适配器模式

- 适配器模式
  - 扩展坞（程序一）
    - 一、应用场景与案例描述
    - 二、案例分析与解决问题
    - 三、各种角色描述与UML
      - 1.目标（Target）
      - 2.被适配者
      - 3.适配器
    - 四、附录
      - 源代码
      - 运行截图
  - 菜单栏（程序二）
    - 一 应用场景与案例描述
    - 二 案例分析与解决问题
    - 三 各种角色描述与UML
      - 1 目标（Target）
      - 2 被适配者
      - 3 适配器
    - 四 附录
      - 4.1源代码
        - CircleMenuLayout.java
        - MainActivity.java
        - activity_main.xml
        - circle_menu_item.xml
      - 4.2运行截图

扩展坞（程序一）

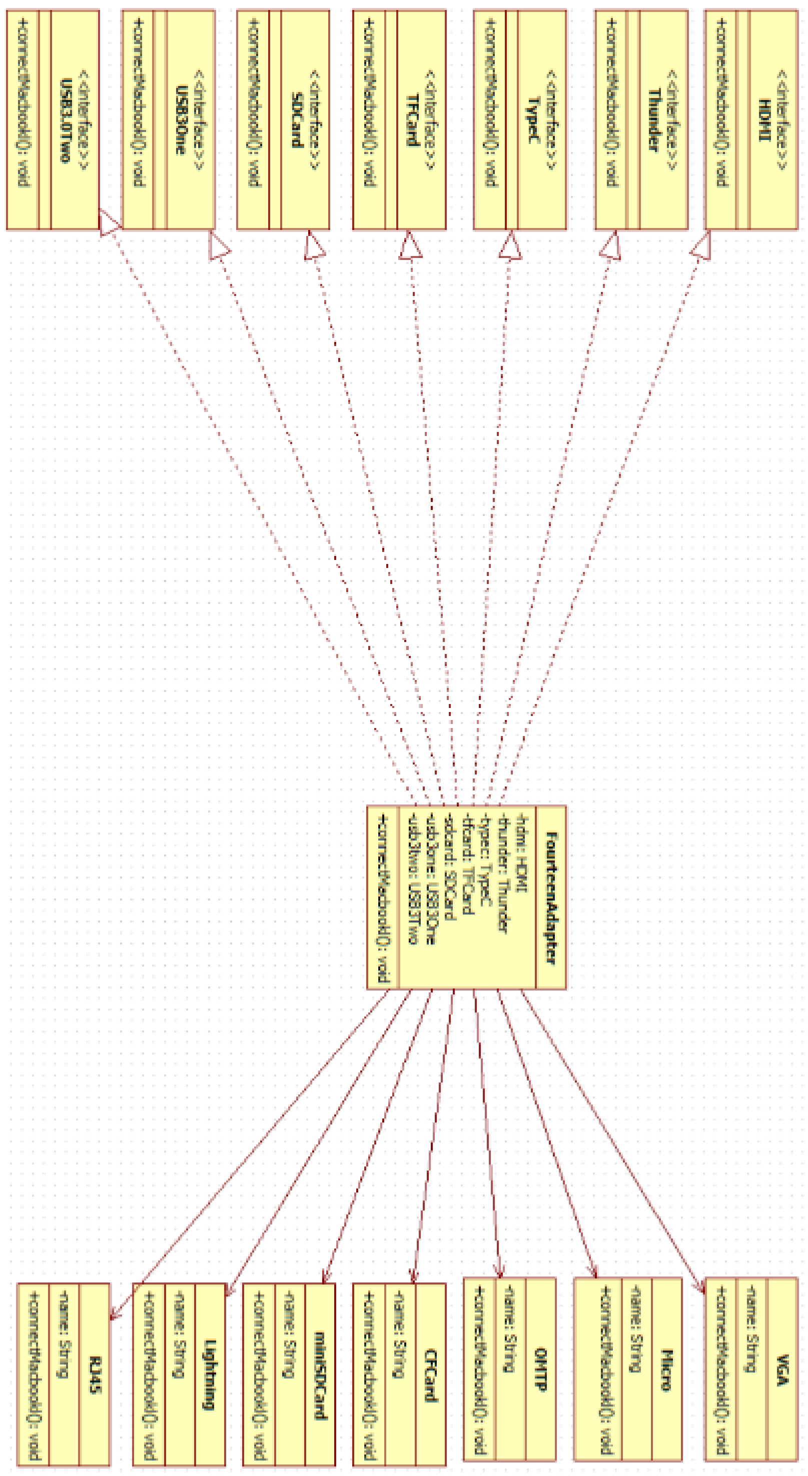
一、应用场景与案例描述

新款Macbook那就是只有一个USB-C接口，对于需要同时进行充电、传输数据或连接打印机、投影仪等外设的用户来说实在不方便。则需要使用扩展坞进行接口的转换。则使用适配器的原理对接口进行转换。

二、案例分析与解决问题

在上面的场景中，接口的转换使用了适配器的模式。 适配器的适配程度：由于扩展坞的目标接口中的方法只有两个雷神3的Type-C接口，而被适配的接口有HDMI接口，雷电3接口，Type-c 3接口，TF/SD卡槽，两个USB3.0接口，所以为不完全适配。 实现原理：将接口协议A、B进行统一，将所有端的协议全部替换成新协议，使用一套新协议；

