嵌入式开发综合

课程设计报告

设计题目：推箱子小游戏

专　　业　计算机科学与技术

班　　级　计 191

学　　生　3190673013　吴世杰

　　3190913021　徐俊豪

指导教师　王海晟

西安理工大学

2021　年　秋　学期

设计任务

吴世杰: 游戏核心功能的设计,比如 Activity 和 Activity 之间的交互过程，绘制游戏地图界面，小人移动的设计，帮助视频，小人移动音效，关卡选择对话框，设计了小人和箱子移动的逻辑判断和对于之前地图状态的保存以便于回退上一步，设计了对于退出游戏时状态的保存，在继续游戏时只需加载退出游戏时保存的状态. 设计了背景图选择的函数,并且使背景图一直生效.完成了报告的排版和前期设计的内容和流程图的编写。

徐俊豪: 游戏菜单和按钮的设计,主要参与了 XML 的设计,对于菜单和进入游戏界面按钮的布局,绑定按钮调用的函数和按钮的布局,寻找了背景图和音频等素材.设计了背景音乐播放的功能和游戏信息的编写.

评语(教师填写)

目录

Part I

总体说明

一、 选题理由

随着智能手机一步步走进普通人的生活，它将成为人们获取信息的主要设备。因此，手机的娱乐应用服务将会有很大的发展空间，游戏也是其中之一。本系统主要是实现基于 Android 的推箱子游戏，推箱子是一款来自日本的古老游戏，其设计目的是训练人的逻辑思维能力。游戏场景一般是设定在空间狭小的仓库中，要求把箱子摆放到指定位置。这就要求玩家巧妙的运用有限的空间和通道，合理的安排箱子的位置和移动次序才可能完成任务。随着计算机游戏的发展，很多编程爱好者基于该游戏的思想开发出了各种版本、各种类型的推箱子。这其中也包括很多手机版本的实现，伴随着 Android SDK 技术的不断发展，一些基于 Android 应用也不断推陈出新，很快推箱子游戏便进入了千家万户。此 Android 推箱子游戏是基于 Android SDK2.1 基础上设计的，主要分为开始游戏、声音开关、游戏说明、退出游戏四个功能模块的设计与实现。本文首先论述了 android 系统的背景和研究现状，接着简要的介绍了 android 的技术及对 android 分析，并且介绍了 android 应用程序的结构。在程序开发中，采用了先设计好游戏的类框架，然后按照各个类的实现进行代码的编写与实现功能。

二、 题目要求

功能：推箱子是源自日本的一种古老的游戏，游戏场景一般是设定在空间狭小的仓库中，要求把箱子摆放到指定位置。要求玩家巧妙的运用有限的空间和通道，合理的安排箱子的位置和移动次序，才可能完成任务

1. 游戏运行后，首先进入欢迎动画界面。
2. 要求设计出菜单界面，至少有四种菜单供用户选择。
3. 要求在适当的位置加上游戏的音效，并且可以对音效进行开启和关闭的设置。
4. 要求有帮助界面，简单介绍游戏的基本功能和玩法。

三、 可行性分析

本次安卓推箱子游戏开发，我们经过讨论后选择了 Android studio 作为开发工具，其优点在于对于 SDK 和 JDK 开发环境的配置相对来说较为自动化，省去了配置开发环境的时间。本次课设的难度主要在于 android 开发环境的不熟悉，对于各个组件之间的控制与功能实现之间的联系，反而对于推箱子的功能的算法到并不是很复杂。

1. 技术可行性: 采用 AndroidSDK2.7 和 Android Studio 进行开发
2. 经济可行性: 随着计算机游戏的发展，很多编程爱好者基于该游戏的思想开发了出各种版本、各种类型的推箱子。这其中也包括很多手机版本的实现，伴随着手机与计算机的普及，很快推箱子游戏便进入了千家万户。因此，从经济上来说，开发推箱子游戏不需要很大的投入，硬件上只需普通的 PC 电脑一台，附加配置好模拟器的运行环境即可，有条件的话可以再配备 Android 系统的真机

四、 需求分析

实现功能如下:

1. 游戏菜单界面: 此界面主要提供用户功能选择
2. 玩家定制功能: 通过提供游戏关卡选择，使玩家根据自己的情况选择不同的关卡进子游戏。
3. 继续游戏功能: 通过字符串参数传递和保存来实现对之前游戏状态的重载
4. 游戏背景音乐设置: 主要实现让该游戏提供音乐播放功能
5. 游戏背景图片选择功能: 提供若干图片供用户选择
6. 游戏帮助: 当用户遇到难关时可以查看帮助视频来更好的过关
7. 游戏说明: 对游戏进行简要的说明.

五、 总体功能设计

经典的推箱子自日本的古老游戏, 用于训练人的逻辑思维和思考能力。实现的功能是在一个规定的有限的区域中, 把木箱从初始位置移动到指定位置, 移动过程中稍不小心就会出现箱子无法移动或者通道被堵住的情况, 而且箱子只能推, 不能拉, 所以需要巧妙的利用有限的空间和通道, 合理安排移动的次序和位置, 才能顺利的完成任务。

游戏规则如下:

1. 游戏开始后, 人物可以往四个方向移动: 上、下、左、右点击屏幕可以控制任务的移动, 在前进方向上没有墙阻挡时, 可以一次移动“人”, 当“人”和“箱子”相连接, 如果对应方向没有墙阻挡, 按方向键, 则可以将箱子推动一格
2. 当‘人’将箱子推入到空位置后, 空格位置被人占据, 空格消失
3. 当所有的箱子被推入目的地, 游戏结束, 并且进入下一关
4. 游戏共设有 5 关, 第 1 关的游戏最简单, 然后难度依次递增, 第 5 关难度最大, 如果到了最后一关游戏结束, 则可以返回第一关或者退出游戏

