概述

随着经济社会发展，电子通信技术在人们的日常生活中起着越来越重要的作用,地图服务不仅使人们的日常生活便利，更重要的是其在经济和社会发展方面对效率和收益所做出的贡献。手机地图规避了电脑端的不方便携带和GPS复杂的坐标参数的缺点，为广大普通用户所喜爱，因此开发和研究使用的手机地图服务具备很大的现实需求和实用价值。

1. 引言
   1. 课题研究背景
      1. Android系统的发展现状

随着科学技术的发展,只能用在PC机和主机上的技术移植到移动终端上,不仅芯片的性能越来越高,而且芯片的能耗也越来越低，加上多点触屏，高清摄像头，大容量内存的迅速问世，使得移动端设备以智能手机为旗舰变得越来越强大。智能手机因其便携性,在人们的日常生活中也变得越来越重要。它已成为人们日常生活中不可缺少的工具。

开放式手机的出现推动智能手机的发展。真正意义上的不仅是一款完全开放和开放的手机厂商和开发工具,而且是底层的源码。谷歌于2007十一月正式推出新安卓系统并对外开放其源代码,是真正意义上的第一个真正完整的移动平台。Android是基于Linux内核，具有很高的灵活度便于根据需求进行定制，同时还具备完善的网络支持和可移植性。加之其免费向开发人员提供的源代码，成本的降低和其灵活，自由度不断吸引着世界上越来越多的开发人员加入Android的阵营。这不仅使得Android应用市场越来越丰富，也同时推动着Android系统本身逐渐走向成熟和完善。目前世界上使用Android系统的设备已经超过10亿台。  
 Android是java语言开发。对于有一定基础的人来说,使用Android开发是一件比较简单的事情。加上Android世界丰富的可查阅资料，和开源的底层代码都不断促进着Android的发展。Android平台还支持个性化定制可以替换底层源码构建自己的开发框架，在操作和UI方面作出自己的创意，这一切都似使得Android从发布开始就以非常迅速的态势不断迭代更新。

**1.1.2** 位置检索技术背景介绍

在以往的非智能移动端上开发移动地图应用是非常有难度的课题，但智能手机系统的出现让移动端地图的开发变得容易起来。以往的非智能平台不仅需要向地图厂商付高额的服务费，而且当时的gps传感器无论是能耗性能还是成本都不足以支持在非智能平台上开发地图服务。和Android平台,因为谷歌从一开始就非常关注它的Web性能,所以Android在开发web应用程序方面有着固有的优势，加上其优秀的浏览器和多媒体的支持，和其丰富多彩的图形系统，是得其基于web开发移动地图应用比起直接操作传感器要方便得多。Google公司本身的地图服务支持也使得基于Android平台的移动地图服务的开发成本大大减少。

在通信技术迅猛的发展势头下，传统的手机功能已经不能满足人们日常生活的需要，硬件水平的发展使得gps系统也逐渐成为手机上的标配，手机已经不仅仅是一个通信系统也成为了一个gps系统。

在通信网络方面，随着我国3G,4G网络的不断普及，基于基站的定位技术也越来越成熟。现在的地图厂商大多采用结合gps,基站，wi-fi信号的形式，在发生定位请求时根据请求所在地的综合因素（gps信号情况，网络连接等）定位和返回其最佳结果。

1.2课题任务

本课题是基于Android平台开发出的位置检索系统，不仅能实现本机定位，而且还可以根据建筑物名称进行定位，显示行走路径等功能。同时对安卓系统精心剖析学习，总结Android程序的开发流程，结合百度公司的开放API，设计基于Android平台的地图应用。

1. 在Android平台上实现根据建筑物名称进行定位，显示行走路径的功能。  
   2.保证软件信息的实时性、准确性、及时性。
2. Android平台的相关技术介绍及SDK分析

本章主要介绍了Android平台的特点和基于Android应用的开发环境，应用程序的架构以及Android开发SDK和地图定位所需的相关理论。

* 1. Android平台特性

Android平台是基于Linux操作系统内核而来的移动平台操作系统。它采用多层软件体系结构,可分为三个部分:底层是基于Linux内核的,是系统工作的基础，主要提供系统功能；应用以上是由外部世界的发展,主要使用java语言。Android平台支持多媒体，包含音视频和静态图像，网络方面支持3G,4G,wi-fi技术以及蓝牙连接。相机，方向传感器，加速度传感器等硬件设备也在Android平台得到优秀的支持。组件的可重用也让Android平台上的应用程序开发变得简便高效。其集成的浏览器基于WebKit系统，Android平台还自带其优化图形库，包括可自行定义的2D图形库，和三维的OpenGL ES1.0。硬件设备优化方面采用了Dalvik虚拟机。

* 1. Android平台特性

图2.1是Android系统的重要组件展示, 从低到高分别是Linux内核层、系统运行时层、应用程序框架层和应用层。蓝色的部分是由java程序,和黄色的部分是虚拟机的实现,这是用来构建java运行平台，绿色为Android系统自带的库文件是由C/C++语言实现的，红色部分为Linux内核。在Application Framework环境中通过JNI调用C/C++库文件从Android系统通过JNI。每个部件如下所述。

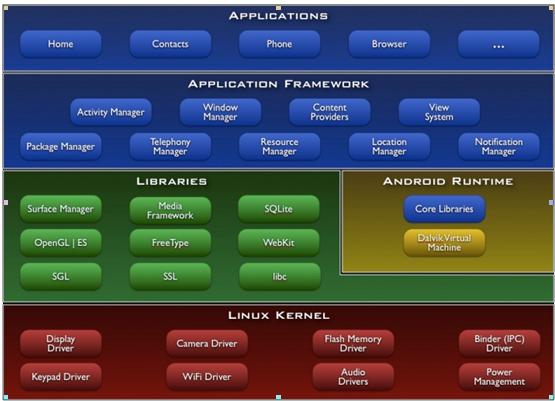
****

图2.1 Android系统架构图

1. 应用程序：

是在Android系统上集成的核心应用程序,随Android系统发布,包括浏览器、联系人、邮件、短信短信、日历Android包。所有的应用程序都是用java语言写的。

应用框架:为了简化组件重用的开发,开发人员可以访问API的核心应用程序,每个应用程序可以发布它的框架和任何其他应用程序,并且可以释放功能块。这样的程序复用机制使得程序组件可以更容易的被更换。一般来说，每一个应用程序中同时存在大多数的系统服务。主要包括一下方面：用于在网页浏览器上创建丰富的可扩展的应用程序,包括表格、文本框、按钮和支持。

