目录

一．应用简介

1.1 背景

当前的智能手机高度发展，娱乐信息唾手可得，网上冲浪也变得方便快捷。同时，现代生活节奏也越来越快，碎片化时间很多，但是未能得到充分有效的利用。手机就像是有一种魔力，牢牢地抓住了人们的双手，我们只能看着时间从自己的指缝间溜走。想要认真坐下来静心工作，却总在不经意间打开手机摸鱼，严重影响工作与学习的效率。

因此我们决定开发一款时间管理与保持专注的 app，来减少人们工作学习被手机影响的情况出现。

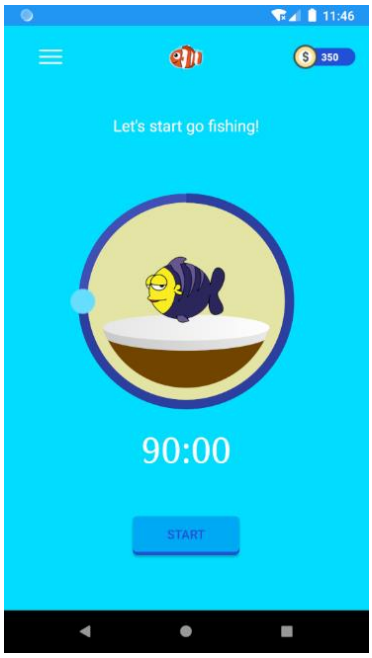
1.2 与同类 APP 的对比

目前市场上已有类似的 app，但是大多数都为付费软件。例如，常见的同类型产品有 forest，番茄 todo 等等。

首先最为人熟知的 forest，是 ios 平台的一款应用。forest 的界面简介，功能丰富，有好友系统，可以共享森林，可视化成果。但是它的缺点就是需要付费使用，并且有些时候使用还会出现 bug，使得限制使用手机的功能失效。

其次是番茄 todo 这款 app，它是一款免费 app，支持安卓和 ios 平台，而且它的各种统计数据非常详细。支持好友系统，也可以创建自习室，共享学习区等等。但是他的缺点就是页面略显冗余。

而我们的 app 取长补短，实现了严格的手机管理，计时期间切出 app 时间超过五秒就会失败，不会出现 forest 的失效的 bug，也实现了成果的可视化，并且界面简洁美观。而且当然，我们的 app 也是免费的。并且我们还在考虑添加其他功能，比如可以拍摄并上传专注的视频来分享，实现好友系统，切换主题等等。但是由于时间原因没有来得及实现。



1.3 运行环境

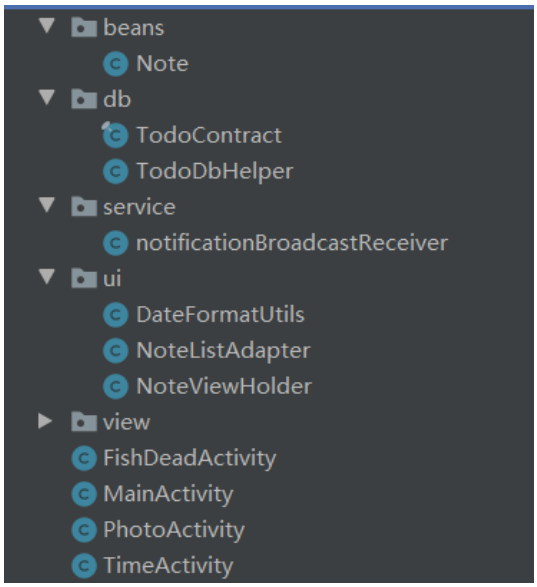
Android Studio 4.1.0/3.6.0 均可

API 25~27 均可，API 28 和 29 创建的虚拟机里面，设置里面没有设置相应权限的选项，会抛出异常
如果打开 APP 提示需要获取权限，请点击确认，会跳转到设置权限的界面。在设置完毕之后，在切换回到 APP 方可继续运行。

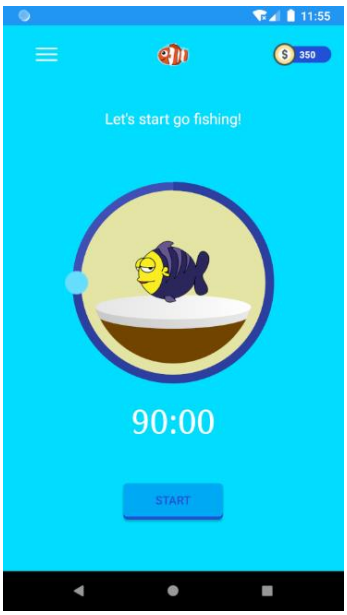
二．总体架构及功能设计

（本部分只做思路介绍，具体的代码讲解请看第三部分）

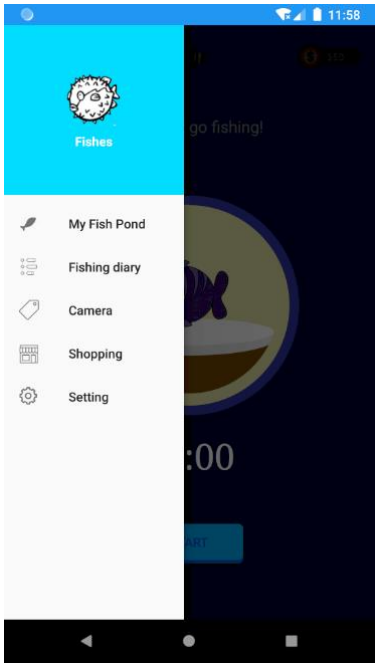
2.1 总体架构



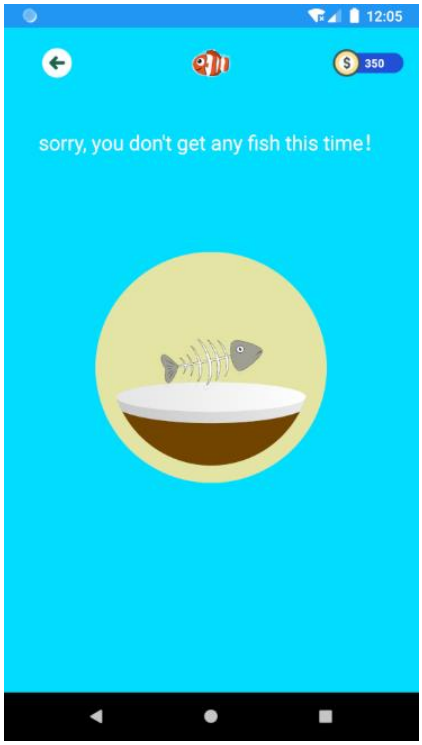
这就是我们的 app 的包的结构图。
其中 MainActivity 就是我们程序的主界面，我们可以在这个界面设置专注的时长，以及展开菜单，设置鱼的风格等等。效果如下