三、各种角色描述与UML



1.目标（Target）

LCD（显示器），Earphone（耳机），Phone（手机），TF（TF卡），SD（SD卡），UsbDisk（U盘），HardDisk（移动硬盘）等设备

2.被适配者

多功能扩展坞的各种接口

3.适配器

TreatyAdapter（多功能扩展坞）

四、附录

源代码

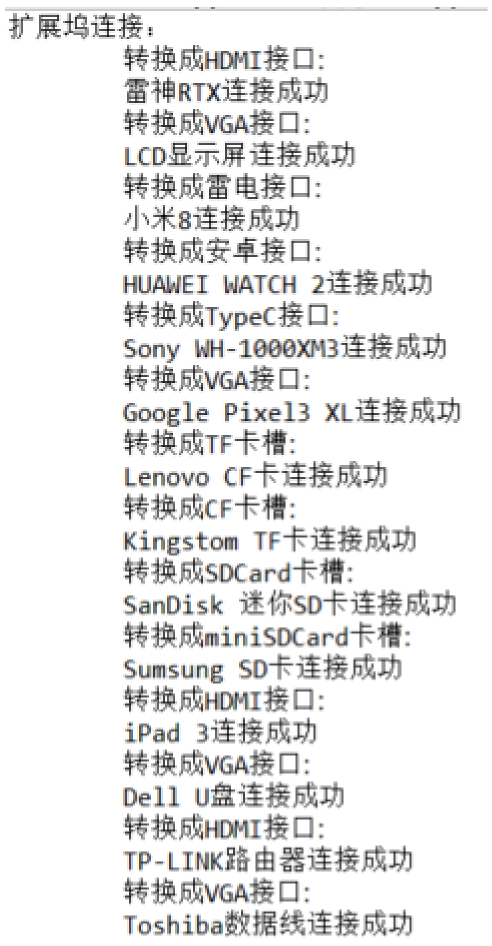
```
//HDMI接口
public interface HDMI{
    public abstract void connectMacbook();
}

//VGA接口
public interface VGA{
    public abstract void connectMacbook();
}

//Thunder接口
public interface Thunder{
    public abstract void connectMacbook();
}

//Micro接口
public interface Micro{
    public abstract void connectMacbook();
}

//TypeC接口
public interface TypeC{
    public abstract void connectMacbook();
}

//OMTP接口
public interface OMTP{
    public abstract void connectMacbook();
}

//TF卡槽
public interface TFCard{
    public abstract void connectMacbook();
}

//CF卡槽
public interface CFCard{
    public abstract void connectMacbook();
}

//SD卡槽
public interface SDCard{
    public abstract void connectMacbook();
}

//miniSD卡槽
public interface miniSDCard{
    public abstract void connectMacbook();
}

//USB3One接口
public interface USB3One{
    public abstract void connectMacbook();
}

//Lightning接口
public interface Lightning{
    public abstract void connectMacbook();
}

//USB3Two接口
public interface USB3Two{
    public abstract void connectMacbook();
}

//RJ45接口
public interface RJ45{
    public abstract void connectMacbook();
}

//十四向适配器模式
public class FourteenAdapter implements HDMI, VGA, Thunder, Micro, TypeC,
OMTP, TFCard, CFCard, SDCard, miniSDCard, USB3One, Lightning, USB3Two,
RJ45{
    HDMI hdmi;
VGA vga;
    Thunder thunder;
    Micro micro;
    TypeC typec;
    OMTP omtp;
    TFCard tfcard;
    CFCard cfcard;
    SDCard sdcard;
    miniSDCard minisdcard;
    USB3One usb3one;
    Lightning lightning;
    USB3Two usb3two;
    RJ45 rj45;
    FourteenAdapter(HDMI hdmi, VGA vga, Thunder thunder, Micro micro,
TypeC typec, OMTP omtp, TFCard tfcard, CFCard cfcard, SDCard sdcard,
miniSDCard minisdcard, USB3One usb3one, Lightning lightning, USB3Two
usb3two, RJ45 rj45){
        this.hdmi=hdmi;
        this.vga=vga;
        this.thunder=thunder;
        this.micro=micro;
        this.typec=typec;
        this.omtp=omtp;
        this.tfcard=tfcard;
        this.cfcard=cfcard;
        this.sdcard=sdcard;
```

```
            this.minisdcard=minisdcard;
            this.usb3one=usb3one;
            this.lightning=lightning;
            this.usb3two=usb3two;
            this.rj45=rj45;
        }
    public void connectMacbook(){
        if(this instanceof HDMI) {
            System.out.println("\t转换成HDMI接口:");
            vga.connectMacbook();
        }
        if(this instanceof VGA) {
            System.out.println("\t转换成VGA接口:");
            hdmi.connectMacbook()
        }
        if(this instanceof Thunder) {
            System.out.println("\t转换成雷电接口:");
            micro.connectMacbook();
        }
        if(this instanceof Micro) {
            System.out.println("\t转换成安卓接口:");
            thunder.connectMacbook();
        }
        if(this instanceof TypeC) {
            System.out.println("\t转换成TypeC接口:");
            omtp.connectMacbook();
        }
        if(this instanceof OMTP) {
            System.out.println("\t转换成VGA接口:");
            typec.connectMacbook();
        }
        if(this instanceof TFCard) {
            System.out.println("\t转换成TF卡槽:");
            cfcard.connectMacbook();
        }
        if(this instanceof CFCard) {
            System.out.println("\t转换成CF卡槽:");
            tfcard.connectMacbook();
        }
        if(this instanceof SDCard) {
            System.out.println("\t转换成SDCard卡槽:");
            minisdcard.connectMacbook();
        }
        if(this instanceof miniSDCard) {
            System.out.println("\t转换成miniSDCard卡槽:");
            sdcard.connectMacbook();
        }
        if(this instanceof USB3One) {
            System.out.println("\t转换成HDMI接口:");
            lightning.connectMacbook();
        }
        if(this instanceof Lightning) {
            System.out.println("\t转换成VGA接口:");
            usb3one.connectMacbook();
        }
        if(this instanceof USB3Two) {
            System.out.println("\t转换成HDMI接口:");
            rj45.connectMacbook();
        }
        if(this instanceof RJ45) {
            System.out.println("\t转换成VGA接口:");
            usb3two.connectMacbook();
        }
      }
}

public class Application{
    public static void main(String args[]){
       HDMI hdmi;
       Thunder thunder;
       TypeC typec;
       TFCard tfcard;
       SDCard sdcard;
       USB3One usb3one;
       USB3Two usb3two;
       LCD lcd=new LCD(); //显示器
       ThundeRobot thunderobot=new ThundeRobot();//主机
       Watch watch=new Watch();//手表
       XiaoMi xiaomi=new XiaoMi();//安卓手机
       Pixel pixel=new Pixel(); //TypeC手机
       Sony sony=new Sony(); //耳机
       TF tf=new TF();  //TF卡
       CF cf=new CF();  //CF卡
       SD sd=new SD();  //SD卡
       miniSD minisd=new miniSD();  //miniSD卡
       UsbDisk usbdisk=new UsbDisk();   //U盘
       iPad ipad=new iPad();    //平板
       HardDisk harddisk=new HardDisk();    //移动硬盘
       WIFI wifi=new WIFI();    //路由
       FourteenAdapter adapter = new FourteenAdapter(lcd, thunderobot,
watch, xiaomi, pixel, sony, tf, cf, sd, minisd, usbdisk, ipad, harddisk,
wifi);
       System.out.println("扩展坞连接: ");
       adapter.connectMacbook();
    }
}
class LCD implements HDMI{
    String name;
    LCD (){
        name = "LCD显示屏";
    }
    LCD (String name){
        this.name = name;
    }
    public void connectMacbook() {
        System.out.println("\t" + name + "连接成功");
    }
}
```

```
class ThundeRobot implements VGA{
    String name;
    ThundeRobot (){
        name = "雷神RTX";
    }
    ThundeRobot (String name){
        this.name = name;
    }
    public void connectMacbook() {
        System.out.println("\t" + name + "连接成功");
    }
}
class Watch implements Thunder{
    String name;
    Watch (){
        name = "HUAWEI WATCH 2";
    }
    Watch (String name){
        this.name = name;
    }
    public void connectMacbook() {
        System.out.println("\t" + name + "连接成功");
    }
}
class XiaoMi implements Micro{
    String name;
    XiaoMi (){
        name = "小米8";
    }
    XiaoMi (String name){
        this.name = name;
    }
    public void connectMacbook() {
        System.out.println("\t" + name + "连接成功");
    }
}
class Pixel implements TypeC{
    String name;
    Pixel (){
        name = "Google Pixel3 XL";
    }
    Pixel (String name){
        this.name = name;
    }
    public void connectMacbook() {
        System.out.println("\t" + name + "连接成功");
    }
}
class Sony implements OMTP{
    String name;
    Sony (){
        name = "Sony WH-1000XM3";
    }
    Sony (String name){
        this.name = name;
    }
    public void connectMacbook() {
        System.out.println("\t" + name + "连接成功");
    }
}
class TF implements TFCard{
    String name;
    TF (){
        name = "Kingstom TF卡";
    }
    TF (String name){
        this.name = name;
    }
    public void connectMacbook() {
        System.out.println("\t" + name + "连接成功");
    }
}
class CF implements CFCard{
    String name;
    CF (){
        name = "Lenovo CF卡";
    }
    CF (String name){
        this.name = name;
    }
    public void connectMacbook() {
        System.out.println("\t" + name + "连接成功");
    }
}
class SD implements SDCard{
    String name;
    SD (){
        name = "Sumsung SD卡";
    }
    SD (String name){
        this.name = name;
    }
    public void connectMacbook() {
        System.out.println("\t" + name + "连接成功");
    }
}
class miniSD implements miniSDCard{
    String name;
    miniSD (){
        name = "SanDisk 迷你SD卡";
    }
    miniSD (String name){
        this.name = name;
    }
    public void connectMacbook() {
        System.out.println("\t" + name + "连接成功");
    }
}
```

```
class UsbDisk implements USB3One{
    String name;
    UsbDisk (){
        name = "Dell U盘";
    }
    UsbDisk (String name){
        this.name = name;
    }
    public void connectMacbook() {
        System.out.println("\t" + name + "连接成功");
    }
}
class iPad implements Lightning{
    String name;
    iPad (){
        name = "iPad 3";
    }
    iPad (String name){
        this.name = name;
    }
    public void connectMacbook() {
        System.out.println("\t" + name + "连接成功");
    }
}
class HardDisk implements USB3Two{
    String name;
    HardDisk (){
        name = "Toshiba数据线";
    }
    HardDisk (String name){
        this.name = name;
    }
    public void connectMacbook() {
        System.out.println("\t" + name + "连接成功");
    }
}
class WIFI implements RJ45{
    String name;
    WIFI (){
        name = "TP-LINK路由器";
    }
    WIFI (String name){
        this.name = name;
    }
    public void connectMacbook() {
        System.out.println("\t" + name + "连接成功");
    }
}
```

