

**本科生实验报告**

**实验课程 移动应用开发技术\_ \_\_\_\_\_\_\_\_**

**学院名称 网络安全学院\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**学生姓名 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**班内序号 2班15号 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**学生学号 201613160616\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**指导教师 刘恒\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**实验成绩\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**二〇 18 年 12 月**

目录

[实验一：初探连连看 4](#_Toc533971897)

[内容 4](#_Toc533971898)

[要求 4](#_Toc533971899)

[目的 4](#_Toc533971900)

[实现 4](#_Toc533971901)

[1.实现Fragment的切换 4](#_Toc533971902)

[2.全局变量（MyApplication.java） 6](#_Toc533971903)

[3.生成游戏地图 8](#_Toc533971904)

[4.菜单 9](#_Toc533971905)

[5.bfs用于判断点击的Image是否连通 10](#_Toc533971906)

[6.判断是否可消除 10](#_Toc533971907)

[7.重新排列 12](#_Toc533971908)

[运行结果 14](#_Toc533971909)

[实验二：多线程+广播 17](#_Toc533971910)

[内容 17](#_Toc533971911)

[要求 17](#_Toc533971912)

[目的 17](#_Toc533971913)

[实现 17](#_Toc533971914)

[1.多线程背景音乐 17](#_Toc533971915)

[2.多线程点击音效 18](#_Toc533971916)

[3.广播接收器 19](#_Toc533971917)

[4.注册广播接收器 19](#_Toc533971918)

[5.弹出对话框 20](#_Toc533971919)

[6.发送广播 20](#_Toc533971920)

[运行结果（假设两次点击消除是全部消除） 21](#_Toc533971921)

[实验三：持久化存储 23](#_Toc533971922)

[内容 23](#_Toc533971923)

[要求 23](#_Toc533971924)

[目的 23](#_Toc533971925)

[实现 23](#_Toc533971926)

[1. preference存储 23](#_Toc533971927)

[2.本地存储 24](#_Toc533971928)

[3. DatabaseHelper 24](#_Toc533971929)

[4.SQLite存储 25](#_Toc533971930)

[运行结果 27](#_Toc533971931)

# 实验一：初探连连看

## 内容

设计完成一个8\*9的连连看小游戏。

## 要求

1、具备25种不同的连连看动物（或其他图标）

2、当手指点击相同的动物，动物消失

3、当手指点击不同的动物，动物不消失

4、小图标出现的时候是随机出现的

5、可以设置不同的难度，不同难度级别对应不同的动物种类数量，难：25种，中等,15种，简单10种

6、遵守连连看规则，如果图片无阻塞的方能连接（可以不出现关联的线段）

7、具备胜负判断、打乱重拍的功能

8、在界面上点击右键菜单，选择打乱重拍功能

9、系统选项菜单中可以选择难度

10、在导航中可以切换显示游戏说明和游戏界面以及参数设置界面。

## 目的

1、掌握图片文件的加载和布局

2、掌握鼠标单击事件的发生和判断

3、熟悉利用fragment和导航

4、掌握选项菜单、弹出菜单（或上下文菜单）的设计开发。

## 实现

### 1.实现Fragment的切换

|  |
| --- |
| public void onClick(View v) {  FragmentTransaction fTransaction = fManager.beginTransaction();  myApplication.hideAllFragment(fTransaction);  switch (v.getId()){  case R.id.txt\_game:  setSelected();  txt\_game.setSelected(true);  if(myApplication.getFgGame() == null){  GameFragment fgGame = new GameFragment();  myApplication.setFgGame(fgGame);//修改全局变量  fTransaction.add(R.id.ly\_content,fgGame);  }else if(myApplication!=null && myApplication.getLevle()!=level){  fTransaction.remove(myApplication.getFgGame());  level = myApplication.getLevle();  GameFragment fgGame = new GameFragment();  myApplication.setFgGame(fgGame);  fTransaction.add(R.id.ly\_content,fgGame);  }  else{  fTransaction.show(myApplication.getFgGame());  }  if(sp.getBoolean("swhBgMusic",false)){  bmt = new BgMusicThread();  bmt.start();  }  break;  case R.id.txt\_rank:  setSelected();  txt\_rank.setSelected(true);  if(myApplication.getFgRank() == null){  RankFragment fgRank = new RankFragment();  myApplication.setFgRank(fgRank);  fTransaction.add(R.id.ly\_content,fgRank);  }else{  fTransaction.show(myApplication.getFgRank());  }  if(bgMusic!=null){  stopMusic();  }  break;  case R.id.txt\_info:  setSelected();  txt\_info.setSelected(true);  if(myApplication.getFgInfo() == null){  InfoFragment fgInfo = new InfoFragment();  myApplication.setFgInfo(fgInfo);  fTransaction.add(R.id.ly\_content,fgInfo);  }else{  fTransaction.show(myApplication.getFgInfo());  }  if(bgMusic!=null){  stopMusic();  }  break;  }  fTransaction.commit();  } |

