目　录

致谢

当前文档 《Android实验指导书》 由 进击的皇虫 使用 书栈(BookStack.CN) 进行构建，生成于 2018-05-08。

书栈(BookStack.CN) 仅提供文档编写、整理、归类等功能，以及对文档内容的生成和导出工具。

文档内容由网友们编写和整理，书栈(BookStack.CN) 难以确认文档内容知识点是否错漏。如果您在阅读文档获取知识的时候，发现文档内容有不恰当的地方，请向我们反馈，让我们共同携手，将知识准确、高效且有效地传递给每一个人。

同时，如果您在日常生活、工作和学习中遇到有价值有营养的知识文档，欢迎分享到 书栈(BookStack.CN) ，为知识的传承献上您的一份力量！

如果当前文档生成时间太久，请到 书栈(BookStack.CN) 获取最新的文档，以跟上知识更新换代的步伐。

文档地址: http://www.bookstack.cn/books/android-experiment-guide

书栈官网: http://www.bookstack.cn

书栈开源: https://github.com/TruthHun

分享，让知识传承更久远！ 感谢知识的创造者，感谢知识的分享者，也感谢每一位阅读到此处的读者，因为我们都将成为知识的传承者。

引言

引言

可以作为高年级本科实验教学指导书。

实验方案

课时及实验内容设置。

参与编写人员

1. @zengsn （发起人，指导老师）
2.

来源(书栈小编注)

> *https://github.com/zengsn/android-experiment-guide*

游戏开发

游戏开发

1. 黑白块

完整代码参见：https://github.com/hzuapps/android-labs/tree/master/app/src/main/java/edu/hzuapps/androidworks/homeworks/net1314080903206

实验1：搭建开发环境

实验1: 搭建开发环境

1. 在Windows下搭建Android Studio开发环境

简要说明.....

步骤.....

2. 在Windows下搭建Eclipse开发环境

简要说明.....

步骤.....

截图

实验2：Android组件编程

实验2: Android组件编程

实验3：Android资源使用

实验3: Android资源使用

实验4：Android界面设计

实验4: Android界面设计

8.1 知识点

1.

2.

8.2 实例讲解

1. 设计仿微信导航界面

简要说明.....

详细步骤.....

2. 制作黑白棋游戏界面

简要说明:我做的是一个黑白棋的应用，在主界面是用垂直布局来弄的，中间会涉及到几个相对布局和表格布局，还是相对来说比较简单

的。

详细步骤：

1. android:background=”@drawable/net13140803138background” //一开始用了这一句调用了drawable里的背景图片作为背景。
2. android:orientation=”vertical”//然后这里是说明我是垂直方向进行线性布局。
3. 然后我就开始按照布局放控件了，其中：

   <ImageView

```
android:layout_width="match_parent"
android:layout_weight="1"
android:layout_height="0dp"
android:scaleType="fitXY"
android:id="@+id/top1"
android:src="@drawable/net13140803138top"/>
```

   这一部分代码即我最上面的一个ImageView，调用了drawable/net13140803138top图片。

```
<ImageView
    android:layout_width="0dp"
    android:layout_weight="2"
    android:layout_height="match_parent"
    android:scaleType="fitCenter"
    android:src="@drawable/net13140803138daojian"
    />
```

```
</LinearLayout>
```

这里我是用了一个水平方向的线性布局，这里值得一提的是，我放了3个位置，但是我只放了一个控件，调用了drawable/net13140803138daojian图片，如果我后面增加了新的

功能，可以有位置添加进去。

1. 这里则是由12个黑白棋子组成的一个布局，由12个黑白棋图片组成，代码会有点长，但是代码其实是比较简单的

```
<GridLayout
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:rowCount="3"
    android:columnCount="4"
    android:layout_centerInParent="true"
    >

    <ImageView
        android:id="@+id/iv1"
        android:layout_width="@dimen/width_of_chess1"
android:layout_margin="@dimen/chess_margin1"
        android:layout_height="@dimen/width_of_chess1"
        android:scaleType="fitCenter"
        android:src="@drawable/net13140803138white_chess"/>

    <ImageView
        android:id="@+id/iv2"
        android:layout_width="@dimen/width_of_chess1"
android:layout_margin="@dimen/chess_margin1"
        android:scaleType="fitCenter"
        android:layout_height="@dimen/width_of_chess1"
        android:src="@drawable/net13140803138black_chess"
        />

    <ImageView
        android:id="@+id/iv3"
        android:layout_width="@dimen/width_of_chess1"
android:layout_margin="@dimen/chess_margin1"
        android:scaleType="fitCenter"
        android:layout_height="@dimen/width_of_chess1"
        android:src="@drawable/net13140803138black_chess"
        />
```

```
31.
32.     <ImageView
33.         android:id="@+id/iv4"
34.         android:layout_width="@dimen/width_of_chess1"
    android:layout_margin="@dimen/chess_margin1"
35.         android:scaleType="fitCenter"
36.         android:src="@drawable/net13140803138white_chess"
37.         android:layout_height="@dimen/width_of_chess1" />
38.
39.     <ImageView
40.         android:id="@+id/iv5"
41.         android:layout_width="@dimen/width_of_chess1"
    android:layout_margin="@dimen/chess_margin1"
42.         android:scaleType="fitCenter"
43.         android:layout_height="@dimen/width_of_chess1"
44.         android:src="@drawable/net13140803138black_chess"/>
45.
46.     <ImageView
47.         android:id="@+id/iv6"
48.         android:layout_width="@dimen/width_of_chess1"
    android:layout_margin="@dimen/chess_margin1"
49.         android:scaleType="fitCenter"
50.         android:layout_height="@dimen/width_of_chess1"
51.         android:src="@drawable/net13140803138white_chess"
52.         />
53.
54.     <ImageView
55.         android:id="@+id/iv7"
56.         android:layout_width="@dimen/width_of_chess1"
    android:layout_margin="@dimen/chess_margin1"
57.         android:scaleType="fitCenter"
58.         android:layout_height="@dimen/width_of_chess1"
59.         android:src="@drawable/net13140803138white_chess"
60.         />
61.
62.     <ImageView
63.         android:id="@+id/iv8"
64.         android:layout_width="@dimen/width_of_chess1"
```

```
    android:layout_margin="@dimen/chess_margin1"
65.         android:scaleType="fitCenter"
66.         android:src="@drawable/net13140803138black_chess"
67.         android:layout_height="@dimen/width_of_chess1" />
68.
69.     <ImageView
70.         android:id="@+id/iv9"
71.         android:layout_width="@dimen/width_of_chess1"
    android:layout_margin="@dimen/chess_margin1"
72.         android:scaleType="fitCenter"
73.         android:layout_height="@dimen/width_of_chess1"
74.         android:src="@drawable/net13140803138black_chess"/>
75.
76.     <ImageView
77.         android:id="@+id/iv10"
78.         android:layout_width="@dimen/width_of_chess1"
    android:layout_margin="@dimen/chess_margin1"
79.         android:scaleType="fitCenter"
80.         android:layout_height="@dimen/width_of_chess1"
81.         android:src="@drawable/net13140803138white_chess"
82.         />
83.
84.     <ImageView
85.         android:id="@+id/iv11"
86.         android:scaleType="fitCenter"
87.         android:layout_width="@dimen/width_of_chess1"
    android:layout_margin="@dimen/chess_margin1"
88.         android:layout_height="@dimen/width_of_chess1"
89.         android:src="@drawable/net13140803138black_chess"
90.         />
91.
92.     <ImageView
93.         android:id="@+id/iv12"
94.         android:layout_width="@dimen/width_of_chess1"
    android:layout_margin="@dimen/chess_margin1"
95.         android:scaleType="fitCenter"
96.         android:src="@drawable/net13140803138white_chess"
97.         android:layout_height="@dimen/width_of_chess1" />
```

```
    </GridLayout>

</RelativeLayout>
```

这一段的代码我是用了一个相对布局，里面在用一个表格布局放在这个相对布局的中间，调用了12个ImageView放在表格布局里面，其中，调用了drawable/net13140803138white_chess和drawable/net13140803138black_chess两个黑白棋子的图片。

1. 这里我要加一个声音的开关放在右下角，所以要加一个相对布局

```
<CheckBox
    android:id="@+id/soundSwitch"
    android:layout_alignParentRight="true"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Sound"
    android:textColor="#FF000000"/>
```

所以这里我用了一个相对布局，然后放一个CheckBox在这相对布局里面。

2. <ImageButton

```
android:layout_width="match_parent"
android:layout_height="0dp"
android:layout_weight="1"
android:contentDescription="@string/app_name"
android:id="@+id/startButton"
android:src="@drawable/net13140803138start"
android:scaleType="fitXY"
android:background="#00000000"
android:layout_marginBottom="6dp"/>
```

这里也是调用了一个ImageButton作为一个开始按钮，其中调用了drawable/net13140803138start图片。
至此，我的主界面就完成了。
```

```
###3. 制作练练看游戏界面
简要说明：海贼王连连看游戏主界面是由Net1314080903126main.xml 文件通过调用Net1314080903126CtrlView.java Net1314080903126GameView.java Net13140809031260nePieceGame.java 来显示主界面的以及游戏规则和游戏玩法的等。
详细步骤：
1.   游戏的主界面是通过Net1314080903126main.xml 来显示的，代码：
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
android:layout_width="match_parent"
    android:layout_height="wrap_content">
    <TableLayout
xmlns:android="http://schemas.android.com/apk/res/android"
         android:layout_width="fill_parent"
android:layout_height="wrap_content">
         <TableRow>
            <ProgressBar android:id="@+id/pb"
android:layout_width="fill_parent"
                android:layout_height="wrap_content" style="?
android:attr/progressBarStyleHorizontal"
                android:layout_weight="9"/>
            <TextView android:layout_height="wrap_content"
android:layout_width="wrap_content"
                android:text="@string/remain_time"
android:layout_weight="1"/>
            <TextView android:layout_height="wrap_content"
android:layout_width="wrap_content"
                android:id="@+id/show_remainTime"
android:layout_weight="1"/>
        </TableRow>
```
```

```
21.     </TableLayout>
22.
23.
    <edu.hzuapps.androidworks.homeworks.net1314080903126.Net1314080903126
24.         android:id="@+id/cv"
25.         android:layout_width="wrap_content"
    android:layout_height="fill_parent" />
26. </LinearLayout>
27.
28. 其中：
    <edu.hzuapps.androidworks.homeworks.net1314080903126.Net1314080903126
29.         android:id="@+id/cv"
30.         android:layout_width="wrap_content"
    android:layout_height="fill_parent" />
31.         这是显示海贼王人物的以及人物界面布局。
32. 然后：<TableRow>
33.             <ProgressBar android:id="@+id/pb"
    android:layout_width="fill_parent"
34.                 android:layout_height="wrap_content" style="?
    android:attr/progressBarStyleHorizontal"
35.                 android:layout_weight="9"/>
36.             <TextView android:layout_height="wrap_content"
    android:layout_width="wrap_content"
37.                 android:text="@string/remain_time"
    android:layout_weight="1"/>
38.             <TextView android:layout_height="wrap_content"
    android:layout_width="wrap_content"
39.                 android:id="@+id/show_remainTime"
    android:layout_weight="1"/>
40.         </TableRow>
41.         这是显示时间进度条的，以及时间 剩余时间（秒）和倒计时300.
42.         这是海贼王连连看的界面的大致介绍。
43. 2.  下面详细解释一下海贼王连连看。
44.  (1) .  public class Net1314080903126GameView extends View {
45.
46.     public final int row = 10;
47.     public final int col = 10;
48.     public float width;
```

```
49.     public float height;
50.     private int selY;
51.     private int selX;
52.     public boolean isLine = false;
53.     public int grid[][] = new int[row][col];
54.     private Rect selRect = new Rect();
55.     public int lineType = 0;
56.     public final int V_LINE = 1;
57.     public final int H_LINE = 1;
58.     public final int ONE_C_LINE = 2;
59.     public final int TWO_C_LINE = 3;
60.     public int much = 0;
61.     Point[] p;
62.     public int[] imageType = new int[] {
    R.drawable.net1314080903126aa, R.drawable.net1314080903126bb,
63.             R.drawable.net1314080903126cc,
    R.drawable.net1314080903126dd, R.drawable.net1314080903126ee,
    R.drawable.net1314080903126ff,
64.             R.drawable.net1314080903126gg,
    R.drawable.net1314080903126hh, R.drawable.net1314080903126ii,
    R.drawable.net1314080903126jj,
65.             R.drawable.net1314080903126kk,
    R.drawable.net13140809031261l, R.drawable.net1314080903126mm,
    R.drawable.net1314080903126nn,
66.             R.drawable.net1314080903126oo,
    R.drawable.net1314080903126pp};
67.     public Bitmap[] image;
68.     public List<Integer> type = new ArrayList<Integer>();
69.     这个是Net1314080903126GameView.java文件中，调用图片。public final
    int row = 10;
70.     public final int col = 10;图片的布局分布十行十列显示。以及显示大小尺
    寸。调用drawable文件下的图片显示在界面上。
71.
72. (2). <TextView android:layout_height="wrap_content"
    android:layout_width="wrap_content"
73.                 android:text="@string/remain_time"
    android:layout_weight="1"/>
74.     这是显示剩余时间（秒）
```

```
75. (3). <ProgressBar android:id="@+id/pb"
    android:layout_width="fill_parent"
76.                 android:layout_height="wrap_content" style="?
    android:attr/progressBarStyleHorizontal"
77.                 android:layout_weight="9"/>
78.     这是显示倒计时时间进度条的。
79. (4). <TextView android:layout_height="wrap_content"
    android:layout_width="wrap_content"
80.                 android:id="@+id/show_remainTime"
    android:layout_weight="1"/>
81.         这是显示300时间的跑秒。
82. (5).public boolean onCreateOptionsMenu(Menu menu) {
83.         // TODO Auto-generated method stub
84.         menu.add(0, START_ID, 0, R.string.newgame);
85.         menu.add(0, REARRARY_ID, 0, R.string.rearrage);
86.         menu.add(0, END_ID, 0, R.string.exit);
87.         return super.onCreateOptionsMenu(menu);
88.     }
89.     这是页面上方选项菜单，可以选择（新游戏，重排列以及退出游戏）这三个功能。
90. (6).public AlertDialog dialogForSucceed() {
91.         AlertDialog.Builder builder = new
    AlertDialog.Builder(this);
92.
    builder.setIcon(R.drawable.icon).setMessage(R.string.succeedInfo)
93.                 .setPositiveButton(R.string.next,
94.                         new DialogInterface.OnClickListener() {
95.                             @Override
96.                             public void onClick(DialogInterface
    dialog,
97.                                     int which) {
98.                                 // TODO Auto-generated method stub
99.                                 dormant = dormant - 300;
100.                                newPlay();
101.                            }
102.
    }).setNeutralButton(R.string.again_challenge,
103.                        new DialogInterface.OnClickListener() {
104.                            @Override
```

```
105.                                public void onClick(DialogInterface
    dialog,
106.                                        int which) {
107.                                    // TODO Auto-generated method stub
108.                                    newPlay();
109.                                }
110.                            });
111.            return builder.create();
112.        }
113.
114.        public AlertDialog dialogForFail() {
115.            AlertDialog.Builder builder = new
    AlertDialog.Builder(this);
116.
    builder.setIcon(R.drawable.icon).setMessage(R.string.failInfo)
117.                    .setPositiveButton(R.string.again_challenge,
118.                            new DialogInterface.OnClickListener() {
119.                                @Override
120.                                public void onClick(DialogInterface
    dialog,
121.                                        int which) {
122.                                    // TODO Auto-generated method stub
123.                                    newPlay();
124.                                }
125.                            }).setNegativeButton(R.string.exit,
126.                            new DialogInterface.OnClickListener() {
127.                                @Override
128.                                public void onClick(DialogInterface
    dialog,
129.                                        int which) {
130.                                    // TODO Auto-generated method stub
131.                                    isCancel=false;
132.                                    finish();
133.                                }
134.                            });
135.            return builder.create();
136.        }
137.        这是显示300秒内完成游戏提示通往下一关，以及没有完成游戏提示你不气馁。
```

```
以上就是海贼王练练看看游戏界面的详细实现过程。


###4. 制作扫雷游戏界面
简要说明：这里介绍的是扫雷游戏的初始界面和游戏进行界面的实现方法，它是通过获取每个格子的id值来调用显示相应的背景图片。
详细步骤：
1. 先获取每个格子对象，再给每个格子对象编号或下标获取id值，代码如下：
```

```
public int getCount() {
return levellevel;
}
/*
```

```
 * 方法：获取每个格子对象
 * @param position 格子编号，位置下标
 * @return 格子类型的GameGroundEntity
 * */
@Override
public GridEntity getItem(int position) {
```

```
// 调用GameGroundEntity中的getEntity方法获取格子对象
return gameGround.getEntity(position);
}
/**
```

```
 * 方法：通过适配器给每个格子对象编号或下标获取id值
 * @return long类型，在java中，byte和short可自动转换为int，int可自动转换为long
 * */
@Override
public long getItemId(int position) {
    return position;
}
```

```
2.设置格子对象的背景图片， 不同状态下设置不同的背景图片（i00,i13等是图片的名字）
```

