# 毕业论文初稿

## 论序

### 1.1课题背景

由青光眼导致的失明是不可逆的，而且多数患者都没有明显的症状，因此青光眼的早期诊断显得尤为重要，视野检测是处理青光眼的基本手段，定期进行检测有助于发现，预防青光眼以及追踪病情进展，但是现如今只能在有医疗设备条件的医院进行检测，对于处在偏远地区的人来说是十分不方便的，基于这个需求，开发设计了一款移动端的快速筛查系统。

### 1.2国内外现状

随着VR技术的不断成熟，VR设备的价格也大大降低

SITA快速程序只用FastPac一半的时间，却能提供相同的结果，明显优于旧策略

如今的移动医疗app相关体制和法律不健全，未能得到受中信赖

### 1.3 本文的主要功能

此系统使用了java进行app框架搭建，使用C#进行图像处理，使用python进行服务端搭建，实现了视野检测，结果展示与存储。

## 需求分析

### 2.1 功能性需求

为了保证用户能够再次查看自己的测试结果，不至于在app杀掉之后数据丢失，故需要将测试结果进行存储，在此可以有两种选择方式，一是使用android自己的sqlite数据库，这种方式是将数据存在本地，每当你进行查询时，他就会查询本地数据库匹配返回结果，二是将数据库建立在服务端，这种方式是将数据库存在服务器上，每次查询会发送一个get请求，服务器接收到请求后会查询数据库匹配返回结果。这里我选择了第二种方式。

为了检测测试过程中的视心是否发生偏移，我创建了两个数组进行验证，一个是记录着视标位置信息的二维数组location[][]，一个是记录位置是否可见的以为数组status[]，当时标所处的位置status[i]为false是，说明这个位置是不可见的，但如果此时检测者做出了反应，那么就说明视心发生了偏移，医学上称之为固视丢失。

为了检测测试过程中是否存在作假操作，作假操作分为两种情况，一是视标还没有进行显示，但检测者做出了反应，二是视标的亮度比前一次响应时的亮度要亮，但却没有做出反应。对情况一，我建立了一个视标的显示状态ifdisplay，如果测试者进行了响应，但此时ifdisplay=false，表明视标还没有出现，那就说明测试者进行了虚假操作，医学上称之为假阳性。对于情况二，我建立了一个一维数组bright[],数组中的每个值的初始值都为1，如果进行了一次成功的检测(视标出现在正确的位置上，测试者进行响应)，则对应的bright[i]的值加1，如果某个位置上测试者没有进行响应，但此时对应位置上的bright[i]>1,则说明之前已经对更弱亮度的视标进行了响应，医学上称之为假阴性

为了将测试结果转换成图片,我使用了unity中内置的GL类，他封装了OpenGL,也就是说，我使用OpenGL对图像做的处理，为了准确的画出结果图，我将位置数组中每两个点之间的距离设置为1f，而在使用OpenGL画图的过程中，我使用的元素类型(void glBegin(GLenum mode))为四边形(GL\_QUADS),以每个位点为中心点，分别向上下左右增加0.5f作为四边形的四个顶点,这样每个位点画出来的四边形刚好是相接的，而不是重叠的。

为了加快测试进行的速度，我进行了两种方式的处理，一是使用SITA快速策略，如果检测位点的视野阈值降低，那么其相邻的9个位点的视野阈值也会降低，将原本需要检测的72个位点缩短为检测54个位点。二是对检测者的响应机制做了调整，当测试者进行响应时，如果上次响应的结果计算已经结束，那么会立即开始本次响应结果计算。

为了保证检测结果的唯一性，使用了身份证号和眼睛作为联合主键的形式，进行结果存储。

为了使检测者能够执行查看测试结果，对ckeckbox做了监听，当选中对应的checkbox时，会显示响应眼睛的结果，也可以同时选中，那么将会显示左右眼睛合并后的结果，合并的处理就是读取所有的图片为bitmap，然后在将所有的bitmap写入到canvas中，并为之添加文字，最后储存为新的图片。

为了使生成的结果能够分享给他人，我们使用了Android自身为我们提供的分享功能，其实他也是以Intent的方式从一个Activity跳转到了另一个Activity，这里我们只需要指定我的的Action为ACTION\_SEND，指定数据及类型就可以。

### 2.2 非功能性需求

由于此app的开发使用的多种语言进行开发，为了减少运行环境对程序的影响，故对开发语言做以下规定

|  |  |
| --- | --- |
| 语言 | 版本号 |
| java | 1.8 |
| python | 3.9 |
| C# | 8.8.3 |

为了减轻开发难度，需要使用多种开发框架，为了减少运行环境对程序的影响，故对开发框架做以下规定

|  |  |
| --- | --- |
| 开发框架 | 版本号 |
| Android API | 30 |
| .NET | 5.0 |

为了减轻开发难度，且运用了多种开发语言，故也需要使用多种开发工具，为了减少运行环境对程序的影响，故对开发工具做以下规定

|  |  |
| --- | --- |
| 开发工具 | 版本号 |
| Android Studio | 3.5.2 |
| Unity Hub | 2.4.14 |
| Unity | 2018.4.29f1 |
| Visual Studio | 2019 |
| Idea | 2020.2 |
| Sqilte | 3.32.3 |

