Android 移动应用开发期中项目报告

——三国字典

组号：103

小组成员	姓名	学号	邮箱	qq 号
组长	赖贤城	15352155	754578682@qq.com	754578682
组员	黄洪彬	15352125	410542141@qq.com	410542141
组员	何伟杰	15352110	1219437376@qq.com	1219437376
组员	黄路	15352128	845758437@qq.com	845758437

目录

一、概述

我们的期中 proj 是一个联网的 app 应用，实现了三国人物词典应该具有的基础增删改查功能，同时也增加了一些拓展功能，包括以下：

1、使用云服务器 tomcat 和数据库 mysql

2、针对人物简介进行关键词全文检索

3、对人物点赞、并根据点赞数生成排行榜

4、增加 app 启动页动画

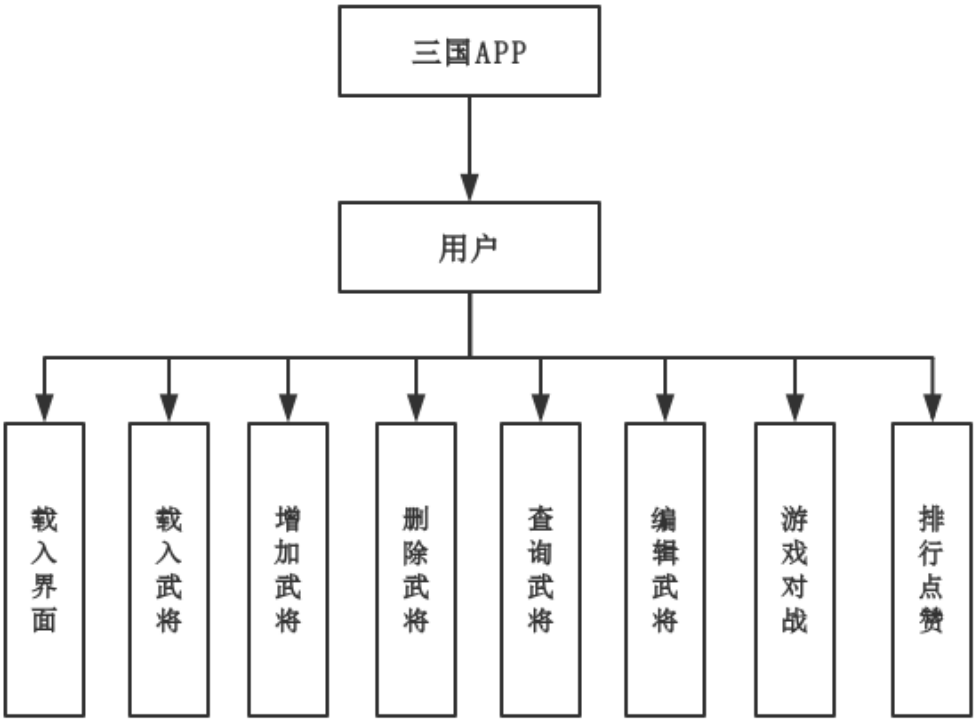
5、基于“田忌赛马”回合制的联网游戏对战功能

6、人物播放录音功能

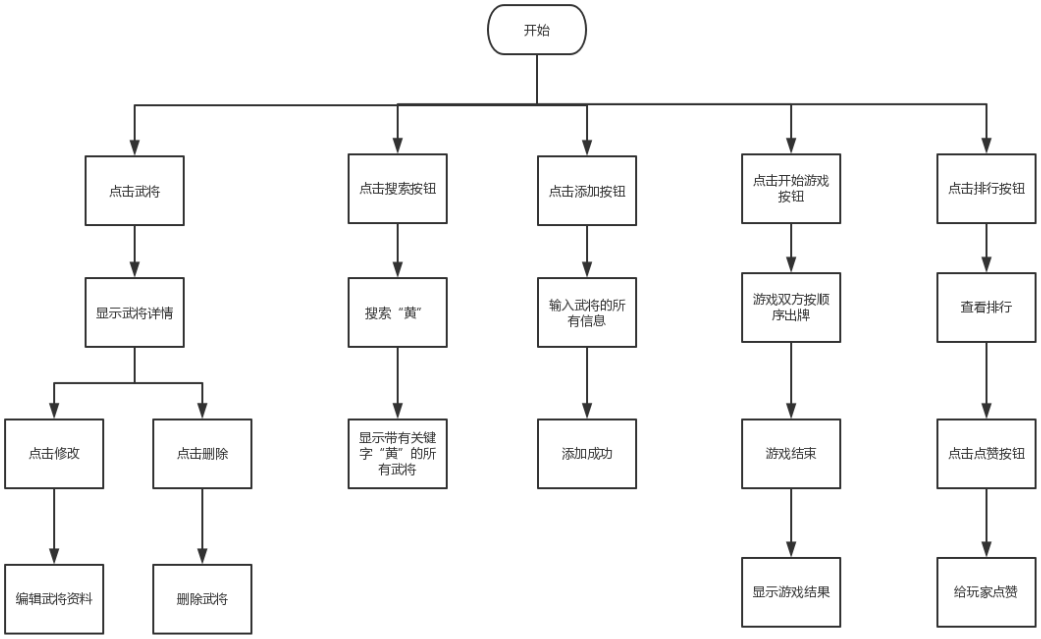
这些功能增强了我们 app 的娱乐性与实用性：全文检索可以使得用户通过关键字模糊定位到向搜索的对象，十分方便快捷；丰富的动画渲染，结合五十多个不同人物的配音，能给用户带来强烈的视听冲击；为喜欢的角色点赞，增强了使用 app 时的趣味性；最后，基于“田忌赛马”结合周易五行的回合制游戏，是我们的核心娱乐功能，玩家可以两两对战，用武将牌进行一场场策略的比拼。