内容提供器（content providers）在应用数据共享的前应用程序的数据数据可以由其他应用程序实现。

资源管理器（resource manager）包括非代码资源,如布局文件、图形本地化字符串等。

通知管理器（notification manager）可以显示在系统状态页上显示的应用程序的动态消息通知。

活动管理器（activity manager）为所有应用程序提供通用的导航回退,并管理每个应用程序的生命周期。

1. 程序库：

开发人员在使用Android系统框架的不同组件时，Android程序库的一些C/C++库可以为我们提供一些服务。标准C库中,继承了BSD,能够提供基于Linux的嵌入式设备服务。表面管理器可以提供2D和3D的图层集成来显示管理子系统。基于PacketVideo的OpenCore的媒体库可以支持多种音视频，它还支持静态图像文件的各种编码格式。libwebcore是一个新的Web浏览器引擎,可以嵌入一个网页在浏览器。公司是一个一二维图形引擎的原型。一个OpenGL API建立在es1.0 3D库库可以使用3D硬件加速和优化的3D软件。FreeType提供矢量和位图字体显示。在Android系统中还提供了一个强大的轻量级的关系数据库引擎的所有应用,SQLite。

（3）Android运行库：

Dalvik虚拟机的设计是在一台机器上运行多个虚拟系统的可靠性,每个Android应用程序运行在自己的进程中,许多图书馆核心java编程语言的特点是可以实现的。Dalvik虚拟机的Linux从底层的内存和线程管理机制,Dalvik平台的实施是适用于所有的java文件形式存储系统的可执行文件格式优化的基础上编制登记然后SDK中的“dx”工具。由虚拟机执行的Low DEX格式。

1. Linux内核：

在核心服务,如安全、进程管理、内存管理、网络协议栈和驱动模型，Android基于Linux内核。

* 1. Android应用开发环境

（1）系统和环境要求

当使用Android SDK代码工具开发应用程序时,您需要一个合适的软硬件开发环境,需要以下操作系统：

* Windows XP或 Vista或Windows 7
* Mac OS X 10.4.8及更高版本
* Linux

（2）所需的开发工具

之前Android应用程序开发通常采用Eclipse作为其集成开发环境。Eclipse是一个基于java的集成开发平台,支持扩展。Eclipse是一个开源的IDE,专注于开发集成，全能的商业品质平台。

另一个开发工具是目前谷歌Android官方网站推荐的Android开发环境集成Android工作室。Android Studio是一个应用程序的开发和调试工具基于IntelliJ IDEA,类似于Eclipse的ADT。基于IDEA的Android studio提供了一套应用Gradle组件的自动修复和重构的工具。在解决兼容性和性能都问题时为开发人员提供提示。我们知道因为世界上诸多以Android为系统平台的手机厂商大多都有其自行定制的Android系统版本，所以各版本之间的兼容问题一度成为困扰开发人员的一大棘手问题。Android Studio还提供了应用签名，ProGuard的开发工具以及应用程序创建模板和布局管理器，布局编辑器允许开发人员通过拖动UI组件来预览布局效果。

在Android Studio相比于Eclipse提供的更为强大的调试合和设计的灵活性，而在谷歌官方推荐下,在这个项目中,我将使用Android Studio作为一个集成的开发环境。

（3）安装SDK

首先,在谷歌Android官方网站下载SDK的ZIP文件,和良好的SDK文件下载到android\_sdk\_ <平台> \_ <释放> \_ <建设>文件夹。包含工具/样品/注释等。下来将SDK工具文件夹的路径添加到系统环境变量。如果是在Linux系统下,你修改~ / bash\_profile文件和~ / .bashrc文件。配置环境变量,如果你找不到这个设置,添加路径= $ { }:<路径your\_sdk\_dir > /工具作为新的线在上面的文件。在Mac环境,在个人文件夹中找到Mac环境,bash\_profile文件,然后做同样的操作和Linux。在Windows系统中,在“系统变量”部分,单击路径,然后添加自己的工具文件夹的完整路径。

（4）安装ADT

如果使用Android Studio,跳过此步骤。为Eclipse IDE(ADT)安装Android开发工具。该插件用于运行和调试Android程序,其中包括一个功能强大,高速,易于构建的Android模拟器。启动Eclispe之后选择“帮助”，找到软件更新选项，点击查找并安装。按引导安装完成后重启Eclipse后更新为指向SDK文件Eclispe首选项。从面板中选择Android，点击Apply就完成了ADT的安装。

* 1. 上层应用结构分析

在开发Android应用程序时,从Android程序的角度来看,可以将它们分成两个层次,即应用层和低级硬件。

一个完整的Android应用程序通常由四个组成部分:活动,服务(服务),广播接收器(广播,接收器),内容提供商(内容,供应商)。并非所有的应用程序同时需要这四个组件，要根据程序的实际需要进行选择。AndroidManifest.xml正是为Android应用提供住址管理的文件，我们需要登记我们需要使用AndroidManifest.xml的成分。如图2.2所示，Android应用程序的组成。

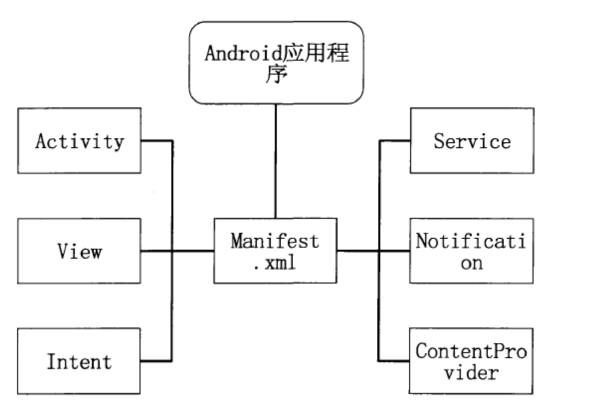


图2.2 Android应用组件

（1）活动（Activity）：是应用中最基本的组成部分，属于应用中的表示层。用户界面和用户事件形成一个UI界面,成为一个活动,它使用视图来显示信息并通过显示界面来响应用户操作,该界面与应用程序中的用户交互的窗口。Activity是应用程序中关键的组成，开发者必须清楚理解Activity的生命周期等各方面特征。图2.3是Activity的生命周期流程图。类比j2EE中的servlet技术，我们要实现一个自己的servlet类需要实现相关接口并重写其中的函数，继承相关类。在servlet容器中,这些方法是自动调用的。活动的操作机制类似于servlet。