@param grid :格子对象

```
 * */
public int getRes(GridEntity grid){
```

```
// 设置格子对象的背景图片的ID为0
int resID=0;
// 判断，如果格子对象被标记了且标记正确
if(grid.isFlag()&&!grid.isFlagWrong()){
resID=R.drawable.i_flag;
}
// 判断，如果格子对象被标记了但标记不正确
else if(grid.isFlag()&&grid.isFlagWrong()){
resID=R.drawable.i14;
}
// 判断，如果格子对象没有被点击，isShow()属性为false
else if(!grid.isShow()){
resID=R.drawable.i00;
}
// 判断，格子对象是地雷且非自动显示
else if(grid.isBoom()&&!grid.isAutoShow()){
resID=R.drawable.i13;
}
// 判断，格子对象是地雷，自动显示
else if(grid.isBoom()&&grid.isAutoShow()){
resID=R.drawable.i12;
```

```
}
// 判断，格子周围没有地雷，是空白格
else if(grid.getBoomsCount()==0){
resID=R.drawable.i09;
}
// 判断，格子中卫有地雷，个数为1-8个
else if(grid.getBoomsCount()!=0){
// 动态拼接图片名，格式为图片名称，图片类型，资源所在包名
resID=context.getResources().getIdentifier(“i0”+grid
.getBoomsCount(),”drawable”,context.getPackageName()
);
}
return resID;
}
```

1. 源代码：!https://github.com/hzuapps/android-labs/tree/master/app/src/main/java/edu/hzuapps/androidworks/homeworks
2.
3.
4. ###5. 制作围住神经猫游戏界面
5. 简要说明：简单版的围住神经猫游戏的背景是一排排的圆圈，可以用二维数组创建，而这些圆圈有三种状态，分别为猫可以走并可以设置路障的位置（灰色），猫所处的位置（红色），猫不能走并已开启路障的位置（橘色），这些状态定义在Dot类中，显然其应该有坐标X和Y；游戏的背景等我都放在Playground类中，背景是基于SurfaceView开发的，而里面还有很多功能，如猫和已设路障的初始化位置，猫如何躲路障，如何判断路可走，判断处在边界位置等等；最后在Activity中调用Playground。
6.
7. 详细步骤：
8. 1. 建立Dot类，用来记录每个场景中的元素它的X,Y坐标点的状态。并不会直接参与界面的响应和界面的绘制。每一个点都是一个抽象的对象，需要把每一个点抽象为一个类，然后让每一个圆圈继承于这个类，或者直接把它实现为这个类的实例。每个点有三个状态：灰色-猫可走的路径；橘色-路障的状态
9. 无法改变；红色-猫当前的位置。

public class Net13140809031420t {

```
1.  int x,y;//当前点的X, Y坐标
2.  int status;//记录这个点的状态
3.  //三个表征圆点状态静态常量
4.  public static final int STATUS_ON = 1;//已经开启路障的状态
5.  public static final int STATUS_OFF = 0;//代表灰色可走路径
6.  public static final int STATUS_IN = 9;//猫当前的位置
7.  //三个数字不同即可，具体用哪个数字无所谓
8.
9.
10.    //指定X, Y坐标
11. public Net13140809031420t(int x, int y) {
12.     super();
13.     this.x = x;
14.     this.y = y;
15.     status = STATUS_OFF;
16. }
17.
18.    //指定geter和sette方法
19. public int getX() {
20.     return x;
21. }
22.
23. public void setX(int x) {
24.     this.x = x;
25. }
26.
27. public int getY() {
28.     return y;
29. }
30.
31. public void setY(int y) {
32.     this.y = y;
33. }
34.
35. public int getStatus() {
```

```
36.     return status;
37. }
38. 
39. public void setStatus(int status) {
40.     this.status = status;
41. }
42. 
43. //同时设置X, Y坐标的方法
44. public void setXY(int x,int y) {
45.     this.y = y;
46.     this.x = x;
47. }
```

}

```
1. 
2. 2.建立Playground类，用来绘制界面还有实现游戏的各种算法。以下只详细介绍界面的绘制还有每个圆圈状态转化的过程。
```

public class Net1314080903142Playground extends SurfaceView implements OnTouchListener{
//界面的响应和界面的绘制在SurfaceView完成，触摸事件的响应通过OnTouchListener接口实现

```
1. private static  int WIDTH = 40;
2. private static final int ROW = 10;//行高：每行储存10个元素
3. private static final int COL = 10;//列宽：每列储存10个元素
4. private static final int BLOCKS = 15;//默认添加的路障数量
5. 
6. 
7. private Dot Net1314080903142matrix[][];//声明二维数组来存放点元素
8. private Dot Net1314080903142cat;//声明猫这个点
9. 
10. public Net1314080903142Playground(Context context) {
11.     super(context);//使用Context创建当前类构造函数
12.     getHolder().addCallback(callback);//将Callback对象指定给getholder
```

```
13.     matrix = new Net1314080903142Dot[ROW][COL];//将行高，列宽传递进去，指定数组大小
14.     for (int i = 0; i < ROW; i++) {//循环添加数据
15.         for (int j = 0; j < COL; j++) {
16.             matrix[i][j] = new Net1314080903142Dot(j, i);/*X，Y坐标值和行列值是相反的。
17.             即通过查找列值获得X坐标，查找行值获得Y坐标*/
18.         }
19.     }
20.     setOnTouchListener(this);//设定为自己的触摸监听器
21.     initGame();//调用游戏初始化
22. }
23.
24.
25. //坐标反转：封装一个getDot函数实现X，Y坐标反过来传递，所有的操作通过X，Y调用
26. private Net1314080903142Dot getDot(int x,int y) {
27.     return matrix[y][x];
28. }
29.
30.
31.    //实现猫移动到下一个点
32.    private void MoveTo(Net1314080903142Dot one) {
33.     one.setStatus(Net1314080903142Dot.STATUS_IN);//one的状态设置为猫所处的点
34.     getDot(cat.getX(), cat.getY()).setStatus(Net1314080903142Dot.STATUS_OFF);;//将猫当前点的状态复位
35.     cat.setXY(one.getX(), one.getY());//将猫移动到新的点
36. }
37. //猫的移动
38. private void move() {
39.     if (isAtEdge(cat)) {
40.         lose();return;/猫处于游戏边缘，失败
41.     }
42.     Vector<Net1314080903142Dot> avaliable = new Vector<Net1314080903142Dot>();//avaliable容器记录可用点
43.     Vector<Net1314080903142Dot> positive = new Vector<Net1314080903142Dot>();//positive容器记录这个方向上可以直接到达屏
```

```
    幕边缘的路径
44.     HashMap<Net1314080903142Dot, Integer> al = new
    HashMap<Net1314080903142Dot, Integer>();//al容器记录方向
45.     for (int i = 1; i < 7; i++) {//如果当前猫被6个邻点围住
46.         Net1314080903142Dot n = getNeighbour(cat, i);
47.         if (n.getStatus() == Net1314080903142Dot.STATUS_OFF) {
48.             avaliable.add(n);//如果相邻点可用，把它添加到avaliable记录器
    中
49.             al.put(n, i);//为al传入方向i
50.             if (getDistance(n, i) > 0) {
51.                 positive.add(n);//当它有一个路径可以直接到达屏幕边缘，把n
    传递进positive中
52.
53.             }
54.         }
55.     }
56.             //移动算法的优化
57.     if (avaliable.size() == 0) {
58.         win();//周围的6个点都不可走，没有可用点，成功围住猫
59.     }else if (avaliable.size() == 1) {
60.         MoveTo(avaliable.get(0));//只有一个方向可走，可用点有一个，移动到
    这个可用点上
61.     }else{//有多个方向可走
62.         Net1314080903142Dot best = null;
63.         if (positive.size() != 0 ) {//存在可以直接到达屏幕边缘的走向
64.             System.out.println("向前进");
65.             int min = 999;//999远大于场景中的所有可用步长，其他数也可
66.             for (int i = 0; i < positive.size(); i++) {
67.                 int a = getDistance(positive.get(i),
    al.get(positive.get(i)));
68.                 if (a < min) {
69.                     min = a;//把最短路径长度传给min
70.                     best = positive.get(i);//选出拥有最短路径的点
71.                 }
72.             }
73.             MoveTo(best);
74.         }else {//所有方向都存在路障
75.             System.out.println("躲路障");
```

```
76.             int max = 0;
77.             for (int i = 0; i < avaliable.size(); i++) {
78.                 int k = getDistance(avaliable.get(i),
    al.get(avaliable.get(i)));
79.                 if (k <= max) {//所有方向都存在路障，距离k为负数
80.                     max = k;
81.                     best = avaliable.get(i);//选出拥有最短路径的点
82.                 }
83.             }
84.             MoveTo(best);//移动到最短路径的下一点
85.         }
86.     }
87. }
88.
89.
90.
91. //实现界面绘制，在redraw方法中将所有元素以图形化显示出来，也就是将它绘制在
    Canvas对象上
92.     private void redraw() {
93.     Canvas c = getHolder().lockCanvas();//锁定画布
94.     c.drawColor(Color.LTGRAY);//设置颜色为浅灰色
95.     Paint paint = new Paint();//创建Paint对象
96.     paint.setFlags(Paint.ANTI_ALIAS_FLAG);//开启抗锯齿，优化视频质量
97.
98.             //用两个For循环嵌套将所有的点显示到界面中来
99.     for (int i = 0; i < ROW; i++) {
100.         int offset = 0;//引入偏移量
101.         if (i%2 != 0) {
102.             offset = WIDTH/2;//对偶数行进行缩进
103.         }
104.         for (int j = 0; j < COL; j++) {
105.             Net13140809031420t one = getDot(j, i);//将坐标赋值给内部
    变量one
106.                                 //由于每个点对应的三种状态颜色不一样，要用一个
    switch语句
107.             switch (one.getStatus()) {
108.             case Net13140809031420t.STATUS_OFF:
109.                 paint.setColor(0xFFEEEEEE);//STATUS_OFF状态时设置颜色
```

```
为浅灰色
                    break;
                case Net1314080903142Dot.STATUS_ON:
                    paint.setColor(0xFFFFAA00);//STATUS_ON状态时设置颜色为
橘色
                    break;
                case Net1314080903142Dot.STATUS_IN:
                    paint.setColor(0xFFFF0000);//STATUS_IN状态时设置颜色为
红色
                    break;

                default:
                    break;
                }
                c.drawOval(new RectF(one.getX()*WIDTH+offset,
one.getY()*WIDTH,
                        (one.getX()+1)*WIDTH+offset,
(one.getY()+1)*WIDTH), paint);
                                /*在Canvas画布上画椭圆并界定它的上下左右边界
宽度且有错位*/
                            }

        }
        getHolder().unlockCanvasAndPost(c);//取消Canvas的锁定，吧绘图内容更
新到界面上
    }

    //为Surfaceview添加Callback
    Callback callback = new Callback() {//声明并实例化一个Callback接口

        @Override
        public void surfaceDestroyed(SurfaceHolder arg0) {
            // TODO Auto-generated method stub

        }

        @Override
        public void surfaceCreated(SurfaceHolder arg0) {
```

```
141.          // TODO Auto-generated method stub
142.          redraw();//执行redraw函数，在界面第一次显示时将指定的内容显示到界面
      上
143.      }
144.
145.      @Override
146.              //使用surfaceChanged方法来适配不同的屏幕尺寸
147.      public void surfaceChanged(SurfaceHolder arg0, int arg1, int
      arg2, int arg3) {
148.
149.                      //surfacechanged方法包含四个参数：SurfaceHolder
      holder, int format, int width, int height
150.          // TODO Auto-generated method stub
151.          WIDTH = arg2/(COL+1);//需要修改width，即arg2。
152.          redraw();//重绘界面
153.      }
154.  };
155.  //游戏初始化：分别对可走路径位置，猫的位置和路障位置进行初始化
156.  private void initGame() {
157.              //用for循环将所有点设置为STATUS_OFF，即可用状态
158.      for (int i = 0; i < ROW; i++) {
159.          for (int j = 0; j < COL; j++) {
160.              matrix[i][j].setStatus(Net1314080903142Dot.STATUS_OFF);
161.          }
162.      }
163.      cat = new Net1314080903142Dot(4, 5);//设置猫的起始点
164.      getDot(4, 5).setStatus(Net1314080903142Dot.STATUS_IN);//把猫的起
      始点的状态设置为STATUS_IN，才能记录猫的位置
165.
166.              //用for循环随机的指定15个点的坐标作为路障
167.      for (int i = 0; i < BLOCKS;) {
168.          int x = (int) ((Math.random()*1000)%COL);
169.          int y = (int) ((Math.random()*1000)%ROW);//随机获取1对坐标点
170.          if (getDot(x, y).getStatus() ==
      Net1314080903142Dot.STATUS_OFF) {//对当前可用路径点进行选择
171.              getDot(x,
      y).setStatus(Net1314080903142Dot.STATUS_ON);//并把这个点设置为路障
172.              i++;//循环内自加避免当前路障被重复添加
```

```
173.             //System.out.println("Block:"+i);
174.         }
175.     }
176. }
177. 
178. @Override
179.     //触摸事件的处理
180. public boolean onTouch(View arg0, MotionEvent e) {
181.     if (e.getAction() == MotionEvent.ACTION_UP) {/当用户触摸之后手离开屏幕释放的瞬间才对事件进行响应
182.          //    Toast.makeText(getContext(), e.getX()+":"+e.getY(), Toast.LENGTH_SHORT).show();
183.                     //将屏幕的坐标转换为游戏的坐标
184.         int x,y;
185.         y = (int) (e.getY()/WIDTH);//横向状态下，奇、偶数行有坐标偏移，而纵向的Y值是不变的，将y进行转换
186.         if (y%2 == 0) {
187.             x = (int) (e.getX()/WIDTH);//奇数行直接将屏幕的X坐标转换成游戏的X坐标
188.         }else {
189.             x = (int) ((e.getX()-WIDTH/2)/WIDTH);//偶数行偏移半个元素宽度，故需减去WIDTH/2
190.         }
191.                     //数组越界异常时，对坐标进行保护
192.         if (x+1 > COL || y+1 > ROW) {
193.             initGame();//触摸超出边界时初始化游戏
194.         }else if(getDot(x, y).getStatus() == Net13140809031420Dot.STATUS_OFF){
195.             getDot(x, y).setStatus(Net13140809031420Dot.STATUS_ON);//当这个点可用时被点击之后设定为路障状态
196.             move();
197.         }
198.         redraw();//将改变更新到界面
199.     }
200.     return true;
201. }
```

```
}
```

```
3.最后创建Activity,调用Playground。
```

```
public class Net1314080903142Activity extends
Activity {
```