二、项目说明

(一) 功能模块



(二) 逻辑流程



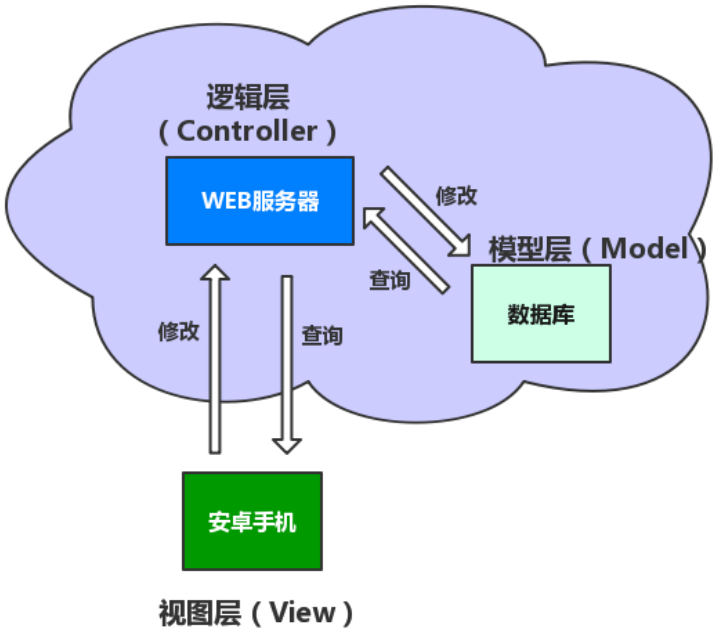
(三) 技术说明

一、 后端技术说明：

本次 app 应用用到了云服务器，后端语言为 java，web 服务器为 tomcat，数据库为 mysql，部署在阿里云上面。后端应用采用了 Spring MVC + Spring + JDBC 的三层架构，另外还使用了 websocket、redis、solr 搜索引擎的技术。

- **Web 应用的 MVC 三层架构**

作为一个 web 应用，app 端和服务器后端共同遵循 MVC 三层架构的规范，app 端作为 View 层，通过位于服务器的逻辑层 Controller，与数据库 Model 层进行交互。本次用到一些 JavaEE 的开发框架有 Spring MVC 和 Spring。SpringMVC 作为 MVC 框架，实现前端控制器、分派器、逻辑处理层以及数据库操作层的分离。作为常用的 web 框架，Spring MVC 作为 app 服务端也比较便于书写代码，在 controller 中书写 ajax 数据接口，直接将数据返回给视图层（安卓端）；Spring 作为依赖注入和控制反转（IOC）的框架，提供了统一的 Java Bean 对象管理，加快了代码开发和测试的流程，在事务管理方面，运用到了 Spring 的事务管理切面，实现了面向切面编程的事务管理。



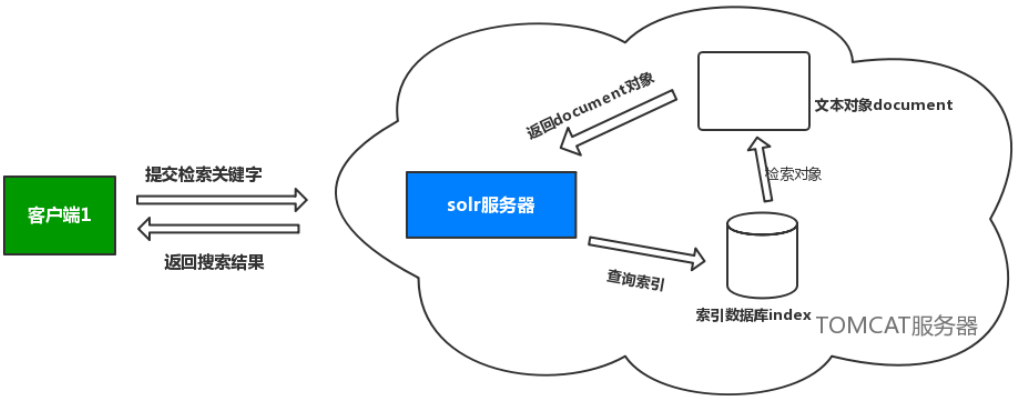
➢ WebSocket 技术

WebSocket protocol 是 HTML5 一种新的协议。它实现了浏览器与服务器全双工通信。与低效的轮询相比，WebSocket 让浏览器和服务器只需要要做一个握手的动作，然后，浏览器和服务器之间就形成了一条快速通道。两者之间就直接可以数据互相传送。Spring4 中新增了 websocket 相关的 api，我们使用该 websocket api 实现安卓客户端与服务端的相互通信，进为我们的游戏对战功能提供了通信基础。实现方式如下，服务器作为 server 端，手机端作为 client 端，两个 client 与 server 建立连接后，在一轮游戏中，client 发送游戏相关信息给 server，server 根据游戏规则进行判断后，将游戏结果、另外一个玩家的游戏信息主动推送给 client，即 server 作为消息中转站，实现了两个 client 端的实时通信。

- Solr 搜索引擎服务器

Solr 是 apache 一个基于 lucene 的全文检索引擎，可直接运行在 tomcat 服务器上。在本次我们的 app 应用中，个人信息除了存储在 mysql 关系型数据库外，还存储在 solr 的索引数据库中。通过 solr 的全文检索，可以迅速查询到对应用户。Solr 的基础为 lucene，一个基于 java 的全文检索工具包，这种搜索引擎的核心是索引，索引即是从非结构化数据中提取出的然后重新组织的信息，先建立索引，再对索引进行搜索的过程就叫全文检索。对于输入的一段文本信息，先进行分词，然后建立倒排索引。Lucene 中使用了 field 的概念，用于表达信息所在位置，在建索引中，该 field 信息记录在词典文件中，每个关键词都有一个 field 信息(因为每个关键字一定属于一个或多个 field)。假设要查询单词 “live”，lucene 先对词典二元查找、找到该词，通过指向频率文件的指针读出所有文章号，然后返回结果。词典通常非常小，因而，整个过程的时间是毫秒级的。如果使用普通的顺序匹配算法，不建索引，而是对所有文章的内容进行字符串匹配，这个过程将会相当缓慢。

本次实验中，我为每个对象 Person 定义了一个唯一的建 uuid，作为存储在 solr 中的主键，然后配置中文分词器，从 mysql 数据库导入数据，对个人简介信息 description 进行分词，建立索引，在 app 端界面中可以输入一些关键词来搜索该对象。



- Redis 数据库

本次我们的应用中有一个根据点赞数进行人物人气排名的功能，这个功能不是通过 mysql 排序得到的，而是用到了 redis 的 zset 数据结构。Redis 是一个 key-value 型的内存数据库，它支持存储的 value 类型包括 string(字符串)、list(链表)、set(集合)和 zset(有序集合)，作为一个高性能的内存数据库，是对 mysql 这一类存储在硬盘的关系型数据库很好的补充。这次期中项目我们使用 redis 作为后端的缓存，结合 spring 的 RedisTemplate 进行操作，减少了 mysql 被访问的压力。另外，zset 作为排序 set，可以便捷的用来更新我们的人物人气排行榜信息。

二、app 端技术说明：

1、详情页

详情页采用各种布局嵌套,详情活动 DetailActivity 的数据来自主活动 MainActivity，接收到相应的人物数据后进行显示。

- 编辑：

在初始状态时，各个 EditText 的功能被禁用，即不可对其进行编辑操作，其中简介是通过设置 EditText 和 TextView 的 Visibility 来控制的。