### 2.全局变量（MyApplication.java）

|  |
| --- |
| package club.anlan.lab1;  import android.app.Application;  import android.support.v4.app.FragmentTransaction;  public class MyApplication extends Application {  private int levle;  private boolean isConunterPaused = false;  private InfoFragment fgInfo;  private RankFragment fgRank;  private GameFragment fgGame;  private ChangLevelFragment fgLevel;  private static MyApplication instance;  public MyApplication(){  }  @Override  public void onCreate() {  super.onCreate();  this.setLevle(10);  instance = this;  }  public static MyApplication getInstance() {  return instance;  }  public int getLevle() {  return levle;  }  public void setLevle(int levle) {  this.levle = levle;  }  public InfoFragment getFgInfo() {  return fgInfo;  }  public void setFgInfo(InfoFragment fgInfo) {  this.fgInfo = fgInfo;  }  public RankFragment getFgRank() {  return fgRank;  }  public void setFgRank(RankFragment fgRank) {  this.fgRank = fgRank;  }  public GameFragment getFgGame() {  return fgGame;  }  public void setFgGame(GameFragment fgGame) {  this.fgGame = fgGame;  }  public ChangLevelFragment getFgLevel() {  return fgLevel;  }  public void setFgLevel(ChangLevelFragment fgLevel) {  this.fgLevel = fgLevel;  }  public boolean isConunterPaused() {  return isConunterPaused;  }  public void setConunterPaused(boolean conunterPaused) {  isConunterPaused = conunterPaused;  }  //隐藏所有Fragment  public void hideAllFragment(FragmentTransaction fragmentTransaction){  if(this.getFgInfo() != null)  fragmentTransaction.hide(this.getFgInfo());  if(this.getFgLevel() != null)  fragmentTransaction.hide(this.getFgLevel());  if(this.getFgGame() != null)  fragmentTransaction.hide(this.getFgGame());  if(this.getFgRank() != null){  fragmentTransaction.hide(this.getFgRank());  }  }  } |

### 3.生成游戏地图

|  |
| --- |
| int curImageViewNum = imageViewIds.length;  while (curImageViewNum > 0) {  // 随机选择一张图片  level = 2;  MyApplication application = (MyApplication) MyApplication.getInstance();  if (application != null) {  level = application.getLevle();  }  int imageInArray = (int) (Math.random() \* level); //难度  // 随机选择两个ImageView并加载这同一张图片  for (int i = 0; i < 2; i++) {  int ImageViewInArray = (int) (Math.random() \* curImageViewNum);  ImageView view = getActivity().findViewById(curImageViews[ImageViewInArray]);  curImageViewNum -= 1;  curImageViews[ImageViewInArray] = curImageViews[curImageViewNum]; //已经有Image的ImageView 不再参与随机  view.setImageResource(imageIds[imageInArray]);  view.setTag(imageIds[imageInArray]); // 用于判断点击的是否是同一张图片  view.setOnClickListener(new View.OnClickListener() {  @Override  public void onClick(View v) {  GameFragment.this.onClick(v);  }  });  }  } |

### 4.菜单

|  |
| --- |
| btnMenu = getActivity().findViewById(R.id.btn\_menu);  btnMenu.setOnClickListener(new Button.OnClickListener() {  @Override  public void onClick(View v) {  final MyApplication myApplication = MyApplication.getInstance();  myApplication.setConunterPaused(true);  PopupMenu popup = new PopupMenu(getActivity(), v);  popup.inflate(R.menu.menu\_main);  //绑定菜单项的点击事件  popup.setOnMenuItemClickListener(new PopupMenu.OnMenuItemClickListener() {  @Override  public boolean onMenuItemClick(MenuItem item) {  switch (item.getItemId()) {  case R.id.rePlay:  rePlay();  break;  case R.id.changeLevel:  changeLevel();  break;  case R.id.reSet:  reSet();  break;  default:  break;  }  myApplication.setConunterPaused(false);  return false;  }  });  popup.show();  }  }); |