运行截图



```
扩展坞连接:
        转换成HDMI接口:
        雷神RTX连接成功
        转换成VGA接口:
        LCD显示屏连接成功
        转换成雷电接口:
        小米8连接成功
        转换成安卓接口:
        HUAWEI WATCH 2连接成功
        转换成TypeC接口:
        Sony WH-1000XM3连接成功
        转换成VGA接口:
        Google Pixel3 XL连接成功
        转换成TF卡槽:
        Lenovo CF卡连接成功
        转换成CF卡槽:
        Kingstom TF卡连接成功
        转换成SDCard卡槽:
        SanDisk 迷你SD卡连接成功
        转换成miniSDCard卡槽:
        Sumsung SD卡连接成功
        转换成HDMI接口:
        iPad 3连接成功
        转换成VGA接口:
        Dell U盘连接成功
        转换成HDMI接口:
        TP-LINK路由器连接成功
        转换成VGA接口:
        Toshiba数据线连接成功
```

菜单栏（程序二）

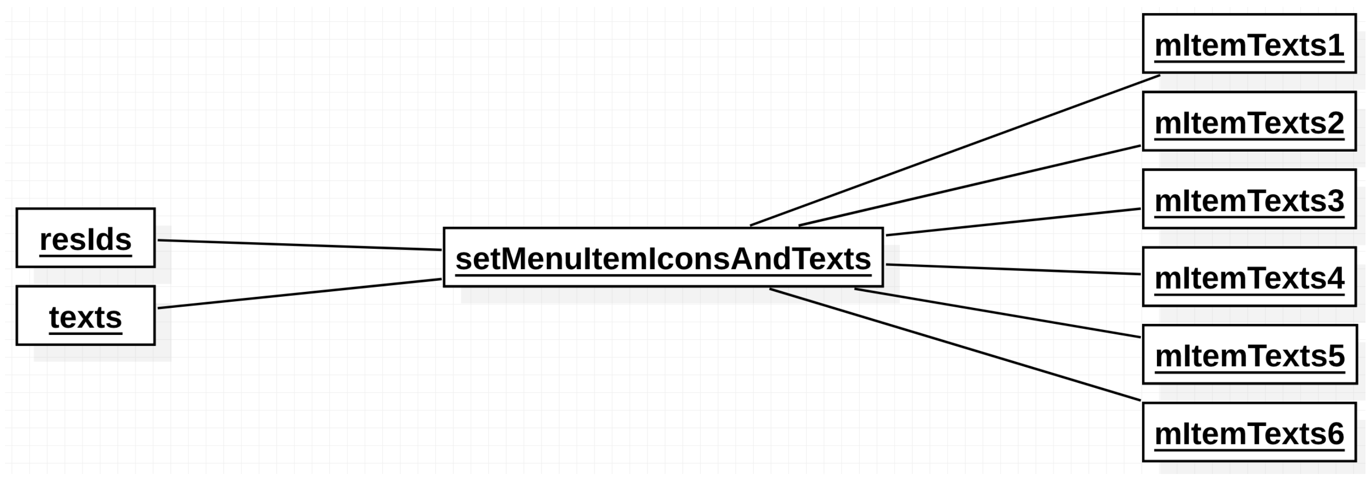
一 应用场景与案例描述

现在人们的生活离不开App，而稍微复杂一点的App都会有一个最基础的组件-菜单栏，菜单栏各式各样，而现在的菜单栏按钮文本和图片至少设置其一，而现在很多菜单项都是两者兼具。使用适配器将图片和文字进行适配。