六、 概要设计

6.1 总体模块划分

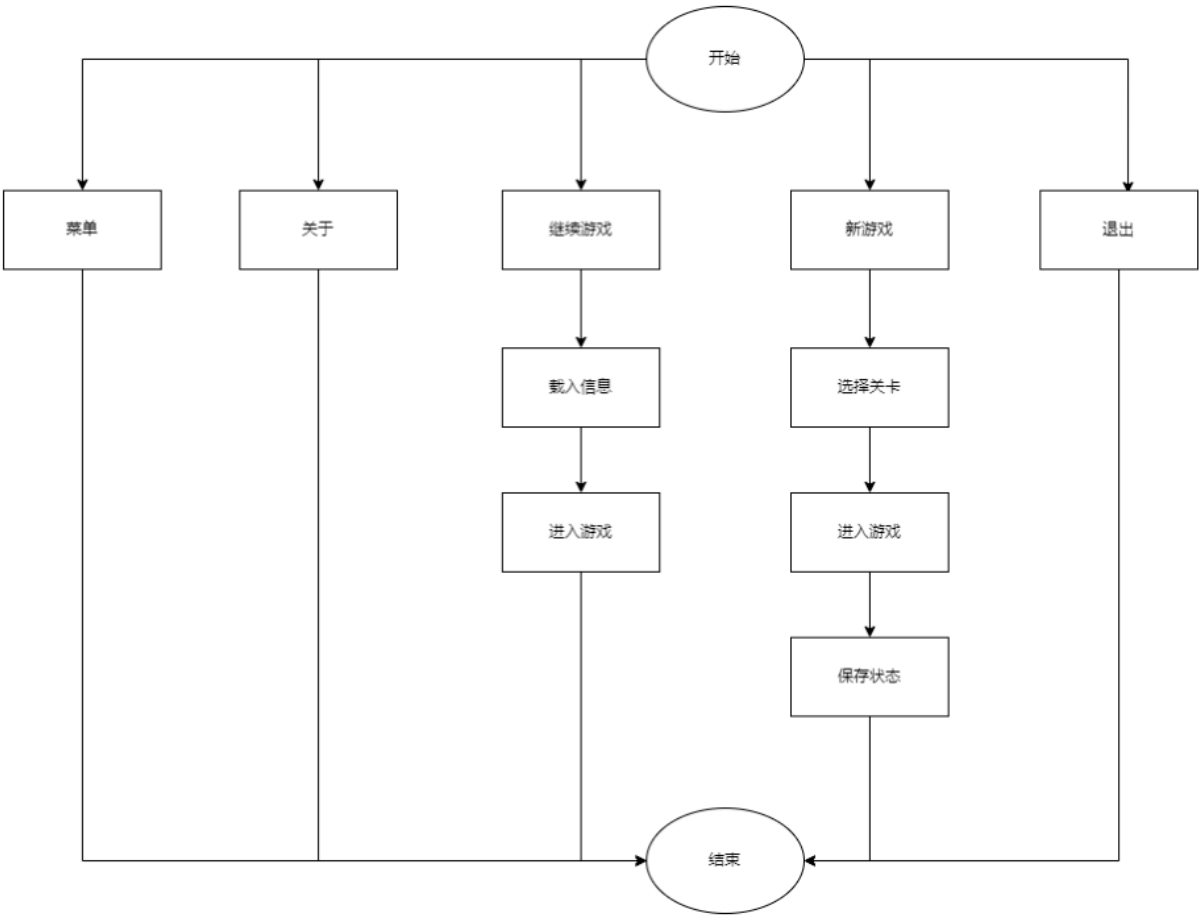


图 1　功能流程图

其中菜单的功能如下:

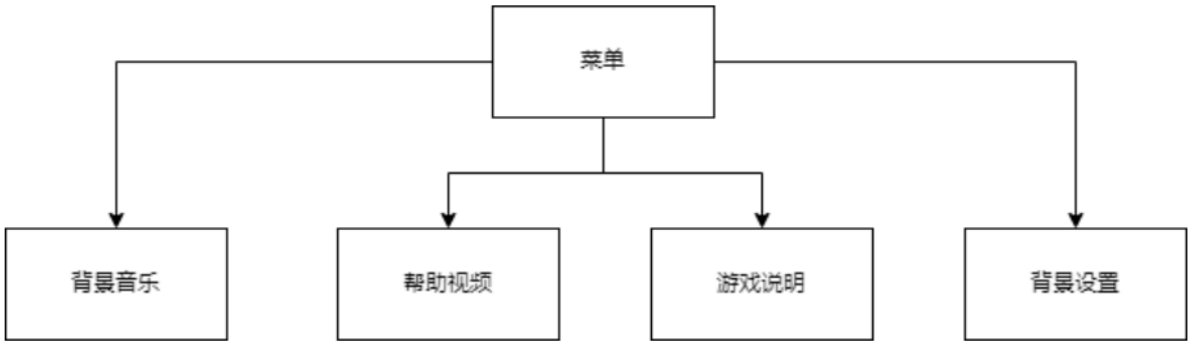


图 2　菜单功能

Part II

吴世杰

一、 设计思路

游戏大致分为界面布局、人物移动和逻辑判断三个部分。

1. 界面布局: 界面使用二维数组布局，页面布局时，为了界面显示的友好性，需要注意计算窗体以及图片素材的大小，调整显示位置
2. 人物移动: 小人移动功能的实现需要给窗体添加键盘监听事件，判断玩家控制小人的移动方向，同时通过切换小人图片，实现小人移动的动态效果，优化页面显示
3. 逻辑判断: 通过判断二维数组中的数据，达到小人遇到障碍物、箱子前有障碍物、2个箱子肩并肩时不能继续向前移动、小人随箱子一起移动，实现推箱子功能，并且在小人移动时不断更换二维数组中数据进行判断

二、 功能实现

详细的设计过程为: 游戏界面设计布局 (窗体、小人、障碍物等图片素材) → 功能实现 (小人移动以及小人图片更换，箱子移动) → 逻辑判断处理 → 判断是否闯关成功。

需要注意的是，游戏整体设计过程中，最重要是逻辑判断及控制箱子移动，判断箱子什么情况下可以移动，什么情况下无法移动，什么时候闯关成功，需要理清逻辑关系。设计过程中一些重要变量

```
1     public int gate=Game.gate;
2     private int manx;
3     private int many;
4     public int maprow=0;
5     public int mapcolumn=0;
6     private int width=0;
7     private int heith=0;
8     private Bitmap pic[]=null;
9     final int WALL=1;
10    final int GOAL=2;
11    final int ROAD=3;
12    final int BOX=4;
13    final int BOXATGOAL=5;
14    final int MAN=6;
15    private Paint paint;
16    private Game game=null;
17    public  int [][]map=null;
18    private int [][]tem;//原来的地图
19    float currentx;
20    float currenty;
```

2.1 游戏界面设计

使用二维数组对页面排版，遍历二维数组，数字 1 代表墙体、数字 2 代表目的地、数字 3 代表道路、数字 4 代表箱子，数字 5 代表箱子到达目的地的状态，数字 6 代表小人，定义了一个 Bitmap(位图) 类型的数组，每一位都通过通过掉用 loadPic 函数来绑定图片资源用于界面的绘制，例如将墙体图片绑定到 WALL 变量上,loadPic(WALL,r.getDrawable(R.drawable.wall));WALL 的值是 1 其实就是绑定到了 pic[1] 上，添加图片，添加背景图、小人图、箱子、目的地图片，页面整体效果预览，如图 (1) 所示。