监听器监听到点击编辑按钮后，上方状态栏变为确认修改和放弃修改。各个 EditText 的功能恢复，这时记录下来他们的内容，当放弃修改时，使用 SetText 将内容恢复，当保存修改时，各个 EditText 和 TextView 的内容更新，且再次将他们的功能禁用。

- 修改头像：

当处于编辑状态时，点击头像可以调用访问相册和相机，对图片进行裁剪、显示处理，最后返回给数据库。

- 删除：

监听到删除按钮被点击，则设置 isDelete 为 true，直接结束活动。

- 点赞：

监听到点赞事件时，点赞图标变为被点赞图标，且告知数据库该武将被点赞。

- 结束事件：

DetailActivity 将 isDelete、isEdit 和更新后的 Person 类传回给 MainActivity，活动结束。

2、主页

技术实现：RecyclerView 自定义 Adapter，活动之间数据的传递，上拉加载更多，沉浸式状态栏。

(1) RecyclerView 是相对于 ListView 来说性能比较好的第三方控件，要使用该控件需要自定义 Adapter。

本次的自定义 Adapter 首先先新建了一个 RoleItemAdapter，然后在类里面定义内部类 ViewHolder 继承 RecyclerView.ViewHolder，该内部类用于存放每个 item 的控件这样就不用每次加载 item 的时候都要重新加载控件。然后必须重写 onCreateViewHolder()和 onBindViewHolder()这两个抽象方法。onCreateViewHolder()在 ViewHolder 实例被创建的时候调用，在这个方法中加载自定义的 item 的布局，然后绑定布局中的控件。

onBindViewHolder()方法是用于对 RecyclerView 子项的数据进行赋值的，会在每个子项被滚到屏幕中的时候执行，传入的参数有子项的 viewhoder、子项的位置 position。这里可以通过获取列表相应位置的项的数据对 viewholder 中的控件进行设置。

然后为了能让外界得到点击项的位置，我们需要在自定义的 Adapter 中新建一个点击事件的接口。当子项被点击时接口就会接收到相应的子项的 position，外界就可以接收到。

(2)要在 Activity 中传递数据，需要用到 Intent。

项目中多处使用活动间数据传递，比如点击列表中的子项进入详情页，详情页中进行修改之后返回修改值。为了能够通过主活动传递对象到详情页界面，需要把要传递的对象在定义的时候继承 Serializable 即可。Serializable 是序列化的意思，表示将一个对象转换成可存储或可传输的状态。这样就能够直接通过 Intent 传递对象了。

要接收来自启动的活动的返回值，需要使用 startActivityForResult()方法。然后在要

传回数据的活动中使用 setResult()方法返回数据，主活动则通过重写 onActivityResult()方法在其中对返回的数据进行操作。这样就实现了活动间的数据传递。

(3)上拉加载

项目中通过上拉加载可加载更多的人物列表项。要实现这个功能通过原生的控件是不能够实现的，必须自己继承 SwipeRefreshLayout 重写里面的方法。不过由于时间关系，本次项目直接使用了第三方开源库 SmartRefreshLayout 来实现上拉刷新。由于第三方开源库封装得很好，我们使用的时候只需要增加一该控件，然后通过 setRefreshFooter()方法设置样式，再通过 setOnLoadmoreListener()方法监听上拉事件以刷新列表。

(4)沉浸式状态栏

一般我们的 APP 的进入后状态栏是我们设置的 Theme 的 colorPrimaryDark 的颜色，而无论设成什么颜色和主界面的背景图都不是很搭。所以就使用了沉浸式状态栏，把主界面的背景图延伸到状态栏。首先需要把状态栏的颜色设为透明，而这只状态栏为透明只能在 Android5.0(API 21)之后才能操作，所以需要新建一个 values-v21 的目录，里面自定义 style HomeTheme 设置状态栏颜色。然后使用 CoordinateLayout 包裹在需要背景图延伸的 Activity 的布局的最外层，然后给 CoordinateLayout 和里面的布局添加属性 android:fitsSystemWindows="true"，最后把该 Activity 在注册的时候添加属性 android:theme="@style/HomeTheme"就大功告成了。

3、启动动画和游戏实现

启动页动画

功能概述：像用毛笔在涂抹一样，将背景图一行一行的慢慢涂抹出来。涂抹过程中的截图和涂抹结束后的截图见下图

涂抹过程中的截图

涂抹结束的截图

实现方法概述：

1.自定义可涂抹的 view，涂抹的意思是图片一开始被遮挡，通过涂抹使得图片显示出来。

2.使用数值动画（ValueAnimator）控制每次涂抹的点，用路径（Path）将涂抹的点保存起来，将上一步自定义的 view 沿着路径涂抹出来

下面也分为这两点讲解实现过程

A. 实现自定义可涂抹的 view

自定义可涂抹 view 的实现思路：

1. 重写 onDraw 方法，绘制图片，然后在图片之上再绘制一层纯白色作为蒙版
2. 然后再用一个画笔作为“橡皮擦”在纯白色蒙版上进行对蒙版的擦除（使之变透明，就显示出下方的图片了）
3. 将画笔变成“橡皮擦”的方法是设置画笔的图形混合方式，将其设置为 PorterDuff.Mode.XOR 模式，这样画笔画的地方与底层重叠部分就能变为透明

具体实现过程：

1. 在自定义 view 的构造器方法中

①将图片资源解码为 Bitmap，对应图中的第一个红框

②初始化擦除蒙版的画笔，主要是设置 PorterDuff.Mode，对应图中的第二个红框

```
public FadeInView(Context context, AttributeSet attrs) {
    super(context, attrs);
    mBitmapPaint = new Paint();//Bitmap的画笔
    //背景图
    mBitmapBackground = BitmapFactory.decodeResource(getResources(), R.mipmap.sanguo_launch);
    mErasurePaint = new Paint();
    //抗锯齿
    mErasurePaint.setAntiAlias(true);
    mErasurePaint.setColor(Color.WHITE);
    //画笔宽度
    mErasurePaint.setStrokeWidth(350);
    //设置图形混合方式，这里使用PorterDuff.Mode.XOR模式，与底层重叠部分设为透明
    PorterDuffXfermode mode = new PorterDuffXfermode(PorterDuff.Mode.XOR);
    mErasurePaint.setXfermode(mode);
    mErasurePaint.setStyle(Paint.Style.STROKE);
    mErasurePaint.setStrokeCap(Paint.Cap.ROUND);
    mErasurePaint.setStrokeJoin(Paint.Join.ROUND);

    //绘制蒙版的画笔
    mPaintRect = new Paint();
    mPaintRect.setAntiAlias(true);
    mPaintRect.setColor(Color.WHITE);

    //记录擦除点的路径
    mPath = new Path();
}
```

