《拍死那只蚊子》技术开发文档

目录

1. 整体架构 3
2. 主要部分关键技术实现 7
3. 完成时间进度情况
4. 本游戏创意与特色
5. 开发团队介绍

# 整体架构



1. **公共类**（com.public.Class包）：

首先，该包下有常量类（Constant类）和主控制类（GameActivity类）。

1. Constant类

该类声明了游戏中用到的所有常量，例如，屏幕尺寸，各关的逻辑地图，蚊子行走的步长等。这部分示例代码如：

//屏幕参数

**public** **static** **int** *SCREEN\_WIDTH*;//屏幕的宽度

**public** **static** **int** *SCREEN\_HEIGHT*;//屏幕的高度

使用如此的声明方式，保证在整个工程中都可以使用类似Constant.*SCREEN\_WIDTH*的方式引用到相应的常量。

另外Constant类中还包含两个方法：

1. initConst()方法，该方法在主控制类（GameActivity类）中调用，接受主控制类获取的屏幕的尺寸，然后计算出适应全屏的长缩放比和宽缩放比，以适应不同尺寸的设备。

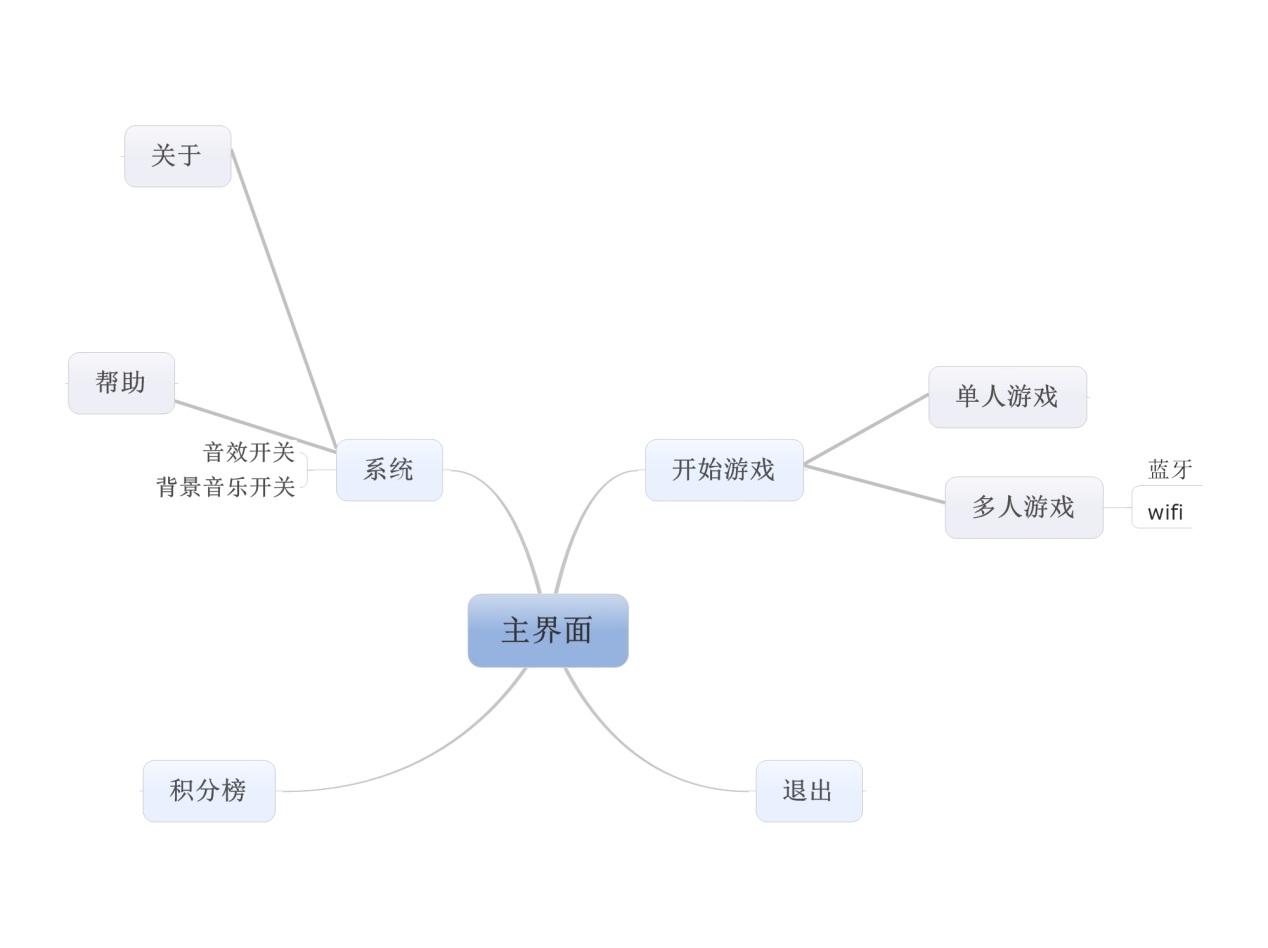
2. isPointInRect()方法,该方法会在用户界面类中调用，用来完成判断用户触控。