为了减轻后期维护成本，开发遵循软件工程开发规范

为了提升用户的体验，应在界面中给予一定的提示，色彩应尽量柔和

### 2.3 设计约束

Android api版本必须大于19

## 概要设计

### 3.1 总体设计

此系统遵循着MVC模式的设计理念，主要有仨大模块组成，展示层，控制层，数据层

展示层完全由android处理，通过加载xml以及接收处理过后的数据将结果展示在app上，

控制层一部分使用java，一部分使用C#，其中java部分主要进行与服务器交互，加工结果，以及控制分享，C#部分主要进行算法检测，数据层就是控制着数据的存储与查询

### 3.2 接口设计

KeyEvent.Callback中的onKeyDown,该方法捕捉按键事件

View.OnClickListener中的onCheckedChanged,该方法监听改变

View.OnClickListener中的onClick,该方法监听点击

Activity中的onCreate,该方法用于初始化Activity中的view

Fragment中的onCreateView，该方法用于初始化Fragment中静态的view

Fragment中的onActivityCreated，该方法用于初始化Fragment中非静态的view

### 3.3 数据库设计

关联表 association

关联表中主要存放people表中用户id之间的联系，当同一个人使用两个id进行检验时，如果这两个id进行了关联，那么在people表中只会产生一条结果，如果这两个id没有进行关联，那么在people表中会产生两条结果，但是这两条结果是同一个人产生的，所以为了保证一致性，需要进行id关联

用户表 people

用户表中主要存放着用户的相关信息，包括id，眼睛（eye），固视丢失率（sightingLoseRatio），假阴性率（falseNegativeRatio），假阳性率（falsePositiveRatio），以及72个位点的视野阈值情况，由于眼睛分为左眼和右眼，故为了保证唯一性，使用id和eye组成联合主键来确保每条数据都是唯一的

# 详细设计

## 4.1 展示层

### 4.1.1功能描述

LoginActivity用于加载activity\_login.xml中的内容，是app启动后展示的第一个页面，其中包括两部分，一是拥有一个用于填写id的EditText，以及一个用于发送id的Button，另一个是只有一个选择不发送id的Button

MainActivity用于加载activity\_main.xml中的内容，是在LoginActivity中进行点击Button后跳转后的页面，其中包括三部分，第一部分是导航栏，用于选择Fragment，第二部分和第三部分分别对应着选择之后的ChoiceFragment和ResultFragment

ChoiceFragment用于加载fragment\_choice.xml中的内容，是当在MainActiviy的导航栏中选择主页后展示的页面，主要包括两部分，一是用于给予用户使用提示的TextView，一是用于选择测试眼睛的Button

ResultFragmnet用于加载fragment\_result.xml中的内容，是当在MainActivity的导航栏中选择结果后展示的页面，主要包括四部分，一是用于选择眼睛测试结果的checkbox，二是用于加载控制层传输过来的数据的TextView，三是用于加载控制层传输过来的图片的ImageView，四是用于分享的Button

UnityPlayerActivity用于展示VR效果下的视野检测组件，是当在ChoiceFragment中选择好测试的眼睛后，在控制器上确认开始检测后展示的页面，主要包括三部分，一是用于给予用户使用提示的Text，二是视野检测中使用到的背景板，三是视野检测中使用到的视标

### 4.1.2 类图设计

### 4.1.3 关键流程设计

在LoginActivity中使用onCreate加载activity\_login.xml对LoginActivity进行初始化

在MainActivity中使用onCreate加载activity\_main.xml对MainActivity进行初始化，使用

setTableSelection选择加载哪一个Fragment，使用clearSelection清除所有Fragment选择状态，使用hideFragments隐藏所有的Fragment，防止多个Fragment显示在界面上

在ChoiceFragment中使用onCreateView加载fragment\_choice.xml，使用onActivityCreated对ChoiceFragment中的组件进行初始化。

在ResultFragment中使用onCreateView加载fragment\_result.xml，使用onActivityCreated对ResultFragment中的组件进行初始化，使用shareStatus控制分享按钮是否隐藏，使用

setImageView设置图片存在与不存在时的情况

在UnityPlayerActivity中使用onCreate对UnityPlayerActivity进行初始化，使用onkeyDown来响应控制器

## 4.2 控制层

### 4.2.1 功能描述

LoginController用于处理LoginActivity中的逻辑，其中包括两部分内容，一是用于检测EditText中的输入是否合法的editTextCheck，一是用于监听按钮点击事件的onClickLogin

MainContyoller用于处理MainActivity中的逻辑，包含用于监听View点击事件的onClickMain

ChoiceController用于处理ChoiceFragment中的逻辑，其中包括两部分，一是用于检测控制器是否连接的ifExistAir,一是用于监听按钮点击事件的onClickChoice

ReaultController用于处理ResultFragment中的逻辑，其中包括四部分，一是用于加载图片的getImg，二是用于合并图片的mergeBitmap，三是用于分享的onPress，四是用于更新数据的updateVal