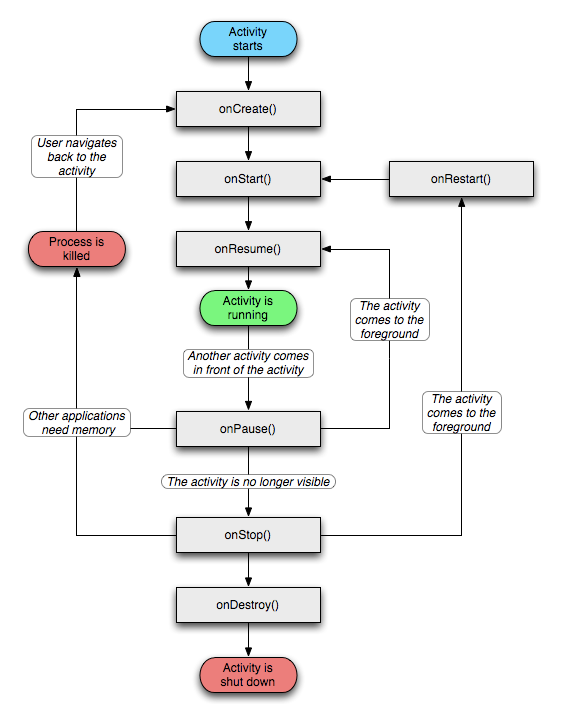
。 

图2.3 Activity生命周期展示

1.开始Activity：系统首先调用重写onCreate方法,然后调用onResume,使活动开始运行。

2.onPause方法被用于在从当前页面跳出到其他页面是，或者屏幕锁定状态下。

3.在从提出的页面转回之前页面时会调用onResume方法，在解除屏幕锁定时也是使用的该方法。

4.onPause方法被用于跳转到新的Activity，在返回系统主界面时也是使用该方法（Home键流程）。

5. 用户将通过返回键返回前的活动模式,系统会调用onRestart方法。

6. 当你退出,结束一个活动,你调用onPause,onStop,反过来ondestory方法。

以上就是对Activity生命周期的总结。在一个Android应用程序中往往存在着多个活动这也就是说会存在有多个Activity，比如系统界面，主菜单界面，功能界面等。各个Activity之前不是相互独立的，它们之前存在着数据传输交换，界面之间跳转等联系。所以Android底层系统处理Activity时维护了一个堆栈的模型，将跳转过程中前一个Activity压入堆栈，当又要跳转回这个页面时再将它从堆栈中弹出。

这个跳转的过程是由意图(Intent)实现的，意图描述了一个Activity想要干什么，即活动的目的。它包含了意图的动作和它所传递的数据。传递数据可以直接专递也可以经过Baundle包装之后再进行传递。常用的动作类型有：View,Edit,Pick。意图过滤器（Intent Filter）用来描述一个活动可以处理怎么样的意图。

（2）服务（Service）：是应用程序中隐藏的角色。服务试运行在后台的是不可见的，他可以更新数据源和可见的活动，并触发通知。服务是不能自行运行的，他通过各其他组件的绑定来工作，和活动的几倍差不多。服务一般用来处理不愿意暴露给用户的操作。启动服务的方式有两种，StartService和BindService。两者的差别在于，StartService是和其他组件没有联系的，只是通过起来组件来启动，而BindService是和其他组件相绑定的，如果该组件停止工作，那么这个服务也将停止。服务一般用来处理程序中的一些阻塞式工作，比如查询，下载等耗时任务。服务是不占用主线程的，它通过后台独立线程工作这样可以将阻塞时间面向用户屏蔽掉，以改善用户体验。服务同样需要在AndroidMainFest.xml文件中进行注册。

（3）广播接收器（Broadcast Receiver）:是广播意图的接受者。广播接收器是用来处理应用程序之间消息传输的一个组件，用于监听特定过滤器的意图并控制启动应用程序来响应该意图。是时间驱动程序的最佳选测。

（4）内容提供器（Content Provider）:内容提供器是多个应用之间的数据共享传输组件。应用程序将数据文件存储在手机上，SQLite数据库中，以及手机的存储卡上，这个过程就是依靠本内容传输器来完成的。比如，手机上的联系人名单，一个应用程序修改并通过内容提供器将该名单存储在手机上，当其他应用需要调用该名单时再通过内容提供器获取做自己的处理。内容提供器充当着应用程序之间共享的数据库的角色。

以上的各种结构都是通过AndroidMainFest.xml文件结合在一起的，从而形成一个完整的Android结构。

* 1. 底层应用结构分析

硬件抽象层（Hardware Abstraction Layer）是位于底层具体硬件和操作系统之间的借口层，是连接底层硬件电路和上层操作系统的桥梁。我们知道在Linux中有一个概念叫“一切皆文件”即硬件在Linux中的展现也是一个文件。Android是基于Linux的一种操作系统，所以在Android系统中也存在这样的抽象。硬件抽象层的目的就是屏蔽掉硬件电路的具体细节，只向操作系统提供其虚拟平台接口。这样做的好处就是为操作系统在各种硬件平台上的移植提供了很大的便利性，因为其屏蔽掉了硬件平台间的具体差异。硬件抽象层向上为应用程序提供接口，向下调用底层Linux硬件驱动，是一个承上启下的作用。硬件抽象层为Android系统应用框架层所调用。是使用C/C++语言编写的。Android硬件驱动程序包括以下部分：

1. 显示（Display）：是显示屏的驱动，应用程序可以操作手机屏幕。
2. 用户输入(Input)：虚拟键盘驱动，应用程序可以对键盘操作。
3. 多媒体编码解码(Codec)：提供多媒体的编码和解码，包括硬件Codec驱动和Codec插件，上层的播放器应用可以调用硬件播放视频。
4. 3D加速(3D Accelerator)：用于3D图形处理，从而进行3D游戏开发，由OpenGL驱动和OpenGL插件组成。
5. 音频(Audio)：音频驱动，包括播放音频和通过麦克风录制音频。
6. 视频输出(Video Out)：视频显示驱动，以及Overlay硬件抽象层，使得Android设备可以播放视频。
7. 摄像头(Camera)：摄像头驱动，使得Android设备可以动过摄像头进行拍摄，一般是v412。
8. 电话(Phone)：打电话发短信等功能驱动，包裹Modem驱动程序和RIL库。
9. 定位系统(GPS)：包括GPS驱动，一般是串口驱动，为Android提供定位服务。