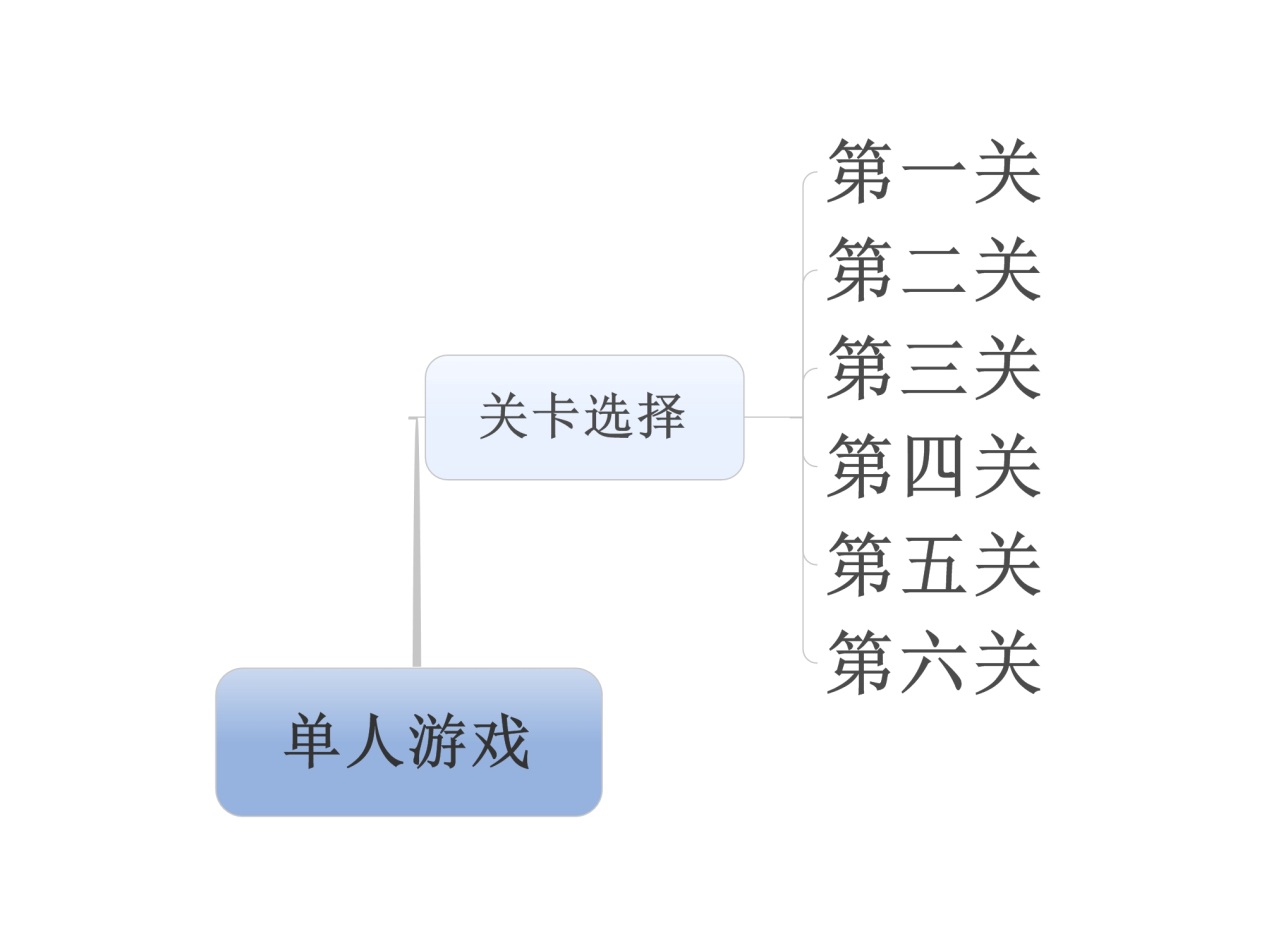
1. GameActivity类：

该类为本游戏的主控制类，由它来负责控制各个游戏界面的跳转，即本游戏采取由一个activity控制多个SurfaceView之间的跳转。另外，

本游戏中的wifi、蓝牙及蓝牙可见性的打开，以及用于判断蓝牙和wifi状态的广播接收器均在本类中，而且本类还包括积分榜的功能实现。（跳转实现，wifi、蓝牙部分和积分榜功能实现都将在<关键技术实现>部分详细介绍，这里就先一笔带过）。