2. 重写 onDraw 方法，

①画出图片，这里使用的变换矩阵 matrix 将在下一步讲解（第一个红框）

②画出图片之上的蒙版（第二个红框）

③在蒙版对应的 bitmap 上面绘制路径，使得绘制的地方变成透明（第三个红框）

```
@Override
protected void onDraw(Canvas canvas) {
    super.onDraw(canvas);
    //将bitmapBackground设置该View画布的背景
    canvas.drawBitmap(mBitmapBackground,matrix, paint: null);

    //然后画布添加背景的基础上绘制蒙版
    canvas.drawBitmap(maskingBitmap, left: 0, top: 0, mBitmapPaint);
    bitmapCanvas.drawRect( left: 0, top: 0, width, height, mPaintRect);

    //在蒙版对应的bitmap上绘制路径，绘制的地方会变成透明
    bitmapCanvas.drawPath(mPath, mErasurePaint);
}
```

3. 重写 onMeasure，获取测量得到的高度和宽度（第一个红框），获得的高度宽度

用于设定蒙版的宽高（第二个红框）以及图片的缩放，就是上一步的变换矩阵（第三个红框）

```
@Override
protected void onMeasure(int widthMeasureSpec, int heightMeasureSpec) {
    super.onMeasure(widthMeasureSpec, heightMeasureSpec);

    width = getDefaultSize(getSuggestedMinimumWidth(), widthMeasureSpec);
    height = getDefaultSize(getSuggestedMinimumHeight(), heightMeasureSpec);
    setMeasuredDimension(width, height);

    //创建一个Bitmap作为蒙版
    maskingBitmap = Bitmap.createBitmap(width, height, Bitmap.Config.ARGB_8888);
    bitmapCanvas = new Canvas(maskingBitmap);//该画布为bitmap

    matrix.reset();
    //变换矩阵，使得图片大小与屏幕匹配
    matrix.postScale( sx: (float)width/mBitmapBackground.getWidth(),
            sy: (float)height/mBitmapBackground.getHeight());
}
```

4. 最后，留出一个接口供外部调用以进行绘制（其实应该说是擦除）。这个函数将多次的外部输入点保存成一个路径，供上述第 2 步中在 onDraw 函数里面沿着这个路径进行擦除

```
public void setDrawPosition(float x, float y){
    //将移动的轨迹画成直线，很多个点就构成路径
    if (!isFirst)
        mPath.lineTo(x, y);
    isFirst = false;
    mPath.moveTo(x, y);
    invalidate();//更新画面
}
```

B. 使用数值动画（ValueAnimator）逐行擦除蒙版

实现思路：

1. 确定逐行擦除的行间距，用行间距计算出总共要擦除的行数（记为 R）
2. 获得 view 的宽度 w，R*w（行数*每行宽度）就是绘制过程中要经过的像素数量
3. 让数值动画从 0 一直增加到 R*w，在数值更新的时候（onAnimationUpdate）就可以判定当前对应第几行的第几个像素点
4. 根据行数奇偶性进行绘制方向的调整，这是为了形成 S 型的绘制效果。然后调用自定义的 view 的 setDrawPosition 函数执行当前点的擦除
5. 动画结束之后开启 MainActivity

具体实现过程：

1. 设定行间距为 200px（第一个红框），这个设定与自定义 view 中的画笔宽度有关系，用行间距计算出总共要擦除的行数（第二个红框），记为 R。设置数值动画的变化范围与用时（第三个红框）

```
//每一个绘制行之间的距离（px）
final int GAP = 200;

private void startAnim(final FadeInView fadeInView) {
    final int mWidth = fadeInView.getWidth();
    int mHeight = fadeInView.getHeight();

    //计算总共要绘制的行数
    final int lineNum = mHeight/GAP;

    //用数值动画对绘制点进行控制
    anim = ValueAnimator.ofFloat(0f, lineNum*mWidth);
    anim.setDuration(2000);
```

2. 添加数值更新监听器，在 onAnimationUpdate 回调函数中计算当前要绘制的点，计算过程根据当前行的奇偶性进行绘制方向的调整

```
//数值更新监听器，控制绘制的点
anim.addUpdateListener(new ValueAnimator.AnimatorUpdateListener() {
    //onAnimationUpdate回调的次数
    int cnt = 0;
    @Override
    public void onAnimationUpdate(ValueAnimator animation) {
        float currentValue = (float) animation.getAnimatedValue();
        cnt++;
        //隔两次才绘制一次，这样减少了绘制的压力，而且画笔宽度足够大不用担心有地方绘制不到
        if (!(cnt % 3 == 0)){
            return;
        }
        //计算当前行数
        if (currentValue > curLine*mWidth)
            curLine++;
        //控制x轴的折返（从左往右，然后变成从右往左，然后左往右……）
        float x = curLine % 2 == 1 ? currentValue % mWidth : mWidth - (currentValue % mWidth);
        float y = curLine==1 ? 100 : curLine*GAP - 100;
        fadeInView.setDrawPosition(x, y);
    }
});
```

3. 添加动画结束监听器，当动画结束之后执行开启 MainActivity 的动作

```
//数值动画结束的监听器，动画结束开启MainActivity
anim.addListener(new AnimatorListenerAdapter() {
    @Override
    public void onAnimationEnd(Animator animation) {
        super.onAnimationEnd(animation);
        openMainActivity();
    }
});
//匀速动画
anim.setInterpolator(new LinearInterpolator());
anim.start();
```

至此，启动页动画完成

4. 联机对战游戏功能

功能概述：当两个客户端同时连接上游戏服务器的时候两者进入游戏状态，服务器给双方随机分配 5 个武将，对战双方都持有武力值 1~5 的武将各一张，除了武力值，每个武将都有他的五行属性——金木水火土，服务器根据属性判定输赢，将其结果发送给客户端，客户端使用动画和播放音频的形式将结果展示出来。

游戏未开始以及游戏过程中的截图如下

等待其他玩家加入的状态

游戏过程中的状态

实现方法概述：

1. 对人物卡牌的布局设计，自定义 View，作为游戏中的人物卡牌的 View
2. 监听“左上角”问号点击事件，使用 AlertDialog 展示规则
3. 使用对象动画旋转“阴阳鱼”，作为等待时的动画
4. 使用数值动画一直变更擂台中间的武器，作为对战时的动画
5. 使用对象动画改变透明度，实现结果（“胜”，“负”，“平”）的淡入淡出，以及计分
6. 使用 MediaPlayer 播放对战结果呈现时我方武将的声音音效（每个人物分别有“输”，“赢”两种声音），以及对战当中的音效播放