（10）无线局域网(WI-FI)：包括WLAN驱动和相关通信协议，为Android设备提供上网功能。

（11）蓝牙(Blue Tooth)：包括蓝牙驱动和相关协议，是的Android设备可以通过蓝牙进行数据传送。

（12）传感器(Sensor)：Android设备内置的一系列传感器，如温度传感器，方向传感器，光传感器，速度传感器等。

（13）震动(Vibrator)：震动模块，对应Android震动提示功能。

（14）背光和指示灯(Light)：LED驱动和指示灯驱动，对应屏幕背光和手机状态指示灯。

（15）警报和时钟(Alarm&RTC)：时钟用于获取当前时间，警报驱动使Android设备可以调用报警系统。

（16）电池(Battery)：包括电源的驱动和电池的抽象层，对应获取电池电量和充电的功能。

硬件抽象层大大方便了开发人员，使得开发者不在需要关注底层硬件，在使用时直接调用其API接口便可操作相应模块。

* 1. **Android SDK分析**
     1. Android API核心包分析

从安装系统平台的最底层到最高层的核心API包如下：

1. Android.util:底层辅助类，eg:XML等工具，容器类型等。
2. Android.os:提供用户基本活动，进程间通信IPC和消息传递。
3. Android.graphics:图片渲染相关。
4. Android.text、Android.text.util、Android.text.style、Android.text.method:支持富文本，输入模式的文本处理工具。
5. Android.database:底层数据库API，用于处理数据库表及数据。
6. Android.content:用于访问设备上安装的应用程序数据和相关资源。
7. Android.view:展示界面关键框架。
8. eg:list,button,layout,managers，界面的组成元素。
9. Android.app:上层应用程序，实现对Avtivity的调用。

(10) Android.privide:调用content providers的接口。

(11) Android.telephony:通话接口和API交互。

(12) Android.webikit:web层面工作API。

(13) Android.widget:标准展示界面组件元素。

在本课题中除了以上核心包之外还需要用到Android提供的可选包GPS的API-LBS用于为位置检索服务提供定位。原理在于驱动集成的GPS芯片接受卫星信号，从而定位手机位置。

* + 1. Android SDK提供的工具包

Android SDK提供了一系列用于调试，打包和安装的工具，用来配合Google公司为Android提供的模拟器等开发工具插件，为开发人员提供便利。

1. Android系统模拟器（Android Emulator）：运行在计算机上的虚拟Android系统，用于设计，运行，调试应用程序。
2. 集成开发环境插件：是集成在开发环境中的各种开发工具插件，方便协助开发人员创建，测试，调试工作。
3. 调试系统监控：是集成在Dalvik虚拟机上的用于管理在虚拟机上运行的程序的系统，可以查看运行程序的进程，内存堆栈的使用情况，生产数据情况等。
4. Android调试桥：用于将标准的调试器连接到虚拟机或者手机设备上，也可以向模拟机或者手机设备发送应用程序apk。
5. Android系统资源打包工具：可以通过Aapt创建.apk文件。包括资源和二进制可执行文件。
6. Android系统接口：产生进程接口代码。
7. SQLite3数据库：Android系统可以产生开发人员和用户方便访问的SQLite数据文件。
8. 跟踪显示工具：Android系统可以产生跟中数据文件和图形分析视图。
9. 创建SD卡工具：创建位于模拟器外部的存储卡或者手机设备SD卡上的磁盘镜像。

(10) DX工具：将class字节码反编译为Android系统字节码。

（11）生成Ant构建文件：是一个用来生成Ant构建文件的脚本，Ant构建文件用来编译Android应用程序。

（12）Android系统虚拟设备：Android系统虚拟设备都模拟了一个虚拟平台用来运行Android系统。

1. Android平台的位置检索系统设计

手机地图服务是指通过手机上集成的GPS模块和移动基站的方式通过卫星定位技术，移动通信技术并结合地理信息技术而成的地图应用形式。通过以上技术可以得到准确的用户定位信息并将位置信息展示在手机端地图应用上，使用户可以通过手机方便的进行定位即时位置，查询地图，检索目标地点，出行线路规划等活动，极大的方便了人们日常生活。

**3.1** 地图软件技术市场现状

目前市场上存在着众多的地图厂商其中占据主流市场的主要有高德，百度，Google，搜狗等，其中绝大多数都是有免费API提供给个人和企业级开发者的。同时市面上还存在着诸多的其他类型但是在某些模块需要地图功能支持的应用软件，如美团，饿了么，携程，滴滴等。这些软件的做法大多是通过调用现有的地图服务提供商的API。国内目前的地图产业链基本可分为三大环节，四大链路，依次为共有数据链、软件链、互联网和移动端服务链，具体如表4.1所示：

|  |  |  |
| --- | --- | --- |
| 链路 | 服务 | 厂商 |
| 基础地图数据提供 | 兴趣点（POI）道路等地图信息，底层地图数据库 | 百度，MAPABC |
| 本地导航 | 传统离线地图和线路规划，核心是利用独立存在的导航引擎 | 高德，凯立德 |
| 大型互联网手机地图  大型互联网手机地图使用者（通过调用其他厂商API） | 调用第一种链路服务提供商的数据来提供服务 | 谷歌，搜狗，百度 |
| 调用第二种链路所提供的地图API来提供服务，开发出基于地图的应用 | 携程，链家网，美团，滴滴等。 |

表4.1 地图产业链展示

其中Google公司提供同时也是Google公司在Android系统中集成并在Android SDK中提供的谷歌地图处于产业链的第三种，是通过调用第一种产业链的厂商提供的API开发为用户提供地图服务的。在2011年之前，谷歌地图一直占据着地图产业的绝大部分市场，但谷歌一直没有拿到国内资质牌照，所以目前国内的大部分有关地图的应用软件都是采用国内地图厂商所提供的服务。所以在本课题的实现中没有采用Android集成的谷歌地图。

百度地图API是一套为开发人员提供的免费的基于百度地图的用于构建基于地图服务的应用程序接口，包括JavaScript、IOS、Android、静态地图、Web端等多种平台提供定位，基础地图，LBS数据，出行线路规划等多种服务。百度地图依托百度公司强大的团队近几年来发展势头迅猛，目前已经发展的相当成熟，所以经以上分析本课题之后的实现部分将采用百度地图SDK进行应用软件的开发。

**3.2** 地图软件技术市场现状

基于本本课题的任务要求和前期的用户行为调研，本课题的开发主要分为以下几个部分：

1. 基础地图：提供世界地图和国内详细城市地图。
2. 地图操作：提供基本的地图界面操作，如放大缩小，地图方向转换，卫星地图、交通地图等模式的切换等。
3. 目标地点检索：根据目标地点如建筑物名称定位，以及常用的兴趣点（POI）搜索，如公共卫生间，加油站，便利店等。
4. 路线规划：显示行走路径，并提供导航信息。
5. 定位：通过GPS模块和移动基站等信号源定位用户当前位置并在地图上展示。