1. **用户游戏界面类**（com.userView包）：





首先，所有的游戏界面类都是继承自android.view.SurfaceView类。

1. WelcomeView类：

欢迎界面，通过逐渐改变所显示的图片的透明度来实现LOGO动画显示。

1. MainMenuView类：

主菜单界面，用户在这里选择接下来将要跳转到的界面。带有动画效果。

1. StartGameView类：

开始游戏界面，用户在这里选择单人模式还是双人模式。

1. SystemView类：

系统设置界面，用户在这里可以打开帮助界面和关于界面，也可以选择关闭背景音乐和游戏音效。

1. ScoreView类：

积分榜界面，用户可以在这里查看排行前五的积分。

1. ExitView类：

退出界面，用户可以在这里选择是否退出（在主菜单界面直接按返回键也是可以退出的）

1. ChooseView类：

关卡选择界面，用户可以在这里选择他想要玩的关卡（当然，关卡需要一关一关地解锁）

1. CoopView类：

合作界面，用户可以在这里选择使用蓝牙还是wifi来和其他玩家进行联机（本游戏采用WiFi Direct,需要android 4.0以上支持）。

1. GameViewOne——GameViewSix类：

游戏界面，即本游戏的第一关到第六关。

1. WinView和FailView类：

分别为游戏的胜利和失败界面。

1. AboutView类：

关于界面，主要是游戏版本信息和团队信息。

1. HelpView类：

游戏帮助界面，这个界面将讲解游戏的规则。

1. **蚊子类**（mosquito包）：

该包中包含本游戏中蚊子的封装类——Mosquito类，该类封装了蚊子的各种属性，蚊子的寻路方法，以及蚊子的绘制方法。

1. **道具类**（tool包）：

本游戏中道具（需要用户一步一步通过过关来解锁）包括以下几种：

1. 蚊香（ExpelIncense类）:

该类封装了蚊香的各种属性例如，作用范围、持续时间等，同时封装了对逻辑地图的修改方法和蚊香类的绘制方法（包括蚊香一点点燃尽的动画效果）。

1. 驱蚊草（ExpelPlant类）：

该类封装了驱蚊草的各种属性例如，作用范围、持续时间等，同时封装了对逻辑地图的修改方法和驱蚊草类的绘制方法（包括驱蚊草一点点枯萎的动画效果）。

1. 青蛙（Frog类）：

该类封装了青蛙的各种属性例如，CD时间、作用范围、持续时间等，同时封装了对逻辑地图修改的方法和青蛙的绘制方法（包括青蛙跳来跳去的动画效果）。关于类似魔兽争霸中CD效果的实现将会在关键技术实现中介绍，这里不再赘述。

1. 灭蚊剂（killspray类）：

该类封装了灭蚊剂的各种属性例如，CD时间、作用范围、持续时间等，同时封装了对逻辑地图修改的方法和灭蚊剂的绘制方法（包括灭蚊剂烟雾逐渐消失的动画效果）。关于类似魔兽争霸中CD效果的实现将会在关键技术实现中介绍，这里不再赘述。

1. 灭蚊灯（KillLamp类）：

该类封装了灭蚊剂的各种属性例如，作用范围，同时封装了对逻辑地图修改的方法和灭蚊灯的绘制方法（灭蚊灯是本游戏中唯一个不会消失的道具）。

1. 蜘蛛网（SpiderWeb类）：

该类封装了蜘蛛网的各种属性例如，作用范围、持续时间等，同时封装了对逻辑地图修改的方法和蜘蛛网的绘制方法（包括蜘蛛网逐渐消失的动画效果）。

1. 电蚊拍（Swatter类）：

该类封装了电蚊拍的各种属性例如，作用范围等，同时封装了对逻辑地图修改的方法和电蚊拍的绘制方法（电蚊拍是一个一次性的道具）。

1. **辅助类**（utility包）：
2. 客户端线程类（ClientThread类）：

该类封装了蓝牙的socket连接。

1. 管理线程类（ManagerThread类）：

该类中包括通过已建立的socket，来建立IO流的方法，通过使用死循环来不断监听是否有数据写入。同时，提供一个向外发送数据的方法。

1. 服务端线程类（ServerThread类）：

该类封装了蓝牙serverSocket的建立。

1. IO传输对象类（Target类）：

本游戏中多人联机采用Socket模式，传输的为Target类的对象，该类封装了多人游戏中使用到的相关指令，当另一方收到这些相关指令后，会相应更新UI，同时指令还可以保证双人游戏的同步性。

1. 时间线程类（TimeThread——TimeThread6类）

该类主要负责相应关卡的计时，包括用户放置道具的持续时间、一些道具的CD时间和关卡本身倒计时所显示的时间。采用handler来与UI通信，效果相当不错。这部分将在关键代码实现中详细介绍，这里不再赘述。