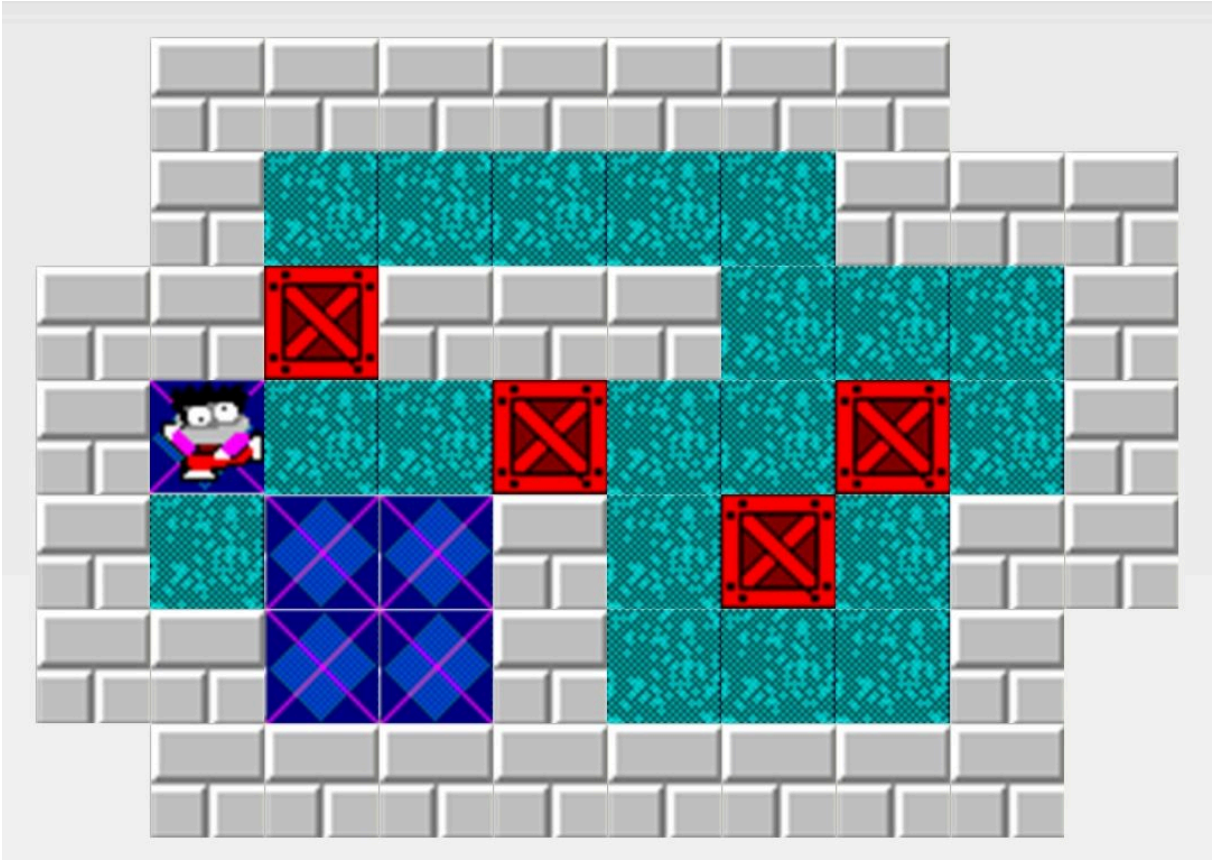


图 1　绘制效果

绘制代码如下:

```
    protected void onDraw(Canvas canvas)
    {
        super.onDraw(canvas);
        for (int i = 0; i < maprow; i++)
            for (int j = 0; j < mapcolumn; j++)
                if (map[i][j] != 0)
                    canvas.drawBitmap(pic[map[i][j]], xoff + j * tilesize,
                            yoff + i * tilesize, paint);
    }
```

对于不同大小的行和宽, 比如地图为 8×9 还是 6×9 的地图都可以同一的进行绘制, 对于不同地图大小只需要计算出每一个格子的宽因为多少即可, 计算代码如下:

```
int t=maprow>mapcolumn?maprow:mapcolumn;
int s1=(int) Math.floor((width-2*xoff)/t);
int s2=(int) Math.floor((heith-xoff)/t);
tilesize=s1<s2?s1:s2;
```

首先获得地图的行和列, 然后通过获取屏幕的宽和高来计算出每一行中一格的宽度和每一列中一格的宽度, 然后取最小即可.

2.2 小人移动设置

通过 MotionEvent 来监听点击时间，通过 getX() 和 getY() 来获取获得触摸点在当前 View 的 X 轴坐标和 Y 轴坐标，通过与当前小人所在位置的坐标做对比，并改变当前数组的值，利用 invalidate() 重画当前界面，来改变小人在地图中的位置，控制小人移动方向，实现小人的移动功能，改变箱子的位置。下面代码展示了判断小人向右和向左移动时的逻辑, 向上, 向下逻辑基本一致. 移动之后需要判读是否取得胜利, 并且要重画 View 树来显示.

```
public boolean onTouchEvent(MotionEvent event)
{
    soundPool.play(musicId.get(1),1,1, 0, 0, 1);
    currentx=event.getX();
    currenty=event.getY();
    float x=(float)(xoff+many*tilesize);
    float y=(float)(yoff+manx*tilesize);
    if(currenty>y&&(currenty<y+tilesize))
    {
        if(currentx>x+tilesize){  moveRight();  }
        if(currentx<x){    moveLeft();   }
    }
    //向上和向下就不展示了
    if(finished()){  nextGate();  } //移动后判断是否成功.
    this.invalidate(); //重画View树
    return super.onTouchEvent(event);
}
```

2.3 逻辑判断

判断小人是否能够移动要满足以下规制:

1. 判断小人前否有障碍物: 如果存在，不允许小人向前移动; 不存在，则小人可以向前移动。这里只需要判断二维数组中小人移动方向的前一个位置的数字即可，当其为

1 时，小人可以移动，否则不能移动

2. 判断小人前是否有箱子存在: 当存在箱子时 (数字为 4) ，还要判断箱子的前面是否有障碍物和另一个箱子 (1 或 4) ，存在时，小人无法推着箱子向前移动，否则可以推箱子移动。
3. 在二维数组中，如果小人的移动方向的下一个位置的数据是 4，代表其前面有 1 个箱子存在。如果箱子的下一个位置的数字不等于 1 或 4，代表其前面没有障碍物和箱子———不能同时推动两个 (以上) 箱子，小人和箱子可以同时移动———即推箱子前进，否则不能向前移动

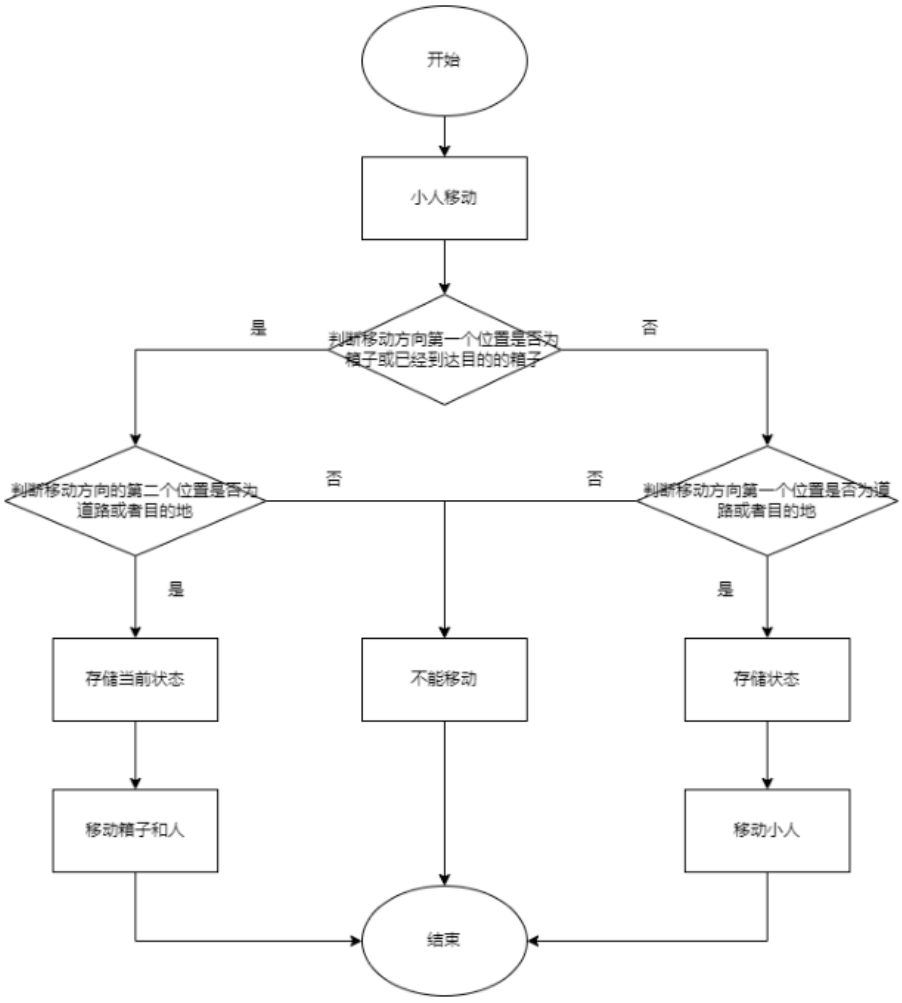


图 2　逻辑判断流程图

基于以上规则可以编写出小人移动的代码, 对于小人的移动方向无论是上下左右判断逻辑都是一致的, 以小人向上移动为例, 首先要判断下一个位置是不是箱子或者已经到达目的地的箱子, 如果是的话在判断箱子是否能移动, 如果箱子能移动那么小人跟着箱子移动, 如果不是的话只需要判断小人的下一个位置是不是可以移动的即是否是目的地或者道路. 向上移动代码如下所示.

```
    private void moveup()
    {
        if(map[manx-1][many]==BOX||map[manx-1][many]==BOXATGOAL)
        {
            if(map[manx-2][many]==GOAL||map[manx-2][many]==ROAD)
            {
                storyMap(map);
                map[manx-2][many]=map[manx-2][many]==GOAL?BOXATGOAL:BOX;
                map[manx-1][many]=MAN;
                map[manx][many]=roadorgoal(manx,many);
                manx--;
            }
        }
        else
        {
            if(map[manx-1][many]==ROAD||map[manx-1][many]==GOAL){
                storyMap(map);
                map[manx-1][many]=MAN;
                map[manx][many]=roadorgoal(manx,many);
                manx--;
            }
        }
    }
```