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);

    setContentView(new Net1314080903142Playground(this));//Conte
}
```

```
}
```

```
备注：实现该游戏的其他算法未列出，有兴趣的可以在该网站看全部代码：https://g

###6. 自动滚动Banner图片

####简要说明：
本程序实现的是图片轮播banner，主要运用得到的控件为ViewPager,具体有以下功能

>a.定时功能，每隔5S切换下一张图片。

>b.手动切换，可手势左右滑动选择上一张或下一张图片。
>
>c.跳转功能，点击跳转对应Activity

####详细步骤

1.指示器圆点，有两个状态，分别为选中和为选中。用两个shape实现。存放在drawab
```

```
- 未选中状态：net1314080903118_dot_normal.xml

```xml

    <?xml version="1.0" encoding="utf-8"?>
    <shape xmlns:android="http://schemas.android.com/apk/res/and
        android:shape="oval" >

        <solid android:color="#33000000" />

        <corners android:radius="5dip" />

    </shape>
```

- 选中状态：net1314080903118_dot_press.xml

```
    <?xml version="1.0" encoding="utf-8"?>
    <shape xmlns:android="http://schemas.android.com/apk/res/and
        android:shape="oval" >

        <solid android:color="#aaFFFFFF" />

        <corners android:radius="5dip" />

    </shape>
```

2.布局文件，主要用于显示图片的是ViewPager,九个View分别对应九个圆点，即可同时存放九张图片用于轮播。View的 `background` 使用上面定义的选中未选中状态。这里我单独抽取出来，使用时在主布局文件用 `include` 引入即可。

- net1314080903118_banner.xml

```
```
```

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="vertical" >

    <FrameLayout
        android:layout_width="match_parent"
        android:layout_height="160dp" >

        <android.support.v4.view.ViewPager
            android:id="@+id/vp"
            android:layout_width="match_parent"
            android:layout_height="160dp" />


        <LinearLayout
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginTop="10dip"
            android:layout_gravity="bottom|center_horizontal
            android:layout_marginBottom="10dp"
            android:gravity="center" >

            <View
                android:id="@+id/v_dot0"
                android:layout_width="5dip"
                android:layout_height="5dip"
                android:background="@drawable/net131408090311
                android:layout_marginLeft="1.5dip"
                android:layout_marginRight="1.5dip"
                android:visibility="invisible" />

            <View
                android:id="@+id/v_dot1"
                android:layout_width="5dip"
                android:layout_height="5dip"
                android:background="@drawable/net131408090311
```

```
            android:layout_marginLeft="1.5dip"
            android:layout_marginRight="1.5dip"
            android:visibility="invisible"/>

        <View
            android:id="@+id/v_dot2"
            android:layout_width="5dip"
            android:layout_height="5dip"
            android:background="@drawable/net13140809031
            android:layout_marginLeft="1.5dip"
            android:layout_marginRight="1.5dip"
            android:visibility="invisible"/>

        <View
            android:id="@+id/v_dot3"
            android:layout_width="5dip"
            android:layout_height="5dip"
            android:background="@drawable/net13140809031
            android:layout_marginLeft="1.5dip"
            android:layout_marginRight="1.5dip"
            android:visibility="invisible"/>

        <View
            android:id="@+id/v_dot4"
            android:layout_width="5dip"
            android:layout_height="5dip"
            android:background="@drawable/net13140809031
            android:layout_marginLeft="1.5dip"
            android:layout_marginRight="1.5dip"
            android:visibility="invisible" />
        <View
            android:id="@+id/v_dot5"
            android:layout_width="5dip"
            android:layout_height="5dip"
            android:background="@drawable/net13140809031
            android:layout_marginLeft="1.5dip"
            android:layout_marginRight="1.5dip"
            android:visibility="invisible" />
```

```
                <View
                    android:id="@+id/v_dot6"
                    android:layout_width="5dip"
                    android:layout_height="5dip"
                    android:background="@drawable/net131408090311
                    android:layout_marginLeft="1.5dip"
                    android:layout_marginRight="1.5dip"
                    android:visibility="invisible" />
                <View
                    android:id="@+id/v_dot7"
                    android:layout_width="5dip"
                    android:layout_height="5dip"
                    android:background="@drawable/net131408090311
                    android:layout_marginLeft="1.5dip"
                    android:layout_marginRight="1.5dip"
                    android:visibility="invisible" />
                <View
                    android:id="@+id/v_dot8"
                    android:layout_width="5dip"
                    android:layout_height="5dip"
                    android:background="@drawable/net131408090311
                    android:layout_marginLeft="1.5dip"
                    android:layout_marginRight="1.5dip"
                    android:visibility="invisible" />
            </LinearLayout>
        </FrameLayout>

    </LinearLayout>
```

3.关键代码

> 定时切换用到一个类：*ScheduledExecutorService*
>
> 作用是定时执行任务，我们这里要做的定时任务是，*5*秒执行一次图片切换

```
    private void startAd() {
            scheduledExecutorService = Executors.newSingleThread
```

```
            // 当Activity显示出来后，每两秒切换一次图片显示
            scheduledExecutorService.scheduleAtFixedRate(new Scr
                    TimeUnit.SECONDS);
        }
```

> 定义一个线程,滑动时通知*handle*响应,同时更改圆点索引号

```
    private class ScrollTask implements Runnable {

            @Override
            public void run() {
                synchronized (adViewPager) {
                    currentItem = (currentItem + 1) % imageViews
                    handler.obtainMessage().sendToTarget();
                }
            }
        }
```

> 通过*handle*来通知*ViewPager*进行视图切换

```
    private Handler handler = new Handler() {
            public void handleMessage(android.os.Message msg) {
                adViewPager.setCurrentItem(currentItem);
            };
        };
```