点击左上角会弹出菜单，可以点击菜单来切换到其他的界面：

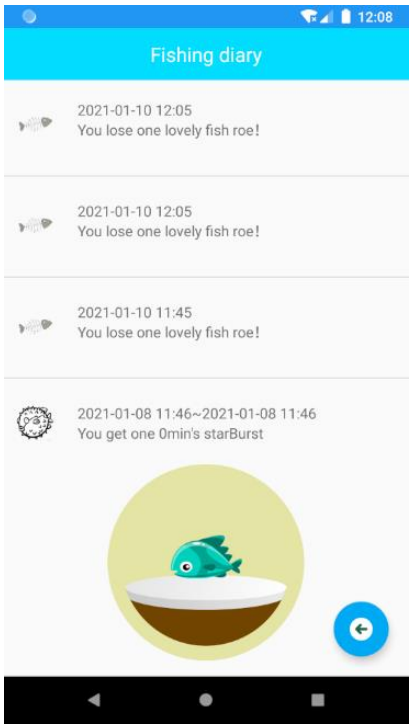


而 FishDeadActivity 则对应着鱼死后的界面。当鱼死之后则跳转到这个界面。



PhotoActivity 则是对应着点击左上角弹出菜单中，点击 Camera 跳转的界面。提供录制视频的功能。

最后的 TimeActivity 则是点击左上角菜单后的查看历史专注记录的界面，效果如下



而 Note，TodoContract，TodoDbHelper 个文件则是与数据库相关的，定义了存放历史纪录的数据库。

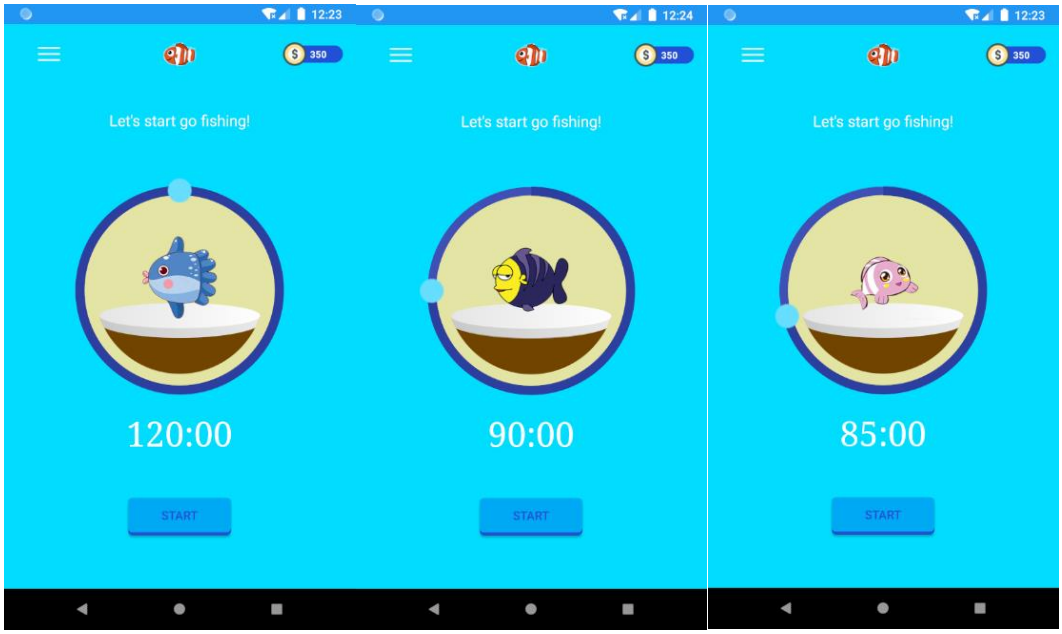
notificationBroadcastReceiver.java 则是用来接受鱼死亡的信号的。

剩下的部分则是与 ui 界面相关的,不再赘述。

2.2 计时功能设计

我们在主界面通过一个环形进度条，可以拖动设置倒计时的时长，而后点击 start 即可开始倒计时。

当设置时间的时候，根据设置时间的长短，我们可以看到倒计时结束之后鱼成长的结果。不同的时间长度对应不同大小的鱼。



而在点击 start，倒计时开始之后，我们会看到鱼从小长到大的过程。（由于这一部分花费时间较长，不在不同阶段截图，改为贴上对应的实现代码）

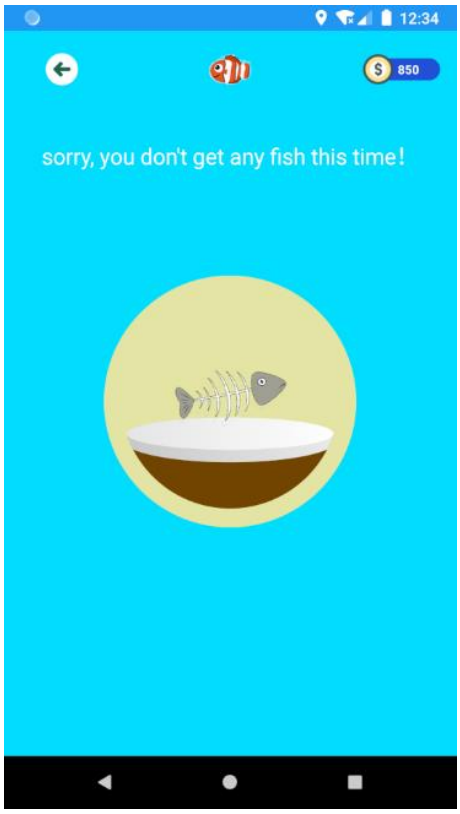
```
int minute = progress/60;
if(minute<1)
    level = 1;
else if(minute<5)
    level = 2;
else if(minute<10)
    level = 3;
else if(minute<60)
    level = 4;
else if(minute<90)
    level = 5;
else if(minute<120)
    level = 6;
else
    level = 7;
fishName = fishName + level;
```

2.3 限制切换出 APP 功能设计

为了达到限制使用手机的目的，我们在倒计时开始后，限制切换出此 APP。当切出去的时候，用户会在屏幕底端看到提示，需要在 5 秒钟内返回 APP，如果用户没有在 5 秒内返回，则鱼会死亡，用户也会收到通知。

这部分设计的思路就是检测用户当前是否在使用此 APP，如果不是则有个计时器开始倒计时，很直接了当的实现。

当用户专注失败之后，鱼死亡，用户切回来的时候看到如下的界面



2.4 商店功能设计

为了提高趣味性与吸引力，我们还设计了商店系统。我们设计了三种鱼的风格（皮肤），用户可以用积累的金币来解锁。

用户首先要去左上角的菜单点击 Shopping，然后会弹出来上面的选择对话框。用户可以点击后两种鱼的图片来花费金币解锁（第一种鱼是默认拥有的）。如果金币不够，则会提示金币不足，如果金币足够则会提示成功解锁。

然后用户可以回到主界面，点击中间圆形的鱼的图片，然后又会弹出上面的选择对话框。如果用户点击了未解锁的鱼的图片，则会提示尚未解锁对应的鱼。如果用户点击的鱼

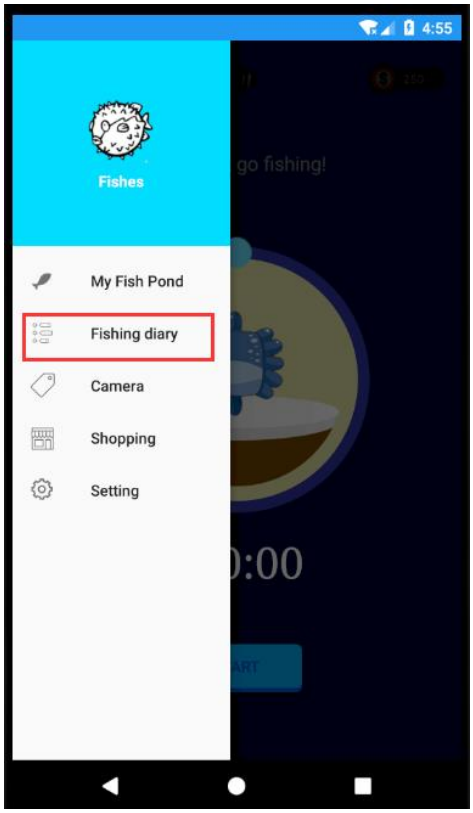
已经解锁，则会提示切换成功，并且切换到用户选择的鱼的种类，然后回到主界面。

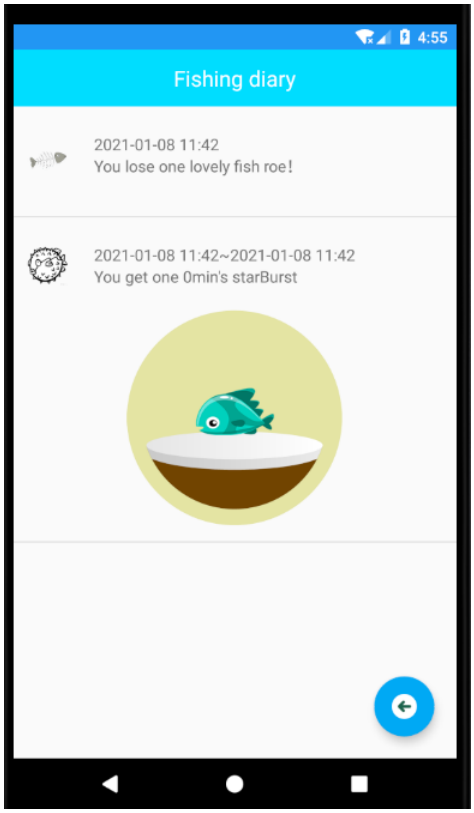
三．关键数据结构/算法

3.1 数据库的建立与使用

在本程序中，使用了数据库 Android：SQLiteOpenHelper 类。在程序中的 src/main/java 下可以看到 db 包，包含了项目中的数据库建立、索引、插入、查询等部分。

这个数据库主要用于对用户使用应用的历史信息进行存储以及查询写入。在 main 中我们通过这个数据库来查询用户过去使用本 app 时哪些 fishing 任务是完成的，以及完成时间、完成的鱼类种类等。同时也会保存 fishing 的失败数据，在主界面左侧的“Fishing Diary”查看具体信息。具体操作及结果可见下图演示截图：