下面根据以上 6 点分点叙述具体实现

A 卡牌布局与自定义卡牌 View

1. 使用线性布局的 weight 方式按比例分配头像、姓名、武力、五行的高度，线性布局外层嵌套一个 CardView，用以实现卡牌的圆角和阴影（代码略）
2. 继承 FrameLayout，在构造器中将上一步做好的布局 inflate 进来。获得卡牌中对应 view 的引用

```
public VersusRoleView(Context context) { this(context, attrs: null); }

public VersusRoleView(Context context, @Nullable AttributeSet attrs) {
    super(context, attrs);
    mContext = context;
    LayoutInflater.from(context).inflate(R.layout.role_in_versus, root: this);
    tvName= findViewById(R.id.role_name);
    tvFiled= findViewById(R.id.role_filed);
    tvAbility= findViewById(R.id.role_ability);
    ivAvatar = findViewById(R.id.iv_avatar);
}
```

3. 留出一个函数以供外部进行设置卡牌内容的操作，根据传入的 Person 的内容更新卡牌的显示内容。其中头像用著名的开源库 glide 进行加载（不再展开）

```
public void setPerson(Person person) {
    this.person = person;
    this.tvName.setText(person.getName());
    this.tvFiled.setText("五行:"+fileds[person.getPerson_field()-1]);
    this.tvAbility.setText("武力: "+person.getAbility());
    Glide.with(mContext).load(person.getHead_url()).into(ivAvatar);
}
```

B 监听“左上角”问号点击事件，使用 AlertDialog 展示规则

```
    ivHelper = findViewById(R.id.helper);
    ivHelper.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) { showHelpDialog(); }
    });
}

private void showHelpDialog() {
    final AlertDialog.Builder builder = new AlertDialog.Builder( context: GameActivity.this);
    builder.setTitle("游戏规则");
    builder.setMessage(R.string.rules);
    builder.setPositiveButton( text: "确定", listener: null);
    builder.show();
}
```

C 使用对象动画旋转“阴阳鱼”，作为等待时的动画

在 onAttachedToWindow 函数中设置对象动画的播放次数以及循环模式，效果就是“阴阳鱼”一直旋转，直到代码显式的将其 stop

```
@Override
public void onAttachedToWindow() {
    super.onAttachedToWindow();

    //等待图标的旋转效果
    waitAnimator = ObjectAnimator.ofFloat(ivWait, propertyName: "rotation", ...values: 0f, 360f);
    waitAnimator.setDuration(1000);
    waitAnimator.setInterpolator(new LinearInterpolator());
    waitAnimator.setRepeatCount(ValueAnimator.INFINITE);
    waitAnimator.setRepeatMode(ValueAnimator.RESTART);
    waitAnimator.start();
}
```

D 使用数值动画一直变更擂台中间的武器，作为对战时的动画

当接收到服务器发来的对手的卡牌的时候，用数值动画进行控制，隔一段时间就换一次舞台中间的武器图，造成对战的动画效果

```
ValueAnimator valueAnimator = ValueAnimator.ofInt(0,versusTime);
valueAnimator.setDuration(versusTime);
valueAnimator.addUpdateListener(new ValueAnimator.AnimatorUpdateListener() {
    int cnt = 0;
    int imageCnt = 0;
    @Override
    public void onAnimationUpdate(ValueAnimator animation) {
        cnt++;
        if (cnt % 10 != 0)
            return;
        ivAnimation.setImageResource(versusIds[imageCnt % 4]);
        imageCnt++;
    }
});
```

E 对战结束后，实现结果（“胜”，“负”，“平”）的淡入淡出展示效果，以及计分

1. 首先根据判定结果从“胜”，“负”，“平”三张图片中选择相应的图片作为下一步的动画展示

```
switch (res) {
    case Win:
        resId = R.drawable.win;
        rawId = resources.getIdentifier( name: getPackageName()+":raw/"+name_pinyin+"_win",
                defType: null,  defPackage: null);
        increaseScore(playerEnum.Myself);
        break;
    case Lose:
        resId = R.drawable.lose;
        rawId = resources.getIdentifier( name: getPackageName()+":raw/"+name_pinyin+"_lose",
                defType: null,  defPackage: null);
        increaseScore(playerEnum.Opponent);
        break;
    case Tied:
        rawId = R.raw.tie;
        resId = R.drawable.tied;
        break;
}
```

选择图片

进行计分

2. 将上一步的图片用对象动画实现淡入淡出效果

```
//动画展示结果 缓慢出现然后缓慢消失
ivResult.setVisibility(View.VISIBLE);
ivResult.setImageResource(resId);
ObjectAnimator animator = ObjectAnimator.ofFloat(ivResult,  propertyName: "alpha",  ...values: 0f, 1f, 0f);
animator.setDuration(2000);
animator.setInterpolator(new LinearInterpolator());
```

3. 上述动画结束之后判定是否整局已经结束（代码略）

F 使用 MediaPlayer 播放对战结果呈现时我方武将的声音音效（每个人物分别有“输”，“赢”两种声音），以及对战当中的音效播放

1. 对战当中播放攻击音效（此处简单，代码略）
2. 展示对战结果的时候根据我方武将的名字以及胜负情况选择对应的音频进行播放（例如曹操赢的时候音频是“吾好梦中杀人”）。 音频名字设置成与人物名字一一对应，因此这里使用 getIdentifier 函数根据名字获得资源 id，然后将其播放出来

```
Person myPerson = player1.getPerson();
Resources resources = getResources();
String name_pinyin = Name2Pinyin.getPinyin(myPerson.getName());

switch (res) {
    case Win:
        resId = R.drawable.win;
        rawId = resources.getIdentifier( name: getPackageName()+":raw/"+name_pinyin+"_win",
                defType: null,  defPackage: null);
        increaseScore(playerEnum.Myself);
        break;
    case Lose:
        resId = R.drawable.lose;
        rawId = resources.getIdentifier( name: getPackageName()+":raw/"+name_pinyin+"_lose",
                defType: null,  defPackage: null);
        increaseScore(playerEnum.Opponent);
        break;
    case Tied:
        rawId = R.raw.tie;
        resId = R.drawable.tied;
        break;
}

//读取成功
if (rawId > 0){
    mp = MediaPlayer.create( context: this, rawId);
    mp.start();
}
```

根据结果选择音频

4、联网接口：okhttp 和 websocket

➢ Okhttp 实现 http 请求

以一条从服务器获取一段人物信息的请求为例。首先，传入参数包含有通知主线程更新 ui 的 handler，向服务器指明需要第几页的 page，以及搜索关键字 keyword。首先创建一个