> 异步线程加载网络图片，具体代码我封装在 `Net1314080903118MyRequest.java` 中，详情可查看最后的链接。

4.具体代码如下：

```
    public class BannerTestActivity extends AppCompatActivity {

        private Context mContext;
```

```
        private ViewPager adViewPager;
        private List<ImageView> imageViews;// 滑动的图片集合
        private List<View> dots; // 图片标题正文的那些点
        private List<View> dotList;

        private int currentItem = 0; // 当前图片的索引号
        // 定义的五个指示点
        private View dot0,dot1,dot2,dot3,dot4,dot5,dot6,dot7,dot

        // 定时任务
        private ScheduledExecutorService scheduledExecutorServic

        // 轮播banner的数据
        private List<String> mDataList = null;

        private Handler handler = new Handler() {
            public void handleMessage(android.os.Message msg) {
                adViewPager.setCurrentItem(currentItem);
            };
        };

        @Override
        protected void onCreate(@Nullable Bundle savedInstanceSt
            super.onCreate(savedInstanceState);
            setContentView(R.layout.net1314080903118);
            setTitle("Net1314080903118");
            mContext = BannerTestActivity.this;
            initDatas();
            initView();
            startAd();
        }

        /**
         * 添加九张图片的数据，用作模拟数据
         */
        private void initDatas() {
            mDataList = new ArrayList<>();
            mDataList.add("http://image.wufazhuce.com/Fi7nIAI3UF
```

```
            mDataList.add("http://image.wufazhuce.com/Fn8JdRZ1g9
            mDataList.add("http://image.wufazhuce.com/FjJPOSVbyp
            mDataList.add("http://image.wufazhuce.com/FiyEgeiSFJ
            mDataList.add("http://image.wufazhuce.com/Fg-EwcAuXv
            mDataList.add("http://image.wufazhuce.com/Fi7nIAI3UF
            mDataList.add("http://image.wufazhuce.com/Fn8JdRZ1g9
            mDataList.add("http://image.wufazhuce.com/FjJPOSVbyp
            mDataList.add("http://image.wufazhuce.com/FiyEgeiSFJ
            mDataList.add("http://image.wufazhuce.com/Fg-EwcAuXv
            mDataList.add("http://image.wufazhuce.com/Fi7nIAI3UF
        }


        private void initView() {
            imageViews = new ArrayList<ImageView>();
            // 点
            dots = new ArrayList<View>();
            dotList = new ArrayList<View>();
            dot0 = findViewById(R.id.v_dot0);
            dot1 = findViewById(R.id.v_dot1);
            dot2 = findViewById(R.id.v_dot2);
            dot3 = findViewById(R.id.v_dot3);
            dot4 = findViewById(R.id.v_dot4);
            dot5 = findViewById(R.id.v_dot5);
            dot6 = findViewById(R.id.v_dot6);
            dot7 = findViewById(R.id.v_dot7);
            dot8 = findViewById(R.id.v_dot8);
            dots.add(dot0);
            dots.add(dot1);
            dots.add(dot2);
            dots.add(dot3);
            dots.add(dot4);
            dots.add(dot5);
            dots.add(dot6);
            dots.add(dot7);
            dots.add(dot8);

            adViewPager = (ViewPager) findViewById(R.id.vp);
```

```
        loadViewPager();
    }

    private void addDynamicView() {
        // 动态添加图片和下面指示的圆点
        // 初始化图片资源
        for (int i = 0; i < mDataList.size(); i++) {
            ImageView imageView = new ImageView(mContext);
            new Net1314080903118MyRequest(mContext).getImage
            imageView.setScaleType(ImageView.ScaleType.CENTE
            imageViews.add(imageView);
            dots.get(i).setVisibility(View.VISIBLE);
            dotList.add(dots.get(i));
        }

    }


    /**
     * 定时任务，5s更换一次
     */
    private void startAd() {
        scheduledExecutorService = Executors.newSingleThread
        // 当Activity显示出来后，每两秒切换一次图片显示
        scheduledExecutorService.scheduleAtFixedRate(new Scr
                TimeUnit.SECONDS);
    }


    /**
     * 自定义线程，滑动时通知handle响应
     */
    private class ScrollTask implements Runnable {

        @Override
        public void run() {
            synchronized (adViewPager) {
```

```
                currentItem = (currentItem + 1) % imageViews
                handler.obtainMessage().sendToTarget();
            }
        }
    }

    @Override
    public void onStop() {
        super.onStop();
        // 当Activity不可见的时候停止切换
        scheduledExecutorService.shutdown();
    }

    /**
     * 重写ViewPager，改变圆点是否选中状态
     * 在这里我们不需要重写ViewPager的滑动动作
     */
    private class MyPageChangeListener implements ViewPager.

        private int oldPosition = 0;

        @Override
        public void onPageScrollStateChanged(int arg0) {

        }

        @Override
        public void onPageScrolled(int arg0, float arg1, int

        }

        @Override
        public void onPageSelected(int position) {
            currentItem = position;
            dots.get(oldPosition).setBackgroundResource(R.dr
            dots.get(position).setBackgroundResource(R.drawa
            oldPosition = position;
        }
```

```
        }

        private void loadViewPager() {
            addDynamicView();
            // 设置填充ViewPager页面的适配器
            adViewPager.setAdapter(new Net1314080903118MyAdapter
            // 设置一个监听器，当ViewPager中的页面改变时调用
            adViewPager.setOnPageChangeListener(new MyPageChange
        }

        /**
         * 自定义ViewPager的适配器，具体点击图片的跳转逻辑在instantiate
         */
        public class Net1314080903118MyAdapter extends PagerAdap

            private List<String> DataList;
            private List<ImageView> imageList;
            private Context mContext;

            public Net1314080903118MyAdapter(Context context,Lis
                this.DataList = adList;
                this.imageList = imageViews;
                this.mContext = context;
            }

            @Override
            public int getCount() {
                return DataList.size();
            }

            @Override
            public Object instantiateItem(ViewGroup container, i
                ImageView iv = imageList.get(position);
                ((ViewPager) container).addView(iv);
                // 在这个方法里面设置图片的点击事件
                iv.setOnClickListener(new View.OnClickListener()

                    @Override
```

```
                public void onClick(View v) {
                    // 处理跳转逻辑
                }
            });
            return iv;
        }

        @Override
        public void destroyItem(View arg0, int arg1, Object 
            ((ViewPager) arg0).removeView((View) arg2);
        }

        @Override
        public boolean isViewFromObject(View arg0, Object ar
            return arg0 == arg1;
        }

        @Override
        public void restoreState(Parcelable arg0, ClassLoade

        }

        @Override
        public Parcelable saveState() {
            return null;
        }

        @Override
        public void startUpdate(View arg0) {

        }

        @Override
        public void finishUpdate(View arg0) {

        }

    }
```

```
    }
```

> 备注：完整代码请访问以下链接

https://github.com/hzuapps/android-labs/issues/139

实验5：Android文件存储

实验5: Android文件存储

首选项

文件

实验6：Android数据库编程

实验6: Android数据库编程

实验7：Android网络编程

实验7: Android网络编程

7.1 知识点

1.

2.

7.2 实例步骤

1. 根据位置信息从中国天气网获取天气信息

简要说明.....

详细步骤.....

2. Socket编程

简要说明

这个例子只用了TCP协议下的Socket编程，因此这里只讨论TCP的情况。

在java中，服务器端socket、bind、listen等操作被封装在ServerSocket类库中，客户端socket、connect等操作被封装在Socket类库中。如需了解这些操作细节，可学习C语言下的socket编程。

详细步骤

服务器端步骤

1). servs = new ServerSocket(port);　　//创建套接字，port 为指定的端口号

2). socket = servs.accept();　　//等待连接，主线程会阻塞在这里

3). 为每个连接创建线程来服务；

4). 线程中进行读写socket【通信】。

客户端步骤

1). socket = Socket(“ip_str”, port);　　//创建套接字，ip_str 为服务器端IP，port为端口号，和服务器端指定的要一样

2). 读写socket。　　//1).中已经执行连接服务器的操作，所以接下来可以直接读写。

注意:

1). 客户端连接服务器的操作要放在新线程里面。

2). 需要添加权限:

```
1.     <uses-permission android:name="android.permission.INTERNET"/>
```

3). 真机调试的时候，真机和主机要在同一个局域网。

核心代码

服务器端:（所有源码）

```
1.  public class Myserver {
2.      //定义保存所有的Socket的ArrayList
3.      public static ArrayList<Socket> socketList = new
    ArrayList<Socket>();
4.
5.      public static void main(String[] args)
6.          throws IOException
7.      {
8.          ServerSocket ss = new ServerSocket(9402);               //
    端口号为9402
9.          while (true)
10.         {
11.             Socket s = ss.accept();
12.             socketList.add(s);                  //把新连接加入
    ArrayList中。
13.             //每连接一个客户端就开一个线程为之服务
14.             new Thread(new ServerThread(s)).start();
15.         }
16.     }
17. }
```

客户端代码：(所有源码)

```
1.      public void run()
2.      {
3.          try {
4.              s = new Socket("192.168.240.22", 9402);           //IP
    是服务器IP,  端口号和服务器一致
5.              br = new BufferedReader((new
    InputStreamReader(s.getInputStream())));
6.              os = s.getOutputStream();
7.
8.              //do something
9.
10.         }
11.         catch (...)
12.         {
```

```
            //...
        }
    }
```

另一个例子：https://github.com/hzuapps/android-labs/tree/master/app/src/main/java/edu/hzuapps/androidworks/homeworks/net1314080903204/tcp_tester

3. 从aqicn.org获取PM2.5信息@ZhengQZ123

简要说明……

利用aqicn.org提供的接口，显示pm2.5的数值

详细步骤……

1. 获取网络权限

```
<uses-permission android:name="android.permission.INTERNET"/>
```

2.读取http://aqicn.org/publishingdata/json提供的json数据

```
                BufferedReader reader = new BufferedReader(new InputStreamReader(new URL("http://aqicn.org/publishingdata/json").openStream(), "utf-8"));
                String line=null;
                StringBuffer content =new StringBuffer();
                while((line=reader.readLine())!=null)
                {
                   content.append(line);
                }
```

3.解析获取到的json数据

```
                try {
```

```
                    JSONArray jsonarr=new JSONArray(s);
                    JSONObject
firstJO=jsonarr.getJSONObject(0);
                    JSONArray
pollutants=firstJO.getJSONArray("pollutants");
                    JSONObject
firstPollutants=pollutants.getJSONObject(0);
                    System.out.println("cityName="
firstJO.getString("cityName") ",local="
firstJO.getString("localName"));
                    String cityName=
firstJO.getString("cityName");
                    String localName=
firstJO.getString("localName");
                    Double
pollutant=firstPollutants.getDouble("value");
                    String a=cityName+"
"+localName+":"+pollutant;

firstJO.getString("localName"),firstPollutants.getDouble("value"
                    tvPmData.setText(a);
                } catch (JSONException e) {
                    e.printStackTrace();
                }
```

4.将获取到的数据按照需要显示出来 tvPmData.setText(a)

activity代码: (https://github.com/ZhengQZ123/android-labs/blob/master/app/src/main/java/edu/hzuapps/androidworks/homeworks/net1314080903247/Net1314080903247Activity.java)

AndroidManifest代码: (https://github.com/ZhengQZ123/android-labs/blob/master/app/src/main/java/edu/hzuapps/andro

idworks/homeworks/net1314080903247/AndroidManifest.xml)

activity xml代码:
(https://github.com/ZhengQZ123/android-labs/blob/master/app/src/main/res/layout/activity_net1314080903247.xml)

实验8：Android设备编程

实验8：Android设备编程

8.1 知识点

1.

添加权限

2.

设置布局

3

获取GPS信息

8.2 实例讲解

1. 获取设备当前位置信息

简要说明

获取手机的GPS信息，显示获取信息中的经纬度信息。当位置变化时，会重新获取手机的PGS信息，显示最新获取到的经纬度信息。

详细步骤

1.1 修改AndroidManifest.xml，添加权限

```
1.    <uses-permission Android:name="android.permission.INTERNET"/>
2.     <uses-permission
    android:name="android.permission.ACCESS_FINE_LOCATION"/>
3.     <uses-permission
    android:name="android.permission.ACCESS_COARSE_LOCATION" />
```

1.2 判断GPS是否正常启动

```
1.  if(!lm.isProviderEnabled(LocationManager.GPS_PROVIDER)){
2.              Toast.makeText(this, "请开启GPS导航...",
    Toast.LENGTH_SHORT).show();
```

```
            //返回开启GPS导航设置界面
            Intent intent = new
Intent(Settings.ACTION_LOCATION_SOURCE_SETTINGS);
            startActivityForResult(intent,0);
            return;
        }
```

3设置查询条件

```
        String bestProvider = lm.getBestProvider(getCriteria(),
true);
        //获取位置信息
        //如果不设置查询要求，getLastKnownLocation方法传人的参数为
LocationManager.GPS_PROVIDER
        Location location= lm.getLastKnownLocation(bestProvider);
        updateView(location);
        //监听状态
        lm.addGpsStatusListener(listener);
        //绑定监听，有4个参数
        //参数1，设备：有GPS_PROVIDER和NETWORK_PROVIDER两种
        //参数2，位置信息更新周期，单位毫秒
        //参数3，位置变化最小距离：当位置距离变化超过此值时，将更新位置信息
        //参数4，监听
        //备注：参数2和3，如果参数3不为0，则以参数3为准；参数3为0，则通过时间
来定时更新；两者为0，则随时刷新

        // 1秒更新一次，或最小位移变化超过1米更新一次；
        //更新位置
        lm.requestLocationUpdates(LocationManager.GPS_PROVIDER,
5000, 10, locationListener);
    }
```

1.4 位置监听

```
1.  private LocationListener locationListener=new LocationListener() {
2.
3.               public void onLocationChanged(Location location) {
4.              updateView(location);
5.
6.                  ContentValues cv = new ContentValues();
7.
    cv.put("longitude",String.valueOf(location.getLongitude()));
8.                  cv.put("latitude",
    String.valueOf(location.getLatitude()));
9.                  dbWrite.insert("whereYou", null, cv);
10.                 refresh();//刷新数据库
```

1.5 GPS开启/关闭时触发

```
1.   public void onProviderEnabled(String provider) {
2.              Location location=lm.getLastKnownLocation(provider);
3.              updateView(location);
4.          }
5.
6.    public void onProviderDisabled(String provider) {
7.              updateView(null);
8.          }
```

1.6 状态监听

```
1.  GpsStatus.Listener listener = new GpsStatus.Listener() {
2.          public void onGpsStatusChanged(int event) {
3.              switch (event) {
4.                  //第一次定位
5.                  case GpsStatus.GPS_EVENT_FIRST_FIX:
6.                      Log.i(TAG, "第一次定位");
```

```
                    break;
                //卫星状态改变
                case GpsStatus.GPS_EVENT_SATELLITE_STATUS:
                    Log.i(TAG, "卫星状态改变");
                    //获取当前状态
                    GpsStatus gpsStatus=lm.getGpsStatus(null);
                    //获取卫星颗数的默认最大值
                    int maxSatellites =
gpsStatus.getMaxSatellites();
                    //创建一个迭代器保存所有卫星
                    Iterator<GpsSatellite> iters =
gpsStatus.getSatellites().iterator();
                    int count = 0;
                    while (iters.hasNext() && count <=
maxSatellites) {
                        GpsSatellite s = iters.next();
                        count++;
                    }
                    System.out.println("搜索到："+count+"颗卫星");
                    break;
                //定位启动
                case GpsStatus.GPS_EVENT_STARTED:
                    Log.i(TAG, "定位启动");
                    break;
                //定位结束
                case GpsStatus.GPS_EVENT_STOPPED:
                    Log.i(TAG, "定位结束");
                    break;
            }
        };
    };
```