创建数据库之前，我们要先建立自己的数据格式，也就是我们的事件如何存，存储哪些信息等。这一部分的定义在 beans 包下的 Note 完成。源代码如下：（仅截取定义部分）

```
public class Note {
    public final long id;        //事件在数据库中存储条目对应的唯一 id
    private String deadline;     //事件的截止日期
    private String scheduled;    //事件的记录日期
    private String state;        //事件的状态,分为 Todo,Done 和 None
    private String caption;      //事件的标题
    private String time;         //事件的设定时间
}
```

在 TodoContract 部分中完成了对数据库的定义以及列、表数据的输入部分，具体代码如下：

```
public final class TodoContract {
    //新建表
    public static final String SQL_CREATE_NOTES =
            "CREATE TABLE " + TodoNote.TABLE_NAME
                    + "(" + TodoNote.COLUMN_ID + " INTEGER PRIMARY KEY
AUTOINCREMENT, "
                    + TodoNote.COLUMN_TIME + " TEXT, "
                    + TodoNote.COLUMN_DEADLINE + " TEXT, "
                    + TodoNote.COLUMN_STATE + " TEXT, "
                    + TodoNote.COLUMN_SCHEDULED + " TEXT, "
                    + TodoNote.COLUMN_CAPTION + " TEXT)";

    private TodoContract() {
    }
    //TodoNote 的列名对应
    public static class TodoNote implements BaseColumns {
        public static final String TABLE_NAME = "fish";
        public static final String COLUMN_ID = "_id";
        public static final String COLUMN_DEADLINE = "deadline";
        public static final String COLUMN_SCHEDULED = "scheduled";
        public static final String COLUMN_STATE = "state";
        public static final String COLUMN_TIME = "time";
        public static final String COLUMN_CAPTION = "caption";
    }
}
```

在 TodoContract 自定义好我们所需的表后即可通过 TodoDbHelper 类完成数据库和表的创建，以方便后续程序对数据的读取与写入。数据库存放在根目录下，名称使用 todo1.db：

```
public class TodoDbHelper extends SQLiteOpenHelper {
    private static final String DB_NAME = "todo1.db";
    private static final int DB_VERSION = 2;
    public TodoDbHelper(Context context){
        super(context,DB_NAME,null,DB_VERSION);
    }//构造数据库

    @Override
    public void onCreate(SQLiteDatabase db){
        db.execSQL(TodoContract.SQL_CREATE_NOTES);
    }//创建表

    @Override
```

```
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVer
sion) {
    }
}
```

就此，我们建立了一个数据库，方便我们独写用户的使用信息。

3.2 时间部分的获取与处理

既然是一个类似 to-do-list 的应用，那自然少不了对时间数据的处理。

在获取系统时间时，我们使用了安卓的系统调用函数，但是返回的是 long 类型的数据，那么为了后续的使用方便，我们在 src/main/java 下定义了 UI 包，包内的 DataFormatUtils 实现了对时间数据的处理部分。

在本程序中，我们定义了两种类型的时间格式，分别是年月日、年月日时分秒。这两种时间数据与系统返回的时间数据之间主要使用函数重载完成 long 与 string 之间的转换。精简部分代码显示如下：

```
public class DateFormatUtils {
    private static final String DATE_FORMAT_PATTERN_YMD = "yyyy-MM-dd";
    private static final String DATE_FORMAT_PATTERN_YMD_HM = "yyyy-MM-
dd HH:mm";

    /**
     * 时间戳转字符串
     *
     * @param timestamp     时间戳
     * @param isPreciseTime 是否包含时分
     * @return 格式化的日期字符串
     */
    public static String long2Str(long timestamp, boolean isPreciseTime
) {
        /* …… */
    }

    private static String long2Str(long timestamp, String pattern) {
        /* …… */
    }

    /**
     * 字符串转时间戳
     * @param dateStr       日期字符串
     * @param isPreciseTime 是否包含时分
     * @return 时间戳
     */
    public static long str2Long(String dateStr, boolean isPreciseTime)
{
```

```
        /* …… */
    }

    private static long str2Long(String dateStr, String pattern) {
        /* …… */
    }

    private static String getFormatPattern(boolean showSpecificTime) {
        /* …… */
    }
}
```

3.3 安卓应用权限获取

本应用是类似 forest 应用的监控手机使用情况的 APP，所以需要向用户申请特定的权限。

在本应用中，我们在 src/main/java 中的 UI 包定义了 PermissionDialog 类，实现对权限的申请与处理。在这个类中，我们主要完成申请权限部分时的弹窗以及消息显示部分，同时我们使用安卓的内部 dialog 类实现弹窗。源代码部分如下：

```
public class PermissionDialog extends Dialog {
    private Context context;
    private TextView jumpButton;
    private jumpClickListener jumpListener;

    public PermissionDialog(Context context){
        super(context, R.style.Dialog);
        this.context = context;
    }
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.dialog_permission);
        setCanceledOnTouchOutside(true);
        jumpButton = findViewById(R.id.jump);
        jumpButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                jumpListener.onClick();
            }
        });
    }
    public interface jumpClickListener{
        public void onClick();
    }
```

```
    public void setJumpClickListener(jumpClickListener jumpClickListene
r){
        this.jumpListener = jumpClickListener;
    }
}
```

此外，我们在 src/main/res/layout 中创建 permission_dialog.xml 文件，定义显示部分的属性，显示出错信息。具体代码部分截取如下：

```
    <RelativeLayout
        android:orientation="vertical"
        android:layout_width="match_parent"
        android:layout_height="wrap_content">
        <TextView
            android:layout_width="match_parent"
            android:layout_height="40dp"
            android:text="In order to use program properly, please give
 Fishes fully permissions!"
            />
```

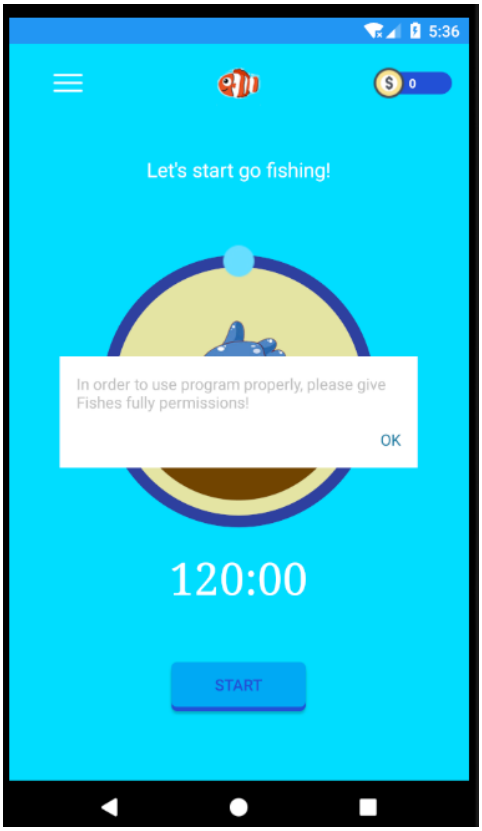
好的，我们现在定义好了获取权限的弹窗以及消息显示类，那么就需要在应用最开始时调用这个类，并完成向用户获取权限的功能。这部分在 MainActivity 中实现。

在 MainActivity 初始被创建时，也就是 onCreate 函数内，我们首先要做的就是检测用户是否是第一次使用，是否赋予了相应的权限。所以我们调用定义好的 checkGetAppInfPermission()函数来检测当前的用户权限，当返回为 false 时说明未获得对应权限，则调用上文提到的 PermissionDialog 弹窗，提醒用户分配相应权限，并在获取相应权限后重启 APP 即可正常使用。具体的实现代码如下所示：