### 5.bfs用于判断点击的Image是否连通

|  |
| --- |
| protected boolean bfs(int x, int y, int targetX, int targetY){  initUsedNode();  int nx = 0;  int ny = 0;  Queue<Node> q = new LinkedList<Node>();  Node node = new Node(x,y,null);  q.add(node);  while (!q.isEmpty()){  Node curNode = q.poll();  for(int i=0;i<4;i++){  nx = curNode.x+moveXY[i][0];  ny = curNode.y+moveXY[i][1];  if (nx < 0 || nx > 8 || ny < 0 || ny > 7) {  continue;  } else if (curMap[nx][ny] == true && usedNode[nx][ny] == false) {  usedNode[nx][ny] = true;  node = new Node(nx,ny,curNode);  q.add(node);  }  if (nx==targetX && ny ==targetY){  return true;  }  }  }  return false;  } |

### 6.判断是否可消除

|  |
| --- |
| protected boolean judge(ImageView first, ImageView second){  boolean flag = false;  if(first.getId() == second.getId()){ // 两次点击的是一个地方  second = null;  }  else if(first.getTag().equals(second.getTag())){ // 两次点击的图片是一样的  int idFirst = first.getId();  int idSecond = second.getId();  int firstX=0,firstY=0;  int secondX=0,secondY=0;  // 得到两次点击的图片在虚拟地图的位置  for(int i=0;i<map.length;i++){  for(int j=0;j<map[i].length;j++){  if(idFirst==map[i][j]){  firstX = i;  firstY = j;  }  else if(idSecond==map[i][j]){  secondX = i;  secondY = j;  }  }  }  // 判断这两张图片是否是可连通的  if((firstX==0 && (secondX==0 || secondY==0 || secondY==7)) || (firstX==8 && (secondX==8 || secondY==0 || secondY==7)) || (firstY==0 && (secondY==0 || secondX==0 || secondX==8)) || (firstY==7 && (secondY==7 || secondX==0 || secondX==7))){ // 如果都在四周  flag = true;  }  else{  curMap[secondX][secondY] = true;  flag = bfs(firstX,firstY,secondX,secondY);  curMap[secondX][secondY] = false;  }  if(flag==true){  curMap[firstX][firstY] = true;  curMap[secondX][secondY] = true;  }  }  return flag;  } |

### 7.重新排列

|  |
| --- |
| protected void reSet() {  if (imgFirstClicked != null) {  imgFirstClicked.setBackgroundResource(0);  }  int imageNum = curImageNum; //得到此时未消除的图像数目  curImages = new int[imageNum];  // 获取所有的Image id  for (int i = 0; i < curMap.length; i++) {  for (int j = 0; j < curMap[i].length; j++) {  if (curMap[i][j] == false) {  ImageView view = getActivity().findViewById(map[i][j]);  imageNum--;  curImages[imageNum] = (int) view.getTag();  }  }  }  // 打乱顺序  for (int i = 0; i < curImages.length; i++) {  int swapId = (int) (Math.random() \* curImages.length);  int temp = curImages[swapId];  curImages[swapId] = curImages[i];  curImages[i] = temp;  }  // 将乱序后的id赋给ImageView  int allImageViewNum = 0;  for (int i = 0; i < map.length; i++) {  for (int j = 0; j < map[i].length; j++) {  if (curMap[i][j] == false) {  ImageView view = getActivity().findViewById(map[i][j]);  view.setImageResource(curImages[imageNum]);  view.setTag(curImages[imageNum]);  imageNum++;  }  }  }  } |

8.选择关卡

|  |
| --- |
| protected void changeLevel(){  MyApplication myApplication = MyApplication.getInstance();  FragmentTransaction fTransaction = getActivity().getSupportFragmentManager().beginTransaction();  myApplication.hideAllFragment(fTransaction);  ChangLevelFragment fgChangeLevel;  if(myApplication.getFgLevel()==null){  fgChangeLevel = new ChangLevelFragment();  fTransaction.add(R.id.ly\_content,fgChangeLevel);  myApplication.setFgLevel(fgChangeLevel);  }  else  fgChangeLevel = myApplication.getFgLevel();  fTransaction.show(fgChangeLevel);  fTransaction.commit();  } |