1. 图片缩放帮助类（PicScaleHelper类）

该类封装了计算图片缩放比的方法。

1. 图片缩放结果类（ScreenScaleResult类）

该类封装了图片缩放比计算后的结果。

1. 图片缩放实用类（ScreenScaleUtil类）

该类提供了缩放图片的方法。

# 主要部分关键技术实现

1. **游戏界面的跳转实现简介**：

首先每个游戏界面类的成员变量中都含有主控制类（GameActivity类）的引用，通过在构造方法中初始化。代码示例如下：

**public** AboutView(GameActivity activity) {

**super**(activity);

**this**.activity=activity;

//获得焦点并设置为可触控

**this**.requestFocus();

**this**.setFocusableInTouchMode(**true**);

getHolder().addCallback(**this**);//注册回调接口

}

主控制类（GameActivity类）中含有一个处理跳转的handler，代码示例如下：

//定义用于处理页面跳转的handler

Handler mHandler=**new** Handler(){

**public** **void** handleMessage(Message msg){

**switch** (msg.what) {

//每个游戏界面在Consant类中都有唯一的常量标识

**case** Constant.*GOTO\_ABOUT\_VIEW*:

//调用去相应界面的方法

goAboutView();

**break**;

………………………………………

}

同时主控制类提供一个向该handler发送消息的方法，各个游戏界面中，通过对用户的触控进行判断，使用类似activity.sendMessage（Constant.用户选中的游戏界面）的语句进行跳转。

1. **语音操作实现简介**：

首先通过复写onKeyDown()方法，监听搜索键，即当用户按下搜索键时，调用手机自带的语音识别工具。示例代码如下：

//下面为监听用户按“返回键”和搜索键

@Override

**public** **boolean** onKeyDown(**int** keyCode, KeyEvent e) {

**switch** (keyCode) {

//监听返回键

………………………………..

//监听搜索键

**case** KeyEvent.*KEYCODE\_SEARCH*:

**try**{

Intent i=**new** Intent(RecognizerIntent.*ACTION\_RECOGNIZE\_SPEECH*);

i.putExtra(RecognizerIntent.*EXTRA\_LANGUAGE\_MODEL*, RecognizerIntent.*LANGUAGE\_MODEL\_FREE\_FORM*);

i.putExtra(RecognizerIntent.*EXTRA\_PROMPT*, "开始语音" );

startActivityForResult(i,1234);

}**catch**(Exception ee){

Toast.*makeText*(**this**, "找不到语音设备", **Toast.*LENGTH\_SHORT*).show();**

}

return ture;

}

return false;

}

然后通过复写onActivityResult()方法，进行相应的匹配。

代码示例如下：

**protected** **void** onActivityResult(**int** requestCode, **int** resultCode, Intent data) {

**……………………………………………..**

//捕获语音识别结果

**else** **if**(requestCode==1234){

**if**(resultCode==*RESULT\_OK*){

ArrayList<String> result=data.getStringArrayListExtra(RecognizerIntent.*EXTRA\_RESULTS*);

action=result.get(0)

//如果当前在主菜单界面，只能跳转到 start\_game/system/exit/score

//防止用户说法不一，这里进行了模糊查询处理

**if**(action.equals("开始游戏")||action.equals("游戏")){

goStartGameView();

}

**if**(currentView==Constant.*GOTO\_START\_GAME\_VIEW*){

**if**(action.equals("单人模式")||action.equals("单人")){

goChooseView();

}

………………………………………………………………………………….

}

}**else**{

Toast.*makeText*(**this**,"要使用语音功能，请先连接无线网络", Toast.*LENGTH\_LONG*).show();

}

}

}

1. **积分榜功能实现简介**：

主控制类（GameActivity类）提供一个在首次运行游戏时执行的方法，她用来创建本游戏的配合文件，代码示例如下：

//第一次启动游戏时创建配置文件

**public** **void** createConfig(){

Editor config=getSharedPreferences("config", 0).edit();

config.putString("game", "拍死那只蚊子");

config.putString("author", "山东大学android实验室11级");

config.putString("player1", 0+"");

config.putString("player2", 0+"");

config.putString("player3", 0+"");

config.putString("player4", 0+"");

config.putString("player5", 0+"");

config.putString("current", 0+"");

config.putString("achievement", 1+"");

config.commit();

}

主控制类提供用于记录当前积分的方法和将当前积分与排在前五名的积分进行比较并同时更新前五名积分的方法的方法，部分代码如下：

//在配置文件中缓存此时游戏积分的方法

**public** **void** trackGrade(**int** i){

Editor config=getSharedPreferences("config", 0).edit();

config.putString("current",i+"");

config.commit();