UnityController用于处理UnityPlayerActivity中的逻辑，其中包括三部分，一是用于向服务器端发送数据的insertVal，二是用于接收unity部分传输过来的数据的accept，三是向unity部分传输数据的send

ChooseEye用于处理选择眼睛后的使用的计算逻辑chooseEye，以及启动检测程序ifStartCheck

ResultDisplay使用Opengl的方式将数字结果转化成结果图的OnPostRender

ResultStore用于将结果图存储到特定位置的DrawResult

SightingPostLocationDeal用于控制视标显示的位置，时间，以及视标的亮度sightingPostMove

SwitchTheScreen用于控制横竖屏的切换，横屏switchHorizontal，竖屏switchVertical

ThresholdCalculate用于计算检测结果的thresholdCalculate

### 4.2.2 类图设计

### 4.2.3 关键流程设计

当检测者选择填写editText并提交时，会首先触发点击按钮的监听事件，这个监听首先调用了editTextCheck方法，这个方法是一个字符串的正则表达式匹配如果能过匹配上，就返回ture否则返回false，之后会在监听中接收到这个返回值，如果为true，则跳转到下一个Activity，如果为false则显示输入不合法的提示并返回，如果选择的是不填写，则会直接跳转到下一个Activity

跳转到MainActivity之后，首先选中默认的tab，即主页，之后当选择结果后，会触发点击按钮的监听事件，这里就只调用了setTabSelection方法，在这故方法中会先清除掉之前选中tab的状态clearSelection，隐藏掉所有的fragment，然后在根据选择的tab进行判断，如果选择的fragment之前已经创建则直接将它显示出来，如果没有创建则进行创建

当点击主页fragment中的按钮时，会触发按钮的点击事件，首先会创建一个Intent，然后根据选择的按钮，将对应按钮的信息传入intent中，并在提示文本textView中显示出来，之后会调用ifExistAir来判断控制器是否接入，接入则返回true，没有接入则返回false，之后会在监听中接收到这个返回值，如果为true，则跳转到下一个Activity，如果为false，则在提示文本中显示没有连接

当跳转到UnityPlayerActivity后，会首先显示提示文本，提示测试者点击控制器以开始检测，

首先视标会每隔2秒的时间显示0.2秒，为了加快测试进度，如果检测者在视标本次显示到下次显示的时间范围内，进行了响应，那么视标将在本次结果计算完成后立即显示，当检测着对视标进行响应时，会触发以下操作，判断视标是否在可视范围之内，如果再，跳过，如果不再，对应固视丢失的值加一，如果没有在时间范围内进行响应，对应假阳性的值加一，当检测者没有对视标进行响应时，会触发以下操作，判断视标是否处于有效显示期，如果再，判断这个位置之前有没有被响应，如果有，对应假阴性的值加一，之后为了加快检测，会将其周围的9个点的视野阈值降低，检测将在所有可视范围内的点全部检测完成后结束，视标的位置是随机出现的

检测完成后调用OnPostRender开始进行结果绘制，绘制时会对位点的有效性进行判断，如果正在绘制的点的显示状态为false，则跳过这个点的绘制，这样在72个点位中，只有54个被绘制了，所绘制的点位每两个点的距离是1f，所以为了每个绘制的正方形相连接，均以位点为中心，然后向上下左右增加0.5f，这样绘制正方形的四个点中，每两点的距离为1f

绘制完成后，调用DrawResult，将绘制的结果储存到指定路径下，之后调用SceneManager.LoadScene(0)重新加载场景，调用jumpToMain返回到MainActivity

当选中结果Fragment是，会首先进行一次刷新，将最新的值显示在view上，当点击分享按钮时，会首先调用mergeBitmap将所有的结果重新绘制在一个bitmap上，然后会跳转到其他app的Activity将结果分享

## 4.3 数据层

### 4.3.1 功能描述

当客户端发送get请求携带参数为/select？Id=\*\*\*&eye=\*\*\*时，服务端会调用selectval用于查询所需数据，并将结果以json形式返回，当客户端发送get请求携带参数为/insert？val=\*\*\*，服务端会调用insertval用于向数据库中插入数据

TableTool.desc用于查询数据库中表的结构

TableTool.connect用于连接数据库

TableTool.create用于创建表

TableTool.insert用于向表中插入数据

TableTool.select用于查询表中的数据

TableTool.calculate 用于计算表中数据的占比

TableTool.update用于更新表中的数据

TabelTool.delete用于删除表中的数据

TableTool.drop用于删除表

### 4.3.2 类图设计

### 4.3.3 关键流程设计

当进行完一次检测后，UnityPlayerActivity会调用httpcontroller.insertVal想向服务端发送一条get请求携带/insert，服务端接收到请求后，会将调用TableTool.insert将数据写入到数据库中

当选中结果fragment时，ResultActivity会调用httpcontroller.selectVal向服务端发送一条get请求携带/select，服务端接收到请求后，会调用TableTool.select查询数据，并调用TableTool.calculate计算数据在库中所占的比例，之后会将查询结果与计算结果放入一条json中，然后返回给客户端