OkhttpClient 对象，由于这个请求是 post 请求，所以参数不能直接放在请求 url 后面，而是放到请求实体中，这里用 okhttp 一个请求的 builder 进行构造（建造者模式）。

```
/*分段加载的请求，page代表请求第几页（初始化的时候填0），keyword为搜索的关键字，非搜索的时候keyword填null*/
/*第一次加载page填0，第二次填1，第三次填2。。*/
/*返回List<Person>，如果size=0，说明接下来没有了*/
public static void getTwentyPerson(final Handler handler, final Integer page, final String keyword){
    // 使用okhttp
    (Thread) run() -> {
            OkHttpClient mokHttpClient = new OkHttpClient();
            FormBody.Builder builder = new FormBody.Builder();
            builder.add( name: "page",page.toString());
            if (keyword != null && (!keyword.equals(""))) {
                System.out.println("使用关键字搜索");
                builder.add( name: "keyword",keyword);
            }
            //设置参数
            Request request = new Request.Builder().post(builder.build()).url(Const.IP + "getTwentyPerson.action").build();
```

Okhttp 发出请求后，然后执行 newCall 回调，onFailure 里执行请求异常的方法，这里直接打印异常;onResponse 方法里执行请求成功的方法,这里将服务器返回的列表通过 JSON 包装成 List<Person>对象，暂存在 message 对象里，用 sendMessage 方法传到主线程中。最后，主线程的 handleMessage 方法里将该数据设置到 adapter 的列表对象里。

```
    //设置参数
    Request request = new Request.Builder().post(builder.build()).url(Const.IP + "getTwentyPerson.action").build();
    mokHttpClient.newCall(request).enqueue(new Callback() {
        @Override
        public void onFailure(Call call, IOException e) { e.printStackTrace(); }
        @Override
        public void onResponse(Call call, Response response) throws IOException {
            String ans_str = response.body().string();
            //System.out.println(ans_str);
            JSONArray jsonArray = JSON.parseArray(ans_str);
            List<Person> list = jsonArray.toJavaList(Person.class);
            Message msg = new Message();
            msg.what = 0x1;
            msg.obj = list;
            handler.sendMessage(msg);
        }
    });
```

➢ Websocket 与服务器实现全双工通信

用到了 Java-websocket 包，它能够模拟浏览器，与服务器建立 websocket 连接。作为客户端，首先要定义一个继承自 Websocket 类的 Client 类，里面有各种监听，包括建立连接时、接收到消息时、连接关闭时等等。在 onMessage 方法里，我将服务器传来的 JSON 包装成 GameResponse 对象，然后用 EventBus 发布消息，把该对象传递给主线程。

```
public class Client extends WebSocketClient {

    public Client(URI serverURI, Draft draft) { super(serverURI); }

    @Override
    public void onOpen(ServerHandshake handshakedata) { System.out.println("连接成功"); }

    @Override
    public void onMessage(String message) {
        System.out.println("接收到服务器返回的消息");
        //EventBus发给事件接受者
        try{
            GameResponse response = JSON.parseObject(message,GameResponse.class);
            EventBus.getDefault().post(response);
        }catch (Exception e){
            e.printStackTrace();
            System.out.println("服务器返回消息异常");
        }

    }
```

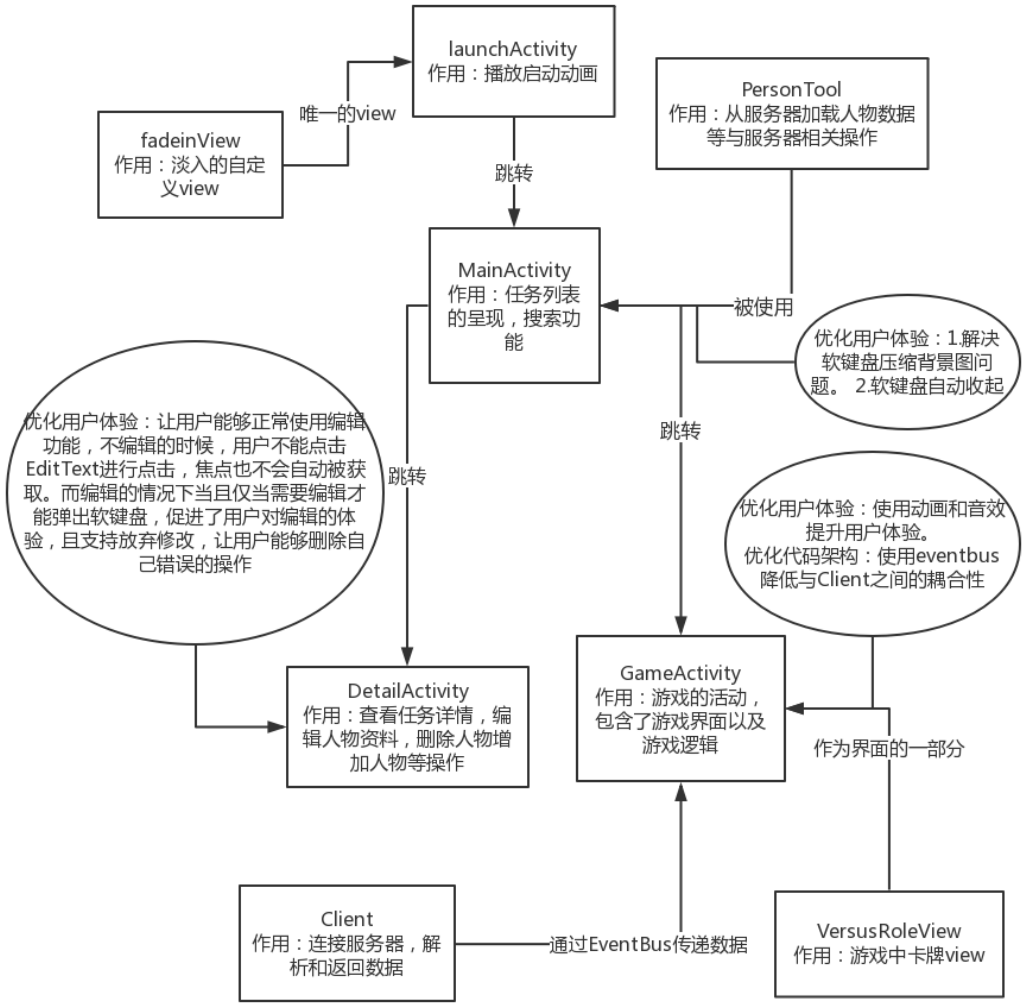
然后在主线程里创建一个 Client 对象，传入的对象分别为 url 以及 websocket 的版本号，connect 方法进行连接，要注意捕捉异常。

```
public void connectServer() {
    try {
        client = new Client(new URI(Const.WEBSOCKET_URI), new Draft_6455());
        client.connect();
        Toast.makeText( context: GameActivity.this,  text: "连接成功", Toast.LENGTH_SHORT).show();
    } catch (URISyntaxException e) {
        Toast.makeText( context: GameActivity.this,  text: "连接失败" + e.getMessage(), Toast.LENGTH_SHORT).show();
        e.printStackTrace();
    }
}
```

