）。遇到问题，我们需要自己多动脑思考，多在网上找解决方法，实在解决不了再考虑请教别人（查字典）。如果对编程的思想掌握起来比较吃力，可以去看看网上的一些开源程序源码（看别人的文章）。久而久之，不仅对常见的API有了了解，也慢慢学会了编程的思想和技巧（自己会写文章了）。编程时，要注意模块化，把常用的一些自己写好的模块封装起来，做好注释，以便以后使用（好词好句记录）。另外，学习的东西记得及时做一些笔记和总结，如果整理的比较好，也可以发表到网上，对别人或许也会有帮助（做笔记和分享）。

软件开发还有个很重要的过程就是程序的调试。在安卓中，由于用的是Java，程序调试手段很多，也非常方便。安卓提供了一个Log接口，可以在关键的地方打印日志，然后在Eclipse的LogCat窗口中查看，对于程序调试会有很大帮助。Java应用在执行出错时，能直接显示出错的代码位置和错误类型，也是在LogCat窗口中显示，然后就很容易找到错误的原因所在了。安卓开发时，可以电脑连接手机在线调试，设置断点，查看变量的值、运行的进程和线程、内存消耗、文件等信息，具体方法请自行搜索。

3.2 基本知识学习

说的具体一点，安卓学习的过程大致是这样的。

首先是搭建开发环境，通常用Java+Eclipse+ADT插件+Android SDK，也可以用Android Studio，具体方法网上有很多参考资料。开发环境中集成了安卓虚拟机，如果你没有安卓设备，可以在虚拟机上运行程序，但是速度较慢，不支持一些传感器等硬件设备；如果有安卓设备，最好在实际设备上运行程序。

然后是掌握安卓四大组件，尤其是Activity和Service及其生命周期（BroadcastReceiver和ContentProvider可以后面再学），Intent实现界面的跳转，Menu菜单；然后是安卓的常用控件、XML布局（Layout）等。这些是安卓最基础的东西，可以通过编写Demo程序的形式去学习。网上有个文档《深入浅出Google Android》，里面就通过一个简单的安卓程序实例，介绍了这些知识。

3.3 进一步学习

到此安卓最基本的基本API就算是学习完成了。然后还有SQLite数据库、各种传感器、动画控制、多媒体、网络通信、GPS定位、电源管理等API，这些API可以先只作简单了解，直接去写实际项目。你可以试着写一些简单应用，例如计算器、音乐播放器、小游戏，或者你所感兴趣的简单应用（一开始难度不要太大）。需要用到的API再去详细的学习，逐步锻炼编程能力，代码要规范，尽可能符合Java命名标准，程序代码尽可能写成模块化的，提高代码复用。记住，完成同样的功能，在保证程序结构清晰、模块化、规范化的基础上，代码量越少越好。

3.4 深入学习，并开发高质量应用

而再到后来，你可能需要更深入的去学习安卓，这个时候可能需要了解一些安卓系统Java层的源码（安卓底层用的C和C++，上层开发包中API用Java编写）。可以在网上下载到，然后在Eclipse中设置关联源代码。需要查看源码时，直接用Eclipse转到函数定义，就能看到安卓系统的Java层源码了。另外你可能需要学习《操作系统》《设计模式》《软件工程》等前面提到的课程知识，加深对软件开发相关知识的理解。

比如你可以自己独立完成或者与别人合作做一些项目，可能涉及到多线程、大量数据的处理、JNI的使用、自定义控件和界面布局，识别特殊的用户手势，游戏引擎等等（可以参考网上的开源项目，以及平时我们用到的各种手机应用）。

我的建议是，后期做安卓应用的时候，直接做功能完整的应用，并且要经过反复测试调整；尤其是要注重用户体验，还有程序的规范性、稳定可靠性（例如Java中空指针的判断、try...catch的使用、线程通信等），这样才能很好的学习移动开发的精髓。如果你只是为了学习那些API，做出来一些体验很不好的Demo级别应用，只能说是学会了安卓，却没有学好。