1.7 更新要显示的文本信息

```
 private void updateView(Location location){
        if(location!=null){
```

```
            editText.setText("位置信息\n\n经度：");
editText.append(String.valueOf(location.getLongitude()));
            editText.append("\n纬度：");
editText.append(String.valueOf(location.getLatitude()));
            editText.append("\n海拔：");
editText.append(String.valueOf(location.getAltitude()));



            editText.append("\n时间：");
            editText.append(str);
        }else{
            //清空EditText对象
            editText.getEditableText().clear();
        }
    }
```

2. 移动定位模式选择

1. 定位精度选择

```
 selectMode = (RadioGroup)findViewById(R.id.selectMode);
//单选按钮组  选择定位精度
  selectMode.setOnCheckedChangeListener(new
OnCheckedChangeListener() {   //监听单选按钮

            @Override
            public void onCheckedChanged(RadioGroup group, int
checkedId) {
                // TODO Auto-generated method stub
                String ModeInformation = null;
                switch (checkedId) {
                    case R.id.radio_hight:
                        tempMode = LocationMode.Hight_Accuracy;
```

```
                        ModeInformation =
getString(R.string.hight_accuracy_desc);
                        //高精度定位模式下，会同时使用GPS、Wifi和基站定
位，返回的是当前条件下精度最好的定位结果
                        break;
                    case R.id.radio_low:
                        tempMode = LocationMode.Battery_Saving;
                        ModeInformation =
getString(R.string.saving_battery_desc);
                        //低功耗定位模式下，仅使用网络定位即Wifi和基站定
位，返回的是当前条件下精度最好的网络定位结果
                        break;
                    case R.id.radio_device:
                        tempMode = LocationMode.Device_Sensors;
                        ModeInformation =
getString(R.string.device_sensor_desc);
                        //仅用设备定位模式下，只使用用户的GPS进行定位。这种
模式下，由于GPS芯片锁定需要时间，首次定位速度会需要一定的时间
                        break;
                    default:
                        break;
                }
                ModeInfor.setText(ModeInformation);
            }
        });
```

2. 定位类型选择

```
 selectCoordinates=
(RadioGroup)findViewById(R.id.selectCoordinates);     //单选按钮组
选择定位类型
  selectCoordinates.setOnCheckedChangeListener(new
OnCheckedChangeListener() {//监听单选按钮

            @Override
            public void onCheckedChanged(RadioGroup group, int
checkedId) {
                // TODO Auto-generated method stub
```

```
                switch (checkedId) {
                    case R.id.radio_gcj02:
                        tempcoor="gcj02";//国家测绘局标准
                        break;
                    case R.id.radio_bd09ll:
                        tempcoor="bd09ll";//百度经纬度标准
                        break;
                    case R.id.radio_bd09:
                        tempcoor="bd09";//百度墨卡托标准
                        break;
                    default:
                        break;
                }
            }
        });
```

3. 设置查询条件

```
private void initLocation(){
        LocationClientOption option = new LocationClientOption();
        option.setLocationMode(tempMode);//可选，默认高精度，设置定位模式，高精度，低功耗，仅设备
        option.setCoorType(tempcoor);//可选，默认gcj02，设置返回的定位结果坐标系，
        int span=1000;
        try {
            span = Integer.valueOf(frequence.getText().toString());
        } catch (Exception e) {
            // TODO: handle exception
        }
        option.setScanSpan(span);//可选，默认0，即仅定位一次，设置发起定位请求的间隔需要大于等于1000ms才是有效的
        option.setIsNeedAddress(checkGeoLocation.isChecked());//可选，设置是否需要地址信息，默认不需要
        option.setOpenGps(false);//可选，默认false,设置是否使用gps
        option.setLocationNotify(true);//可选，默认false，设置是否当gps有效时按照1S1次频率输出GPS结果
        option.setIgnoreKillProcess(true);//可选，默认true，定位SDK内
```

```
    部是一个SERVICE，并放到了独立进程，设置是否在stop的时候杀死这个进程，默认不杀死
16.         option.setEnableSimulateGps(false);//可选，默认false，设置是否需要过滤gps仿真结果，默认需要
17.         option.setIsNeedLocationDescribe(true);//可选，默认false，设置是否需要位置语义化结果，可以在BDLocation.getLocationDescribe里得到，结果类似于“在北京天安门附近”
18.         option.setIsNeedLocationPoiList(true);//可选，默认false，设置是否需要POI结果，可以在BDLocation.getPoiList里得到
19.         mLocationClient.setLocOption(option);
20.     }
```

2. 播放MP3音乐

1.简要说明

1.实现方式

```
1.  使用Android系统本身设备
2.  播放本地SD卡内的MP3音乐
```

2.实现前提

```
1.  Android系统是一个基于Linux平台的开源移动操作系统，换句话说，Android是一种类Linux系统。Linux系统核心部分的权限，自然也被Android系统所继承。简单而言，对于一个普通文件权限属性分为读、写、执行，Android系统中不同开发人员创建的不同文件，文件所有权都被收归系统，而系统的授权分配归类等待授权请求。同时Android系统也发展出了一些与Linux系统不相同的权限，因为该系统的每一个APP理论上都是独立隔离的，APP之间的访问是有限制的（权限）。系统提供的资源都是采取“不请求不启动”的原则，保证体积较小的手机系统运行的速度足够。
```

1.具有对SD卡内容的读写权限

```
1.  在开发过程中，开发人员常常会使用自身项目包内的文件，由于这些文件都是开发人员自行导入生成，所有这些文件的所有者默认便是开发者，开发者对所有的项目包具有完全读写权限，调用修改乃至删除统统符合所有者权限。而当开发人员需要使用到系统其它不属于自己的资源时，则必须申请权限。
```

2.service声明权限

```
1.  service的创建应该是具有限制的，保证不会出现不需要这项service而它偏偏被系统分配空间且长时间占用，系统却认为这种占用是合法的，不予以处理的矛盾情况
```

3.系统API

```
1.  Android系统默认提供了媒体播放的MediaPlayer类，其中有以下API可以使用：
```

```
1.  setDataSource (String path) ;//设置播放文件的地址
2.  setOnSeekCompleteListener(OnSeekCompleteListener);//设置读取完整后的监听器
3.   start();//开始播放
4.   stop();//停止播放
```

4.注意事项

```
1.  在Android 程序开发中，activity里面都不能具有耗时太长的执行语句，因为这会造成图形界面的长时间未响应，让使用者觉得该程序卡死了，于是在Android系统本身便有限制，一旦某个Activity 耗时太长，该程序将会被强制停止
2.  在AndroidManifest.xml的不同地方声明不同部分的权限。
```

2.详细步骤

1.请求SD卡读写权限

```
1.   <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
2.    <uses-permission android:name="android.permission.MOUNT_UNMOUNT_FILESYSTEMS" />
```

2.service声明权限

```
1.  <service android:name=".Net1314080903219MediaPlayerService" android:enabled="true" android:exported="true" />
```

3.播放音频视频的权限

```
<uses-permission android:name="android.permission.MEDIA_CONTENT_CONTROL" />
  <uses-permission android:name="android.permission.BIND_VOICE_INTERACTION" />
  <uses-permission android:name="android.permission.CAPTURE_SECURE_VIDEO_OUTPUT" />
  <uses-permission android:name="android.permission.CAPTURE_VIDEO_OUTPUT" />
```

4.使用MediaPlayer完成设计

```

    MediaPlayer player = new MediaPlayer();
    public Net1314080903219MediaPlayerService() {

      /*  try {
            player.reset();
            player.setDataSource(path);
            player.prepare();
        } catch (IOException e) {
            e.printStackTrace();
        }*/
    }


    @Override
    public IBinder onBind(Intent arg0) {
        // TODO Auto-generated method stub
        return null;
    }

    //在这里我们需要实例化MediaPlayer对象
    public void onCreate(){

        super.onCreate();
        //我们从raw文件夹中获取一个应用自带的mp3文件
```

```
26.
27.         System.out.println("sfsfsfsf  dsf fdfd fsdf sf ffsfs");
28.
29.
30.     }
31.
32.     /**
33.      * 该方法在SDK2.0才开始有的，替代原来的onStart方法
34.      */
35.     public int onStartCommand(Intent intent, int flags, int
    startId){
36.         if(!player.isPlaying()){
37.         //
    System.out.println(String.valueOf(intent.getCharSequenceArrayExtra("s
38.             try {
39.                 player.reset();
40.
    player.setDataSource(intent.getStringExtra("song"));
41.                 player.prepare();
42.                 player.start();
43.             } catch (IOException e) {
44.                 e.printStackTrace();
45.             }
46.         }
47.         return START_STICKY;
48.     }
49.
50.     public void onDestroy(){
51.         //super.onDestroy();
52.         if(player.isPlaying()){
53.             player.stop();
54.         }
55.         player.release();
56.     }
57.
58.
59.
60.
```

```
61.     //后退播放进度
62.     public void haveFun(){
63.         if(player.isPlaying() && player.getCurrentPosition()>2500){
64.             player.seekTo(player.getCurrentPosition()-2500);
65.         }
66.     }
```

5.完全源代码路径

[github]
(https://github.com/helloSingleDog/android-labs/tree/master/app/src/main/java/edu/hzuapps/androidworks/homeworks/net1314080903219)

3. 相机 @WL101ZYF

简要说明

利用此程序，可以调用手机内原有的照相机或者有摄像功能的程序，并以此实现照相功能，最后保存到SD卡或其他存储相片的位置中。

详细步骤

1、建立一个项目。

2、在配置文件中写入下面三条语句:

```
1.  <!--android中使用摄像机的权限  -->
2.  <uses-permission android:name="android.permission.CAMERA" />
3.  <!--android中创建于删除文件的权限  -->
4.  <uses-permission
    android:name="android.permission.MOUNT_UNMOUNT_FILESYSTEMS" />
5.  <!--android中写入SDCARD的权限  -->
6.  <uses-permission
    android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
```

3、在XML文件正确合适的位置里加入以下代码:

```
<SurfaceView
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:id="@+id/surfaceview"/>

    <RelativeLayout
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:visibility="gone"
        android:id="@+id/buttonlayout">

        <Button
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_alignParentRight="true"
            android:layout_alignParentBottom="true"
            android:layout_marginRight="5dp"
            android:text="@string/takepicture"
            android:onClick="takepicture"
            android:id="@+id/takepicture" />



        <Button
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_toLeftOf="@id/takepicture"
            android:layout_alignTop="@id/takepicture"
            android:layout_marginRight="20dp"
            android:text="@string/autofocus"
            android:onClick="takepicture"
            android:id="@+id/autofocus" />
```

4、打开java文件，写入如下代码

```
    private View layout;
    private Camera camera;
```

```
3.
4.
5.      @Override
6.      protected void onCreate(Bundle savedInstanceState) {
7.
8.          super.onCreate(savedInstanceState);
9.          //设置窗口没有标题
10.         requestWindowFeature(Window.FEATURE_NO_TITLE);
11.         //设置窗口全屏
12.
    getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
13.                 WindowManager.LayoutParams.FLAG_FULLSCREEN);
14.         setContentView(R.layout.activity_net1314080903102);
15.
16.
17.         //利用layout方法，找到两个按钮控件
18.         layout = this.findViewById(R.id.buttonlayout);
19.         //获取摄像头窗口
20.         SurfaceView surfaceView = (SurfaceView)
    this.findViewById(R.id.surfaceview);
21.         //将获取的摄像头填满整个窗口
22.
    surfaceView.getHolder().setType(SurfaceHolder.SURFACE_TYPE_PUSH_BUFFE
23.         //设置窗口分辨率
24.         surfaceView.getHolder().setFixedSize(176, 144);
25.         //保持屏幕高亮，不要锁机
26.         surfaceView.getHolder().setKeepScreenOn(true);
27.         //设置摄像头被调用监听事件
28.         surfaceView.getHolder().addCallback(new SurfaceCallback());
29.
30.
31.     }
32.
33.     /*
34.      * 通过switch (v.getId()) 选择拍照事件和对焦事件
35.      */
36.     public void takepicture(View v) {
37.         if (camera != null) {
```

```
38.             switch (v.getId()) {
39.                 case R.id.takepicture:
40.                     //拍照片经过压缩处理后的图片调用MyPictureCallback方法
41.                     camera.takePicture(null, null, new
    MyPictureCallback());
42.                     break;
43.                 case R.id.autofocus:
44.                     //如果不想得到对焦事件，传送NULL事件进去
45.                     camera.autoFocus(null);
46.
47.                 default:
48.                     break;
49.             }
50.
51.
52.         }
53.
54.     }
55.       /*
56.      * 获取图片对象
57.      * */
58.
59.     private final class MyPictureCallback implements
    PictureCallback {
60.         public void onPictureTaken(byte[] data, Camera camera) {
61.
62.             try {
63.                 //将文件存储在SD卡的dcim目录，并以系统时间将文件命名
64.                 File jpgFile = new File("/sdcard/dcim/",
65.                         java.lang.System.currentTimeMillis() +
    ".jpg");
66.                 //文件输出流对象
67.                 FileOutputStream outStream = new
    FileOutputStream(jpgFile);
68.                 //将文件数据存储到文件中
69.                 outStream.write(data);
70.                 //关闭输出流
71.                 outStream.close();
```

```
72.                 //开始预览照片NN
73.                 camera.startPreview();
74.             } catch (IOException e) {
75.                 e.printStackTrace();
76.             }
77.         }
78.
79.     }
80.
81.     /*
82.      * 设置摄像头参数
83.      */
84.     private final class SurfaceCallback implements
    android.view.SurfaceHolder.Callback {
85.
86.         public void surfaceCreated(SurfaceHolder holder) {
87.             try {
88.                 //打开摄像头
89.                 camera = Camera.open();
90.                 //获取摄像头参数对象
91.                 Camera.Parameters parameters =
    camera.getParameters();
92.                 //设置摄像头分辨率
93.                 parameters.setPreviewSize(800, 480);
94.                 //设置摄像头捕获画面的频率为每秒5个画面
95.                 parameters.setPreviewFrameRate(5);
96.                 //设置拍摄照片的大小
97.                 parameters.setPictureSize(1024, 768);
98.                 //设置捕捉图像的JPEG画质
99.                 parameters.setJpegQuality(80);
100.                //把参数返回给摄像头
101.                camera.setParameters(parameters);
102.                //显示摄像头捕获画面
103.                camera.setPreviewDisplay(holder);
104.                //开始预览摄像头
105.                camera.startPreview();
106.                //获取摄像头详细参数，并且打印出来
107.                //Log.i("MainActivity", parameters.flatten());
```

```
108.
109.                 } catch (Exception e) {
110.                     e.printStackTrace();
111.                 }
112.
113.             }
114.
115.             public void surfaceChanged(SurfaceHolder holder, int
     format, int width, int heigh) {
116.
117.             }
118.
119.             public void surfaceDestroyed(SurfaceHolder holder) {
120.                 //如果摄像头不使用时，关闭摄像头
121.                 if (camera != null) {
122.                     camera.release();
123.                     camera = null;
124.                 }
125.
126.
127.             }
128.
129.         }
130.
131.
132.         /*
133.          *屏幕被触摸事件
134.          *屏幕被按下后，显示相对布局里面的两个按钮
135.          */
136.         @Override
137.         public boolean onTouchEvent(MotionEvent event) {
138.             if (event.getAction() == MotionEvent.ACTION_DOWN) {
139.                 layout.setVisibility(ViewGroup.VISIBLE);
140.
141.             }
142.
143.             return super.onTouchEvent(event);
144.
```

```
145.     }
146. }
```