二 案例分析与解决问题

在上面的场景中，菜单项的转换使用了适配器的模式。 适配器的适配程度：同时拥有图片和文字，为完全适配。 实现原理：将函数逻辑进行统一，将所有函数全部替换成新函数；

三 各种角色描述与UML



1 目标（Target）

mItemImgs(安全中心,特色服务, 投资理财,转账汇款, 我的账户, 信用卡),resIds（图片）， texts（文本）

2 被适配者

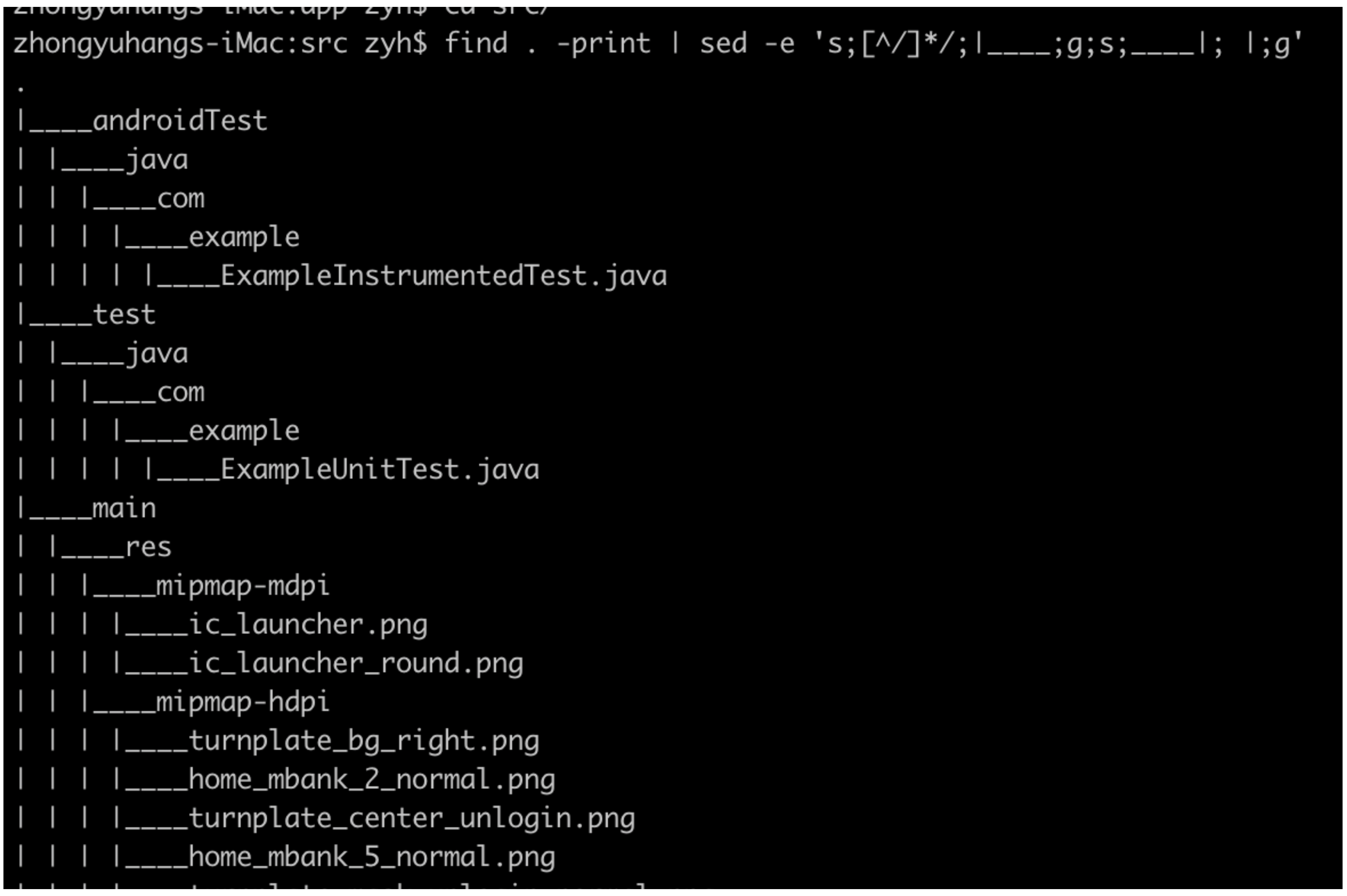
多功能菜单栏函数

3 适配器

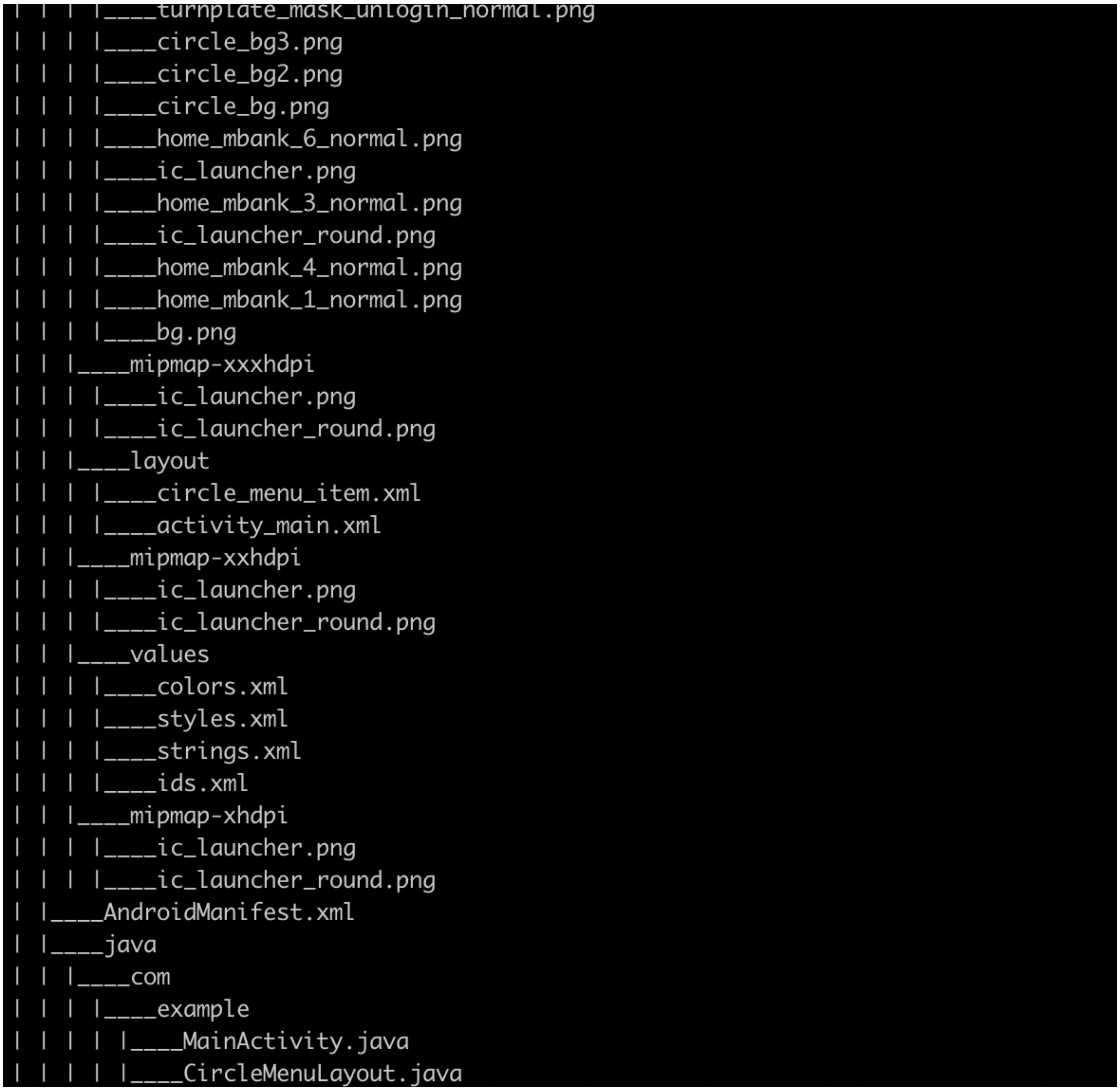
etMenuItemIconsAndTexts（多功能菜单栏）

四 附录

4.1源代码



```
zhongyuhangs-iMac:app zyh$ cd src/
zhongyuhangs-iMac:src zyh$ find . -print | sed -e 's;[^/]*/;|____;g;s;____|; |;g'
.
|____androidTest
| |____java
| | |____com
| | | |____example
| | | | |____ExampleInstrumentedTest.java
|____test
| |____java
| | |____com
| | | |____example
| | | | |____ExampleUnitTest.java
|____main
| |____res
| | |____mipmap-mdpi
| | | |____ic_launcher.png
| | | |____ic_launcher_round.png
| | |____mipmap-hdpi
| | | |____turnplate_bg_right.png
| | | |____home_mbank_2_normal.png
| | | |____turnplate_center_unlogin.png
| | | |____home_mbank_5_normal.png
```



```
| | | |____turnplate_mask_unlogin_normal.png
| | | |____circle_bg3.png
| | | |____circle_bg2.png
| | | |____circle_bg.png
| | | |____home_mbank_6_normal.png
| | | |____ic_launcher.png
| | | |____home_mbank_3_normal.png
| | | |____ic_launcher_round.png
| | | |____home_mbank_4_normal.png
| | | |____home_mbank_1_normal.png
| | | |____bg.png
| | |____mipmap-xxxhdpi
| | | |____ic_launcher.png
| | | |____ic_launcher_round.png
| | |____layout
| | | |____circle_menu_item.xml
| | | |____activity_main.xml
| | |____mipmap-xxhdpi
| | | |____ic_launcher.png
| | | |____ic_launcher_round.png
| | |____values
| | | |____colors.xml
| | | |____styles.xml
| | | |____strings.xml
| | | |____ids.xml
| | |____mipmap-xhdpi
| | | |____ic_launcher.png
| | | |____ic_launcher_round.png
| |____AndroidManifest.xml
| |____java
| | |____com
| | | |____example
| | | | |____MainActivity.java
| | | | |____CircleMenuLayout.java
```