9.重新开始游戏

|  |
| --- |
| protected void rePlay(){  for(int i=0;i<map.length;i++){  for(int j=0;j<map[i].length;j++){  if(curMap[i][j]==true){  ImageView view = getActivity().findViewById(map[i][j]);  view.setBackgroundResource(0);  view.setVisibility(View.VISIBLE);  curMap[i][j]=false;  }  }  }  curImageNum = imageViewIds.length;  GameFragment.this.baseTimer = SystemClock.elapsedRealtime();  } |

## 运行结果

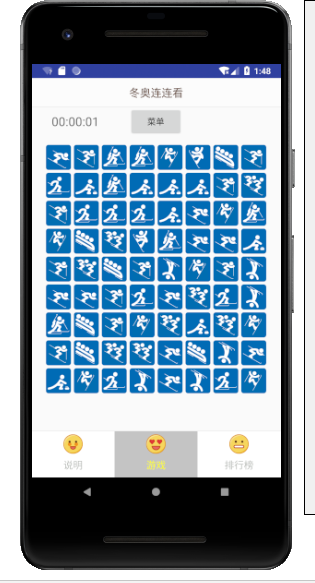


图1-1 游戏界面



图1-2 弹出菜单

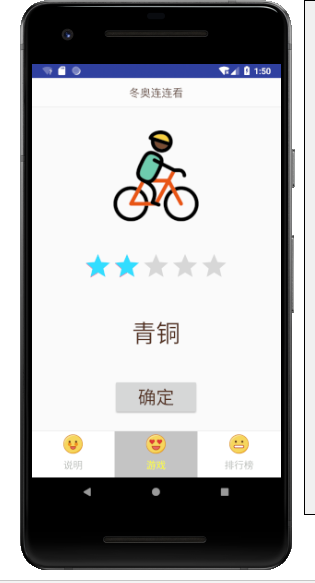


图1-3 选关界面

# 实验二：多线程+广播

## 内容

为连连看添加背景音乐和点击特效，发送广播生成排行榜。

## 要求

在第一次上机的基础上完成

1、利用多线程机制，为连连看程序添加背景音乐（圣诞快乐）和点击音效（点重、消除、点错）

2、当成功完成一局后，发出一个广播，广播内容包含你的个人信息，完成祝贺语，耗时时长，程序收到广播后，显示排名页面（现在排名页面中只显示你自己的个人信息，排名操作属于后续实验操作）

排名页面包括：序号、名字、耗时、上榜时间。

## 目的

1．熟悉多线程的应用。

2. 掌握广播技术。

## 实现

### 1.多线程背景音乐

|  |
| --- |
| public class BgMusicThread extends Thread {  private boolean isAlive;  @Override  public void run() {  if(bgMusic==null){  bgMusic = MediaPlayer.create(MainActivity.this, R.raw.bg\_music);  }  bgMusic.setLooping(true);  bgMusic.start();  }  }  public void stopMusic(){  bgMusic.setLooping(false);  bgMusic.stop();  bgMusic.reset();  bgMusic.release();  bgMusic = null;  } |

### 2.多线程点击音效

|  |
| --- |
| public class ClickMusicThread extends Thread {  private int musicType;  @Override  public void run() {  clickMusic.start();  }  public void setMusciType(int musicType) {  this.musicType = musicType;  switch (musicType){  case 0:  clickMusic = MediaPlayer.create(getActivity(),R.raw.clear);  break;  case 1:  clickMusic = MediaPlayer.create(getActivity(),R.raw.error);  break;  case 2:  clickMusic = MediaPlayer.create(getActivity(),R.raw.repeat);  break;  }  }  public int getMusciType() {  return musicType;  }  } |

### 3.广播接收器

|  |
| --- |
| public class MyReceiver extends BroadcastReceiver {  private MsgCallBack msgCallBack;  @Override  public void onReceive(Context context, Intent intent) {  msgCallBack.setRankInfo(intent.getStringExtra("usedTime"),intent.getStringExtra("userName"));  }  public interface MsgCallBack{  public void setRankInfo(String usedTime, String userName);  }  public void setMsgCallBack(MsgCallBack msgCallBack) {  this.msgCallBack = msgCallBack;  }  } |

### 4.注册广播接收器

|  |
| --- |
| FragmentTransaction fTransaction = fManager.beginTransaction();  if(myApplication.getFgRank()==null){  RankFragment fgRank = new RankFragment();  fTransaction.add(R.id.ly\_content,fgRank);  fTransaction.commit();  myApplication.setFgRank(fgRank);  }  IntentFilter intentFilter = new IntentFilter();  intentFilter.addAction("updateRank");  MyReceiver myReceiver = new MyReceiver();  registerReceiver(myReceiver,intentFilter);  myReceiver.setMsgCallBack(myApplication.getFgRank()); |