4. 执行操作系统命令安装Apk

项目做了一个秒装软件的功能，其实就是静默安装啦。所谓的静默安装，就是不用弹出系统的安装界面，在不影响用户任何操作的情况下不知不觉地将程序装好。

秒装其实是需要ROOT权限的静默安装。静默安装的原理很简单，就是调用Android系统的pm install命令就可以了，但是pm命令系统是不授予我们权限调用的，因此只能在拥有ROOT权限的手机上去申请权限才行。

首先新建一个项目，然后创建一个Net13140809031l2SilentInstall类作为静默安装功能的实现类，代码如下所示：

```
1.  package edu.hzuapps.androidworks.homeworks.net1314080903112;
2.
3.  import android.util.Log;
4.
5.  import java.io.BufferedReader;
6.  import java.io.DataOutputStream;
7.  import java.io.IOException;
8.  import java.io.InputStreamReader;
9.  import java.nio.charset.Charset;
10.
11.
12.
13. public class Net1314080903112SilentInstall {
14.
15.     /**
16.      * 执行具体的静默安装逻辑，需要手机ROOT。
17.      * @param apkPath
```

```
18.      *          要安装的apk文件的路径
19.      * @return 安装成功返回true，安装失败返回false。
20.      */
21.     public boolean install(String apkPath) {
22.         boolean result = false;
23.         DataOutputStream dataOutputStream = null;
24.         BufferedReader errorStream = null;
25.         try {
26.             // 申请su权限
27.             Process process = Runtime.getRuntime().exec("su");
28.             dataOutputStream = new
    DataOutputStream(process.getOutputStream());
29.             // 执行pm install命令
30.             String command = "pm install -r " + apkPath + "\n";
31.
    dataOutputStream.write(command.getBytes(Charset.forName("utf-8")));
32.             dataOutputStream.flush();
33.             dataOutputStream.writeBytes("exit\n");
34.             dataOutputStream.flush();
35.             process.waitFor();
36.             errorStream = new BufferedReader(new
    InputStreamReader(process.getErrorStream()));
37.             String msg = "";
38.             String line;
39.             // 读取命令的执行结果
40.             while ((line = errorStream.readLine()) != null) {
41.                 msg += line;
42.             }
43.             Log.d("TAG", "install msg is " + msg);
44.             // 如果执行结果中包含Failure字样就认为是安装失败，否则就认为安装
    成功
45.             if (!msg.contains("Failure")) {
46.                 result = true;
47.             }
48.         } catch (Exception e) {
49.             Log.e("TAG", e.getMessage(), e);
50.         } finally {
51.             try {
```

```
52.                 if (dataOutputStream != null) {
53.                     dataOutputStream.close();
54.                 }
55.                 if (errorStream != null) {
56.                     errorStream.close();
57.                 }
58.             } catch (IOException e) {
59.                 Log.e("TAG", e.getMessage(), e);
60.             }
61.         }
62.         return result;
63.     }
64.
65. }
```

可以看到，Net13140809031l2SilentInstall类中只有一个install()方法，所有静默安装的逻辑都在这个方法中了，那么我们具体来看一下这个方法。首先调用了Runtime.getRuntime().exec(“su”)方法，在这里先申请ROOT权限，不然的话后面的操作都将失败。然后开始组装静默安装命令，命令的格式就是pm install -r ，-r参数表示如果要安装的apk已经存在了就覆盖安装的意思，apk路径是作为方法参数传入的。接下来的几行就是执行上述命令的过程，注意安装这个过程是同步的，因此我们在下面调用了process.waitFor()方法，即安装要多久，我们就要在这里等多久。等待结束之后说明安装过程结束了，接下来我们要去读取安装的结果并进行解析，解析的逻辑也很简单，如果安装结果中包含Failure字样就说明安装失败，反之则说明安装成功。

接下来搭建调用这个方法的环境，修改net13140809031l2activity_main.xml中的代码，以及新建net13140809031l2activity_file_explorer.xml和net13140809031l2list_item.xml作为文件选择器的布局文件。然后新建Net13140809031l2FileExplorerActivity作为文件选

择器的Activity，接着修改Net1314080903112MainActivity中的代码，如下所示：

```
package edu.hzuapps.androidworks.homeworks.net1314080903112;

import android.content.Intent;
import android.net.Uri;
import android.provider.Settings;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.text.TextUtils;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;
import android.widget.Toast;

import java.io.File;


public class Net1314080903112MainActivity extends AppCompatActiv

    TextView apkPathText;

    String apkPath;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.net1314080903112activity_main);
        apkPathText = (TextView) findViewById(R.id.apkPathText);
    }

    @Override
    protected void onActivityResult(int requestCode, int resultC
        if (requestCode == 0 && resultCode == RESULT_OK) {
            apkPath = data.getStringExtra("apk_path");
            apkPathText.setText(apkPath);
        }
```

```
    }

    public void onChooseApkFile(View view) {
        Intent intent = new Intent(this, Net1314080903112FileExp
        startActivityForResult(intent, 0);
    }

    public void onSilentInstall(View view) {
        if (!isRoot()) {
            Toast.makeText(this, "没有ROOT权限，不能使用秒装", Toast
            return;
        }
        if (TextUtils.isEmpty(apkPath)) {
            Toast.makeText(this, "请选择安装包！", Toast.LENGTH_SHO
            return;
        }
        final Button button = (Button) view;
        button.setText("安装中");
        new Thread(new Runnable() {
            @Override
            public void run() {
                Net1314080903112SilentInstall installHelper = new
                final boolean result = installHelper.install(apkP
                runOnUiThread(new Runnable() {
                    @Override
                    public void run() {
                        if (result) {
                            Toast.makeText(Net1314080903112MainA
                        } else {
                            Toast.makeText(Net1314080903112MainA
                        }
                        button.setText("秒装");
                    }
                });

            }
        }).start();
```

```
    }

    public void onForwardToAccessibility(View view) {
        Intent intent = new Intent(Settings.ACTION_ACCESSIBILITY
        startActivity(intent);
    }

    public void onSmartInstall(View view) {
        if (TextUtils.isEmpty(apkPath)) {
            Toast.makeText(this, "请选择安装包！", Toast.LENGTH_SHO
            return;
        }
        Uri uri = Uri.fromFile(new File(apkPath));
        Intent localIntent = new Intent(Intent.ACTION_VIEW);
        localIntent.setDataAndType(uri, "application/vnd.android
        startActivity(localIntent);
    }

    /**
     * 判断手机是否拥有Root权限。
     * @return 有root权限返回true，否则返回false。
     */
    public boolean isRoot() {
        boolean bool = false;
        try {
            if (Runtime.getRuntime().exec("su").getOutputStream(
                return false;
            } else {
                return true;
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
        return bool;
    }

}
```

可以看到，在Net1314080903112MainActivity中，我们对四个按钮点击事件的回调方法都进行了定义，当点击“选择安装包”按钮时就会调用onChooseApkFile()方法，当点击“秒装”按钮时就会调用onSilentInstall()方法。在onChooseApkFile()方法方法中，我们通过Intent打开了Net1314080903112FileExplorerActivity，然后在onActivityResult()方法当中读取选择的apk文件路径。在onSilentInstall()方法当中，先判断设备是否ROOT，如果没有ROOT就直接return，然后判断安装包是否已选择，如果没有也直接return。接下来我们开启了一个线程来调用Net1314080903112SilentInstall.install()方法，因为安装过程会比较耗时，如果不开线程的话主线程就会被卡住，不管安装成功还是失败，最后都会使用Toast来进行提示。
最后在配置一下AndroidManifest.xml文件即可。

```
## 8.1 基于位置的服务简介
说到只有在移动设备上才能实现的技术， 很容易就让人联想到基于位置的服务 (Locat

## 8.2 实例讲解
### 8.2.1 获取设备当前位置信息
其实，归根结底，基于位置的服务所围绕的核心就是要确定出自己所在的位置，这在
Android 中并不困难， 主要借助 LocationManager 这个类就可以实现了。
下面我们首先学习一下 LocationManager 的基本用法，然后再通过一个例子来尝试
（PS：这里所写的代码建议你都在手机上运行，DDMS 虽然也提供了在模拟器中模拟地

#### 8.2.1.1 LocationManager 的基本用法
毫无疑问， 要想使用LocationManager就必须要先获取到它的实例， 我们可以调用C
LocationManager locationManager = (LocationManager)
getSystemService(Context.LOCATION_SERVICE);`
接着我们需要选择一个位置提供器来确定设备当前的位置。Android 中一般有三种位置
提供器可供选择，GPS_PROVIDER、NETWORK_PROVIDER 和 PASSIVE_PROVIDER。
```

（PS：需要注意的是，定位功能必须要由用户主动去启用才行，不然任何应用程序都无法

接着我们将选择好的位置提供器传入到 getLastKnownLocation()方法中， 就可以

```
String provider = LocationManager.NETWORK_PROVIDER;
Location location = locationManager.getLastKnownLocation(provide
```

这个 Location 对象中包含了经度、纬度、海拔等一系列的位置信息，然后从中取出我

如果有些时候你想让定位的精度尽量高一些，但又不确定 GPS 定位的功能是否已经启用，这个时候就可以先判断一下有哪些位置提供器可用，如下所示：`

```
List<String> providerList = locationManager.getProviders(true);
```

可以看到，getProviders()方法接收一个布尔型参数，传入 true就表示只有启用的

另外，调用 getLastKnownLocation()方法虽然可以获取到设备当前的位置信息，但不用担心，LocationManager 还提供了一个 requestLocationUpdates()方法，写法如下：`

```
locationManager.requestLocationUpdates(LocationManager.GPS_PROVI
Override
public void onStatusChanged(String provider, int status, Bundle
extras) {
}
Override
public void onProviderEnabled(String provider) {
}
Override
public void onProviderDisabled(String provider) {
}
@Override
public void onLocationChanged(Location location) {
}
});
```

这里 requestLocationUpdates()方法接收四个参数，第一个参数是位置提供器的类

好了，关于 LocationManager 的用法基本就是这么多，下面我们就通过一个例子来

8.2.1.2 确定自己位置的经纬度

通过上一小节的学习，你会发现 LocationManager 的用法并不复杂，那么本小节中

新建一个 LocationTest项目，修改 activity_main.xml 中的代码，如下所示：

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/
android:layout_width="match_parent"
```

```
android:layout_height="match_parent" >
<TextView
android:id="@+id/position_text_view"
android:layout_width="wrap_content"
android:layout_height="wrap_content" />
</LinearLayout>`
```

布局文件中的内容实在是太简单了，只有一个 TextView 控件，用于稍后显示设备位置的经纬度信息。

然后修改 MainActivity 中的代码，如下所示：`

```
public class MainActivity extends Activity {
private TextView positionTextView;
private LocationManager locationManager;
private String provider;
@Override
protected void onCreate(Bundle savedInstanceState) throws Securi
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_main);
positionTextView = (TextView) findViewById(R.id.position_text_vi
locationManager = (LocationManager) getSystemService(Context.
LOCATION_SERVICE);
// 获取所有可用的位置提供器
List<String> providerList = locationManager.getProviders(true);
if (providerList.contains(LocationManager.GPS_PROVIDER)) {
provider = LocationManager.GPS_PROVIDER;
} else if (providerList.contains(LocationManager.NETWORK_PROVIDE
provider = LocationManager.NETWORK_PROVIDER;
} else {
// 当没有可用的位置提供器时，弹出Toast提示用户
Toast.makeText(this, "No location provider to use",
Toast.LENGTH_SHORT).show();
return;
}
Location location = locationManager.getLastKnownLocation(provide
if (location != null) {
// 显示当前设备的位置信息
showLocation(location);
}
locationManager.requestLocationUpdates(provider, 5000, 1,
```