CircleMenuLayout.java

```java
package com.example;

import android.content.Context;
import android.util.AttributeSet;
import android.util.DisplayMetrics;
import android.view.LayoutInflater;
import android.view.MotionEvent;
import android.view.View;
import android.view.ViewGroup;
import android.view.WindowManager;
import android.widget.ImageView;
import android.widget.TextView;

/**
```

```
 * Package: example
 * Created by zyh
 * on 2019/5/26
 */

public class CircleMenuLayout extends ViewGroup {

    //变量
    private int mRadius;//直径
    private static final float RADIO_DEFAULT_CHILD_DIMENSION = 1 / 4f;//默
认child item尺寸
    private static final float RADIO__PADDING_LAYOUT = 1 / 12f;  //内边距
    private float mPadding;
    private double mStartAngle = 0;//开始角度

    private String[] mItemTexts;//菜单项文本
    private int[] mItemImgs;//菜单项图片
    private int mMenuItemCount;
    /**
     * MenuItem的点击事件接口
     */
    private OnMenuItemClickListener mOnItemClickListener;
    /**
     * 检测按下到抬起时旋转的角度
     */
    private float mTmpAngle;
    /**
     * 检测按下到抬起时使用的时间
     */
    private long mDownTime;

    /**
     * 判断是否正在自动滚动
     */
    private boolean isFling;

    /**
     * 当每秒移动角度达到该值时，认为是快速移动
     */
    private static final int FLINGABLE_VALUE = 300;
    private int mFlingableValue = FLINGABLE_VALUE;
    /**
     * 如果移动角度达到该值，则屏蔽点击
     */
    private static final int NOCLICK_VALUE = 3;

    private int mMenuItemLayoutId = R.layout.circle_menu_item;

    public CircleMenuLayout(Context context) {
        super(context);
    }

    public CircleMenuLayout(Context context, AttributeSet attrs) {
        super(context, attrs);
        // 无视padding
        setPadding(0, 0, 0, 0);
    }

    public CircleMenuLayout(Context context, AttributeSet attrs, int
defStyleAttr) {
        super(context, attrs, defStyleAttr);
    }

    public CircleMenuLayout(Context context, AttributeSet attrs, int
defStyleAttr, int defStyleRes) {
        super(context, attrs, defStyleAttr, defStyleRes);
    }


    @Override
    protected void onMeasure(int widthMeasureSpec, int heightMeasureSpec)
{
        //测量布局
        measureLayoutView(widthMeasureSpec, heightMeasureSpec);
        //测量子控件
        measureChildViews();
    }

    private void measureLayoutView(int widthMeasureSpec, int
heightMeasureSpec) {
        int resWidth = 0;
        int resHeight = 0;

        //根据传入参数，获得测量值和模式
        int width = MeasureSpec.getSize(widthMeasureSpec);
        int widthMode = MeasureSpec.getMode(widthMeasureSpec);

        int height = MeasureSpec.getSize(heightMeasureSpec);
        int heightMode = MeasureSpec.getMode(heightMeasureSpec);

        //是否设置精确值
        if (widthMode != MeasureSpec.EXACTLY || heightMode !=
MeasureSpec.EXACTLY) {
            resWidth = getSuggestedMinimumWidth();
            resWidth = (resWidth == 0 ? getDefaultWidth() : resWidth);

            resHeight = getSuggestedMinimumHeight();
            resHeight = (resHeight == 0 ? getDefaultWidth() : resHeight);
        } else {
            resWidth = resHeight = Math.min(width, height);
        }
        setMeasuredDimension(resWidth, resHeight);
    }

    /**
     * 获得默认该layout的尺寸
     *
     * @return
     */
    private int getDefaultWidth() {
        WindowManager wm = (WindowManager) getContext().getSystemService(
                Context.WINDOW_SERVICE);
        DisplayMetrics outMetrics = new DisplayMetrics();
        wm.getDefaultDisplay().getMetrics(outMetrics);
        return Math.min(outMetrics.widthPixels, outMetrics.heightPixels);
    }

    private void measureChildViews() {
        mRadius = Math.max(getMeasuredWidth(), getMeasuredHeight());//获取半
径
        final int count = getChildCount();//获取menu item个数
        int childSize = (int) (mRadius *
RADIO_DEFAULT_CHILD_DIMENSION);//menu item 尺寸
        int childMode = MeasureSpec.EXACTLY;

        for (int i = 0; i < count; i++) {
            final View child = getChildAt(i);
            if (child.getVisibility() == GONE) {
                continue;
            }

            //计算menu item 的尺寸，模式，去对item测量
            int makeMeasureSpec = -1;
            makeMeasureSpec = MeasureSpec.makeMeasureSpec(childSize,
childMode);
            child.measure(makeMeasureSpec, makeMeasureSpec);
        }
        mPadding = RADIO__PADDING_LAYOUT * mRadius;
    }

    //设置menu item监听
    public interface OnMenuItemClickListener {
        void itemClick(View view, int position);

        void itemCenterClick(View view);
    }

    public void setOnItemclickListener(OnMenuItemClickListener listener) {
        this.mOnItemClickListener = listener;
    }

    /**
     * 布局view item的位置
     *
     * @param changed
     * @param l
     * @param t
     * @param r
     * @param b
     */
    @Override
    protected void onLayout(boolean changed, int l, int t, int r, int b) {
        int layoutRadius = mRadius;
        final int childCount = getChildCount();
        int left, top;
        int itemwidth = (int) (layoutRadius *
RADIO_DEFAULT_CHILD_DIMENSION); //item尺寸
        // 根据menu item的个数，计算角度
        float angleDelay = 360 / (getChildCount() - 1);
        //遍历布局item
        for (int i = 0; i < childCount; i++) {
            final View child = getChildAt(i);

            if (child.getId() == R.id.id_circle_menu_item_center)
                continue;

            if (child.getVisibility() == GONE) {
                continue;
            }

            mStartAngle %= 360;//菜单的起始角度

            //中心到menu item的距离
            float distanceeFromCenter = layoutRadius / 2f - itemwidth / 2
- mPadding;
            //left坐标
            left = layoutRadius / 2 + (int) Math.round(distanceeFromCenter
* Math.cos(Math.toRadians(mStartAngle)) - 1 / 2f * itemwidth);
            //top坐标
            top = layoutRadius / 2 + (int) Math.round(distanceeFromCenter
* Math.sin(Math.toRadians(mStartAngle)) - 1 / 2f * itemwidth);

            //布局 child view
            child.layout(left, top, left + itemwidth, top + itemwidth);
            //
            mStartAngle += angleDelay;
        }

        // 找到中心的view，如果存在设置onclick事件
        final View centerView =
findViewById(R.id.id_circle_menu_item_center);
        if (centerView != null) {
            centerView.setOnClickListener(new OnClickListener() {
                @Override
                public void onClick(View v) {
                    if (mOnItemClickListener != null) {
                        mOnItemClickListener.itemCenterClick(v);
                    }
                }
            });

            //设置center item位置
            int cl = layoutRadius / 2 - centerView.getMeasuredWidth() / 2;
            int cr = cl + centerView.getMeasuredWidth();
            centerView.layout(cl, cl, cr, cr);
        }
    }

    /**
     * 设置adpater
     */
//    public void setAdapter(ListAdapter adapter) {
//        this.mAdapter = adapter;
//    }

    /**
     * 设置菜单条目的图标和文本
     *
     * @param resIds
     */
    public void setMenuItemIconsAndTexts(int[] resIds, String[] texts) {
        mItemImgs = resIds;
        mItemTexts = texts;

        // 参数检查
        if (resIds == null && texts == null) {
            throw new IllegalArgumentException("菜单项文本和图片至少设置其一");
        }

        // 初始化mMenuCount
        mMenuItemCount = resIds == null ? texts.length : resIds.length;

        if (resIds != null && texts != null) {
            mMenuItemCount = Math.min(resIds.length, texts.length);
        }

        addMenuItems();

    }


    /**
     * 设置MenuItem的布局文件，必须在setMenuItemIconsAndTexts之前调用
     *
     * @param mMenuItemLayoutId
     */
    public void setMenuItemLayoutId(int mMenuItemLayoutId) {
        this.mMenuItemLayoutId = mMenuItemLayoutId;
    }


    /**
     * 构建菜单项
     */
    private void addMenuItems() {
        LayoutInflater mInflater = LayoutInflater.from(getContext());

        //         // 根据用户的参数，初始化menu item
        //         for (int i = 0; i < mAdapter.getCount(); i++) {
```

```
        //            final View itemView = mAdapter.getView(i, null,
this);
        //            final int position = i;
        //            itemView.setOnClickListener(new OnClickListener() {
        //                @Override
        //                public void onClick(View v) {
        //                    if (mOnItemClickListener != null) {
        //                        mOnItemClickListener.onClick(itemView,
position);
        //                    }
        //                }
        //            });
        //            addView(itemView);
        //
        //        }

        for (int i = 0; i < mMenuItemCount; i++) {
            final int j = i;
            View view = mInflater.inflate(mMenuItemLayoutId, this, false);
            ImageView img = (ImageView)
view.findViewById(R.id.id_circle_menu_item_image);
            TextView tv = (TextView)
view.findViewById(R.id.id_circle_menu_item_text);
            if (img != null) {
                img.setVisibility(View.VISIBLE);
                img.setImageResource(mItemImgs[i]);
                img.setOnClickListener(new OnClickListener() {
                    @Override
                    public void onClick(View v) {
                        if (mOnItemClickListener != null) {
                            mOnItemClickListener.itemClick(v, j);
                        }

                    }
                });
            }
            if (tv != null) {
                tv.setVisibility(View.VISIBLE);
                tv.setText(mItemTexts[i]);
            }
            addView(view);
        }
    }

    /**
     * 设置内边距的比例
     *
     * @param mPadding
     */
    public void setPadding(float mPadding) {
        this.mPadding = mPadding;
    }
    /**
     * 记录上一次的x，y坐标
     */
    private float mLastX;
    private float mLastY;
    /**
     * 自动滚动的Runnable
     */
    private AutoFlingRunnable mFlingRunnable;

    /**
     * 自动滚动的任务
     *
     * @author zhy
     */
    private class AutoFlingRunnable implements Runnable {

        private float angelPerSecond;

        public AutoFlingRunnable(float velocity) {
            this.angelPerSecond = velocity;
        }

        public void run() {
            // 如果小于20,则停止
            if ((int) Math.abs(angelPerSecond) < 20) {
                isFling = false;
                return;
            }
            isFling = true;
            // 不断改变mStartAngle，让其滚动，/30为了避免滚动太快
            mStartAngle += (angelPerSecond / 30);
            // 逐渐减小这个值
            angelPerSecond /= 1.0666F;
            postDelayed(this, 30);
            // 重新布局
            requestLayout();
        }
    }

    @Override
    public boolean dispatchTouchEvent(MotionEvent event) {
        float x = event.getX();
        float y = event.getY();
        switch (event.getAction()) {
            case MotionEvent.ACTION_DOWN:
                mLastX = x;
                mLastY = y;
                mDownTime = System.currentTimeMillis();
                mTmpAngle = 0;
                // 如果当前已经在快速滚动
                if (isFling) {
                    // 移除快速滚动的回调
                    removeCallbacks(mFlingRunnable);
```

```
                        isFling = false;
                        return true;
                    }


                    break;
                case MotionEvent.ACTION_MOVE:
                    /**
                     * 获得开始的角度
                     */
                    float start = getAngle(mLastX, mLastY);
                    /**
                     * 获得当前的角度
                     */
                    float end = getAngle(x, y);
                    // 如果是一、四象限，则直接end-start，角度值都是正值
                    if (getQuadrant(x, y) == 1 || getQuadrant(x, y) == 4) {
                        mStartAngle += end - start;
                        mTmpAngle += end - start;
                    } else { // 二、三象限，色角度值是付值
                        mStartAngle += start - end;
                        mTmpAngle += start - end;
                    }
                    // 重新布局
                    requestLayout();
                    mLastX = x;
                    mLastY = y;

                    break;

                case MotionEvent.ACTION_UP:
                    // 计算，每秒移动的角度
                    float anglePerSecond = mTmpAngle * 1000 /
(System.currentTimeMillis() - mDownTime);
                    // 如果达到该值认为是快速移动
                    if (Math.abs(anglePerSecond) > mFlingableValue &&
!isFling) {
                        // post一个任务，去自动滚动
                        post(mFlingRunnable = new
AutoFlingRunnable(anglePerSecond));

                        return true;
                    }
                    // 如果当前旋转角度超过NOCLICK_VALUE屏蔽点击
                    if (Math.abs(mTmpAngle) > NOCLICK_VALUE) {
                        return true;
                    }
                    break;
            }

            return super.dispatchTouchEvent(event);
        }

        /**
```

```
     * 主要为了action_down时，返回true
     */
    @Override
    public boolean onTouchEvent(MotionEvent event) {
        return true;
    }

    /**
     * 根据触摸的位置，计算角度
     *
     * @param xTouch
     * @param yTouch
     * @return
     */
    private float getAngle(float xTouch, float yTouch) {
        double x = xTouch - (mRadius / 2d);
        double y = yTouch - (mRadius / 2d);
        return (float) (Math.asin(y / Math.hypot(x, y)) * 180 / Math.PI);
    }

    /**
     * 根据当前位置计算象限
     *
     * @param x
     * @param y
     * @return
     */
    private int getQuadrant(float x, float y) {
        int tmpX = (int) (x - mRadius / 2);
        int tmpY = (int) (y - mRadius / 2);
        if (tmpX >= 0) {
            return tmpY >= 0 ? 4 : 1;
        } else {
            return tmpY >= 0 ? 3 : 2;
        }

    }
}
```