判断玩家是否闯关成功: 判断二维数组中目的地的数据是否从数字 4 改为数字 5, 当所有目的地的数据更改为数字 5 即不存在数字 4 和数字 2 时，代表玩家闯关成功。

```
    private boolean finished()
    {
        boolean finish=true;
        for(int i=0;i<maprow;i++)
            for(int j=0;j<mapcolumn;j++)
            {
                if(map[i][j]==GOAL||map[i][j]==BOX)
                    finish= false;
            }
        return finish;
    }
```

2.4 退回上一步功能

对于游戏中每一种状态都可以对应一个二维数组, 将二维数组, 将之前的状态转化为二维数组进行存储放入 ArrayList 中进行存储即可恢复之前的状态. 对此定义了一个类 CurrentMap 实现了获取当前状态和返回被存储状态. 每一次返回上一步就取出一个 CurrentMap, 在恢复至取出 CurrentMap 的状态, 如果 ArrayList 为空, 即没有之前状态了, 就用 Toast 来提示 You Cant't back,back 函数如下所示.

```
    public void back()
    {
        if(list.size()>0)
        {
            CurrentMap proMap=list.get(list.size()-1);
            map=proMap.getmap();
            getmanposition();
            list.remove(list.size()-1);
        }
        else
        {
            Toast.makeText(getContext(), "You Can't Back", Toast.LENGTH_LONG).show();
        }
    }
```

2.5 继续游戏功能

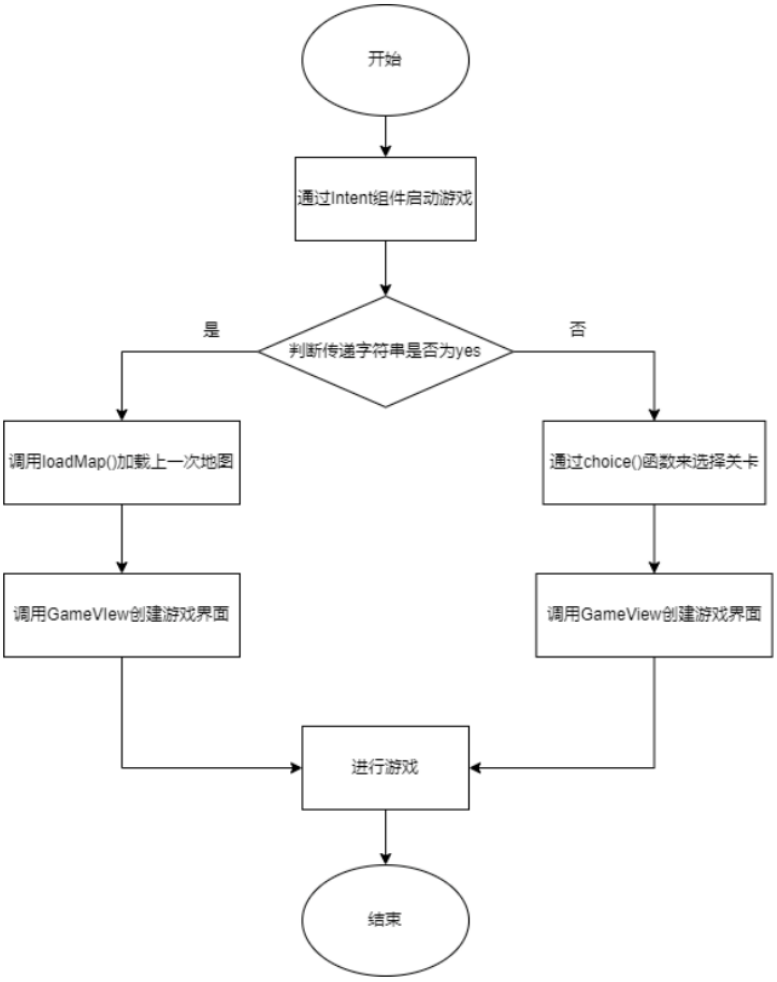


图 3　游戏加载流程图

1. 通过 Intent 组件来从菜单启动 Game

2. 通过 Intent 组件向 Game 传递 name 为”continue” 的字符串
3. 如果”continue” 的值为”yes” 则是继续游戏, 调用 loadMap() 函数加载上一次的游戏状态.
4. 如果”continue” 的值为”no” 则是开启新游戏, 调用 choice() 函数选择关卡
5. 绘制地图.

其中游戏的之前状态是通过 getPreferences(MODE_PRIVATE).edit().putString(”maps”,saveGame()).c来进行存储, 也通过 getPreferences.getString() 来存储当前游戏的状态, 继续游戏时读取上一次保存的字符串将其转化为二维数组来恢复状态.

```
1     public void loadMap()
2     {
3         gate = (int)getPreferences(MODE_PRIVATE).getLong("gate",0);
4         String mapstring =getPreferences(MODE_PRIVATE).getString("maps", first);
5         int ii=(int) getPreferences(MODE_PRIVATE).getLong("maprow", 8);
6         int jj=(int) getPreferences(MODE_PRIVATE).getLong("mapcolumn", 8);
7         int []maps=new int [mapstring.length()];
8         for(int i=0;i<maps.length;i++)
9         {
10            maps[i]=mapstring.charAt(i)-'0';
11        }
12        stored=new int [ii][jj];
13        int a=0;
14        for(int m=0;m<ii;m++)
15            for(int n=0;n<jj;n++)
16            {
17                stored[m][n]=maps[a];
18                a++;
19            }
20    }
```

2.6 帮助视频

对于每一个关卡都设置了帮助视频来帮助用户过关. 通过 AlertDialog.Builder 组件来让用户选择观看第几关的帮助视频. 并加载相应路径的视频进行播放.

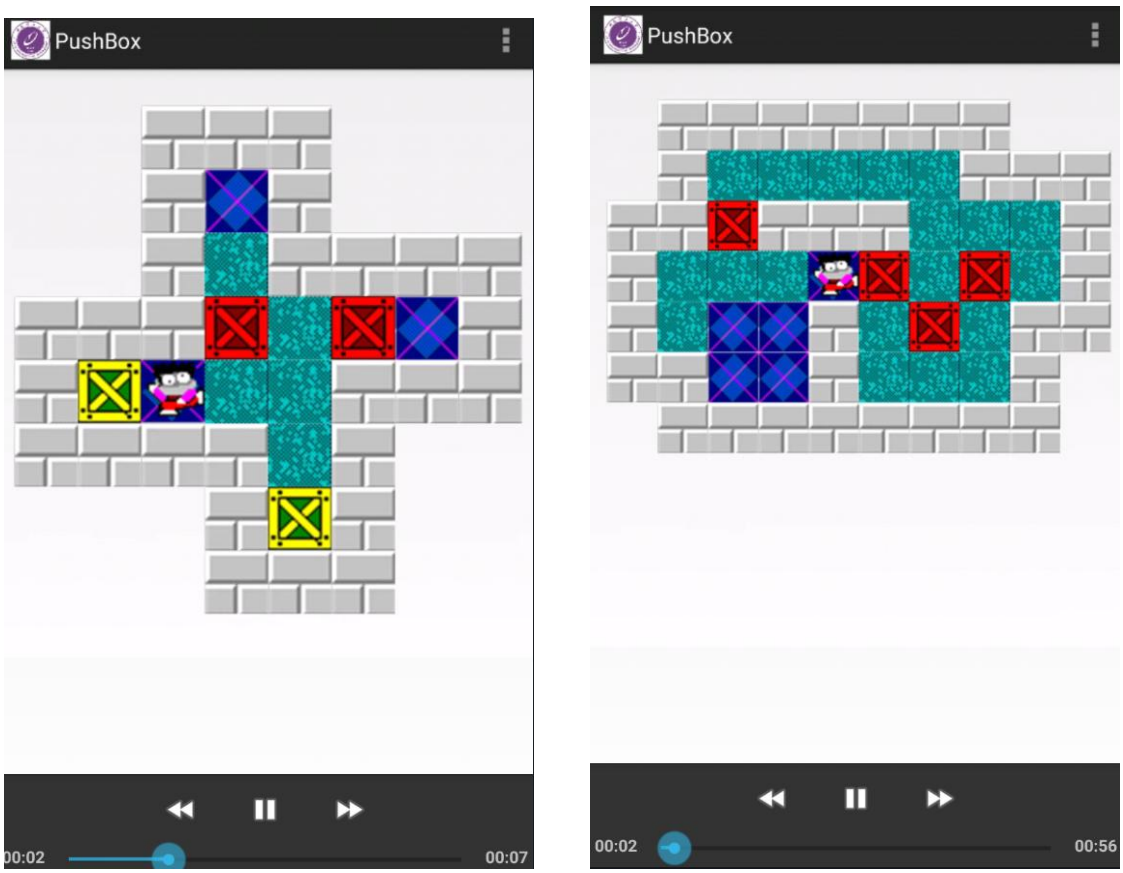
设置对于的视频路径.

```
1       vv.setVideoPath("android.resource://" + "com.example.pushbox" + "/" + R.raw.gate_1);
```