### 5.弹出对话框

|  |
| --- |
| protected void popAlert(){  MyApplication.getInstance().setConunterPaused(true);  String sTime = timerView.getText().toString();  int hour = Integer.parseInt(sTime.substring(0,2));  int minute = Integer.parseInt(sTime.substring(3,5));  int second = Integer.parseInt(sTime.substring(6,8));  int allTime = hour\*3600+minute\*60+second;  usedTime.setText(allTime+"s");  alert.show();  } |

### 6.发送广播

|  |
| --- |
| btnSendBroadCast.setOnClickListener(new View.OnClickListener() {  @Override  public void onClick(View v) {  //将名字存储到本地  FileOutputStream fileOutputStream = null;  try {  fileOutputStream = getActivity().openFileOutput(filename, Context.MODE\_PRIVATE);  fileOutputStream.write(userName.getText().toString().getBytes());  } catch (FileNotFoundException e) {  e.printStackTrace();  } catch (IOException e) {  e.printStackTrace();  }  try {  fileOutputStream.close();  } catch (IOException e) {  e.printStackTrace();  }  Intent intent = new Intent();  intent.setAction("updateRank");  intent.putExtra("usedTime",usedTime.getText().toString());  intent.putExtra("userName",userName.getText().toString());  getActivity().sendBroadcast(intent);  MyApplication.getInstance().setConunterPaused(false);  alert.dismiss();  }  }); |

## 运行结果（假设两次点击消除是全部消除）



图2-1 完成发送广播

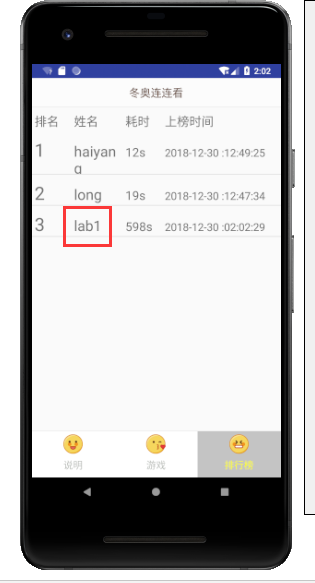


图2-2 收到广播并展示

# 实验三：持久化存储

## 内容

为游戏设置数据存储。

## 要求

1、采用preference 存储用户关于背景音乐的个人设置，允许用户设置默认音效和背景音乐的开关

2、采用本地存储存储用户的个人信息：用户名、头像

3、用SQLite存储排行榜。

## 目的

1. 掌握preference的基本操作。
2. 熟悉并应用本地存储。
3. 实现通过SQLiteOpenHelper管理sqlite数据库。

## 实现

### 1. preference存储

|  |
| --- |
| public void onActivityCreated(Bundle savedInstanceState) {  super.onActivityCreated(savedInstanceState);  sp = getActivity().getSharedPreferences("config", Context.MODE\_PRIVATE);  swhBgMusic = getActivity().findViewById(R.id.swh\_bg);  swhClickMusic = getActivity().findViewById(R.id.swh\_click);  swhBgMusic.setChecked(sp.getBoolean("swhBgMusic",false));  swhClickMusic.setChecked(sp.getBoolean("swhClickMusic",false));  swhBgMusic.setOnCheckedChangeListener(this);  swhClickMusic.setOnCheckedChangeListener(this);  }  @Override  public void onCheckedChanged(CompoundButton buttonView, boolean isChecked) {  switch (buttonView.getId()){  case R.id.swh\_bg:  saveData("swhBgMusic",isChecked);  break;  case R.id.swh\_click:  saveData("swhClickMusic",isChecked);  break;  }  }  private void saveData(String swhBtn, Boolean isChecked) {  SharedPreferences.Editor editor = sp.edit();//获取编辑器  editor.putBoolean(swhBtn, isChecked);//写入数据  editor.commit();//提交  } |

### 2.本地存储

|  |
| --- |
| FileOutputStream fileOutputStream = null;  try {  fileOutputStream = getActivity().openFileOutput(filename, Context.MODE\_PRIVATE);  fileOutputStream.write(userName.getText().toString().getBytes());  } catch (FileNotFoundException e) {  e.printStackTrace();  } catch (IOException e) {  e.printStackTrace();  }  try {  fileOutputStream.close();  } catch (IOException e) {  e.printStackTrace();  } |