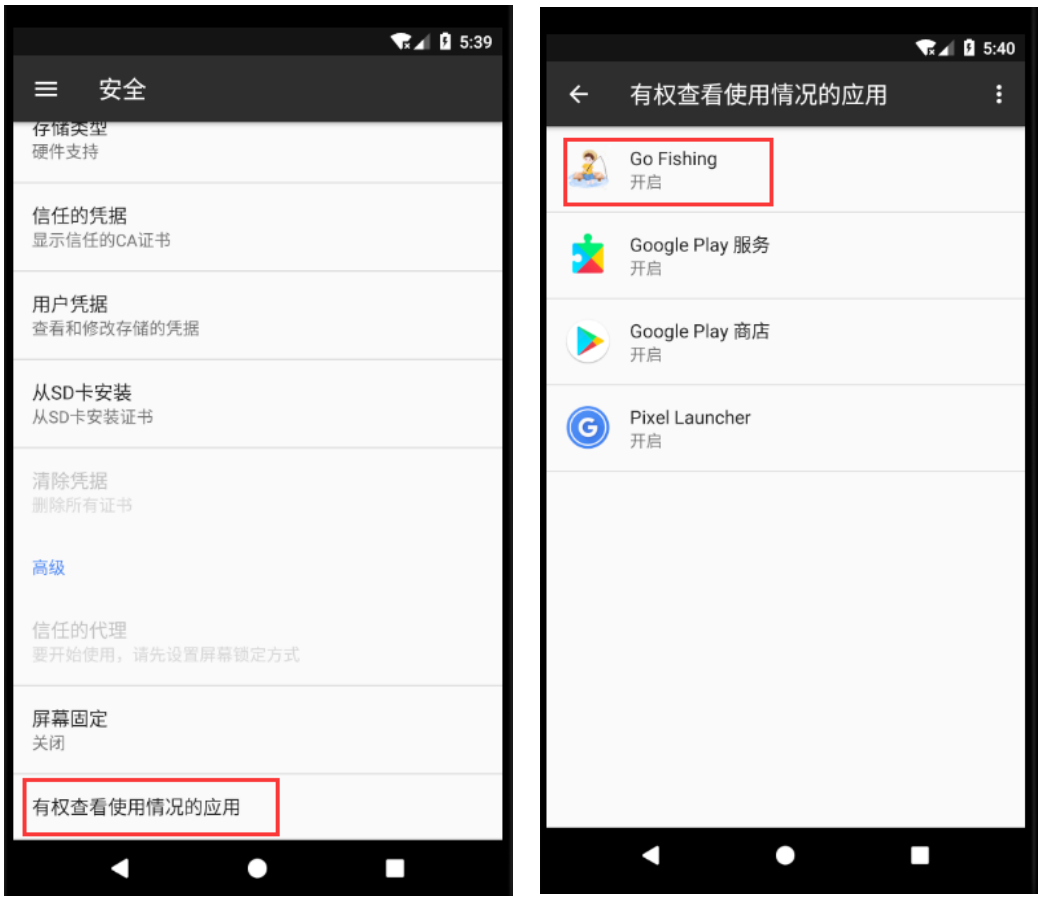
```
protected void onCreate(@Nullable Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        //获取权限
        if(!checkGetAppInfPermission(getApplicationContext())){
            mPermissionDialog = new PermissionDialog(MainActivity.this)
;
            mPermissionDialog.setJumpClickListener(new PermissionDialog
.jumpClickListener() {
                @Override
                public void onClick() {
                    Intent intent = new Intent(Settings.ACTION_USAGE_AC
CESS_SETTINGS);
                    intent.setComponent(new ComponentName("com.android.
settings", "com.android.settings.Settings$SecuritySettingsActivity"));
                    intent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
                    startActivity(intent);
                }
            });
            mPermissionDialog.show();
```

```
        }
    /* ..... */
}
```

以下是具体的使用情况截图：



那么，用户怎样设置权限呢？用户可以通过两种方式，首先是通过点击“OK”，应用会自动跳转至系统设置栏，找到对应的 APP 并选择信任赋予权限项即可。另外还可以通缩 Setting->Secutity->Advanced->Permission 找到 Go Fishing APP 赋予权限即可。



3.4 MainActivity 思路及调用详解

1. 主界面的绘制与鱼类选择

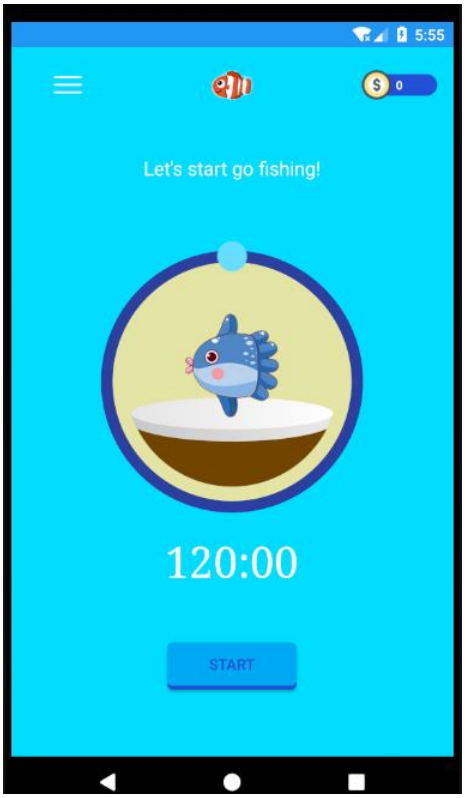
   在主界面，由于我们使用了三个不同的鱼类，所以用户可以在主界面选择自己喜欢并且已解锁的鱼类进行养殖。那么在主界面中就需要对用户选择的鱼类进行 unlock 判断与相应的显示。我们通过设定默认值让程序在启动时显示 style1 类的鱼类，这也代表了用户默认的初始可养殖鱼类。

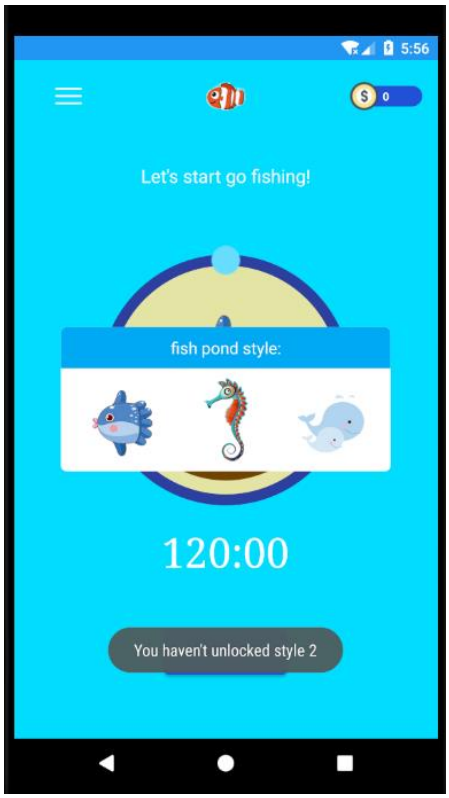
   我们在 MainActivity 中设置了全局的 int 变量 style2 与 style3 来表示用户是否解锁对应鱼类，0 代表未解锁，1 代表已解锁。判断过程部分截取代码如下所示：

```
mBase.setOnClickListener(new View.OnClickListener() {
@Override
    public void onClick(View view) {
    mChooseDialog = new ChooseDialog(MainActivity.this);
    mChooseDialog.setStarBurstOnClickListener(new ChooseDialog.starBurs
tOnClickListener() {
        @Override
            public void onClick() {
            choose = "starBurst";
            mChooseDialog.dismiss();
            update("starBurst", progress);
        }
```

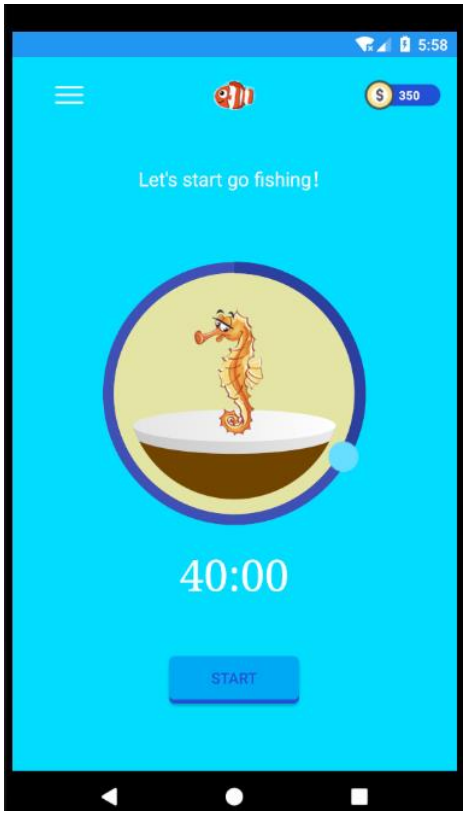
```
    });
    mChooseDialog.setTimeOnClickListener(new ChooseDialog.timeOnClickLi
stener() {
        @Override
            public void onClick() {
            if (style2 == 1) {
                choose = "time";
                mChooseDialog.dismiss();
                update("time", progress);
            }
            else {
                Toast.makeText(MainActivity.this,
                    "You haven't unlocked style 2", Toast.LENGTH_SHORT)
.show();
            }
        }
    });
    mChooseDialog.setStarClickListener(new ChooseDialog.starOnClickList
ener() {
        @Override
            public void onClick() {
            if (style3 == 1) {
                choose = "star";
                mChooseDialog.dismiss();
                update("star", progress);
            }
            else {
                Toast.makeText(MainActivity.this,
                    "You haven't unlocked style 3", Toast.LENGTH_SHORT)
.show();
            }
        }
    });
    mChooseDialog.show();
}
});
mProcessBar.setProgressCallback(new processBar.progressCallback() {
@Override
    public void updateListener(int _progress) {
    progress = _progress;
    update(choose, progress);
}
});
```