首先通过 AlertDialog.Builder 设置对话框,setTitle 设置标题,setItems 设置选项,setVideoPath(”androic+ ”com.example.pushbox” + ”/” + R.raw.gate_1); 设置路径, 效果如图 (4)

图 4　选择帮助视频

对于帮助视频界面的显示如图 (**??**) 所示. 展示了关卡 1 和关卡 2 的帮助视频, 在界面下方有重播, 暂停, 后退等功能.



(a) 关卡 1　　(b) 关卡 2

图 5　帮助视频展示

2.7 关卡设计

定义了 MapList 类, 里面定义了一个三维数组 map 用来存储关卡, 由于在获取关卡属性的时候都是采用 map.length 等自动编译生产的常量, 所以增加关卡只需要在 map 里增加一个二维数组即可, 其他相应部分会自动生成.

三、 编码与测试

3.1 音频加载错误

对每次人物移动绑定音效的时候, 刚开始采用了和播放背景音乐一样的方法, 使用 MediaPlayer 来播放代码如下

```
1       MediaPlayer mPlayer = MediaPlayer.create(getContext(), R.raw.click);
2       mPlayer.start();
```

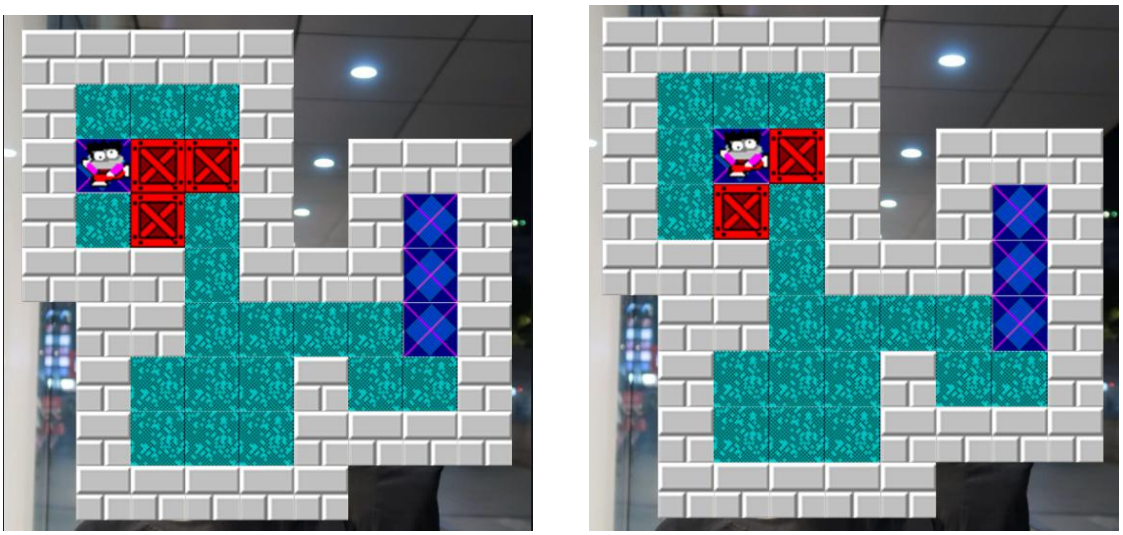
结果音频最多出现十次左右, 查看 run 下的报错 error（1，-19）, 查找原因是因为反复申请了 MediaPlayer 资源而没有得到释放, 可以通过再下一次调用 MediaPlayer 时释放上一次的资源或者采用 soundPool,soundPool 是用于短音频播放的更适合当前状态. 所以采用了 soundPool 来实现本功能.

soundPool 可以绑定多个音频, 每一个音频有一个 ID，将 ID 存入 HashMap 中, 每次通过 HashMap 来返回需要播放音频的 ID。

```
1       soundPool.play(musicId.get(1),1,1, 0, 0, 1);
```

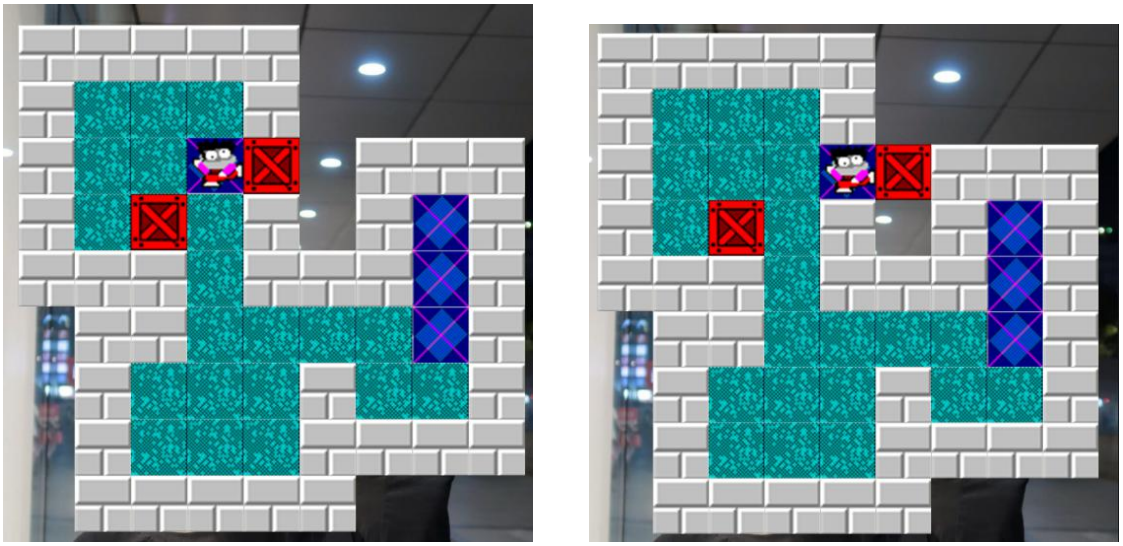
3.2 小人移动问题

出现箱子会重叠问题, 即两个箱子发生碰撞之后会合成一个箱子, 并且会出现将箱子推出游戏界面外的情况.