}

最后主控制类（GameActivity类）提供清零当前积分、获取当前积分、获取前五名积分的方法。用于支持积分的记录与积分榜功能的实现。由于代码比较类似且简单，这里不再赘述。

1. **蚊子移动实现**：

Mosquito对象的构造方法

**public** Mosquito(**int** level, **int** locX, **int** locY, GameView gameView) {

**this**.level = level;

**this**.locX = 8 \* locX;

**this**.locY = 8 \* locY;

// 初始血量、攻击力、积分关于蚊子等级的计算公式

hp = level \* 1 ;

dps = level \* 1 ;

score = level \* 50 + 50;

state = 1;

**this**.gameView = gameView;

setmap(gameViewOne.map);

}

其中要求输入蚊子的种类、其实位置、所出现的View三个关键参数。Setmap方法会从View中读取出逻辑数组，经过如下处理：

map = **new** **int**[8 \* inputmap.length][8 \* inputmap[0].length];

**for** (**int** i = 0; i < inputmap.length - 1; i++) {

**for** (**int** j = 0; j < inputmap[0].length - 1; j++) {

map[8 \* i][8 \* j] = inputmap[i][j];

**for** (**int** k = 1; k < 8; k++) {

map[8 \* i + k][8 \* j] = inputmap[i][j] \* inputmap[i + 1][j];

map[8 \* i][8 \* j + k] = inputmap[i][j] \* inputmap[i][j + 1];

}

}

}

这段代码把原20\*13的数组排列均扩充8倍以方便绘制蚊子的时候计算蚊子的位置。扩充时为了保证路径宽度不增加用乘法处理，并保证但前格与原数组一致。

同时会依据扩充地图产生一个记录地图stepmap：

stepMap = **new** **int**[map.length][map[0].length];

**for** (**int** i = 0; i < map.length; i++) {

**for** (**int** j = 0; j < map[0].length; j++) {

**if** (map[i][j] == 0)

stepMap[i][j] = 0;

**else**

stepMap[i][j] = 100;

}

}

stepMap[locY][locX]--;

可以通过的路径初始化为100，其后走过的地方就会在move方法中记录地图的当前位置-1：

**int**[] retVal = **new** **int**[4];// [0]:x轴位置

// [1]:y轴位置

// [2]:状态：1：正常；:-1：死亡；2：到达终点

// [3]:方向：0：右； 1：下； 2：左； 3：上

**int**[][] direct = { { 1, 0 }, { 0, 1 }, { -1, 0 }, { 0, -1 } };

**int** x = 0, y = 0;

**int** temp[][] = **new** **int**[4][3];

// read the value

**for** (**int** i = 0; i <= 3; i++) {

x = locX + direct[(d + i) % 4][0];

y = locY + direct[(d + i) % 4][1];

**if** (setState() == 2)

temp[i][0] = temp[i][1] = temp[i][2] = 0;

**else** {

temp[i][0] = stepMap[y][x];

temp[i][1] = direct[(d + i) % 4][0];

temp[i][2] = direct[(d + i) % 4][1];

}

}

// compare the minimize

**int** ahead[] = { 0, 0, 0 };

**for** (**int** i = 0; i <= 3; i++) {

**int** dx = temp[i][1];

**int** dy = temp[i][2];

**if**(temp[i][0]>ahead[0] && map[locY + dy][locX + dx]!= 0

&& state != 2 && state != -1) {

ahead = temp[i];

d = (d + i) % 4;

}

}

// move

x = locX + ahead[1];

y = locY + ahead[2];

locX = x;

locY = y;

stepMap[y][x]--;

retVal[0] = x;

retVal[1] = y;

retVal[2] = setState();

**int** dx=ahead[1],dy=ahead[2];

**if**(dx==1&&dy==0)

retVal[3]=0;

**if**(dx==0&&dy==1)

retVal[3]=1;

**if**(dx==-1&&dy==0)

retVal[3]=2;

**if**(dx==0&&dy==-1)

retVal[3]=3;

**return** retVal;

返回值中包含所走的下一步位置（X、Y）、方向、状态（由setState方法依据原始地图信息进行计算），以提供drawself方法绘制：

Bitmap[] move = **new** Bitmap[4];

exchangeXY();

**int**[] i = m.move();

**if** (isSpeed)

i = m.move();

front = i[3];

**switch** (front) {

**case** 0:

move = chooseBitmap(0);

**break**;

**case** 1:

move = chooseBitmap(1);

**break**;

**case** 2:

move = chooseBitmap(3);

**break**;

**case** 3:

move = chooseBitmap(2);

**break**;

}

**if** (delay % 4 == 0)

canvas.drawBitmap(move[0], Sx, Sy, paint);

**if** (delay % 4 == 1)

canvas.drawBitmap(move[1], Sx, Sy, paint);