MainActivity.java

```
package com.example;

/**
 * Package: example
 * Created by zyh
 * on 2019/5/26
 */

import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.view.View;
import android.widget.Toast;

public class MainActivity extends AppCompatActivity {
    private CircleMenuLayout circleMenuLayout;
    private String[] mItemTexts = new String[]{ "安全中心 ", "特色服务", "投资
理财",
            "转账汇款", "我的账户", "信用卡" };

    private int[] mItemImgs = new int[]{R.mipmap.home_mbank_1_normal,
            R.mipmap.home_mbank_2_normal, R.mipmap.home_mbank_3_normal,
            R.mipmap.home_mbank_4_normal, R.mipmap.home_mbank_5_normal,
            R.mipmap.home_mbank_6_normal };

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        circleMenuLayout = (CircleMenuLayout)
findViewById(R.id.id_cirlceMenu);
        circleMenuLayout.setMenuItemIconsAndTexts(mItemImgs, mItemTexts);
        circleMenuLayout.setOnItemclickListener(new
CircleMenuLayout.OnMenuItemClickListener() {
            @Override
            public void itemClick(View view, int position) {
                Toast.makeText(MainActivity.this, "选
中: "+mItemTexts[position], Toast.LENGTH_SHORT).show();
            }

            @Override
            public void itemCenterClick(View view) {
                Toast.makeText(MainActivity.this, "选中: 中间view",
Toast.LENGTH_SHORT);
            }
        });
    }
}
```

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@mipmap/bg"
    tools:context="com.example.MainActivity">