(a) 移动前　　(b) 移动后

图 6　箱子重叠问题



(a) 移动前　　(b) 移动后

图 7　箱子越界问题

出现以上两个问题的原因都是因为只判断了小人移动的下一个位置, 而没有判断小人移动的下下个位置是什么.

修改后判断代码如下:

```
1        if (map[manx-1][many]==BOX||map[manx-1][many]==BOXATGOAL)
2                if (map[manx-2][many]==GOAL||map[manx-2][many]==ROAD)
```

第二行代码为增加的判断.

3.3 资源加载报错

为对话框 AlertDialog.Builder 添加图标时出现 Expected resource of type drawable

```
1      AlertDialog.Builder dll = new AlertDialog.Builder(this);
2      final String[] Bgset = {"跑车1","跑车2","跑车3","跑车4"};
3      dll.setTitle("Which favorite's is select");
4      dll.setIcon(R.id.bgset); //报错
```

原因是因为资源类型不匹配导致 android 编译版本检查无法通过

```
1      Expected resource of type drawable
```

解决办法是在函数头之前面前加上 @SuppressWarnings("ResourceType") 注解.

四、 个人总结

虽然只是制作一个小小的推箱子游戏，却可以让我们更好的理解一个完整程序的设计思路、实现过程和编程思想，循环控制、逻辑判断等应用技巧都在具体的制作过程得到很好的锤炼。其间也会遇到一些问题，需要不断尝试完善，调试解决。通过这样的编程练习，能学到更多知识，在一个快乐的过程中学习，感受编程的魅力，享受满满的成就感。

同样让我了解了 Android 程序的开发流程, 首先画了一天时间看了网上的视频教程, 然后就开始 Activity 的设计和控件的设计, 这次编写过程中, 比较麻烦的是不同 Activity 之间的调用和重画 View 树的时机, 在使用画布和画笔来绘制页面也是这次课程设计的难点之一, 刚开始是通过获取屏幕的绝对位置 getRawX() 但是会出现部分图形缺省和格子大小不对的问题, 后来采用了 getX() 得以解决, 在任务移动的判断逻辑上也经过了反复测试才使得功能实现完整, 对于部分组件如 Toast 内置了常量在调用时需使用其自带的常量否则会出现报错, 总的来说在掌握了控件的使用之后能够比较快的设计出游戏的功能和界面,

Part III

徐俊豪

一、 界面设计

本次实验中关于界面的设计，我进行了游戏主界面、游戏菜单、首页界面、各功能界面（“about”、“music”、“help”、“背景选择”）的设计，采用了 Android 开发中的 xml 技术和 Android、java 中的一些控件。以下为各界面的效果图。