此外，本位置检索系统本着实用性的原则，将着力于以下两个方面：

1. 简洁性：本系统目标功能完整且不赘余，界面简洁操作简单。
2. 流畅性：系统的多项服务都是基于阻塞式的数据和获取，因此各环节采用多线程，通过后台Service完成，积极避免ANR，使本应用运行流畅。

**3.3**界面布局的方式

界面布局是Android应用程序的重要部分，在开发中提前设计好程序界面，然后在资源文件的Layout中写好对应的布局文件，布局文件都是以xml格式存在的。在Android中主要有以下几种布局方式：

1. FrameLayout（框架布局）：层叠方式的布局模式，新添加的布局会覆盖在之前存在的布局之上，因此一般在比较简单的布局中采用。
2. LinearLayout（线性布局）：是一个个元素的罗列相当于在HTML中的DIV模式。分为垂直模式和水平模式，在垂直模式中每行只能有一个对象，在水平模式中每列只能有一个对象。
3. AbsoluteLayout（绝对布局）：用元素在屏幕上像素的绝对位置决定元素的布局，缺乏灵活性。
4. RelativeLayout（相对布局）：是一种灵活的布局方式，用另一已经存在的元素的位置来描述要加入的新的元素的位置，在布局中存在元素较多时比较方便。
5. TableLayout（表格布局）：每个布局中存在表格元素，可以在每个表格元素中定义需要的元素，并设置对齐方式。

以上五种布局方式都可以当做对象嵌套进别的布局方式中，正是这种嵌套机制使得Android应用界面变化丰富，绚丽多彩。

3.4界面布局的方式

经过上述过程，本课题需要开发的应用程序已经比较明确。在进入开发之前我们先要取得百度地图的SDK。所以要到百度官网（http://lbsyun.baidu.com/index.php?title=首页）获取开发库和API Key。

1. 首先我们需要在应用API Key注册页面完成注册（http://lbsyun.baidu.com/apiconsole/key/create）。其中要用到sha1签名和应用程序包名。sha1签名的获取方式如下：在命令行程序中定位到.android文件夹下，输入keytool –list –v –keystore debug.keystore，在获取的认证类型中选择sha1。

（2）然后在类库下载页面选择需要用到的库，如图3.1所示。****

图3.1 在下载页面选择需要使用到的类库

1. 将申请好的API Key放进AndroidMainfest文件中。

<meta-data  
 android:name="com.baidu.lbsapi.API\_KEY"  
 android:value="881ThvzyT1V4t5vtOeWsj2h8BDYxD9oB" />

1. 将下载好的库文件添加进工程中，要注意Eclipse和Android studio工程文件的结构有所不同，Android studio应该将assets文件夹复制到main文件目录下。并将BaiduLBS\_Android.jar和httpmime-4.1.2.jar两个jar文件添加到libs目录，并右键点击add as library。添加成功后会在build.gradle文件中出现：

compile files('libs/BaiduLBS\_Android.jar')  
compile files('libs/httpmime-4.1.2.jar')

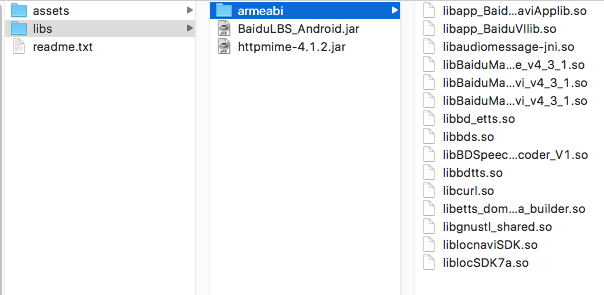


图3.2 库文件结构

1. 定位service首先需要在AndroidMainfest中声明，定位service是每个应用程序独一无二的。

<service android:name="com.baidu.location.f"  
 android:enabled="true"  
 android:process=":remote">

1. 定位相关服务组件权限声明：

<!-- 这个权限用于进行网络定位-->  
<uses-permission android:name="android.permission.ACCESS\_COARSE\_LOCATION"></uses-permission>  
<!-- 这个权限用于访问GPS定位-->  
<uses-permission android:name="android.permission.ACCESS\_FINE\_LOCATION"></uses-permission>  
<!-- 用于访问wifi网络信息，wifi信息会用于进行网络定位-->  
<uses-permission android:name="android.permission.ACCESS\_WIFI\_STATE"></uses-permission>  
<!-- 获取运营商信息，用于支持提供运营商信息相关的接口-->  
<uses-permission android:name="android.permission.ACCESS\_NETWORK\_STATE"></uses-permission>  
<!-- 这个权限用于获取wifi的获取权限，wifi信息会用来进行网络定位-->  
<uses-permission android:name="android.permission.CHANGE\_WIFI\_STATE"></uses-permission>  
<!-- 用于读取手机当前的状态-->  
<uses-permission android:name="android.permission.READ\_PHONE\_STATE"></uses-permission>  
<!-- 写入扩展存储，向扩展卡写入数据，用于写入离线定位数据-->  
<uses-permission android:name="android.permission.WRITE\_EXTERNAL\_STORAGE"></uses-permission>  
<!-- 访问网络，网络定位需要上网-->  
<uses-permission android:name="android.permission.INTERNET" />  
<!-- SD卡读取权限，用户写入离线定位数据-->  
<uses-permission android:name="android.permission.MOUNT\_UNMOUNT\_FILESYSTEMS"></uses-permission>

经过上面的准备工作，已经完成了所有的配置工作，现在可以开始各个模块的开发了。

1. Android平台的位置检索系统实现

4.1开发环境

在系统开发过程中，测试都是在真机上操作的，涉及到的机型和配置如下：

1. 开发工具。笔记本电脑一台，MacBookPro Retina，Intel Core i7四核处理器，主频2.2GHz，内存16G。配置的开发环境为macOS Sierra10.12.5+Android studio2.2.3+Android SDK。
2. 运行测试工具。Android系统手机一部，型号vivo x7，Android版本为Funtouch OS，CPU主频1.8GHz，RAM 4G，ROM 64G。