    <com.example.CircleMenuLayout
```

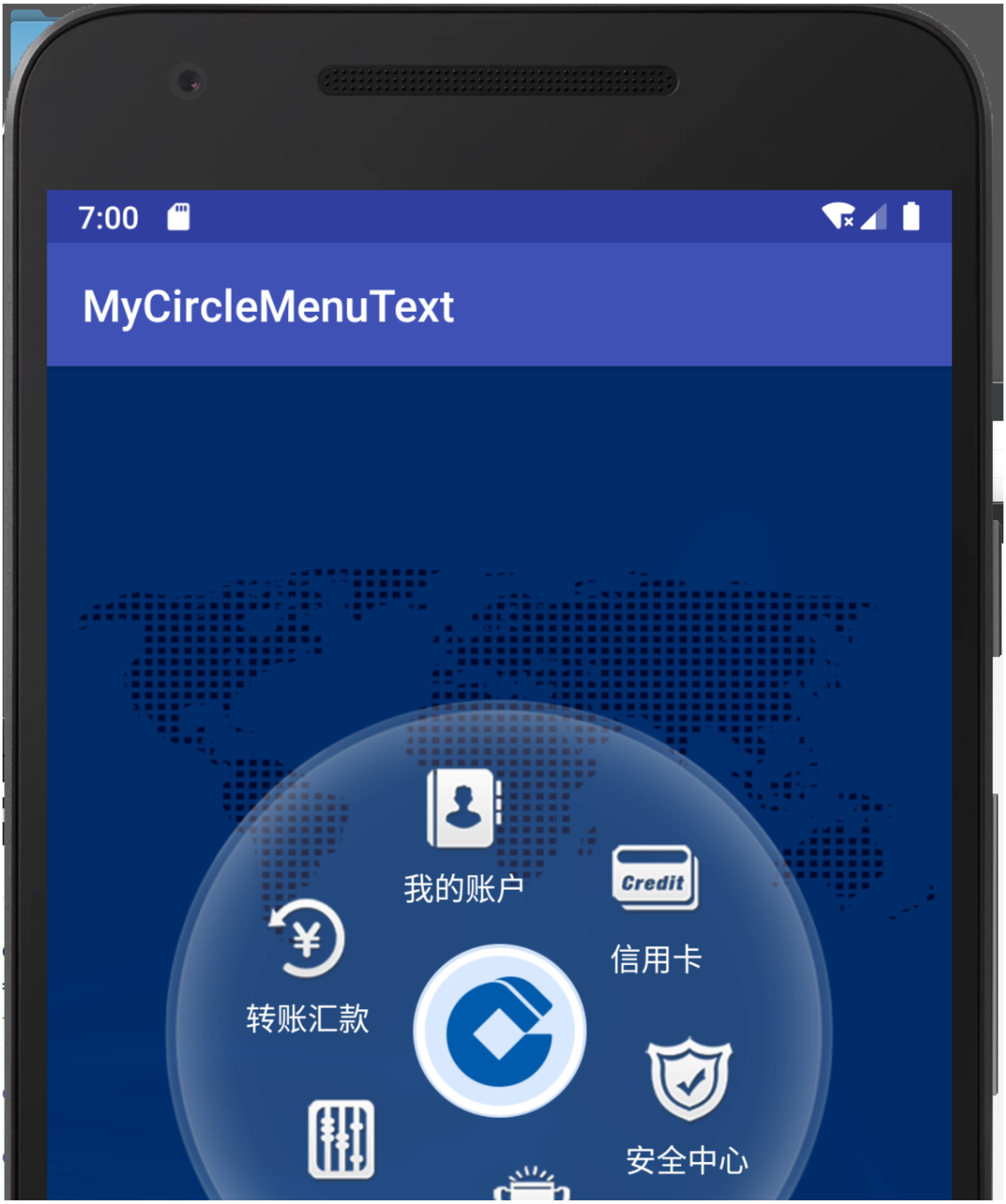
```
        android:id="@+id/id_cirlceMenu"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:background="@mipmap/circle_bg"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent">

        <RelativeLayout
            android:id="@id/id_circle_menu_item_center"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content">

            <ImageView
                android:layout_width="104.0dip"
                android:layout_height="104.0dip"
                android:layout_centerInParent="true"
                android:background="@mipmap/turnplate_center_unlogin"/>

            <ImageView
                android:layout_width="116.0dip"
                android:layout_height="116.0dip"
                android:layout_centerInParent="true"

android:background="@mipmap/turnplate_mask_unlogin_normal"/>
        </RelativeLayout>

    </com.example.CircleMenuLayout>


</android.support.constraint.ConstraintLayout>
```

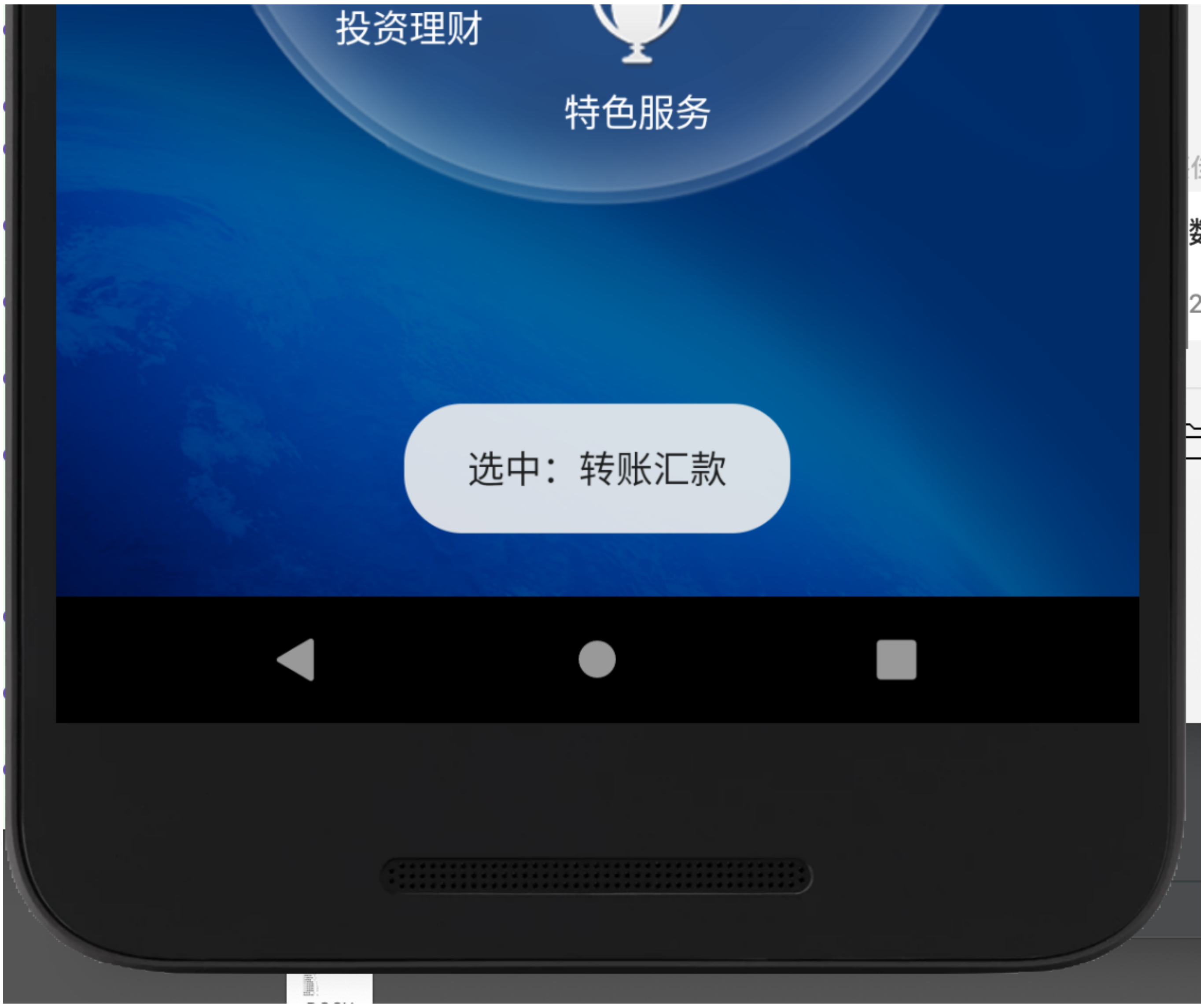
circle_menu_item.xml

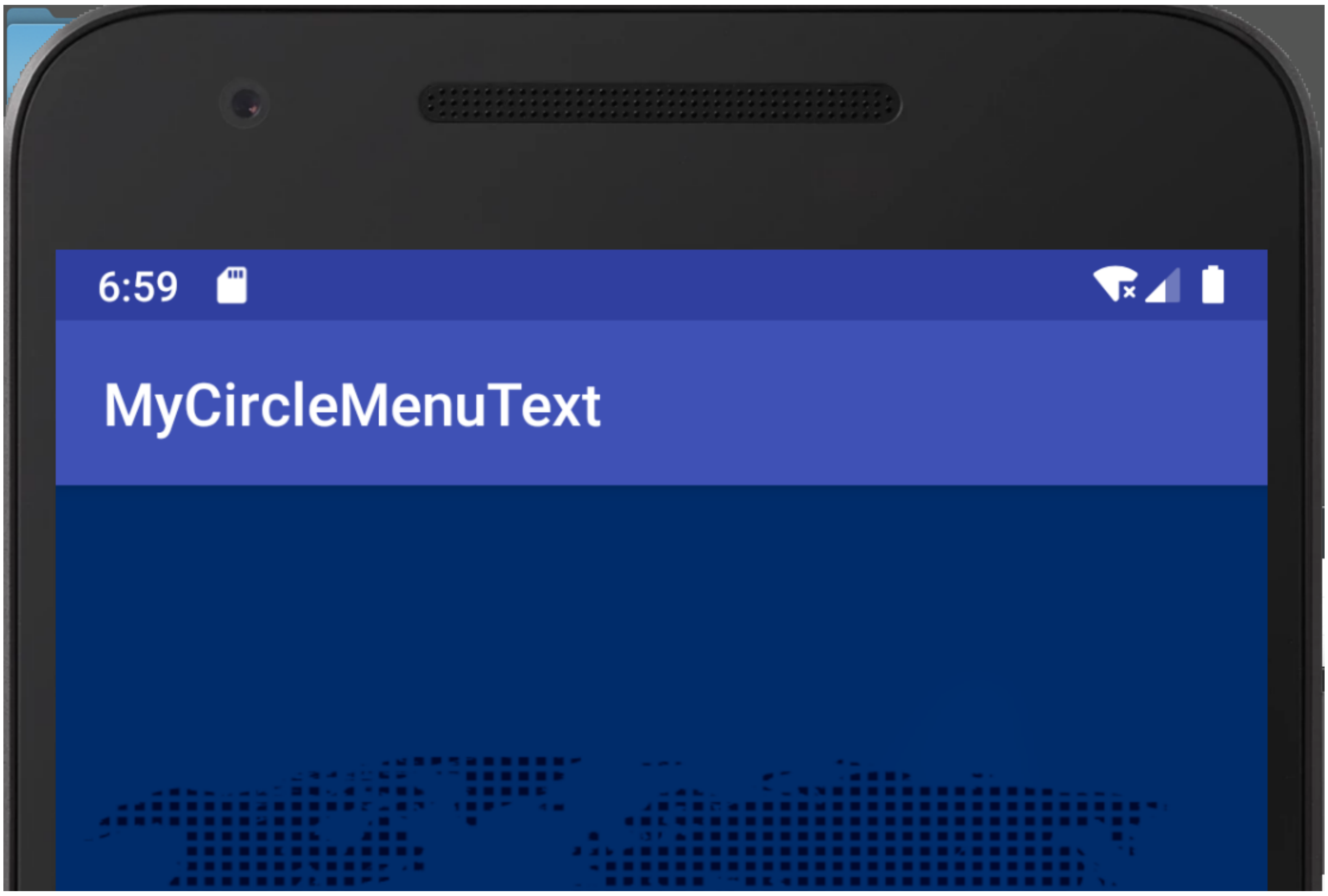
```
<?xml version="1.0" encoding="UTF-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:gravity="center"
    android:orientation="vertical" >