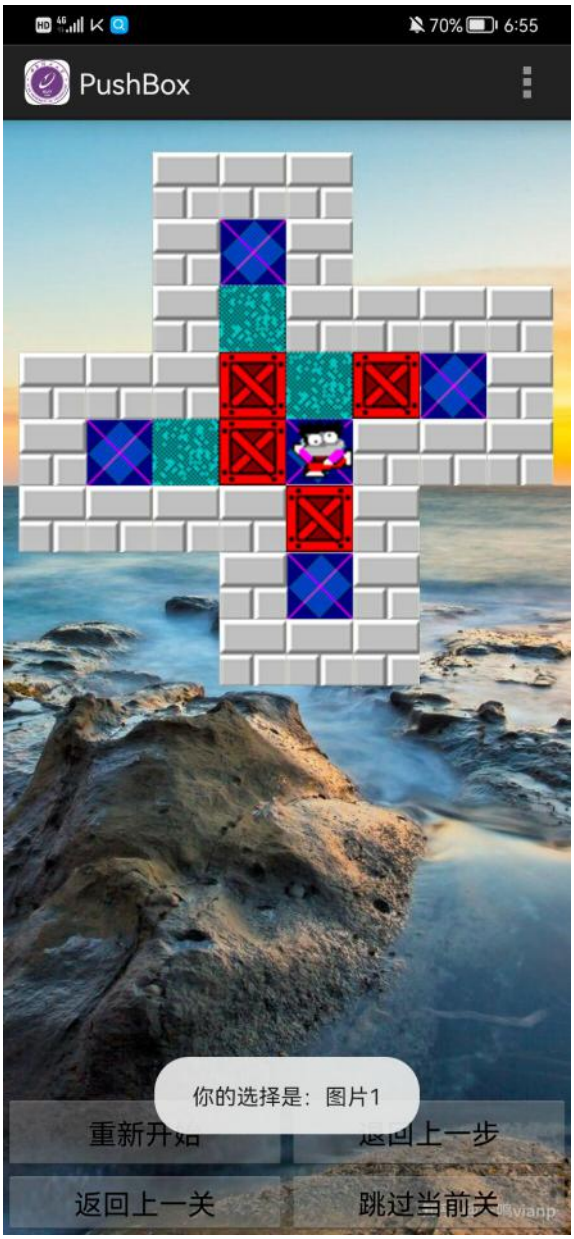
(a) 初始界面

(b) 关卡选择

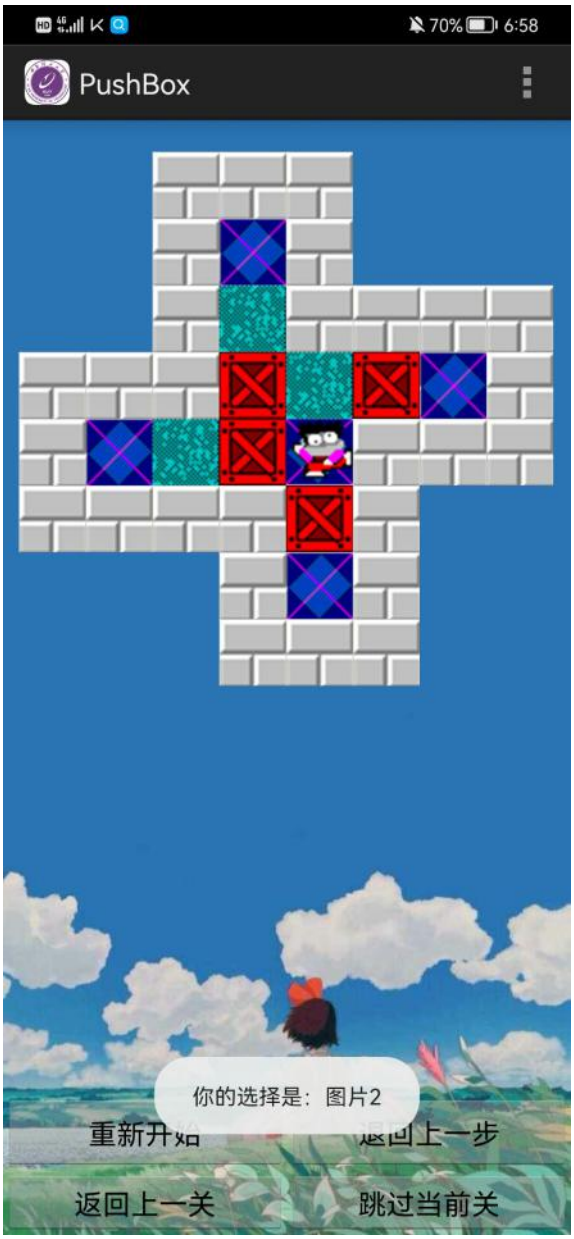
(c) 菜单展示

(d) 游戏说明

另外可以进行游戏背景的选择


PushBox
你的选择是：图片1
重新开始
返回上一步
返回上一关
跳过当前关

(e) 背景 1


PushBox
你的选择是：图片2
重新开始
返回上一步
返回上一关
跳过当前关

(f) 背景 2

二、 子模块设计

2.1 返回上一步

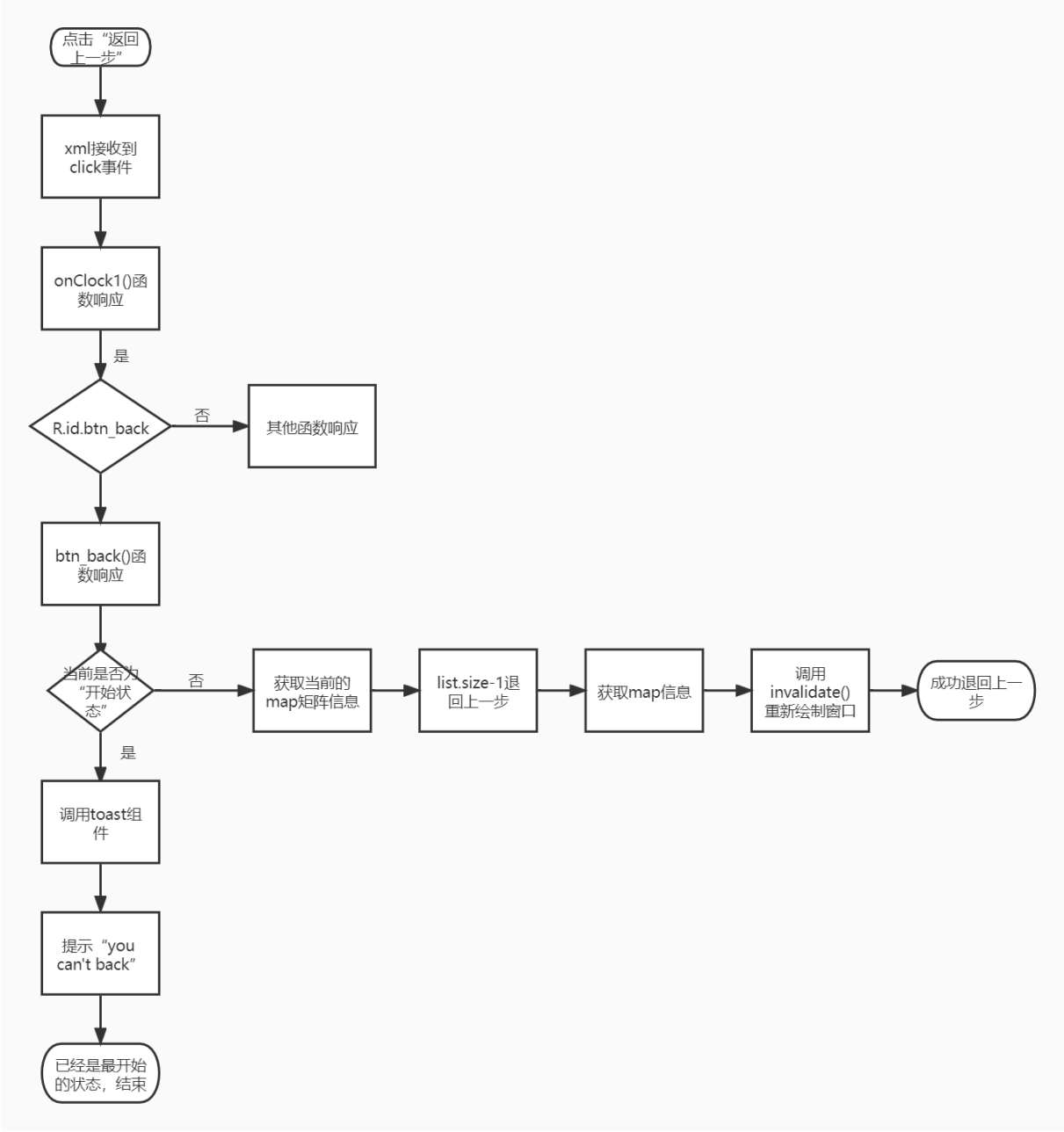


图 1　返回上一步流程图

2.2 重新开始

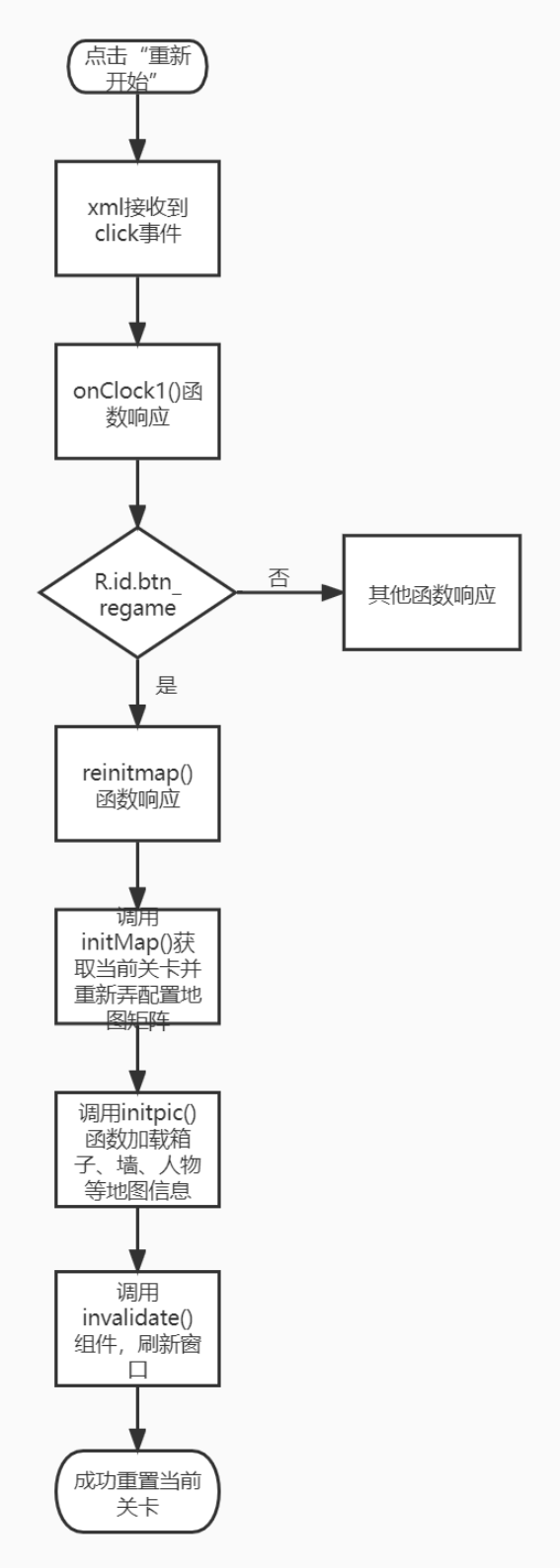


图 2　重新开始流程图

2.3 返回上一关

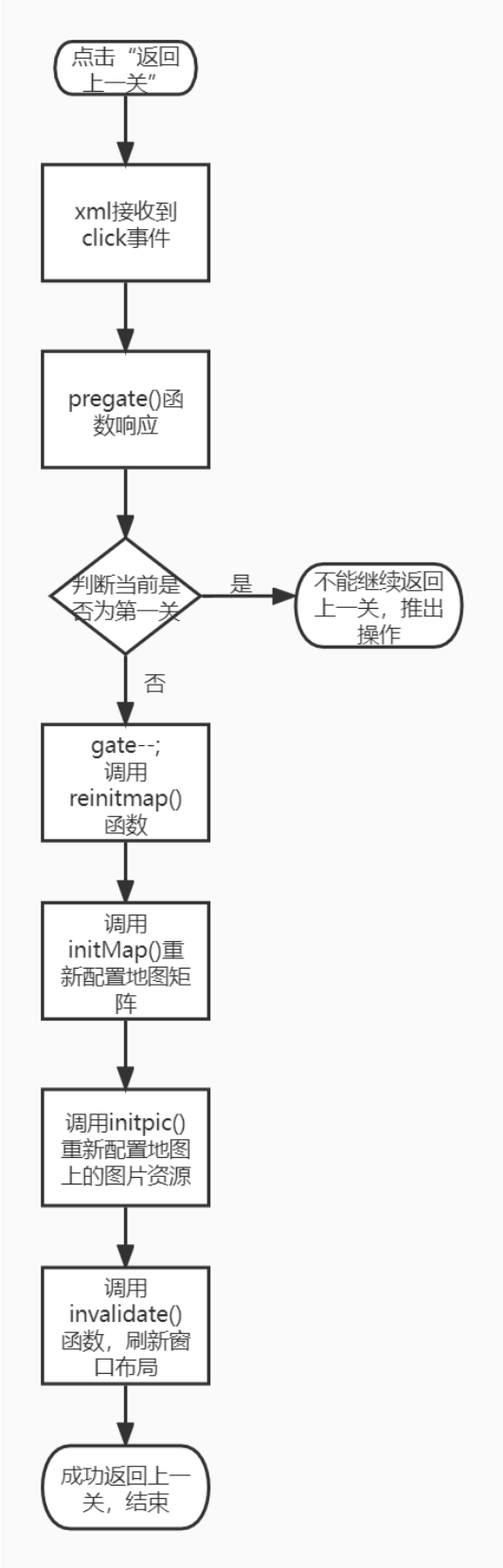


图 3　返回上一关流程图

2.4 跳过当前关

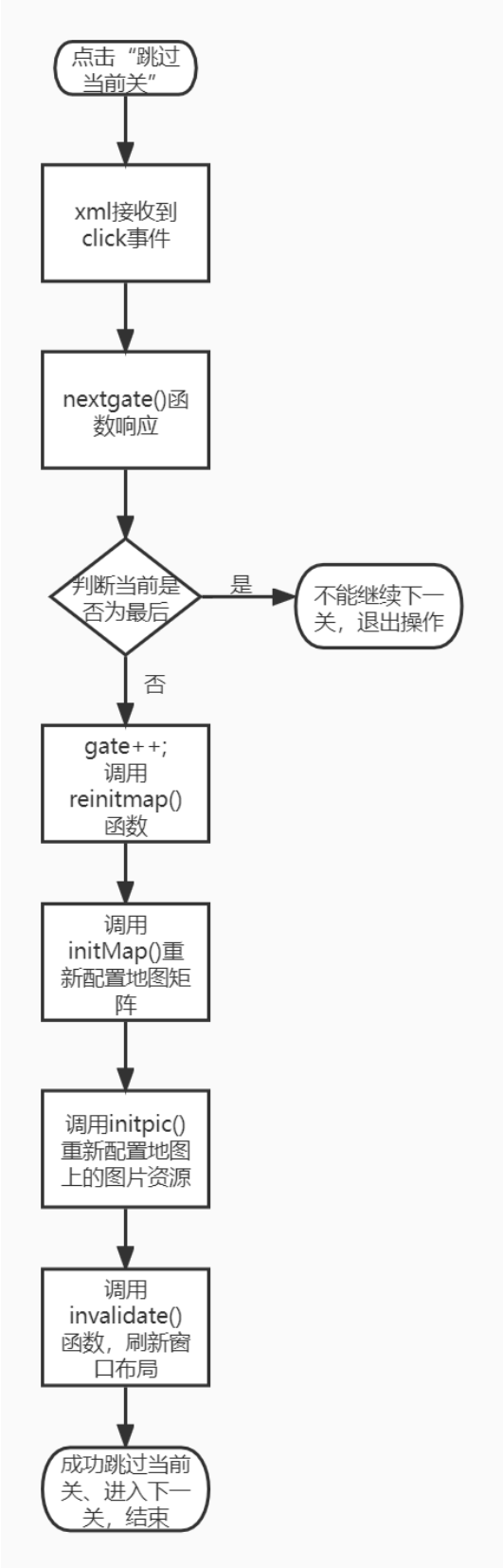


图 4　跳过当前关流程图

2.5 开启背景音乐

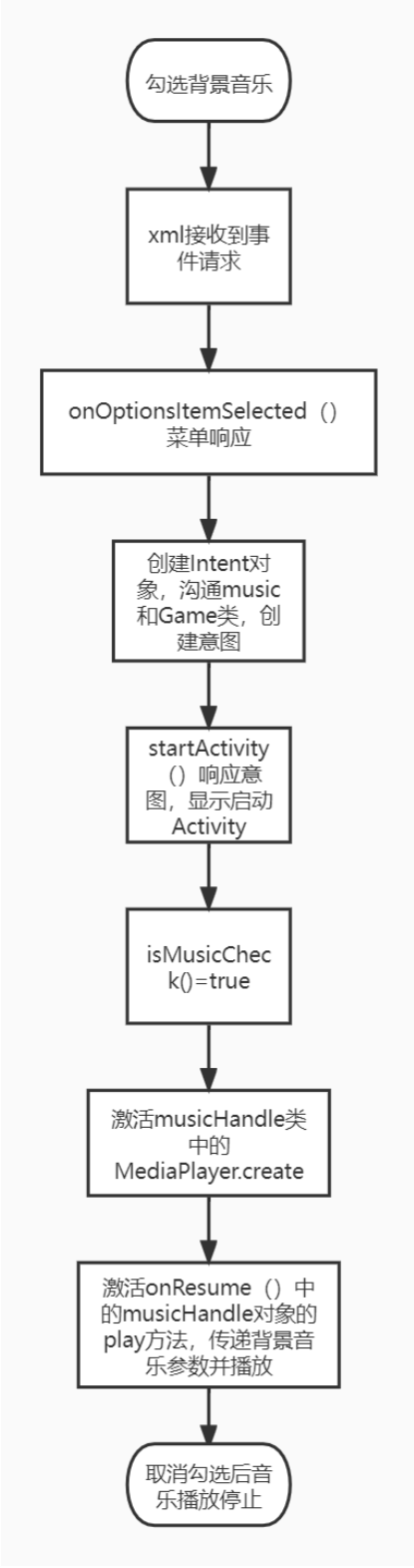


图 5　开启背景音乐流程图

2.6 XML 设置

```
<Buttonandroid:id="@+id/btn_continue"
android:layout_width="@dimen/btn_width"
android:layout_height="wrap_content"
android:text="@string/continue_text"
android:textSize="@dimen/btn_text_size"
android:textColor="@color/btn_text_color"
android:onClick="onClick"/>
```