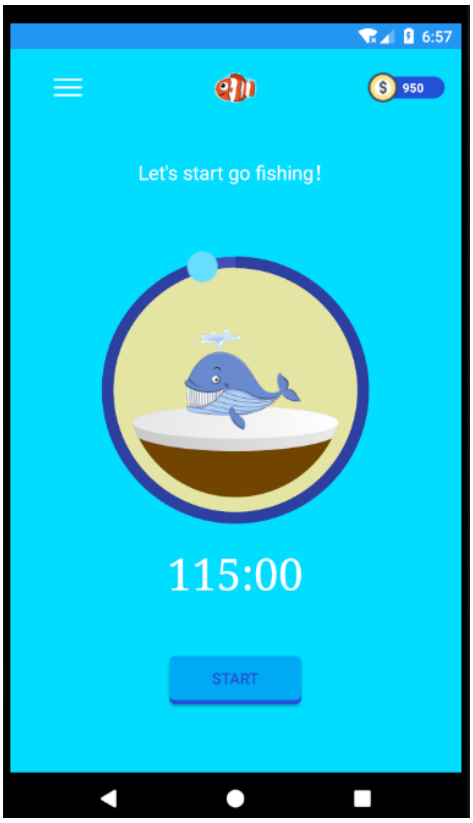
下图显示实际的操作与截图：

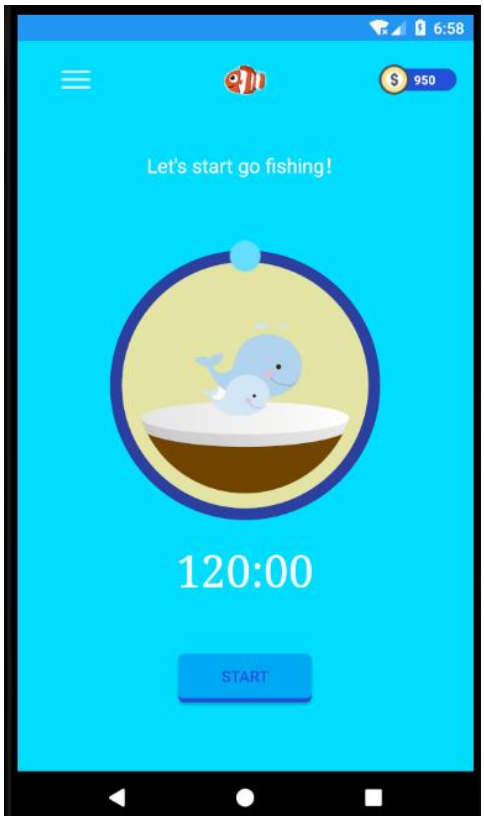




在后续使用过程中，用户会不断积累金币，当用户使用足量的金币在商店中购买对应的鱼类时即可选择对应的鱼类，可选择的情况也如下图所示，用户可以选择丰富多样的鱼类：







2. 滚动拖动时间条的实现与判断

拖动时间条单独使用了 src/java/main 下的 view 包中 processbar 实现。

首先我们设定一个圆，通过初始化，配置圆以及可拖动原点的属性与位置。然后我们需要考虑用户的拖动问题，包括用户拖动到圆圈外的处理情况。最后，我们可以通过计算用户触点与圆中心的距离、角度来推断出可拖动原点的所在位置。主要函数部分截图如下：

```
123 @   public boolean onTouchEvent(MotionEvent event) {...}
153     private int centerX, centerY;
154     private int radius;
155     private int paddingOuterThumb;
156     @Override
157     protected void onSizeChanged(int width, int height, int oldw, int oldh) {...}
166     // 根据点的位置，更新进度
167     private void updateArc(int x, int y) {...}
201
202     //progress callback
203     public interface progressCallback{...}
206     public void setProgressCallback(progressCallback progressCallback){...}
209
210     private int minValidateTouchArcRadius; // 最小有效点击半径
211     private int maxValidateTouchArcRadius; // 最大有效点击半径
212     // 判断是否按在圆边上
213     private boolean isTouchArc(int x, int y) {...}
220     // 计算某点到圆点的距离
221     private double getTouchRadius(int x, int y) {...}
226     public String getTimeText(int progress) {...}
232     public synchronized int getMax() { return max; }
235     public synchronized void setMax(int max) {...}
241     public synchronized int getProgress() { return progress; }
244     public synchronized void setProgress(int progress) {...}
256     public int getCricleColor() { return roundColor; }
259     public void setCricleColor(int cricleColor) { this.roundColor = cricleColor; }
262     public int getCricleProgressColor() { return roundProgressColor; }
265     public void setCricleProgressColor(int cricleProgressColor) {...}
268     public float getRoundWidth() { return roundWidth; }
```

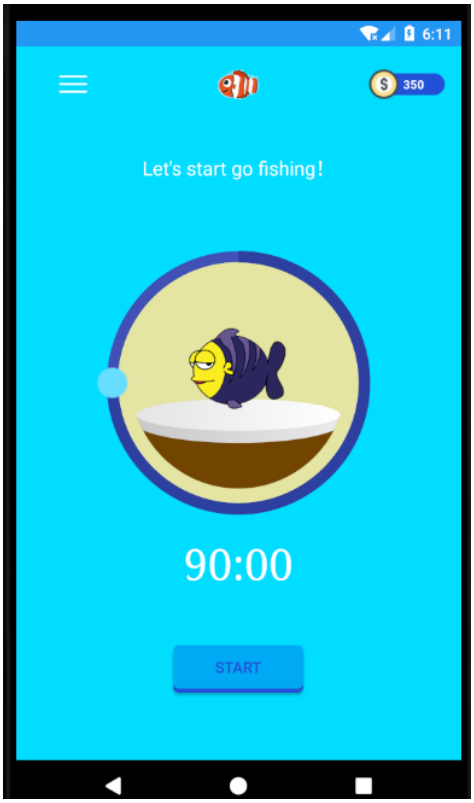
同时，我们还需要将更新返回到 mainActivity 中，来更新我们鱼种的图片。这个判断在 MainActivity 中完成，部分代码截图如下：

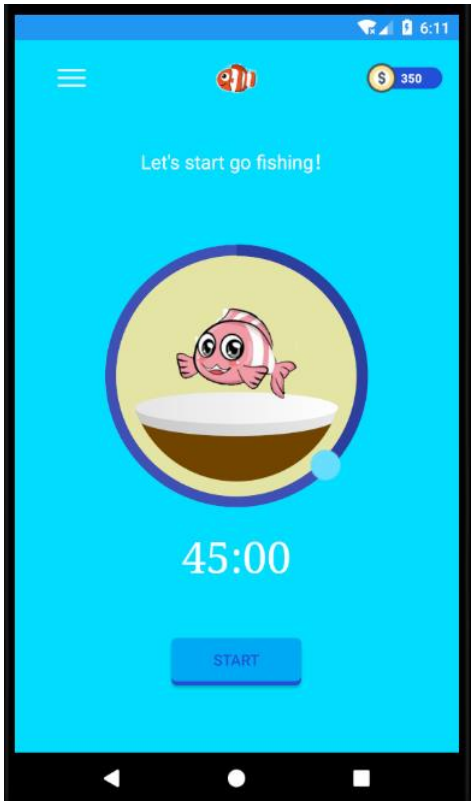
```
//决定显示什么鱼和鱼的生长阶段
private void update(String fishName, int progress){

    int minute = progress/60;
    if(minute<1)
        level = 1;
    else if(minute<5)
        level = 2;
    else if(minute<10)
        level = 3;
    else if(minute<60)
        level = 4;
    else if(minute<90)
        level = 5;
    else if(minute<120)
        level = 6;
    else
        level = 7;
    fishName = fishName + level;
```

```
        if(!flagLock){
            minute = progress/60;
            mTimeText.setText(minute+":00");
        }
        switch(fishName){
            case "starBurst1": mFish.setImageResource(R.drawable.starbu
rst1); break;
            case "starBurst2": mFish.setImageResource(R.drawable.starbu
rst2); break;
            /* …… */
        }
    }
```