系统在上述环境中开发测试，运行流畅，性能良好。

4.2系统界面分析和运行流程

系统运行流程如图4.1：

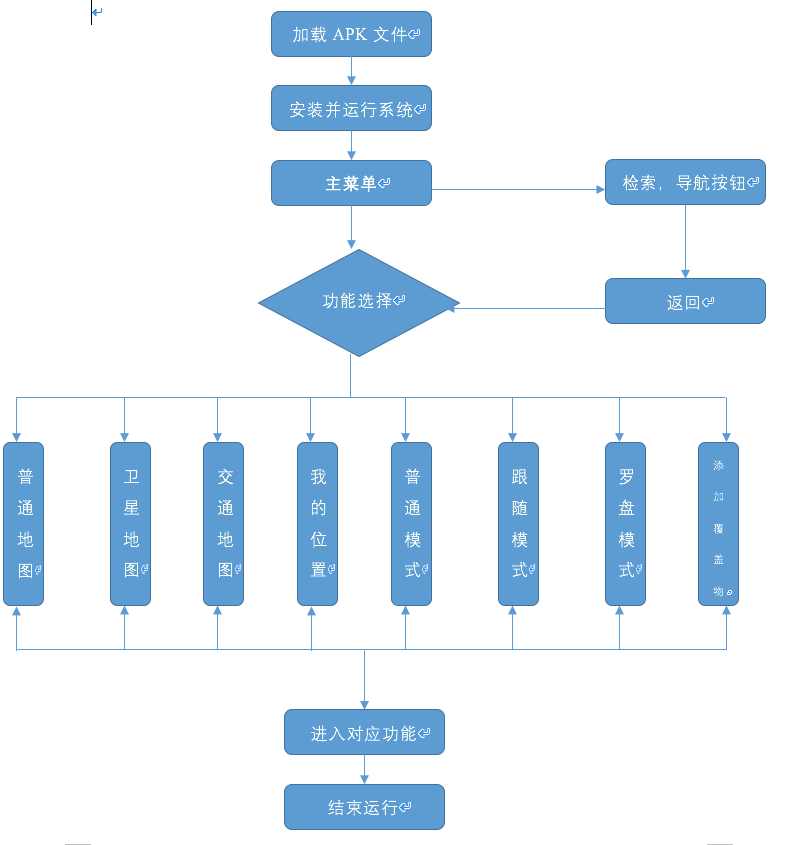


图4.1 课题位置检索系统运行流程

系统运行后界面如图4.2，系统主界面包含了位置检索和导航功能按钮，用户定位和地图模式切换集中在页面右上角功能键以保证主界面简洁。用户在进入各功能具体页面后可以使用返回键返回至主界面。

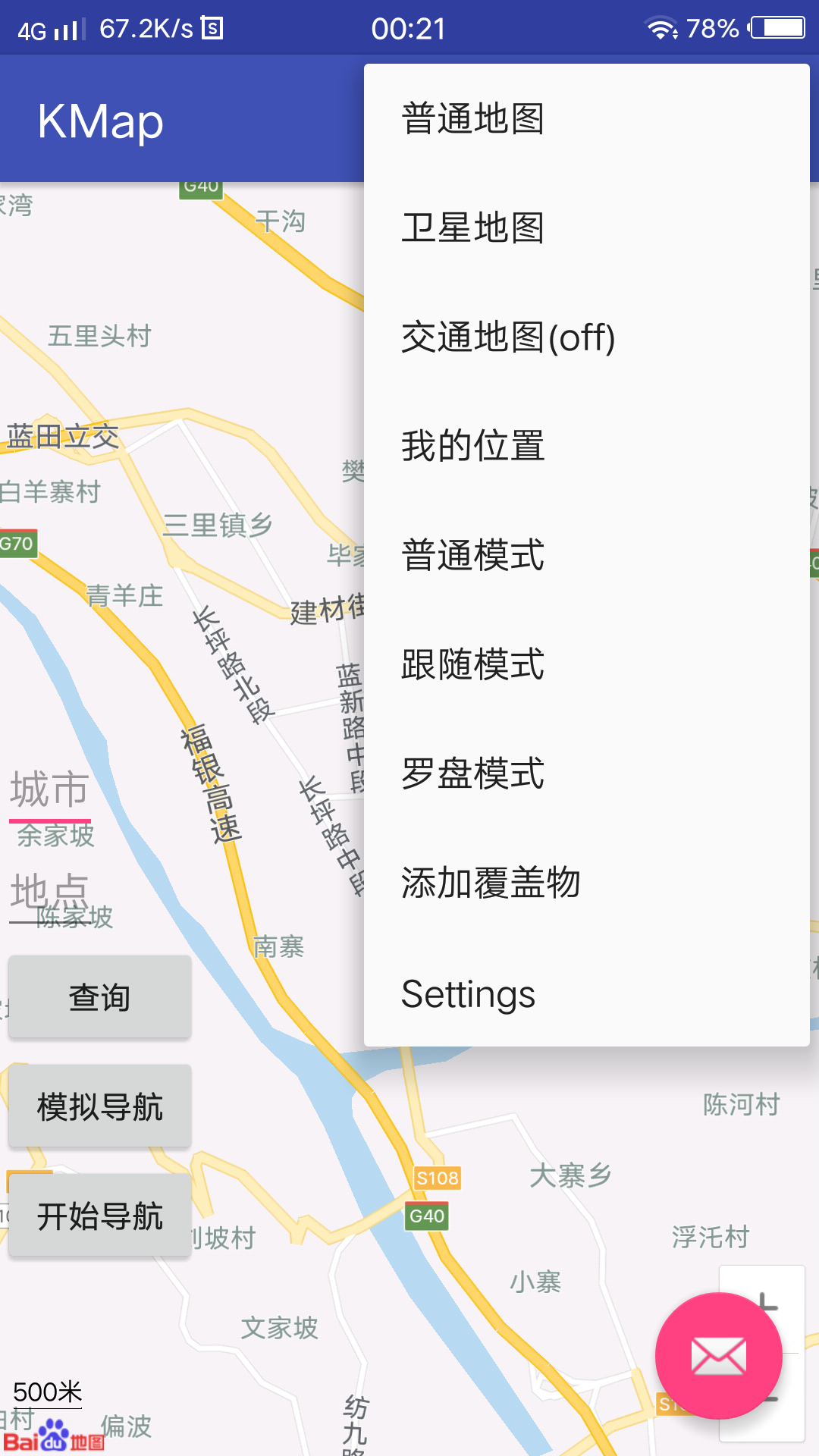
 

图4.2 系统界面展示

Android应用的主界面是应用程序中非常重要的部分，主界面是面向用户的门户，因此保证主界面简洁，功能按钮一目了然是进行系统设计的关键一步。本课题中实现主界面布局，除了使用默认的RelativeLayout之外还使用到了LinearLayout布局模式用于检索，导航按钮的放置。有以下部分代码实现：