    <ImageView
        android:id="@id/id_circle_menu_item_image"
        android:layout_width="wrap_content"
        android:visibility="gone"
        android:layout_height="wrap_content" />

    <TextView
        android:id="@id/id_circle_menu_item_text"
        android:layout_width="wrap_content"
```

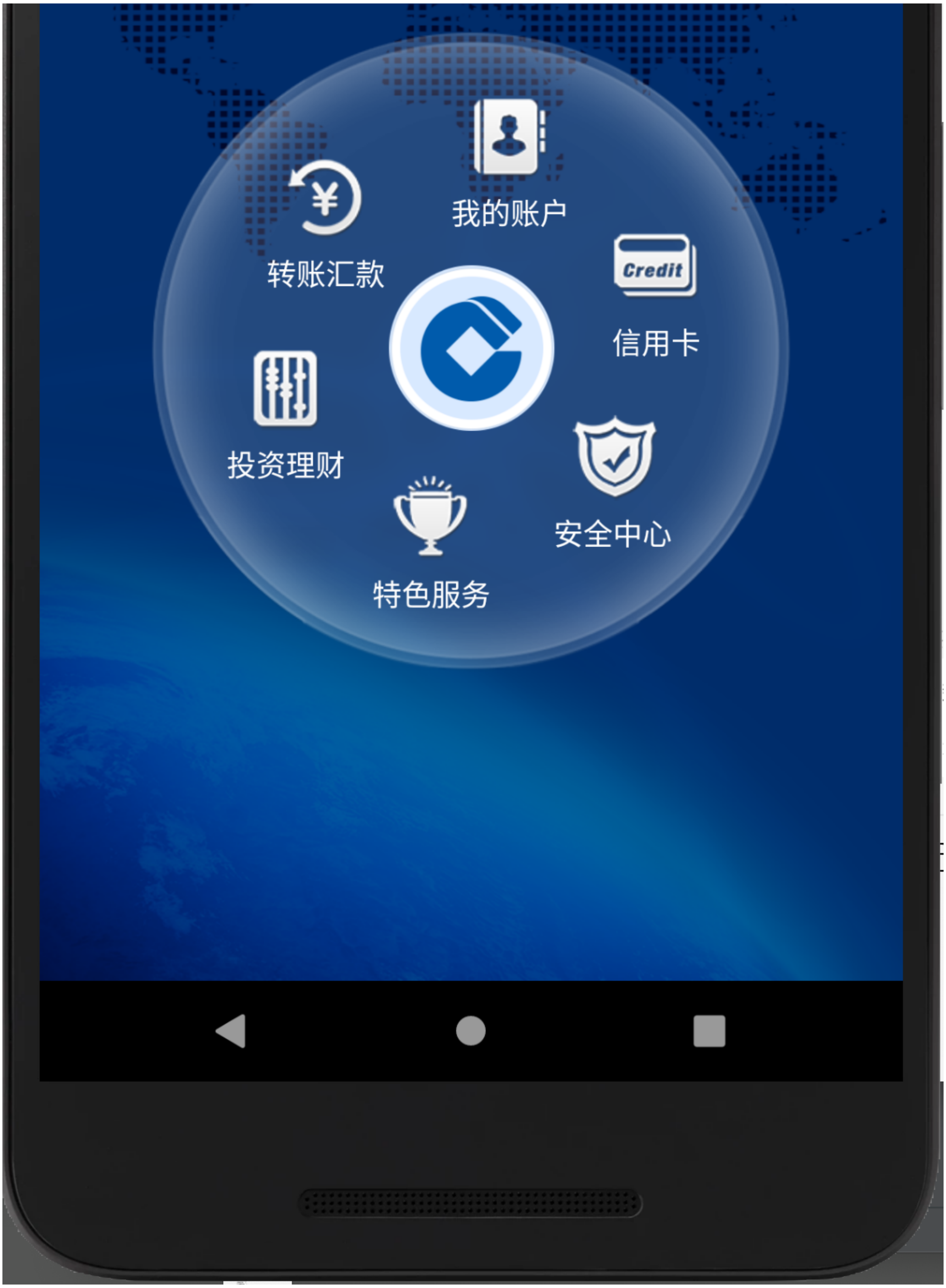
```
            android:visibility="gone"
            android:layout_height="wrap_content"
            android:textColor="@android:color/white"
            android:text="保险"
            android:textSize="14.0dip" />

</LinearLayout>
```

4.2运行截图




投资理财
特色服务
选中：转账汇款


6:59
MyCircleMenuText


我的账户
转账汇款
Credit
信用卡
投资理财
安全中心
特色服务