这样我们就实现了主界面的绘制与可拖动的原点，并更新鱼类数据。实际使用截图如下：

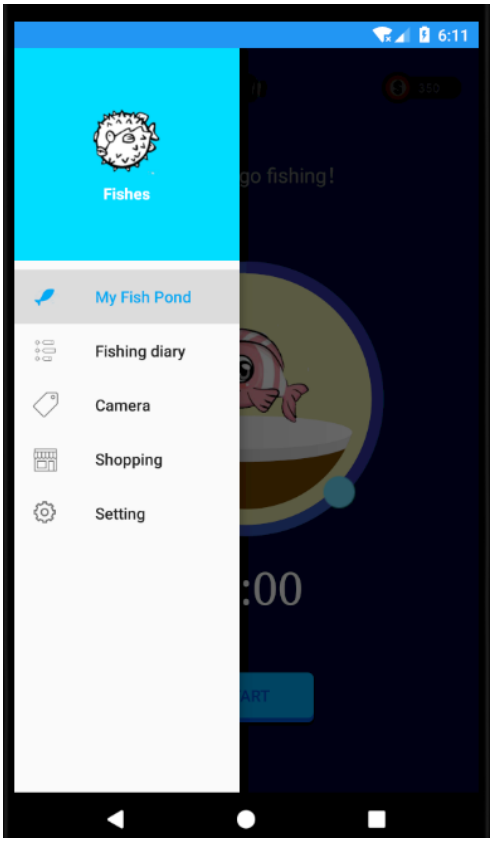




3. 左侧控制栏操作

左侧控制栏包括主界面 My Fish Pond、历史记录 Fishing Diary、相机功能 Camera、商店功能 Shopping 以及设置 Setting。

遗憾的是，由于时间不足，我们只完成了前四项的设定，最后的 Setting 并没有完成，点击后并没有任何效果。



下述是 MainActivity 中对侧栏选择的判断代码的简略展示：

```
switch (item.getItemId()) {
case R.id.menu_fish:
    break;
case R.id.menu_timeline:
    Intent intent = new Intent(MainActivity.this, TimeActivity.class);
    startActivity(intent);
    break;
case R.id.menu_tag:
    Intent intent2 = new Intent(MainActivity.this, PhotoActivity.class)
;
    startActivity(intent2);
    break;
case R.id.menu_store:
    /* ..... */
    break;
case R.id.menu_setting:
    break;
}
mDrawerLayout.closeDrawers();
return true;
```

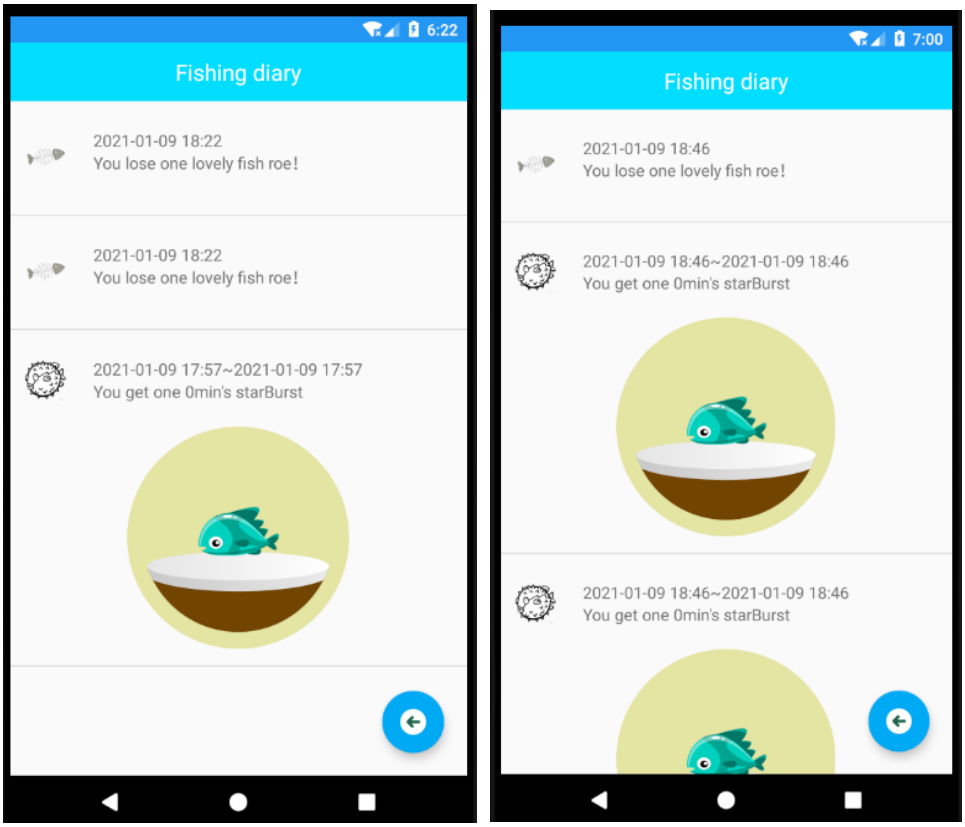
My Fish Pond 功能是返回主界面，并没有具体可实现的。

Fishing Diary 的功能是实现浏览历史记录。正如上述代码所见，我们采用了一个单独的 TimeActivity 来实现这个部分，其中最重要的函数即关于数据库的查询与现实部分如下所示：正如下面代码可见，我们采用上文中实现的数据库来查询具体的数据，并将之存放在一个 List 中。同时，我们按照时间顺序寻找数据，并每次将查询结果存放在 list 的队列首，这样我们最后显示的结果就是按照时间倒叙的，也就是最后插入的数据展示在最上方，符合实际应用场景。

```
private List<Note> loadNotesFromDatabase() {
        if (database == null) {//判断数据库存不存在
            return Collections.emptyList();
        }
        List<Note> result = new LinkedList<>();
        Cursor cursor = null;
        try {
            cursor = database.query(TodoContract.TodoNote.TABLE_NAME, n
ull,
                    null, null,
                    null, null,null);
            while (cursor.moveToNext()) {
                long id = cursor.getLong(cursor.getColumnIndex(TodoCont
ract.TodoNote._ID));
                String caption = cursor.getString(cursor.getColumnIndex
(TodoContract.TodoNote.COLUMN_CAPTION));
                String intState = cursor.getString(cursor.getColumnInde
x(TodoContract.TodoNote.COLUMN_STATE));
                String scheduled = cursor.getString(cursor.getColumnInd
ex(TodoContract.TodoNote.COLUMN_SCHEDULED));
                String time = cursor.getString(cursor.getColumnIndex(To
doContract.TodoNote.COLUMN_TIME));
                String deadline = cursor.getString(cursor.getColumnInde
x(TodoContract.TodoNote.COLUMN_DEADLINE));
                Note note = new Note(id);
                note.setCaption(caption);
                note.setState(intState);
                note.setDeadline(deadline);
                note.setScheduled(scheduled);
                note.setTime(time);
                //Log.d("data",scheduled+","+caption+","+deadline+","+i
ntState);
                result.add(0,note);
            }
        } finally {
            if (cursor != null) {
                cursor.close();
            }
```

```
        }
        return result;
    }
```