1. 主界面类文件MainActivity，继承自AppCompatActivity实现View.OnClickListener接口。

public class MainActivity extends AppCompatActivity implements View.OnClickListener {

@Override  
protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 requestWindowFeature(Window.*FEATURE\_NO\_TITLE*);  
 SDKInitializer.*initialize*(getApplicationContext());  
 setContentView(R.layout.*activity\_main*);

1. 创建主界面的activity\_main.xml布局文件，其中包括一个用于放置查询文本输入框，和查询，导航按钮的LinearLayout组件：

<?xml version="1.0" encoding="utf-8"?>  
<android.support.design.widget.CoordinatorLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:app="http://schemas.android.com/apk/res-auto"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:fitsSystemWindows="true"  
 tools:context="com.example.stand.kmap.MainActivity">  
  
 <com.baidu.mapapi.map.MapView  
 android:id="@+id/bmapView"  
 android:layout\_width="fill\_parent"  
 android:layout\_height="fill\_parent"  
 android:clickable="true" />  
  
  
  
 <LinearLayout  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_alignParentBottom="true"  
 android:layout\_alignParentLeft="true"  
 android:orientation="vertical">  
  
 <EditText  
 android:id="@+id/id\_edit\_city"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginTop="300dp"  
 android:hint="@string/id\_edit\_city"/>  
  
 <EditText  
 android:id="@+id/id\_edit\_house"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:hint="@string/id\_edit\_house"/>  
  
 <Button  
 android:id="@+id/id\_btn\_search"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="检索"/>  
  
 <Button  
 android:id="@+id/id\_btn\_mocknav"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
  
 android:text="模拟导航"/>  
  
 <Button  
 android:id="@+id/id\_btn\_realnav"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="开始导航"/>

1. 新建menu\_main.xml文件用于放置工具栏按钮包括用户当前位置获取，地图模式切换等按钮：

<menu xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:app="http://schemas.android.com/apk/res-auto"  
 xmlns:tools="http://schemas.android.com/tools"  
 tools:context="com.example.stand.kmap.MainActivity">  
 <item  
 android:id="@+id/action\_settings"  
 android:orderInCategory="100"  
 android:title="@string/action\_settings"  
 app:showAsAction="never" />  
 <item  
 android:id="@+id/id\_map\_common"  
 app:showAsAction="never"  
 android:title="普通地图" />  
 <item  
 android:id="@+id/id\_map\_site"  
 app:showAsAction="never"  
 android:title="卫星地图" />  
 <item  
 android:id="@+id/id\_map\_traffic"  
 app:showAsAction="never"  
 android:title="交通地图(off)" />  
  
 <item  
 android:id="@+id/id\_map\_location"  
 app:showAsAction="never"  
 android:title="我的位置" />  
  
 <item  
 android:id="@+id/id\_map\_mode\_common"  
 app:showAsAction="never"  
 android:title="普通模式" />  
  
 <item  
 android:id="@+id/id\_map\_mode\_following"  
 app:showAsAction="never"  
 android:title="跟随模式" />  
  
 <item  
 android:id="@+id/id\_map\_mode\_compass"  
 app:showAsAction="never"  
 android:title="罗盘模式" />  
  
 <item  
 android:id="@+id/id\_add\_overlay"  
 app:showAsAction="never"  
 android:title="添加覆盖物">  
 </item>  
</menu>

1. 初始化地图：初始化地图视图控件MapView，并设置地图缩放比例为15.0f。

private void initView() {  
 mMapView = (MapView) findViewById(R.id.*bmapView*);  
 mBaiduMap = mMapView.getMap();  
 MapStatusUpdate msu = MapStatusUpdateFactory.*zoomTo*(15.0f);  
 mBaiduMap.setMapStatus(msu);  
}

至此基础地图已经可以完成展示，百度地图MapView集成了基础地图放大缩小Button，因此此部分基础地图已经可以完成地图的缩放功能。

4.3定位模块实现

本模块的功能在于即时获取用户的位置信息，并在地图上标识。该模块的实现流程在于先从GPS模块或移动基站模块获取用户目前的位置经纬度信息，然后转换为地图上的点，并用天剑箭头覆盖物。为了保证用户位置信息及时更新，我们设置方法option.setScanSpan(1000);即给过1s时间刷新定位信息。实现代码如下：

private void initLocation() {  
  
 mLocationMode = MyLocationConfiguration.LocationMode.*NORMAL*;  
 mLocationClient = new LocationClient(this);  
 mLocationListener = new MyLocationListener();  
 //注册  
 mLocationClient.registerLocationListener(mLocationListener);  
  
 LocationClientOption option = new LocationClientOption();  
 option.setCoorType("bd09ll");  
 option.setIsNeedAddress(true);  
 option.setOpenGps(true);  
 //刷新间隔  
 option.setScanSpan(1000);  
 mLocationClient.setLocOption(option);  
  
 //初始化图标  
 mIconLocation = BitmapDescriptorFactory.*fromResource*(R.mipmap.*arrow*);  
  
 //初始化方向监听器  
 myOrientationlistener = new MyOrientationlistener(context);  
  
 myOrientationlistener.setOnOrientationListener(new MyOrientationlistener.OnOrientationListener() {  
 @Override  
 public void onOrientationChanged(float x) {  
 mCurrentX = x;  
 }  
 });  
}

在上述代码中，我们先初始化当前地图位置获取控件Client端mLocationClient然后设置位置信息模式为百度经纬度坐标即（bd09ll），之所以选用百度经纬度坐标是因为国际经纬度坐标和国测局坐标以及墨卡托坐标在百度地图基础地图中的转化中误差较大。在上述代码中我们还注册了方向传感器模块用于获取用户即时方向信息，以用户Android设备的屏幕正向为箭头指向。

4.4地址检索模块实现

此功能是由用户输入检索目标地点信息即POI检索城市和目标地点关键词搜索目标地址并在地图上用覆盖物标识出。因为搜索功能比较耗时，且为一个阻塞式任务。因此我们采取多线程的方式，让子线程在后台获取搜索结果，完成搜索后子线程通过Handler将检索的信息发送给主线程，然后主线程负责在基础地图上展示结果以避免ANR所带来的糟糕的用户体验。