```
locationListener);
}
protected void onDestroy() {
super.onDestroy();
try {
 // 关闭程序时将监听器移除
 locationManager.removeUpdates(locationListener);
 } catch (SecurityException e) {
e.printStackTrace();
}
}
LocationListener locationListener = new LocationListener() {
@Override
public void onStatusChanged(String provider, int status, Bundle
extras) {
}
@Override
public void onProviderEnabled(String provider) {
}
@Override
public void onProviderDisabled(String provider) {
}
@Override
public void onLocationChanged(Location location) {
// 更新当前设备的位置信息
showLocation(location);
}
};
private void showLocation(Location location) {
String currentPosition = "latitude is " + location.getLatitude()
positionTextView.setText(currentPosition);
}
}`
```

这里并没有什么复杂的逻辑，基本全是我们在上一小节中学到的知识。

在 onCreate()方法中首先是获取到了 LocationManager 的实例，然后调用 getP

另外，获取设备当前的位置信息也是要声明权限的，因此还需要修改 AndroidManife

中的代码，如下所示：`

```
<manifest xmlns:android="http://schemas.android.com/apk/res/andr
package="com.example.locationtest"
android:versionCode="1"
android:versionName="1.0" >
……
<uses-permission android:name="android.permission.ACCESS_FINE_LO
……
</manifest>`
现在运行一下程序，就可以看到手机当前位置的经纬度信息了，如图 8.1 所示。
![Alt text](./1463559122953.png)

之后如果你拿着手机随处移动， 就可以看到界面上的经纬度信息是会变化的。 由此证实

### 8.2.2 反向地理编码，看得懂的位置信息
话说回来，刚才我们虽然成功获取到了设备当前位置的经纬度信息，但遗憾的是，这种经
#### 8.2.2.1 Geocoding API 的用法
其实 Android 本身就提供了地理编码的 API，主要是使用 GeoCoder 这个类来实现
它可以非常简单地完成正向和反向的地理编码功能， 从而轻松地将一个经纬值转换成看
不过，非常遗憾的是，GeoCoder长期存在着一些较为严重的 bug，在反向地理编码的时
还算比较幸运，谷歌又提供了一套 Geocoding API，使用它的话也可以完成反向地理
本小节中我们只是学习一下 GeocodingAPI的简单用法， 更详细的用法请参考官方文
GeocodingAPI 的工作原理并不神秘，其实就是利用了我们上一章中学习的 HTTP 协
在手机端我们可以向谷歌的服务器发起一条 HTTP 请求， 并将经纬度的值作为参数一同
GeocodingAPI 中规定了很多接口，其中反向地理编码的接口如下：
http://maps.googleapis.com/maps/api/geocode/json?latlng=40.71422
我们来仔细看下这个接口的定义， 其中http://maps.googleapis.com/maps/api
latlng=40.714224,-73.96145 表示传递给服务器去解码的经纬值是北纬 40.714
sensor=true_or_false 表示这条请求是否来自于某个设备的位置传感器，通常指定
如果发送 http://maps.googleapis.com/maps/api/geocode/jsonlatlng=40
"formatted_address" : "277 Bedford Avenue, 布鲁克林纽约州 11211美国
从这段内容中我们就可以看出北纬 40.714224度， 西经 73.96145 度对应的地理位
#### 8.2.2.2 对经纬度进行解析
使用 Geocoding API 进行反向地理编码的流程相信你已经很清楚了，我们先要发送一
修改 MainActivity 中的代码，如下所示：`
public class MainActivity extends Activity {
public static final int SHOW_LOCATION = 0;
……
private void showLocation(final Location location) {
```

```
new Thread(new Runnable() {
@Override
public void run() {
URL urlObject = null;
HttpURLConnection urlConnection = null;
InputStream in = null;
try {
// 组装反向地理编码的接口地址
StringBuilder url = new StringBuilder();
url.append("http://maps.google.com/maps/api/geocode/json?latlng=
url.append(location.getLatitude()).append(",");
url.append(location.getLongitude());
// 指定语言，保证服务器会返回中文数据
url.append("&language=zh-CN&sensor=true");
urlObject = new URL(url.toString());
urlConnection = (HttpURLConnection) urlObject.openConnection();
urlConnection.setRequestMethod("GET");
urlConnection.setRequestProperty("Content-Type", "application/js
urlConnection.connect();
// 判断请求码是否200，否则为失败
if (urlConnection.getResponseCode() == 200) {
in = urlConnection.getInputStream(); // 获取输入流
BufferedReader reader = new BufferedReader(new
InputStreamReader(in));
StringBuilder response = new StringBuilder();
String line;
while ((line = reader.readLine()) != null) {
response.append(line);
}
JSONObject jsonObject = new JSONObject(response.toString());
// 获取result节点下的位置信息
JSONArray resultArray = jsonObject.getJSONArray("results");
if (resultArray.length() > 0) {
JSONObject subObject = resultArray.getJSONObject(0);
// 取出格式化后的位置信息
String address = subObject.getString("formatted_address");
Message message = new Message();
message.what = SHOW_LOCATION;
```

```
message.obj = address;
handler.sendMessage(message);
}
} else {
Log.d("MainActivity", "xyz " + "请求url失败！"); }
} catch (Exception e) {
e.printStackTrace();}
}
}).start();
}
private Handler handler = new Handler() {
public void handleMessage(Message msg) {
switch (msg.what) {
case SHOW_LOCATION:
String currentPosition = (String) msg.obj;
positionTextView.setText(currentPosition);
break;
default:
break;
}
}
};
}`
```

观察 showLocation()方法，由于我们要在这里发起网络请求，因此必须开启一个子线
注意，在url设置中要将语言类型指定为简体中文，不然服务器会默认返回英文的位置信
在得到了这些位置信息后只需要取其中的第一条就可以了， 通常这也是最接近我们位置
不过别忘了，目前我们还是在子线程当中的，因此在这里无法直接将得到的位置信息显示
由于这里我们使用到了网络功能，因此还需要在 AndroidManifest.xml 中添加权限

```
<manifest xmlns:android="http://schemas.android.com/apk/res/andr
package="com.example.locationtest"
android:versionCode="1"
android:versionName="1.0" >
……
<uses-permission android:name="android.permission.ACCESS_FINE_LO
<uses-permission android:name="android.permission.INTERNET" />
……
</manifest>`
```

好了，现在可以重新运行一下程序了，结果如图 8.2 所示。

```
![Alt text](./1463559155247.png)

可以看到，手机当前的位置信息已经成功显示出来了！如果你带着手机移动了较远的距离
当然，在这个例子中我们只是对服务器返回的 JSON 数据进行了最简单的解析，位置信

### 8.2.2 使用第三方的库实现反向地理编码
做到这里，你对整个定位功能应该也了解的差不多了。但是你可能会觉得奇怪，明明按照
其实，这样说就真的是冤枉我了，代码本身是没有错的，但是因为是调用了谷歌的接口，
所以，下面要介绍的就是通过国内的一些地图sdk提供商来实现定位当前位置的功能，这

#### 8.2.2.1 高德SDK开发环境配置
1、注册开发者，创建应用
这个几乎是所有开放平台都通用的做法，无外乎注册帐号，成为开发者，然后创建一个A

2、下载SDK
从网站下载并解压得到定位包“AMap_Location_V2.x.x.jar“。

3、在Android Studio上进行配置
打开Android Studio编译器，切换到project查看方式，如图所示：
![Alt text](./1463642865384.png)

将下载的定位SDK的jar包复制到libs目录下，如果有老版本定位jar包在其中，请删除
![Alt text](./1463642886374.png)

4、配置AndroidMainfest.xml文件
首先，请在application标签中声明service组件,每个app拥有自己单独的定位serv
`<service android:name="com.amap.api.location.APSService"></serv
接下来声明使用权限`
<!--用于进行网络定位-->
<uses-permission android:name="android.permission.ACCESS_COARSE_
<!--用于访问GPS定位-->
<uses-permission android:name="android.permission.ACCESS_FINE_LO
<!--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_
<!--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_ST
<!--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_ST
```

```
<!--用于访问网络，网络定位需要上网-->
<uses-permission android:name="android.permission.INTERNET"></us
<!--用于读取手机当前的状态-->
<uses-permission android:name="android.permission.READ_PHONE_STA
<!--写入扩展存储，向扩展卡写入数据，用于写入缓存定位数据-->
<uses-permission android:name="android.permission.WRITE_EXTERNAL
最后设置Key，在application标签中加入
`<meta-data android:name="com.amap.api.v2.apikey" android:value=
在value后面填入你设定应用时所得到的key，至此，整个高德sdk的配置工作就基本完

#### 8.2.2.2 使用高德SDK实现自动定位
在演示开始之前，我们首先要对高德SDK有一定的了解。
高德定位服务包含GPS和网络定位（Wi-Fi和基站定位）两种能力。定位SDK将GPS、网
高精度定位模式：会同时使用网络定位和GPS定位，优先返回最高精度的定位结果；
低功耗定位模式：不会使用GPS，只会使用网络定位（Wi-Fi和基站定位）；
仅用设备定位模式：不需要连接网络，只使用GPS进行定位，这种模式下不支持室内环境

知道了这些之后，我们就来真正的实现自动定位功能。
第一，我们要初始化定位客户端，设置监听。
（PS：请在主线程中声明AMapLocationClient类对象，需要传Context类型的参数。
//声明AMapLocationClient类对象
public AMapLocationClient mLocationClient = null;
//声明定位回调监听器
public AMapLocationListener mLocationListener = new AMapLocation
//初始化定位
mLocationClient = new AMapLocationClient(getApplicationContext()
//设置定位回调监听
mLocationClient.setLocationListener(mLocationListener);`

第二，我们要配置定位参数，在Activity中的onCreate()中行初始化即可启动定位。
设置定位参数包括：定位模式（高精度定位模式，低功耗定位模式和仅设备定位模式），
//声明mLocationOption对象
public AMapLocationClientOption mLocationOption = null;
//初始化定位参数
mLocationOption = new AMapLocationClientOption();
//设置定位模式为高精度模式，Battery_Saving为低功耗模式，Device_Sensors是
mLocationOption.setLocationMode(AMapLocationMode.Hight_Accuracy)
//设置是否返回地址信息（默认返回地址信息）
```

```
mLocationOption.setNeedAddress(true);
//设置是否只定位一次,默认为false
mLocationOption.setOnceLocation(false);
//设置是否强制刷新WIFI，默认为强制刷新
mLocationOption.setWifiActiveScan(true);
//设置是否允许模拟位置,默认为false，不允许模拟位置
mLocationOption.setMockEnable(false);
//设置定位间隔,单位毫秒,默认为2000ms
mLocationOption.setInterval(2000);
//给定位客户端对象设置定位参数
mlocationClient.setLocationOption(mLocationOption);
//启动定位
mlocationClient.startLocation();`
```

第三，我们要实现AMapLocationListener接口，获取定位结果。
AMapLocationListener接口只有onLocationChanged方法可以实现，用于接收异步

```
public void onLocationChanged(AMapLocation amapLocation) {
    if (amapLocation != null) {
        if (amapLocation.getErrorCode() == 0) {
        //定位成功回调信息，设置相关消息
        amapLocation.getLocationType();//获取当前定位结果来源，如网络
        amapLocation.getLatitude();//获取纬度
        amapLocation.getLongitude();//获取经度
        amapLocation.getAccuracy();//获取精度信息
        SimpleDateFormat df = new SimpleDateFormat("yyyy-MM-dd H
        Date date = new Date(amapLocation.getTime());
        df.format(date);//定位时间
        amapLocation.getAddress();//地址，如果option中设置isNeedAdd
        amapLocation.getCountry();//国家信息
        amapLocation.getProvince();//省信息
        amapLocation.getCity();//城市信息
        amapLocation.getDistrict();//城区信息
        amapLocation.getStreet();//街道信息
                amapLocation.getStreetNum();//街道门牌号信息
        amapLocation.getCityCode();//城市编码
        amapLocation.getAdCode();//地区编码
                amapLocation.getAOIName();//获取当前定位点的AOI信息
    } else {
```

```
                //显示错误信息ErrCode是错误码，errInfo是错误信息，详见错误
        Log.e("AmapError","location Error, ErrCode:"
            + amapLocation.getErrorCode() + ", errInfo:"
            + amapLocation.getErrorInfo());
        }
    }
}`
```

最后，我们要停止自动定位。
停止定位：`
mlocationClient.stopLocation();//停止定位`
销毁定位客户端：
销毁定位客户端之后，若要重新开启定位请重新New一个AMapLocationClient对象。
mlocationClient.onDestroy();//销毁定位客户端。`

看到这里，相信你对于如何运用高德SDK来实现自动定位也已经有了一定的了解了。因为
其实高德sdk的使用远远不止于这些方面，还有很多其他的功能等待你去发掘，比如可以

5.实现指南针基本功能（磁场传感器调用）

简要说明：这是一个简单的指南针应用，实现指南针基本的功能：基本的方向指定。旋转手机就能够在手机界面中看出方向的变换,这种操作在一个Activity中实现。
详细步骤：

1. 第一步：获得传感器管理器