最终显示结果如下所示：

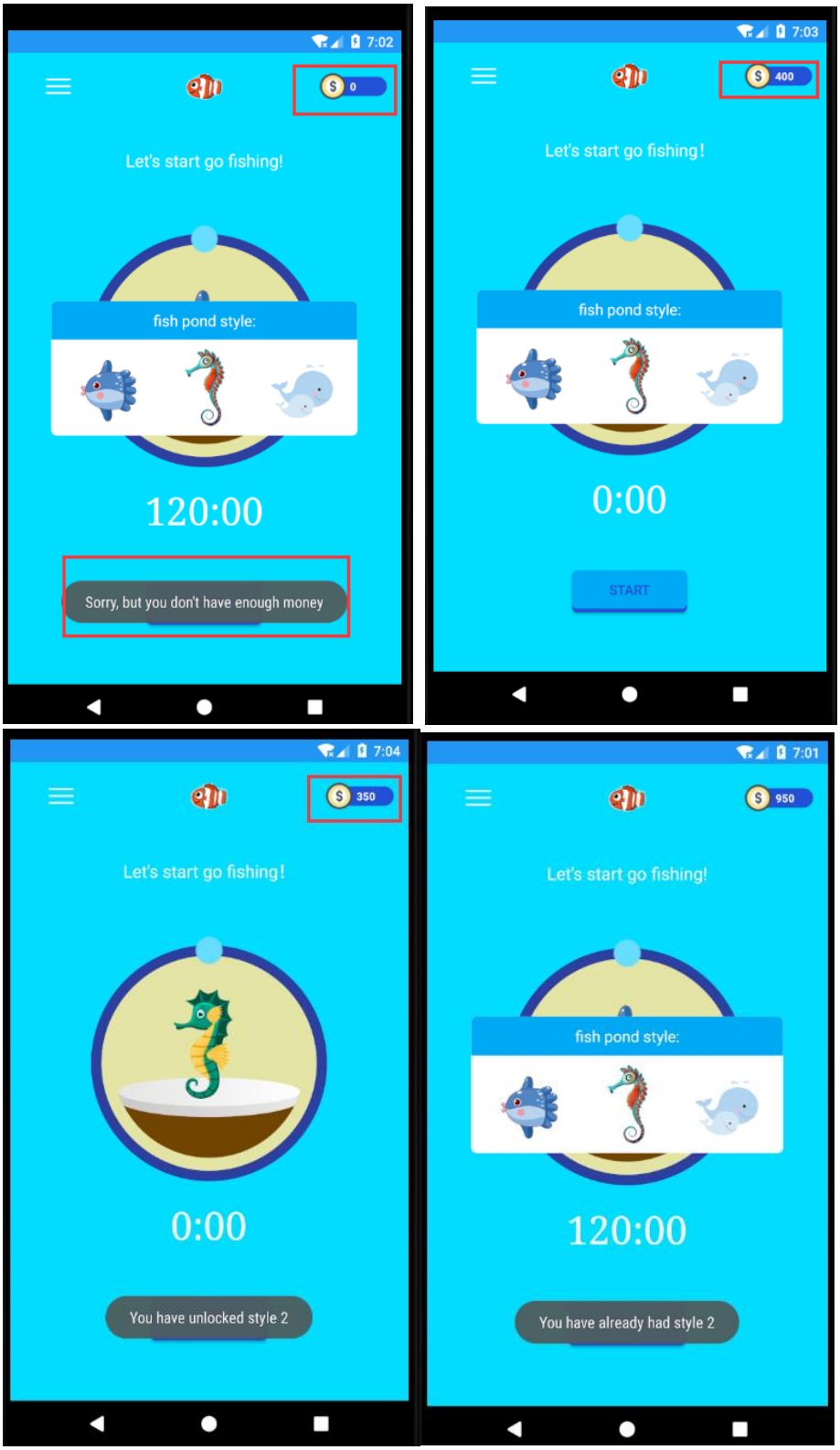


Store 是商店功能，用户需要在商店中购买对应的鱼类，这样才可以在上文中的主界面中实现对鱼类的选择。由于 Store 的实现部分与主界面中选择鱼类的部分类似，所以我们没有再单独建立一个 Activity 来实现具体的商店对象。主要的思路仿照 main 中选择树种即可通过判断全局的 style2、style3 以及金钱数量来购买。具体代码如下：

```
case R.id.menu_store:
mChooseDialog = new ChooseDialog(MainActivity.this);
mChooseDialog.setStarBurstOnClickListener(new ChooseDialog.starBurstOnC
lickListener() {
    @Override
        public void onClick() {
        Toast.makeText(MainActivity.this,
            "You have already had this default style", Toast.LENGTH_SHO
RT).show();
    }
});
mChooseDialog.setTimeOnClickListener(new ChooseDialog.timeOnClickListen
er() {
    @Override
        public void onClick() {
```

```
        if (style2 == 1) {
            Toast.makeText(MainActivity.this,
                "You have already had style 2", Toast.LENGTH_SHORT).sho
w();
        }
        else {
            SharedPreferences preferences1 = getSharedPreferences("user
", Context.MODE_PRIVATE);
            coinNum = preferences1.getInt("coinSum", 0);
            if (coinNum > price2) {
                //buy
                coinNum = coinNum - price2;
                Editor editor1 = preferences1.edit();
                editor1.putInt("coinSum", coinNum);
                style2 = 1;
                editor1.putInt("style2", style2);
                editor1.commit();
                mCoinSum.setText(coinNum + "");
                Toast.makeText(MainActivity.this,
                    "You have unlocked style 2", Toast.LENGTH_SHORT).sh
ow();
                choose = "time";
                mChooseDialog.dismiss();
                update("time", progress);
            }
            else {
                Toast.makeText(MainActivity.this,
                    "Sorry, but you don't have enough money", Toast.LEN
GTH_SHORT).show();
            }
        }
    }
});
mChooseDialog.setStarClickListener(new ChooseDialog.starOnClickListener
() {
    @Override
        public void onClick() {
        if (style3 == 1) {
            /* …… */
        }
    }
});
mChooseDialog.show();
break;
```

商店的使用实际情况如下所示：分别演示了金钱不够、正常购买、购买后自动返回主界面并切换所选 style、重复购买的情况。



4. 全局变量 style、金钱数量 coinNum 的存储

紧接上文，我们来简单一下我们的全局变量的存取问题。

我们的全局变量并不需要保存在上文建立的数据库中，这样太麻烦了，也非常的冗余。相对应的，我们建立一个 json 文件（保存在 src/main/res/raw 下），使用安卓的

内置类 SharedPreferences 以及 Editor 来完成数据的读写。下文展示了我们对 coinSum 的读写，其他的全局变量类似，我们只需要在打开程序时读入对应的数据，并维护这个 json 文件即可。值得注意的一件事是我们保存的是 string 类型，需要经过数据转换转变成我们所需要的 int 类型

```
SharedPreferences preferences = getSharedPreferences("user", Context.MO
DE_PRIVATE);
Editor editor = preferences.edit();
//every minute, you can get a coin
//        coinNum += growProgress/60;
//for test
coinNum += growProgress * 100;
editor.putInt("coinSum", coinNum);
editor.commit();
mCoinSum.setText(coinNum + "");
```

5. 检测用户当前使用 app 与专注提醒

检测用户当前使用的 app 主要通过两个全局的变量：flagDanger 与 timelimit 实现。flagDanger 为 true 时代表用户离开了 Go Fishing APP，timelimit 代表我们设置的可允许离开的时常。当检测到用户离开 APP，timelimit 自减，当自减为 0 时则进入鱼类死亡处理。

下面是对当前 APP 的检测与 flagDanger 的维护，我们只需要判断当前的 APP 包是不是我们的 APP 包以及 tmp 指针即可。值得一提的是，我们的 flagDanger 在每次返回 APP 时都会执行 else 中的程序，也就是复原我们的 timeLimit，这样的意义在于让用户每次离开都有 5s 的返回时间，防止误触：

```
if (!packageName.equals(APP_PACKAGE_NAME) && !temp.equals(APP_PACKAGE_N
AME) && temp != null) {
    if (!flagDanger)
    {
        Toast.makeText(MainActivity.this,
            "you have " + String.valueOf(timeLimit) + " seconds left to
 go back to your fish pond", Toast.LENGTH_SHORT).show();
    }
    flagDanger = true;
}
else {
    flagDanger = false;
    timeLimit = 5;
}
```

下面是对 timelimit 的检测与维护：

```
public void createLockTimer() {
    timer = new CountDownTimer(1000 * timerProgress, 1000){
    @Override
    public void onTick(long l) {
        Log.d("progress:",timerProgress + "");
        timerProgress--;       //倒计时--
```

```
        growProgress++;         //lock 进度++
        Log.d("countdown:",String.valueOf(timeLimit));

        if (flagDanger)           //进入第三方应用
            timeLimit--;        //死亡倒计时
        if (timeLimit == 0) {     //死亡
            flagLock = false;
            mProcessBar.setflagLock(flagLock);
            mProcessBar.invalidate();
            mCancelButton.setVisibility(View.GONE);
            mStartButton.setVisibility(View.VISIBLE);
            mStartButton2.setVisibility(View.VISIBLE);
            setTimeText(progress);
            update(choose, progress);

            Toast.makeText(MainActivity.this,
                    "Your fish died ", Toast.LENGTH_SHORT).show();
            Log.d("debug:","time out");
            if (timer != null)
                timer.cancel();
            if (appListener != null)
                appListener.cancel();

            flagDanger = false;
            mBase.setClickable(true);
            mHintText.setText("Let's start go fishing! ");
            //其余交给 onResume()处理

            boolean succeed = saveNote2Database(choose + level,sche, "0
","0");
            if (succeed) {
                Toast.makeText(MainActivity.this,
                        "Note added", Toast.LENGTH_SHORT).show();
                setResult(Activity.RESULT_OK);
            }
    else {
    Toast.makeText(MainActivity.this,
         "Error", Toast.LENGTH_SHORT).show();
    }
}
```