**if** (delay % 4 == 2)

canvas.drawBitmap(move[2], Sx, Sy, paint);

**if** (delay % 4 == 3)

canvas.drawBitmap(move[3], Sx, Sy, paint);

canvas.drawRect(Sx, Sy + 5, Sx + 5 \* hp, Sy, paint);

delay++;

1. **关卡计时及计时器动画效果、道具持续时间及道具CD实现简介：**

当用户开始游戏，进入到游戏界面后，同时开启一个timeThread线程，该线程为计时线程。

1. **关卡计时实现**

首先，介绍关卡计时的实现，在成员变量中声明一个int值和boolean值（该boolean初始为true），然后在run（）方法中加入while（前面声明的boolean值） ，在while循环中调用Thread.sleep（一定的睡眠时间），每次将int值递增，知道int值达到某值之后，将boolean值设置成false，跳出循环。即该关卡时间已经到了。示例代码如下：

@Override

**public** **void** run(){

**while**(ToDate){

//第一关时间到，调用判断是否游戏胜利的方法

**if**(i >3600){

ToDate=**false**;

}

i++;

**try** {

Thread.*sleep*(50);

} **catch** (InterruptedException e) {

e.printStackTrace();

}

}

1. **计时器动画效果实现：**

计时器中，每一秒数字就要发生变化，所以在timeThread类中，需要持有gameview的引用，然后，在每一秒都要改变gameview中的timer这个成员变量的值，同时在gameview的绘制中，根据timer的改变，改变相应的绘制，即可实现计时器的功能。

绘制方面原理简单，但代码较为繁琐，这里不再赘述。

TimeThread类计量每一秒的代码如下：

@Override

**public** **void** run(){

**while**(ToDate){

//计时器，一秒种变化一次

**if**(time==20){

gameview.timer.subtractTime(1);

time=0;

}

time++;

try {

Thread.*sleep*(50);

} **catch** (InterruptedException e) {

e.printStackTrace();

}

}

1. **道具持续时间的实现**

为了实现道具的持续时间，保证道具在持续时间到了之后消失。我们在TimeThread类中写有一个handler，由于TimeThread线程的开启在GameView中完成，故GameView中持有TimeThread类的引用。

简单介绍原理，即在GameView中复写的onTouchEvent（）方法中，捕获到用户的触控之后，对其进行判断，判断出用户点击了哪种道具，在实例化道具类对象并将其添加到绘制列表的同时，向TimeThread类的handler发送消息。TimeThread收到消息后，判断出道具种类，将相应的boolean值设置为true.

示例代码如下：

**boolean** daojuFlag1=**false**;//标识驱蚊道具一是否开始计时

mhandler=**new** Handler(){

**public** **void** handleMessage(Message msg){

**switch**(msg.what){

//case1,case2道具生成

**case** 1:

daojuFlag1=**true**;

**break**;

…………………………………………………..

}

};

@Override

**public** **void** run(){

**while**(ToDate){

**try** {

Thread.*sleep*(50);

} **catch** (InterruptedException e) {

e.printStackTrace();

}

//第一个驱蚊道具

**if**(daojuFlag1){

ExpelPlant temp = gameview.expelPlantList.get(0);

**if**(temp.level==1)

steady1+=2;

**else**

steady1++;

//第二张图片

**if**(steady1==30){

temp.state = 1;

gameview.expelPlantList.set(0, temp);

}

//第三张图片

**if**(steady1==60){

temp.state = 2;

gameview.expelPlantList.set(0, temp);

}

//第四张图片

**if**(steady1==90){

temp.state = 3;

gameview.expelPlantList.set(0, temp);

}

//持续时间

**if**(steady1==120){

gameview.mhandler.sendEmptyMessage(1);

daojuFlag1=**false**;

}

}

//第二个驱蚊道具

………………………………………………………………………

1. **蓝牙联机实现原理简介：**

首先当用户点击多人游戏，选择蓝牙的时候，将调用检查蓝牙设备是否开启，同时开启蓝牙设备的方法。代码示例如下：

// 获取蓝牙设备，并判断设备是否支持蓝牙

**public** **void** CheckBluetooth() {

mBluetoothAdapter = BluetoothAdapter.*getDefaultAdapter*();

host = mBluetoothAdapter.getName();

**if** (mBluetoothAdapter == **null**) {

// 如果设备上不支持蓝牙

haveBluetooth = -1;

} **else** {

haveBluetooth = 1;

}

}

// 开启蓝牙设备

**public** **void** EnableBluetooth() {

**if** (!mBluetoothAdapter.isEnabled()) {

//向主控制类发送消息，调用开启蓝牙的代码。 activity.sendMessage(Constant.*EnableBluetooth*);

} **else** {

isEnable = 1;