对于 xml 中的 button 按钮，Id 是唯一的，在 java 中通过 id 调用此按钮，text 中引用了另一个 xml 中的变量，亦可以直接给其赋值为 string，onClick 为该按钮接收到事件后响应的触发函数。

三、 编码与测试

3.1 问题 1

```
public void nextGate()
{
    if(gate<MapList.getcount()-1)
    {
        gate++;
    }
    else
    {
        Toast.makeText(this.getContext(), "目前是最后一关,you ...
            win!",Toast.LENGTH_LONG).show();
        this.reinitmap();
        this.invalidate();
    }
}
```

在点击跳过本关和返回上一关的时候，页面没有反应，必须手动点一下才行，后来知道了需要调用一下 invalidate() 函数，以此来刷新窗口布局

3.2 问题 2

```
if(music.isMusicCheck(context))
{   mp=MediaPlayer.create(context, resource);
    mp.setLooping(true);
    mp.start();
}
```

对于背景音乐的设置，需要调用到 android 的 media 库里的 MediaPlayer 函数，以此来创建一个播放器开关。

四、 个人总结

Android 是我们生活中最为常见的一种嵌入式系统，很早之前我便对其开发充满着好奇与向往。本次课设中，一开始选择的开发工具是 eclipse，但是在配置开发环境时出现了一系列问题，最后选择了配置相对自动化的 Android studio。开发过程中，我学习了安卓的 MVC 开发框架，它将业务逻辑、数据、界面显示用分离的方法组织代码，将业务逻辑聚集到一个部件里面，在改进和个性化定制界面及用户交互的同时，不需要重新编写业务