我们设置的 timeLimit 为 5s，当用户离开 APP 时会出现警告，超过时常会显示树苗死亡的信息。实际操作与显示如下图所示：

6. 程序完成与主界面刷新

程序完成包括两种情况：成功与未成功，下图展示了最终的两种结果：

在此我们主要讲述完成时的情况，鱼类死亡时 MainActivity 的动作已在上文进行阐

述（即 timelimit 自减为 0，进入单独分支，另外的鱼类死亡我们单独建立了类，将会在下文中讲到）。下文是对进行完成后的程序：首先重设 main 介面的变量，输出提示信息“CountDown”以及数据库的调用“NoteAdded”，之后完成对金币的结算，返回主界面即可。

```
//完成钓鱼
public void onFinish() {
    mBase.setClickable(true);
    mHintText.setText("Let's start go fishing! ");
    flagLock = false;
    mProcessBar.setflagLock(flagLock);
    mProcessBar.invalidate();
    mCancelButton.setVisibility(View.GONE);
    mStartButton.setVisibility(View.VISIBLE);
    mStartButton2.setVisibility(View.VISIBLE);
    setTimeText(progress);
    update(choose, progress);
    Toast.makeText(MainActivity.this, "CountDownFinish", Toast.LENGTH_S
HORT).show();

    SharedPreferences preferences = getSharedPreferences("user", Contex
t.MODE_PRIVATE);
    Editor editor = preferences.edit();
    //every minute, you can get a coin
    //             coinNum += growProgress/60;
    //for test
    coinNum += growProgress * 100;
    editor.putInt("coinSum", coinNum);
    editor.commit();
    mCoinSum.setText(coinNum + "");

    String time = progress / 60 + "min";
    boolean succeed = saveNote2Database(choose + level, sche, time, "1"
);
    if (succeed) {
        Toast.makeText(MainActivity.this,
            "Note added", Toast.LENGTH_SHORT).show();
        setResult(Activity.RESULT_OK);
    }
    else {
        Toast.makeText(MainActivity.this,
            "Error", Toast.LENGTH_SHORT).show();
    }
}
```

3.5 鱼死亡时的处理

鱼类死亡我们单独封装成了 FishDeadActivity，每次当鱼类死亡时都会调用这个类的 onCreate，完成对后续的操作。

值得一提的是，我们在鱼类死亡时也会根据已专注的时常来计算获得的金币，所以下文的代码种我们需要实现对 coin 数据的维护。之外，我们跳转至鱼类死亡的介面，显示对应的 layout 配置，并监听“返回”按钮，当按钮按下时，我们返回到主界面，并保留原先配置。最后，通过 MainActivity 传递的任务信息，将信息输出到数据库中，完成鱼类死亡的程序。

下文是重要部分的源代码：

```
protected void onCreate(@Nullable Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_fishdead);

    SharedPreferences preferences = getSharedPreferences("user", Contex
t.MODE_PRIVATE);
    coinNum = preferences.getInt("coinSum", 0);

    coinSum = findViewById(R.id.coinSum);
    coinSum.setText(coinNum + "");
    fish = findViewById(R.id.fish);
    backButton = findViewById(R.id.backButton);
    backButton.setOnClickListener(new View.OnClickListener() {
        @Override
            public void onClick(View view) {
            finish();
        }
    });

    Intent intent = getIntent();
    choose = intent.getStringExtra("choose");

    switch (choose) {
    case "starBurst": fish.setImageResource(R.drawable.starburstdead);
break;
    case "time": fish.setImageResource(R.drawable.timedead); break;
    case "star": fish.setImageResource(R.drawable.stardead); break;
    }

    Log.d("debug:", "fishDead onCreate");
}
```

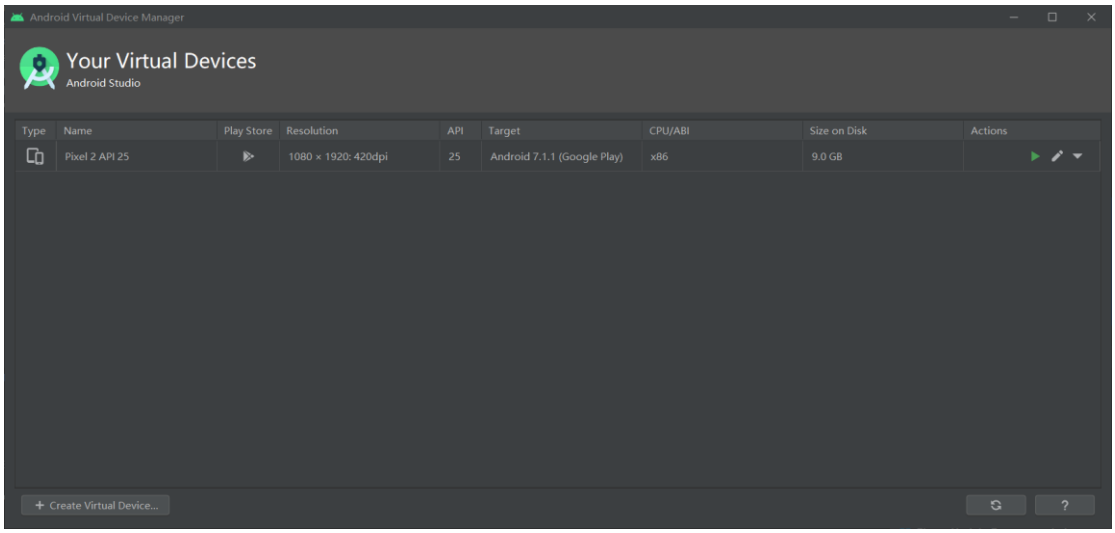
四．开发困难及解决方案

4.1 设置应用权限失败

在检测 APP 时，我们最先使用的安卓模拟机采用了 api 29 的配置，程序会正常的运行，并弹窗显示需要申请对应的权限。

但是在 api29 安卓机中我们并不能获取对应的权限，原因在于 setting 栏并没有上文中提到的“允许应用获取使用情况”的权限设置，这时程序无法正常执行下去，会一直停在获取权限的弹窗下。

解决办法是，我们新建一个较低版本的安卓虚拟机，如下图所示，使用 api 25 即可。



4.2 开发时出现奇怪的 bug

在开发商店功能时，我们遇到了奇怪的情况，一样的代码在两个同学的安卓虚拟机下运行的结果不一致，在检测金币数量、是否已购买某种鱼类时，一个同学的运行结果是无论如何都可以在没有购买权限，或者没有购买金币的情况下都可以在主界面选择切换改鱼种。

在查看了相关的代码之后，我们仍没有发现问题的所在。并且由于预想与实际编写的偏差，一开始的代码编写的也较为混乱，我们只好重构了相关部分的代码，理清逻辑重新编写后，问题消失……我们仍未知道那天所编写的代码 bug 出在了哪里

4.3 项目开发难点

项目的难点实现主要集中在工作量上，由于我们的程序功能较多，并且使用了大量的贴图，所以从网上搜集资源、创建并修改 layout 属性、绘制介面等等就显得非常的繁杂。

除此以外，我们在开发过程中还遇到了一个很奇怪的“bug”，我们的金币显示从正常的数字显示变成了“--”，这时我们最开始认为是我们修改了数据的存储格式或在某些地方删除了代码导致初始化不慎导致的。结果最后经过仔细研究发现于代码没有太大的关系，是在

开发过程中，通过 Android Studio 的 IDE 下显示的 layout 误触，导致直接拖动字符框，导致字符框超出显示范围，偏离显示区域，也就是之前的“--”实际上是数字的一部分。这种低级错误缺也给我们带来了不小的心理负担与工作量。

五．分工

A：完成 APP 中的素材收集、图片整理等，绘制介面。负责各个界面的设计，并处理数据的读写情况，完成数据库的内嵌。同时完成报告的第三、四章内容。
B：实现计时器，完成监控 APP 进程与使用情况的任务，限制切换到其他 app。实现商店功能。同时负责报告的一、二章内容，以及讲解视频的录制。

六．总结

本次项目的开发比预想中要困难一些，虽然平时的课后实验我们都完成了，但是自己真正独立去编写程序的时候还是遇到了很多问题。

而且虽然项目的难度一般，但是整体来讲工作量不小，既要编写后端的逻辑，又要设计前端的界面，整体来讲既考验我们的设计思路，也考验我们的团队协作能力。
而且我们的开发进度相较于预期有些落后，我们还有很多扩展的功能没有添加，比如社交功能，比如录制视频并上传，分享自己的专注时刻（我们已经实现了视频的录制，但是分享和社交的功能没有实现）等等。

总的来看，这个项目达到了我们的预期标准，但是还有继续开发的空间。我们预计将来会对这一项目进行完善，实现更多的功能