}

}

当用户选择搜索时，首先取出，当前设备曾经配对过的设备，放在一个列表中，用于UI绘制。代码示例如下：

// 获取以前曾经连接过的设备列表

**public** ArrayList<BluetoothDevice> getPreviousList() {

Set<BluetoothDevice> pairedDevices = mBluetoothAdapter

.getBondedDevices();

// 如果存在绑定的设备

**if** (pairedDevices.size() > 0) {

PreDeviceList = **new** ArrayList<BluetoothDevice>();

**for** (BluetoothDevice device : pairedDevices) {

PreDeviceList.add(device);

}

**return** PreDeviceList;

} **else** {

**return** **null**;

}

}

当用户发现这个列表中没有他想要配对的设备时，用户可以点击列表下面的搜索按钮，这时，才开始真正的蓝牙设备搜索。这样设计的目的是，尽量避免蓝牙搜索，因为蓝牙搜索耗时多同时导致效率低下。蓝牙设备搜索的代码如下：

// 搜索当前可用设备

**public** **void** getCurrentList() {

mBluetoothAdapter.startDiscovery();

}

需要在主控制类中注册广播接收器来对搜索到的设备进行处理。代码示例如下：

**public** **class** Receiver **extends** BroadcastReceiver{

@Override

**public** **void** onReceive(Context context, Intent intent) {

String action=intent.getAction();

**if**(action.equals(BluetoothDevice.*ACTION\_FOUND*)){

BluetoothDevice device=intent.getParcelableExtra(BluetoothDevice.*EXTRA\_DEVICE*);

coopView.CurDeviceList.add(device);

Log.*e*("CUR", coopView.CurDeviceList.get(0).getName()+"");

Log.*e*("搜索", "新设备");

}

}

}

另一方面，如果用户点击建立房间，执行的操作是打开当前设备的可见性（一般设备蓝牙可见性是默认不打开的）。代码示例如下：

Intent discoverableIntent = **new**

Intent(BluetoothAdapter.*ACTION\_REQUEST\_DISCOVERABLE*);

discoverableIntent.putExtra(BluetoothAdapter.*EXTRA\_DISCOVERABLE\_DURATION*,300);

startActivityForResult(discoverableIntent,*requestCode2*);

下面需要复写**protected** **void** onActivityResult(**int** requestCode, **int** resultCode, Intent data) 方法，处理可见性打开的情况和可见性打开失败的情况。

1. **多人联机游戏的连接与数据交互原理：**

首先分为两个阶段：

1. 连接阶段
2. 数据交互阶段

在连接阶段分为客户端（点击搜索后，加入游戏的）与服务端（点击建立房间的），客户端代码为：

package utility;

import java.io.IOException;

import java.lang.reflect.InvocationTargetException;

import java.lang.reflect.Method;

import java.util.UUID;

import com.userView.CoopView;

import android.bluetooth.BluetoothDevice;

import android.bluetooth.BluetoothSocket;

import android.util.Log;

public class ClientThread extends Thread{

public BluetoothSocket mSocket;

private final BluetoothDevice mDevice;

public CoopView coopView;

boolean flag;

public ManagerThread managerThread;

public ClientThread(BluetoothDevice device,CoopView coopView){

this.coopView=coopView;

mDevice=device;

try{

Log.e(mDevice.getName(), "客户端");

mSocket=mDevice.createRfcommSocketToServiceRecord(UUID.fromString("1539F5CF-0C4D-F0A6-54C8-67FC277A4B5D"));

} catch (SecurityException e) {

// TODO Auto-generated catch block

e.printStackTrace();

} catch (IOException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

}

public void run(){

//取消搜索

coopView.mBluetoothAdapter.cancelDiscovery();

try {

Log.e("connect", "begin");

mSocket.connect();

////开启数据交互线程

managerThread=new ManagerThread(mSocket);

managerThread.start();

coopView.mHandler.sendEmptyMessage(4);

} catch (IllegalArgumentException e) {

// TODO Auto-generated catch block

Log.e("illegal", "s");

} catch (IOException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

}

public void setFlag(boolean flag) {

this.flag = flag;

}

}

服务端代码如下：

package utility;

import java.io.IOException;

import java.lang.reflect.Method;

import java.util.UUID;

import com.publicClass.Constant;

import com.userView.CoopView;

import android.bluetooth.BluetoothAdapter;

import android.bluetooth.BluetoothDevice;

import android.bluetooth.BluetoothServerSocket;

import android.bluetooth.BluetoothSocket;

import android.util.Log;

/\*\*

\*

\* @author 山东大学android实验室刘昭呈

\*

\*/

public class ServerThread extends Thread {

private final BluetoothServerSocket mServerSocket;