### 3. DatabaseHelper

|  |
| --- |
| public class DatabaseHelper extends SQLiteOpenHelper {  static final String userTable = "rank";  static final String colNumber ="number";  static final String colUserName ="username";  static final String colUsedTime ="usedTime";  static final String colCreatedTime ="createdTime";  public DatabaseHelper(Context context, String name, SQLiteDatabase.CursorFactory factory, int version) {  super(context, name, factory, version);  }  @Override  public void onCreate(SQLiteDatabase db) {  db.execSQL("CREATE TABLE "+userTable+"("+colNumber+" INTEGER PRIMARY KEY,"  +colUserName+" Text,"  +colUsedTime+" Text,"  +colCreatedTime+" Text)");  }  @Override  public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {  }  } |

### 4.SQLite存储

|  |
| --- |
| public void onActivityCreated(Bundle savedInstanceState) {  super.onActivityCreated(savedInstanceState);  helper = new DatabaseHelper(getActivity(),"rankDB.db",null,1);  db=helper.getWritableDatabase();  list\_rank = getActivity().findViewById(R.id.list\_rank);  if(userRankList==null){  userRankList = new LinkedList<UserRank>();  Cursor cursor = db.rawQuery("select \* from rank",null);  cursor.moveToFirst();  int dbNumber = 0;  String dbUserName = "";  String dbUsedTime = "";  String dbCreatedTime = "";  while (!cursor.isAfterLast()){  dbNumber=cursor.getInt(0);  dbUserName=cursor.getString(1);  dbUsedTime=cursor.getString(2);  dbCreatedTime=cursor.getString(3);  userRankList.add(new UserRank(dbNumber,dbUserName,dbUsedTime,dbCreatedTime));  cursor.moveToNext();  }  }  myAdapter = new MyAdapter(userRankList, getContext());  list\_rank.setAdapter(myAdapter);  }  @Override  public void setRankInfo(String usedTime, String userName) {  //do  Date date = new Date();  SimpleDateFormat dateFormat= new SimpleDateFormat("yyyy-MM-dd :hh:mm:ss");  String currentTime = dateFormat.format(date);  list\_rank = getActivity().findViewById(R.id.list\_rank);  userRankList.add(new UserRank(userRankList.size()+1,userName,usedTime,currentTime));  Collections.sort(userRankList);  // 插入数据库  ContentValues cv = new ContentValues();  cv.put("number",userRankList.size());  cv.put("username",userName);  cv.put("usedTime",usedTime);  cv.put("createdTime",currentTime);  db.insert("rank",null,cv);  myAdapter = new MyAdapter(userRankList, getContext());  list\_rank.setAdapter(myAdapter);  } |

## 运行结果



图3-1说明与参数设置界面，

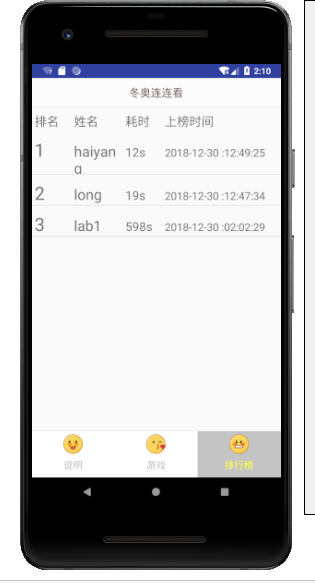


图3-2 排行榜界面

|  |  |
| --- | --- |
| **学生**  **实验**  **心得** | 这学期的移动开发课成学习的挺快乐的，做实验前期挺慢的，虽然难度不大，但是由于不熟悉，总是边做实验边复习知识点，边看老师的示例代码，当遇到自己实在没啥想法时，也会去求助网上的资源；做第二个试验时就好很多了，但是依然在多线程哪里卡了很久，因为想实现切换Fragment时自动实现对背景音乐的停止或者播放，而且由于做第一次实验时计划好，第二次实验修改了很多第一次实验的代码，这是很让人头疼的；第三次实验的时候就得心应手的很多了；最后谢谢老师的悉心教导，喜欢老师这种上课模式。  学生（签名）：龙海洋  2018年12月30日 |
| **指导**  **教师**  **评语** | 成绩评定：  指导教师（签名）：  年 月 日 |