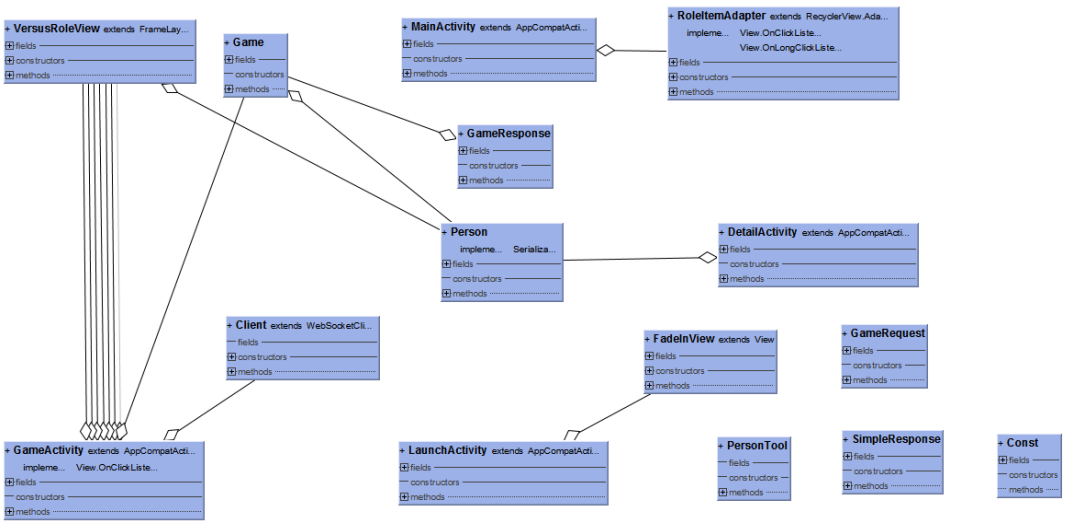
断开连接的时候，先要进行非空判断，同时要注意连接是否打开，因为结束连接意味着服务器要清除会话，同时会通知另外一个用户，若出现异常则会关闭连接失败。

(四) 代码架构

代码架构如下，其中椭圆形内部说明了对应的类的优化情况



以下是类的 UML 关系图，其中连线的“没有菱形的一端”是“有菱形的一端”的成员变量



三、应用效果

1. 启动：

2. 上拉加载

3. 搜索


刘备
男 161-223
关羽
男 ?-219
张飞
男 ?-221
赵云
吕布
吕布
男 ?-198

4. 修改


刘备 (字:玄德)
生卒年份:161-223
性别
男
势力
蜀
籍贯
幽州涿郡涿(
保定市涿州)
简介
刘备，蜀汉的开国皇帝，汉景帝之子中山靖王刘胜的后代。刘备少年孤贫，以贩鞋织草席为生。黄巾起义时，刘备与关羽、张飞桃园结义，成为异姓兄弟，一同剿除黄巾，有功，任安喜县尉，不久辞官；董卓乱政之际，刘备随公孙瓒讨伐董卓，三人在虎牢关战败吕布。后诸侯割据，刘备势力弱小，经常寄人篱下，先后投靠过公孙瓒、曹操、袁绍、刘表等人，几经波


刘备 (字:玄德)
生卒年份:161-223
性别
男
势力
蜀
籍贯
保定市涿州）
简介
刘备，蜀汉的开国皇帝，汉景帝之子中山靖王刘胜的后代。刘备少年孤贫，以贩鞋织草席为生。黄巾起义时，刘备与关羽、张飞桃园结义，成为异姓兄弟，一同剿除黄巾，有功，任安喜县尉，不久辞官；董卓乱政之际，刘备随公孙瓒讨伐董卓，三人在虎牢关战败吕布。后诸侯割据，刘备势力弱小，经常寄人篱下，先后投靠过公孙瓒、曹操、袁绍、刘表等人，几经波
提示
保存修改?
取消
确定

5. 增加


排行榜
刘备
男 161-223
关羽
男 ?-219
张飞
男 ?-221
赵云
男 ?-229
马超
男 176-222
黄月英
女 ?-?
开始游戏
点击设置头像
姓名
性别
男
女
国家势力
魏
蜀
吴
群
刘阿明
未知
生卒年月
?-?
籍贯
广东
简介
此人物是虚构的。


点击设置头像
设置头像
选择本地照片
拍照
取消
广东
此人物是虚构的。
刘阿明
性别 男 女
未知
国家势力 魏 蜀 吴 群
?-?
广东
此人物是虚构的。
排行榜
刘阿明
男 ?-?
刘备
男 161-223
关羽
男 ?-219
张飞
男 ?-221
赵云
男 ?-229
马超
男 176-222
添加成功
开始游戏

6. 删除


刘阿明 (字:未知)
生卒年份:?-?
提示
确定要删除这个武将吗?
取消
确定
简介
此人物是虚构的。
排行榜
刘备
男 161-223
关羽
男 ?-219
张飞
男 ?-221
赵云
男 ?-229
马超
男 176-222
黄月英
女 ?-?
开始游戏

7. 点赞


刘备 (字:玄德)
生卒年份:161-223
性别
男
势力
蜀
籍贯
幽州涿郡涿(
简介
刘备，蜀汉的开国皇帝，汉景帝之子中山靖王刘胜的后代。刘备少年孤贫，以贩鞋织草席为生。黄巾起义时，刘备与关羽、张飞桃园结义，成为异姓兄弟，一同剿除黄巾，有功，任安喜县尉，不久辞官；董卓乱政之际，刘备随公孙瓒讨伐董卓，三人在虎牢关战败吕布。后诸侯割据，刘备势力弱小，经常寄人篱下，先后投靠过公孙瓒、曹操、袁绍、刘表等人，几经波
刘备 (字:玄德)
生卒年份:161-223
性别
男
势力
蜀
籍贯
幽州涿郡涿(
简介
刘备在荆州三顾茅庐，请诸葛亮出山辅助，在赤壁之战中，联合孙权打败曹操，奠定了三分天下的基础。刘备在诸葛亮的帮助下占领荆州，不久又进兵益州，夺取汉中，建立了横跨荆益两州的政权。后关羽战死，荆州
取，刘备大怒，于称帝后伐吴，
战中为陆逊用火攻打得大败，不久病逝于白帝城，临终托孤于诸葛亮。


排行榜
刘备
男 161-223
关羽
男 ?-219
张飞
男 ?-221
赵云
男 ?-229
马超
男 176-222
黄月英
女 ?-?
开始游戏
人气排行榜
刘备
812点赞
关羽
2点赞
张飞
1点赞