CoopView coopView;

boolean flag=true;

public BluetoothSocket socket = null;

public ManagerThread managerThread;

public ServerThread(CoopView coop){

coopView=coop;

BluetoothServerSocket tmp = null;

try{

tmp=coopView.mBluetoothAdapter.listenUsingRfcommWithServiceRecord(Constant.GAME\_NAME,UUID.fromString("1539F5CF-0C4D-F0A6-54C8-67FC277A4B5D"));

}catch(IOException e){

Log.e("serverSocket", "建立不成功");

}

mServerSocket=tmp;

}

public void run(){

try{

socket=mServerSocket.accept();

}catch(IOException e){

Log.e("socket", "连接不成功");

}

if(socket!=null){

//开启数据交互线程

managerThread=new ManagerThread(socket);

managerThread.start();

//发送信息

coopView.mHandler.sendEmptyMessage(3);

}

}

public void cancelSocket(){

}

public void setFlag(boolean flag) {

this.flag = flag;

}

}

数据交互阶段：

在以上的客户端代码和服务端代码中，连接建立后，会开启一个数据交互线程，由此进入第二阶段。

数据交互线程代码示例如下：

package utility;

import java.io.BufferedInputStream;

import java.io.ByteArrayInputStream;

import java.io.DataInputStream;

import java.io.IOException;

import java.io.InputStream;

import java.io.ObjectInputStream;

import java.io.ObjectOutputStream;

import java.io.OptionalDataException;

import java.io.OutputStream;

import com.userView.CoopView;

import com.userView.GameViewOne;

import android.bluetooth.BluetoothSocket;

import android.os.Message;

import android.util.Log;

/\*\*

\*

\* @author 山东大学android实验室刘昭呈

\*

\*/

public class ManagerThread extends Thread {

private BluetoothSocket mSocket;

private InputStream mInStream;

private OutputStream mOutStream;

private ObjectInputStream mOjectInStream;

private ObjectOutputStream mOjectOutStream;

public boolean flag=true;

public CoopView coopView;

public GameViewOne gameView;

public ManagerThread(BluetoothSocket socket){

this.mSocket=socket;

Log.e("sssd", mSocket.toString());

try{

mInStream=mSocket.getInputStream();

mOutStream=mSocket.getOutputStream();

mOjectInStream=new ObjectInputStream(mInStream);

mOjectOutStream=new ObjectOutputStream(mOutStream);

}catch(IOException e){

e.printStackTrace();

}

}

public void run(){

while(flag){

try {

Target receive=null;

receive=(Target)mOjectInStream.readObject();

//发送给相应的surfaceView的handler处理

if(coopView.currentFlag){

Message.obtain(coopView.mHandler, 9, receive);

}

if(!coopView.currentFlag){

//Message.obtain(gameView.mHandler, 0, receive);

}

} catch (OptionalDataException e) {

Log.e("Object\_run", "DataException");

} catch (ClassNotFoundException e) {

Log.e("Oject\_run", "classNotFound");

} catch (IOException e) {

Log.e("Object\_run", "IO");

}

}

}

public void write(Target t){

try {

mOjectOutStream.writeObject(t);

} catch (IOException e) {

Log.e("Oject\_write", "IO");

}

}

public void close(){

try {

if(mInStream!=null)

mInStream.close();

if(mOutStream!=null)

mOutStream.close();

if(mOjectInStream!=null)

mOjectInStream.close();

if(mOjectOutStream!=null)

mOjectOutStream.close();

if(mSocket!=null)

mSocket.close();

} catch (IOException e) {

Log.e("close","eorr");

}

}

}

# 完成时间进度情况

确定游戏主题以及关键部分 6.27-6.28

完成游戏框架，包含页面跳转控制 6.28-6.30

主菜单界面、开始游戏界面、系统设置界面、退出询问界面完成及关联6.30-7.5

蚊子类、道具类、游戏界面结构完成7.5-7.9

游戏绘制完成 7.9-7.11

单机游戏调试及初步优化 7.11-7.16

关卡设计、界面图片设计 7.16-7.17

关卡编制及平衡调试、多人联机界面、蓝牙交互编写 7.17-7.21

Wifi交互编写、全功能调试7.21-7.25

文档完成 7.25-7.27

# 本游戏创意与特色

1. 设计平衡性
2. 游戏多样性
3. 操作多样性
4. 多人娱乐性

# 开发团队介绍

|  |  |  |
| --- | --- | --- |
| **队员姓名** | **所属院系** | **担任职务** |
| 刘昭呈 | 2011级软件工程 | 代码设计 |
| 张昊 | 2011级计算机科学技术 | 代码设计 |
| 马月盺 | 2011级软件工程 | 代码设计 |
| 冯浩宇 | 2011级数媒 | 美工设计 |
| 朱绯云 | 2011级数媒 | 美工设计 |