或许你会觉得有些应用功能很简单，要不了多少时间就能做好，实际上远不是那么简单。一个优秀的安卓应用，不仅用户界面和体验非常好，而且程序规范、稳定可靠、执行效率高，可扩展性强，想做到这一点，非常的不容易。一个优秀的商业安卓应用，主要的代码实现阶段，可能只占了整个应用开发时间的1/3甚至更少。在开发之前，有不少的时间是在进行应用的策划安排；而在开发完之后，又需要大量的时间，对应用进行反复的测试调整更新，最后才能被发布，从而安装到我们的手机上。

这里顺便一提，安卓应用开发相比iOS的一个难点来自安卓系统碎片化问题。安卓系统是开源免费的，这是一大优势，也因此对很多国产和国外手机制造商带来了很大的好处（如果没有安卓，很多手机厂商恐怕都深陷危机之中了，或许移动互联网也不会发展的这么快。不知道现在是不是iOS要称霸天下，或者WP大受欢迎，又或者塞班还会屹立不倒）。但是安卓的这种特点，导致同一款安卓应用至少要同时兼容各种主流手机型号，各种配置，各种屏幕尺寸，各种系统环境。而这也是迄今为止安卓开发者心中永远的痛（ノ__ヽ）。

4、附上我的学习经历和部分作品

这里简单介绍下我的安卓学习经历，如果读者能从中得到一些启发那最好不过了。

4.1 初学安卓

当初学单片机的时候，编程至少是底层到C语言，有时甚至是汇编指令，然后再到底层寄存器、数字电路、模拟电路的理解，几乎所有的东西都得自己实现。因此一开始接触Java和安卓，我很不习惯抽象的上层编程，总感觉那些封装好的函数调用很难理解，因为不知道那些函数做了些什么。不过后来慢慢就习惯了，并且越来越感觉到Java这样的上层语言非常高效好用。

我是从大一下学期开始上选修课学习安卓的。当时只了解C语言，Java并不了解，而安卓主要是由Java开发。Java代码尤其是像接口之类，对于没有学习过面向对象编程语言的人来说不太好理解。比如我当时就一直记不住这样的代码，觉得函数里面又有函数真心很

神奇：

```
button_ok.setOnClickListener(new OnClickListener() {
void onClick(View v) {
// ...
}
});
```

不过现在熟悉了Java以后，这种代码再常见不过了，内部的new关键字在括号中实例化了匿名的接口类实例，并对其抽象方法进行了实现。

当时上选修课我的感觉就是完全听不懂。按照一贯的作风，我没有认真的去听课，而是跟着老师的课程节奏，搭建好开发环境，借了参考书，然后上网找资料，自己学。上课给我带来的好处是学习氛围，还有基本的学习方向。为了交大作业，我用一周的时间做了一个拼图游戏。在当时应该是所有上选修课的人中做的很好的，拿到了95分的高分。不过在现在看来，那个简单的应用只能说是能用，而应用的质量太低，不是因为功能太简单没有新意，而是用户体验不好，兼容性也太差。

4.2 参加谷歌安卓大赛

大一暑假为了参加谷歌大学生安卓大赛（详情百度一下就能找到），借了一本安卓书，花了差不多两个月的时间，几乎每天从早到晚，一边看书一边上网找资料一边写代码。最后做了一个安卓多媒体备忘录应用GoodMemo，获得了谷歌安卓大赛西北赛区优秀奖。当时这个程序算是让我学会了安卓中的各种基本API，同时学到了很多编程思想方面的东西（对于我来说，当时很多东西是靠自己想出来的，过程比较慢，但是锻炼效果很好；当然你可以去参考《设计模式》之类的书籍，或者看网上的开源项目）。要说代码量，Java代码写了两万行，XML没有统计。但是如果现在再让我重写一个一样的程序，肯定要不了那么多代码。现在再看，感觉那个应用的界面还是不够好，对不同手机的兼容性也不够。

这是当时参赛的演示视频 http://v.youku.com/v_show/id_XNDUyMDI5MTE2.html

由于谷歌有段时间没法访问，今年谷歌安卓大赛的主页在这里 http://miac.buu.edu.cn/

4.3 再次参加谷歌安卓大赛

然后就到了今年年初，我决定重新参加一次谷歌安卓大赛，并严格按照软件设计规范，编写一个全新的高质量安卓应用。去年年末设想作品方案，想到要做一个闹铃应用。至于有什么特点，我发布到安智市场了，看介绍就知道（从我一开始做多媒体备忘录GoodMemo以来，就和闹铃应用结下了不解之缘，一个好的创意真的很难 o(╯□╰)o）。