```
    //获得传感器管理器
        manager = (SensorManager) getSystemService(Context.SENSO
```

2. 第二步：为具体的传感器注册监听器

这里使用磁阻传感器方法Sensor.TYPE_ORIENTATION;
SENSOR_TYPE_ORIENTATION这个传感器在android 2.2之后就不

推荐使用了,在Android Studio中可以看到会有条横线横在代码中间,但是仍然能够使用，因此我还是使用这个方法:
int TYPE_ORIENTATION 磁场传感器使用的常量

```
@Override
    protected void onResume() {
        //为具体的传感器注册监听器 ,这里使用磁阻传感器Sensor.TYPE_ORIEN
        Sensor sensor = manager.getDefaultSensor(Sensor.TYPE_ORI
        //注册传感器监听事件
        manager.registerListener(listener, sensor,
                SensorManager.SENSOR_DELAY_GAME);   //SENSOR_DEL
        super.onResume();
}
```

3. 第三步：设置注销传感器监听事件

```
@Override
    //注销传感器监听事件
    protected void onPause() {
        manager.unregisterListener(listener);
        super.onPause();
    }
```

不需要的传感器尽量要解除注册，特别是当activity处于失去焦点的状态时。如果不按照以上去做的话，手机电池很快会被用完。
还要注意的是当屏幕关闭的时候，传感器也不会自动的解除注册。

所以我们可以利用activity 中的 onPause() 方法和onresume()方法。
在onresume方法中对传感器注册监听器，在onPause()方法中解除注册。

4. 第四步：实现具体的监听方法

SensorEventListener接口中定义了两个方法：onSensorChanged和onAccuracyChanged。
当传感器的值发生变化时，例如磁阻传感器的方向改变时会调用onSensorChanged方法。当传感器的精度变化时会调用onAccuracyChanged方法。

onSensorChanged方法只有一个SensorEvent类型的参数event。其中SensorEvent类有一个values变量非常重要，该变量的类型是float[]。但该变量最多只有3个元素，而且根据传感器的不同，values变量中元素所代表的含义也不同。由于在这个Activity中仅仅是实现指南针的基本功能，只需要方向值的改变，因此值选用values[0]这个变量。

values[0]：该值表示方位，也就是手机绕着Z轴旋转的角度。0表示北（North）；90表示东（East）；180表示南（South）；270表示西（West）。如果values[0]的值正好是这4个值，并且手机是水平放置，表示手机的正前方就是这4个方向。

```
private final class SensorListener implements SensorEventListene
     private float predegree = 0;
     public void onSensorChanged(SensorEvent event) {
         float degree = event.values[0];// 存放了方向值
         RotateAnimation animation = new RotateAnimation(predegr
               Animation.RELATIVE_TO_SELF, 0.5f,
               Animation.RELATIVE_TO_SELF, 0.5f);//控件以自身中心为
         //设置动画执行的时间（单位：毫秒）;持续时间为0.2s
         animation.setDuration(200);
         //设置旋转的图片
         imageView.startAnimation(animation);
         predegree = -degree;
     }
     public void onAccuracyChanged(Sensor sensor, int accuracy)
     }
}
```

至此，有关于磁场传感器的方法调用就已经基本实现了。

GitHub代码：https://github.com/hzuapps/android-labs/tree/master/app/src/main/java/edu/hzuapps/androidworks/homeworks/net1314080903146

7. 设置音量

8. 获取短信

9. 相机

10. 手机震动（调用加速度传感器）

简要说明：调用加速度传感器检测摇晃频率，摇晃频率达到速度阀值，手机震动。

1. 修改AndroidManifest.xml，添加控制手机震动及调用加速度传感器的权限

2. 创建加速度传感器

```
public void start(){
//获得传感器管理器
sensorManager=(SensorManager)
mContext.getSystemService(Context.SENSOR_SERVICE);
```

```
if(sensorManager!=null){
//获得加速度传感器
sensor=sensorManager.getDefaultSensor(Sensor.TYPE_AC
CELEROMETER);
//注册加速度传感器
if(sensor!=null){
sensorManager.registerListener(this,sensor,
SensorManager.SENSOR_DELAY_GAME);
}
}
//加速度感应器感应获得变化数据
public void onSensorChanged(SensorEvent event){
//当前检查时间
long currentUpdateTime=System.currentTimeMillis();
//两次检测的时间间隔
long timeInterval=currentUpdateTime-lastUpdateTime;
//判断是否达到了检测时间间隔
if(timeInterval=SPEED_SHRESHOLD)
}
```

3. 震动

```
public void StartVibrato(){
//第一个参数是节奏数组
mVibrator.vibrate(new long[] { 500,200,500,200
},-1);
}
```

实验9：Android综合实验

实验9: Android综合实验

1. 播报数字

+1. 声明并初始化SoundPool类、HashMap类的实例化对象sp和spMap

```
1.  private HashMap<String, Integer> spMap=null;    //用于管理音频流
2.  private SoundPool sp;   // 音频池
3.  private int soundId;    // 音频ID
4.  protected void onCreate(Bundle savedInstanceState) {
5.    .
6.    .
7.    .
8.    // 初始化HashMap<String, Integer>类的实例对象spMap
9.    spMap = new HashMap<>();
10.   // 初始化SoundPool类的实例对象sp，并设置最多可容纳16个音频流
11.   sp = new SoundPool(16, AudioManager.STREAM_MUSIC, 0);
12.   .
13.   .
14.   .
15. }
```

1. 加载音频文件，并用HashMap类来管理加载的音频文件；为按钮添加点击事件

```
1.      // 用SoundPool类的load方法加载指定音频文件，并用soundId保存返回的音频ID。
2.      // 用HashMap类来管理这些音频流
3.      soundId = sp.load(this, R.raw.zero, 1);
```

```
        spMap.put("0", soundId);
        soundId = sp.load(this, R.raw.one, 1);
        spMap.put("1", soundId);
        soundId = sp.load(this, R.raw.two, 1);
        spMap.put("2", soundId);
        soundId = sp.load(this, R.raw.three, 1);
        spMap.put("3", soundId);
        soundId = sp.load(this, R.raw.four, 1);
        spMap.put("4", soundId);
        soundId = sp.load(this, R.raw.five, 1);
        spMap.put("5", soundId);
        soundId = sp.load(this, R.raw.six, 1);
        spMap.put("6", soundId);
        soundId = sp.load(this, R.raw.seven, 1);
        spMap.put("7", soundId);
        soundId = sp.load(this, R.raw.eight, 1);
        spMap.put("8", soundId);
        soundId = sp.load(this, R.raw.nine, 1);
        spMap.put("9", soundId);
        soundId = sp.load(this, R.raw.ac, 1);
        spMap.put("ac", soundId);
        soundId = sp.load(this, R.raw.del, 1);
        spMap.put("del", soundId);
        soundId = sp.load(this, R.raw.div, 1);
        spMap.put("div", soundId);
        soundId = sp.load(this, R.raw.dot, 1);
        spMap.put(".", soundId);
        soundId = sp.load(this, R.raw.equal, 1);
        spMap.put("equal", soundId);
        soundId = sp.load(this, R.raw.minus, 1);
        spMap.put("minus", soundId);
        soundId = sp.load(this, R.raw.mul, 1);
        spMap.put("mul", soundId);
        soundId = sp.load(this, R.raw.plus, 1);
        spMap.put("plus", soundId);
        // 为按钮设置点击监听事件
        findViewById(R.id.btn0).setOnClickListener(this);
        findViewById(R.id.btn1).setOnClickListener(this);
```

```
42.       findViewById(R.id.btn2).setOnClickListener(this);
43.       findViewById(R.id.btn3).setOnClickListener(this);
44.       findViewById(R.id.btn4).setOnClickListener(this);
45.       findViewById(R.id.btn5).setOnClickListener(this);
46.       findViewById(R.id.btn6).setOnClickListener(this);
47.       findViewById(R.id.btn7).setOnClickListener(this);
48.       findViewById(R.id.btn8).setOnClickListener(this);
49.       findViewById(R.id.btn9).setOnClickListener(this);
50.       findViewById(R.id.btnadd).setOnClickListener(this);
51.       findViewById(R.id.btnsub).setOnClickListener(this);
52.       findViewById(R.id.btnmul).setOnClickListener(this);
53.       findViewById(R.id.btndiv).setOnClickListener(this);
54.       findViewById(R.id.btnclr).setOnClickListener(this);
55.       findViewById(R.id.btneq).setOnClickListener(this);
```

2. 根据点击按钮的侦听事件，确定播放哪个音频文件

```
1.  public void onClick(View v) {
2.       switch(v.getId()){
3.           case R.id.btn0:
4.               textView1.append("0");  //在UI界面的TextView中显示 0
5.               sp.play(spMap.get("0"), 1, 1, 0, 0, 1); //播放音频
    流
6.               break;
7.           case R.id.btn1:
8.               textView1.append("1");
9.               sp.play(spMap.get("1"), 1, 1, 0, 0, 1);
10.              break;
11.          case R.id.btn2:
12.              textView1.append("2");
13.              sp.play(spMap.get("2"), 1, 1, 0, 0, 1);
14.              break;
15.          case R.id.btn3:
16.              textView1.append("3");
17.              sp.play(spMap.get("3"), 1, 1, 0, 0, 1);
18.              break;
19.          case R.id.btn4:
20.              textView1.append("4");
```

```
21.                 sp.play(spMap.get("4"), 1, 1, 0, 0, 1);
22.                 break;
23.             case R.id.btn5:
24.                 textView1.append("5");
25.                 sp.play(spMap.get("5"), 1, 1, 0, 0, 1);
26.                 break;
27.             case R.id.btn6:
28.                 textView1.append("6");
29.                 sp.play(spMap.get("6"), 1, 1, 0, 0, 1);
30.                 break;
31.             case R.id.btn7:
32.                 textView1.append("7");
33.                 sp.play(spMap.get("7"), 1, 1, 0, 0, 1);
34.                 break;
35.             case R.id.btn8:
36.                 textView1.append("8");
37.                 sp.play(spMap.get("8"), 1, 1, 0, 0, 1);
38.                 break;
39.             case R.id.btn9:
40.                 textView1.append("9");
41.                 sp.play(spMap.get("9"), 1, 1, 0, 0, 1);
42.                 break;
43.             case R.id.btnadd:
44.                 sp.play(spMap.get("plus"), 1, 1, 0, 0, 1);
45.                 items.add(new
    Item(Double.parseDouble(textView1.getText().toString()),
    Type.num));
46.                 checkAndcompute();
47.                 items.add(new Item(0, Type.add));
48.                 textView1.setText("");
49.                 break;
50.             case R.id.btnsub:
51.                 sp.play(spMap.get("minus"), 1, 1, 0, 0, 1);
52.                 items.add(new
    Item(Double.parseDouble(textView1.getText().toString()),
    Type.num));
53.                 checkAndcompute();
54.                 items.add(new Item(0, Type.sub));
```

```
55.                 textView1.setText("");
56.                 break;
57.             case R.id.btnmul:
58.                 sp.play(spMap.get("mul"), 1, 1, 0, 0, 1);
59.                 items.add(new
    Item(Double.parseDouble(textView1.getText().toString()),
    Type.num));
60.                 checkAndcompute();
61.                 items.add(new Item(0, Type.mul));
62.                 textView1.setText("");
63.                 break;
64.             case R.id.btndiv:
65.                 sp.play(spMap.get("div"), 1, 1, 0, 0, 1);
66.                 items.add(new
    Item(Double.parseDouble(textView1.getText().toString()),
    Type.num));
67.                 checkAndcompute();
68.                 items.add(new Item(0, Type.div));
69.                 textView1.setText("");
70.                 break;
71.             case R.id.btneq:
72.                 sp.play(spMap.get("equal"), 1, 1, 0, 0, 1);
73.                 items.add(new
    Item(Double.parseDouble(textView1.getText().toString()),
    Type.num));
74.                 checkAndcompute();
75.                 textView1.setText(items.get(0).value + "");
76.                 String str = items.get(0).value+"";
77.                 new Test(str).start();  // 启动另一个线程来播放结果
78.                 items.clear();
79.                 break;
80.             case R.id.btnclr:
81.                 sp.play(spMap.get("ac"), 1, 1, 0, 0, 1);
82.                 textView1.setText("");
83.         }
84. }
```

3． 对于结算结果的音频播放：（1）先把计算结果转成字符串型，再

启动一个子线程来处理该字符串；（2）子线程获取传过来的计算结果的字符串后，停止播放的“等于”对应的音频，然后利用for循环语句和String类的substring()方法把计算结果一位一位的截取出来，并播放对应的音频文件。

```

```

```
          .
          .
          String str = items.get(0).value+"";
        new Test(str).start();  // 启动另一个线程来播放结果
          .
        .
```

class Test extends Thread{

```
  private String result;  //计算的结果
  public Test(String s){
      this.result = s;
  }
  @Override
  public void run() {
    String s;
      sp.stop(spMap.get("equal"));
      // 用for循环把计算结果分割成对应spMap中的key
      for(int i=0; i<result.length(); i++){
          try {
              Thread.sleep(350);  //该线程睡350毫秒
          } catch (InterruptedException e) {
              e.printStackTrace();
          }
          s = result.substring(i, i+1);
          sp.play(spMap.get(s), 1, 1, 0, 0, 1);
      }
  }
```

}