8.游戏

四、开发过程中遇到的问题及解决办法

问题一：onCreate 函数中启动 view 动画没有反应

解决：通过复习 activity 生命周期以及查阅相关资料发现 onCreate 调用的时候其实界面是还没呈现出来的，因此在这里开启动画是无效的。activity 还有一个生命周期中的回调函数是 onAttachedToWindow 函数，这个函数调用的时候 view 就已经创建结束呈现出来，此

时就可以开启 view 的动画了

问题二：用动画监听器中监听到动画结束（onAnimationEnd）时调用 openMainAcitvity 函数却发现 openMainAcitvity 函数被调用了三次，也就是说 onAnimationEnd 里面的代码被执行了三次
解决:在网上搜索确实有找到类似的情况,但是都只是讲应对方法而没有讲产生问题的原因，而我自己查看数值动画的源代码看得也不是很理解，主要还是时间比较匆忙就没有深究。因此最后就是的解决方法是自己加了一个标志（boolean），判定 openMainAcitvity 函数是否已经被调用，如果已经调用过了，那么就拒绝再次调用

问题三：EditText 自动获取焦点、无法阻止软键盘的弹出
解决方法：对于 EditText，在他们的父布局加上可获取焦点即可。对于软键盘，需要在每次编辑结束后调用 InputMethodManager 的 hideSoftInputFromWindow 来强制关闭软键盘。

五、思考及感想

赖贤城：

不要停止学习。在做启动页动画的时候有一个很大的收获，以前学 paint 类的时候只知道有 setXfermode 这个方法可以设置两张图片相交时的模式。但是并没有深入学习 PorterDuff.Mode 里面的常量学习其图片混合模式的具体合成方式。这次为了实现擦除蒙版的效果辗转查到了可以通过设置混合模式实现，才去认真学了一下，发现它能做出很炫的效果，而其原理也只是像素与像素执行一个简单的计算公式而已。所以说安卓可以学的东西、神奇的东西还很多，不能停止探索的脚步

不要惧怕看起来很难的东西。这次实验我大量使用了自定义 view 以及动画，说真的这两个都是我以前很怕的东西，总觉得很难。但是真的静下心来认真理解梳理一下发现其实也就那么回事。所以学东西一定不能被它的名字什么的吓倒，或许仔细研究一下就发现不过是只纸老虎

黄路：

这是一次相当奇妙的体验，第一次让我有了做网游的感觉。以前做的都是 web 服务器，服务器只能一次请求一次响应，感觉是被动的。这次运用了 websocket 的技术，服务器终于能主动推送消息给客户端了，也算是有了实现游戏对战的资本了。这次的对战过程做得相当棒，再次感谢鲜橙大佬的杰作，给我“幼小”的心灵带来了深深的震撼。

虽然不是第一次负责 app 的后端代码了，但是这次代码真的挖了很多坑。一味地追求速度，忽略了质量，搞到代码漏洞处处都是，以致于 ddl 的前两天晚上还得熬夜 debug。严格意义上来说，这次服务器用到了三种数据库，关系型是 mysql，以及非关系型的 redis 和 solr 索引数据库，三个数据库都要维护增删改查，加大了开发难度，在某些场景里一些技术还是得谨慎而用，否则真的给自己增添很多的麻烦；此外，自己在写代码的时候感觉脑袋也是不够逻辑清晰，游戏判定规则那里各种搞错，前前后后改了不下 5 次，实在不该。

安卓代码方面，自己还是很多地方没有掌握好的。比如约束布局和相对布局的使用，因为不够熟练而不敢用。还有半学期的安卓，希望接下来能把这些知识都补回来吧。

黄洪彬：

比起平时的实验，这次实验真的体验到了什么叫做 project，其实我们组这个小组是暑假就开始的了，大家都热爱安卓，做自己喜欢的部分。因此我们实现了数据库、动画、

以及一系列的创新功能，因为我们觉得这样才够酷！队友们超给力的，这次实验 100 分的话我给他们打 120 分！

这次实验我对 EditText 的焦点、软键盘的弹入弹出有了更深层次的认识了，他们对于用户来说是一个好的工具，但好像对于开发者并不是那么友善。他们是很不听话的两个小家伙，一个小小的编辑功能竟暗藏如此大的玄机，要管好这两个小家伙，就要请出他们的爸爸——系统来了。通过各种查阅博客，终于明白了为什么 EditText 会如此不听话地召唤出键盘，而键盘又为什么如此不听话地跑出来骚扰用户，因为他们默认有自己的焦点，而我们在不编辑的时候，就要直接禁止他们使用焦点，并强制性把软键盘按下去。实在不行就实用替代品——TextView 来替代 EditText 的显示功能。

除此之外，我们团队这次很好地使用了 GitHub 的功能，大大增加了效率，能够在短短一星期内写出如此高质量的 project 代码，和 GitHub 的同步更新也有很大关系。

但是，这次实验并不是终点，我们将会在期末带来更加震撼的 APP 产品，这是我们团队呕心沥血之作，期待展示的那天能获得更多人的关注！

何伟杰：

本次课程期中项目可以算是“半命题作文”，要求我们实现三国人物词典，以及增删改查功能。

我分配到的部分是做主界面和整体的 UI 图标设计。要说我的工作难也不难，但也不容易，特别是 UI 部分花费了很多时间。主界面其实就是简单放个 RecyclerView 就差不多了，再多几个按钮用于跳转。而因为我们 Lab3 已经实现过 RecyclerView 了，所以界面控件做起来很快。

真正吃力的是要个 APP 选定风格。在我们小组的讨论之后我们觉得三国题材的 APP 用水墨风格比较合适，于是乎就得以水墨风格来设计 UI 界面。比如说主界面背景，要水墨风格，不能太显眼，给人一种似有非有的朦胧感最好。比如说控件图标，网上找不到现成的图标可以使用，只能自己设计出来。而寻找灵感花费了一天一夜。最后找到了毛笔的笔画图，自己 P 图组合笔画形成现在的 UI 图标。

反正现在最大的感想就是，以后绝对不要当 UI 设计师，真的很烦很累的。然后我的部分要实现的代码不多，所以又花了点时间优化了主界面的用户体验，从用户的角度来对待自己的成果，然后发现不舒服的地方就改掉，最后自己的 APP 用起来要非常舒服才算是完成了这次的任务了。最后，我们组的游戏真的很棒，虽然不是我实现的但我参与了里面部分图片的制作，然后被大佬们做的这么棒真心觉得自己的付出是值得的。希望自己也多加学习赶上大佬们的步伐。

六、小组分工

赖贤城	启动页面、游戏设计
黄路	建立数据库、前后端对接
何伟杰	主界面、图片素材收集与制作
黄洪彬	详情界面、音效素材收集与制作

七、参考资料

无