从寒假在家开始进行界面的概念图和交互设计，到实际编程（实际上闹铃数据部分的编程只用了几天就完成了，绝大多数时间用在了界面设计上），再反复的调整和修复各种BUG，包括应用的名称都想了很长时间。最后还是姐姐给我的灵感，就叫做《It's the time》。断断续续直到8月末，才算是基本完成了。然后又找同学帮忙体验和提建议，对界面进行了不少调整。面试时就演示了我的这个作品，感觉还不错。面试结束后又再做了一些调整改进，最后终于发布到安智市场了，可以点击下面的链接安装感受一下。在9月底的时候，提交到了谷歌安卓大赛网站，暂时还没有开始评选。这也是我目前能拿得出手的，充分考虑了用户体验的主要的一个应用。但是兼容性还是不够，貌似在有些同学的小米手机上没法用……深度订制的小米手机系统真折腾人。。。至于功能，没什么好说的……实在想不出什么新意。

http://www.anzhi.com/soft_1856039.html

4.4 黑客编程马拉松

今年早些的时候，四月份的时候，我和另外两个队友参加了一次黑客马拉松编程大赛，我负责所有编程工作。连续30小时，5000行代码，三等奖、最佳设计奖、最努力编程奖。我们的作品Ding，是一个语音控制的闹铃（又是闹铃+_+），其实程序BUG很多，也没有再去修改了，创意一般，实际价值不太大。但是队友设计的界面风格和宣传视频受到了评委一致好评，而我编程的坚持不懈也受到了奖励。其实虽说应用有5000行代码，实际上就和我前面所说，并非全都是现场完成的，而是有一部分之前就做过相关的模块，直接放进去用了。30小时的主要时间都用在了界面的编写和已有代码的移植、修改和完善上了。

4.5 实习经历

还是在今年，从三月份开始到七月份，我在一家公司实习，做的是安卓开发，学了不少之前没用到的东西，多线程、网络、语音识别、各种第三方API（例如天气查询）等等。主要是学了不少技术，项目不再细说。

4.6 最简单的手电筒应用

在一开始打算往安卓发布应用之前，我有一些东西不明白，比如应用签名之类的，个人开发的应用一旦发布出去，成为商业应用，会有什么需要注意的东西呢？当时纠结了很久，也找了不少资料，决定先做个简单的应用试一下。然后在8月份的时候，我做了一个简易的手电筒应用SimpleLamp，发布到市场了。当时感受到安卓碎片化问题确实挺严重，即使是小小一个手电筒，本来简单的一个API就能完成的事，但是由于要兼容不同型号设备，我看网上一些代码，还得对一些特别的手机型号进行判断、特殊对待，相当麻烦，而市场上那些比较受欢迎的安卓手电筒应用，必强调的一个点就是兼容多少种手机型号。所以这里我也来掺和一下，我的手电筒应用经过了更新以后，能兼容很多型号的安卓手机（具体多少种我没法统计，反正我认识的同学下载使用的，都没有出现过无法打开手电筒的现象）。点击下面的链接可以下载体验。

http://www.anzhi.com/soft_1795050.html

以上就是至今为止我的安卓学习经历了，也欢迎大家支持我的作品SimpleLamp和It's the time，再发一遍下载地址……

It's the time http://www.anzhi.com/soft_1856039.html

SimpleLamp http://www.anzhi.com/soft_1795050.html

另外，APP开发的完整流程可以参考知乎上的这个问题和回答：

http://www.zhihu.com/question/19957949

至此，花了将近一天时间写的安卓相关总结就算是写完了，希望对大家有所帮助，也希望大家多多支持。个人水平有限，有任何问题和不对的地方，欢迎指出，或和我交流讨论。也欢迎大家关注我的个人主页 http://www.paincker.com 。大家的支持会是我最大的动力，有了动力，我会认真写更多的文章来分享我的学习经验。