1. 首先需要初始化View.OnClickListener接口的onClick（）方法：

@Override  
public void onClick(View v) {  
 switch (v.getId())  
 {  
 case R.id.*id\_btn\_search*:  
 PoiCitySearchOption poiCitySearchOption = new PoiCitySearchOption().city(city.getText().toString()).keyword(house.getText().toString());  
 poiSearch.searchInCity(poiCitySearchOption);  
 break;  
  
 case R.id.*id\_btn\_mocknav*:  
 if (mDestLocationData == null) {  
 Toast.*makeText*(MainActivity.this, "请先设置目标地点", Toast.*LENGTH\_SHORT*).show();  
 return;  
 }  
 routeplanToNavi(false);  
 break;  
  
 case R.id.*id\_btn\_realnav*:  
 if (mDestLocationData == null) {  
 Toast.*makeText*(MainActivity.this, "请先设置目标地点", Toast.*LENGTH\_SHORT*).show();  
 return;  
 }  
 routeplanToNavi(true);  
 break;  
 }  
}

其中R.id.id\_btn\_search为检索按钮，R.id\_btn\_mocknav和R.id\_btn\_realnav分别为模拟导航和真实导航按钮，之所以设置模拟导航是方便本课题系统的功能演示。

对于检索功能我们首先初始化检索信息对象PoiCitySearchOption将用户在EditText中输入的目标检索信息转化为字符串初始化为PoiCitySearchOption的成员变量。

具体实现检索结果覆盖物展示的是OnGetSearchResultListener类的OnGetPoiResult()方法：

public class MyOnGetPoiSearchResultListener implements OnGetPoiSearchResultListener {  
 @Override  
 public void onGetPoiResult(PoiResult poiResult) {  
 // 将poi结果显示到地图上  
// PoiOverlay poiOverlay = new PoiOverlay(mBaiduMap);  
// poiOverlay.setData(poiResult);  
 if ((poiResult == null)  
 || (poiResult.error == SearchResult.ERRORNO.*RESULT\_NOT\_FOUND*)) {  
 return;  
 }  
 if (poiResult.error == SearchResult.ERRORNO.*NO\_ERROR*) {  
 mBaiduMap.clear();  
 PoiOverlay overlay = new PoiOverlay(mBaiduMap);  
 overlay.setData(poiResult);  
 overlay.addToMap();  
 overlay.zoomToSpan();  
 return;  
 }  
 if (poiResult.error == SearchResult.ERRORNO.*AMBIGUOUS\_KEYWORD*) {  
 }  
 }  
  
  
 @Override  
 public void onGetPoiDetailResult(PoiDetailResult poiDetailResult) {  
  
 }  
  
 @Override  
 public void onGetPoiIndoorResult(PoiIndoorResult poiIndoorResult) {  
  
 }  
 }

4.5路径规划和导航模块实现

路线规划是查询驾车行走路径的即时路线导航，这个功能室友一个框架布局和两个空间完成的，一个是上述位置检索控件，一个是地图显示空间。关键代码如下：

private void initNavi() {  
  
  
 BNOuterTTSPlayerCallback ttsCallback = null;  
  
 // 申请权限  
 if (Build.VERSION.*SDK\_INT* >= 23) {  
  
 if (!hasBasePhoneAuth()) {  
  
 this.requestPermissions(*authBaseArr*, *authBaseRequestCode*);  
 return;  
  
 }  
 }  
  
 BaiduNaviManager.*getInstance*().init(this, mSDCardPath, *APP\_FOLDER\_NAME*, new NaviInitListener() {  
 @Override  
 public void onAuthResult(int status, String msg) {  
 if (0 == status) {  
 authinfo = "key校验成功!";  
 } else {  
 authinfo = "key校验失败, " + msg;  
 }  
 MainActivity.this.runOnUiThread(new Runnable() {  
  
 @Override  
 public void run() {  
 Toast.*makeText*(MainActivity.this, authinfo, Toast.*LENGTH\_LONG*).show();  
 }  
 });  
 }  
  
 public void initSuccess() {  
 Toast.*makeText*(MainActivity.this, "百度导航引擎初始化成功", Toast.*LENGTH\_SHORT*).show();  
 hasInitSuccess = true;  
 initSetting();  
 }  
  
 public void initStart() {  
 Toast.*makeText*(MainActivity.this, "百度导航引擎初始化开始", Toast.*LENGTH\_SHORT*).show();  
 }  
  
 public void initFailed() {  
 Toast.*makeText*(MainActivity.this, "百度导航引擎初始化失败", Toast.*LENGTH\_SHORT*).show();  
 }  
  
 }, null, ttsHandler, ttsPlayStateListener);  
  
}

首先需要初始化百度导航引擎，在上述代码中可见，初始化导航引擎的过程是一个多线程方法，在初始化应用程序主界面的同时进行百度导航引擎的初始化，这样可以减少应用程序启动就绪时间，改善用户体验。

在导航引擎启动之后即可设置目标地点，调用导航方法，实现用户目前定位到目标位置的路线导航功能：

private void routeplanToNavi(boolean mock) {  
 BNRoutePlanNode.CoordinateType coType = BNRoutePlanNode.CoordinateType.*GCJ02*;  
 if (!hasInitSuccess) {  
 Toast.*makeText*(MainActivity.this, "还未初始化!", Toast.*LENGTH\_SHORT*).show();  
 }  
 // 权限申请  
 if (android.os.Build.VERSION.*SDK\_INT* >= 23) {  
 // 保证导航功能完备  
 if (!hasCompletePhoneAuth()) {  
 if (!hasRequestComAuth) {  
 hasRequestComAuth = true;  
 this.requestPermissions(*authComArr*, *authComRequestCode*);  
 return;  
 } else {  
 Toast.*makeText*(MainActivity.this, "没有完备的权限!", Toast.*LENGTH\_SHORT*).show();  
 }  
 }  
  
 }  
 BNRoutePlanNode sNode = null;  
 BNRoutePlanNode eNode = null;  
  
 sNode = new BNRoutePlanNode(mLongtitude, mLatitude, "我的地点", null, coType);  
 eNode = new BNRoutePlanNode(mDestLocationData.longitude, mDestLocationData.latitude, "目标地点", null, coType);  
   
 if (sNode != null && eNode != null) {  
 List<BNRoutePlanNode> list = new ArrayList<BNRoutePlanNode>();  
 list.add(sNode);  
 list.add(eNode);  
 BaiduNaviManager.*getInstance*().launchNavigator(this, list, 1, mock, new DemoRoutePlanListener(sNode));  
 }  
}

上述方法中参数mock及代表真实还是虚拟导航，在演示过程中用到的模拟导航按钮即传入了mock=false。

4.6本章小结

本课题位置检索系统采用了百度地图SDK，项目源码已上传至GitHub托管。地址：<https://github.com/Stand1210/KMap.git>。