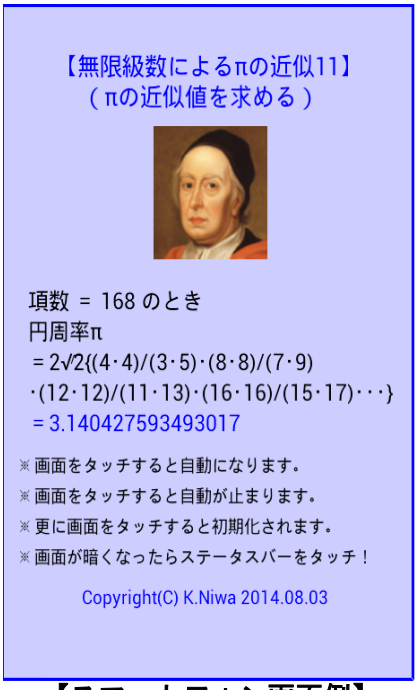
```
//----------------------------------------
//
//          無限級数によるπの近似 11
//            （πの近似値を求める）
//                  Ver9
//       Copyright(C) K.Niwa 2014.08.03
//
//----------------------------------------
```

【エミュレータ画面例】



【スマートフォン画面例】
（Android 4.0）

［アプリの概要］

次の近似式を用いて、円周率πの近似値を求めます。

$$\pi = 2\sqrt{2}\cdot\left(\frac{4\cdot 4}{3\cdot 5}\right)\cdot\left(\frac{8\cdot 8}{7\cdot 9}\right)\cdot\left(\frac{12\cdot 12}{11\cdot 13}\right)\cdot\left(\frac{16\cdot 16}{15\cdot 17}\right)\cdot\cdot\cdot$$

項の数をどんどん多くしていったとき、円周率πの近似値が求まる様子を観察してみましょう。

［1］Kinjiofp11.java

```
package jp.seitoku.kinjiofp11;

import android.content.Context;
import android.content.res.Resources;
import android.graphics.*;          //*に修正
import android.util.AttributeSet;
import android.view.*;              //*に修正

public class Kinjiofp11 extends View {

        private Bitmap bitmap1 = null;

        int flag=0;                 //自動識別子
        int ct=0;                   //分子・分母の項の数
        int count;                  //ループカウンター
        double pai;                 //πの近似値
        double s=1;                 //π／８を求める過程での無限級数

        public Kinjiofp11(Context context, AttributeSet attrs, int defStyle) {
                super(context, attrs, defStyle);
                init(context);
                // TODO 自動生成されたコンストラクター・スタブ
        }

        public Kinjiofp11(Context context, AttributeSet attrs) {
                super(context, attrs);
```

```
		init(context);
		// TODO 自動生成されたコンストラクター・スタブ
	}

	public Kinjiofp11(Context context) {
		super(context);
		init(context);
		// TODO 自動生成されたコンストラクター・スタブ
	}

	private void init(Context context){
		Resources res = context.getResources();
		bitmap1 = BitmapFactory.decodeResource(res, R.drawable.wallis);
	}


	@Override
	protected void onDraw(Canvas canvas) {
		// TODO 自動生成されたメソッド・スタブ

		float a=0;
		float b=0;


		super.onDraw(canvas);
		canvas.drawColor(Color.WHITE);
		Paint paint = new Paint();
		paint.setColor(Color.BLUE);
		paint.setAlpha(50);
		canvas.drawRect((getWidth()/2-240)+10,(getHeight()/2-343)+10,(getWidth()/2-240)+470,(getH
eight()/2-343)+675, paint);

		paint.setAlpha(10000);
		paint.setColor(Color.BLUE);

		for (int i=0;i<3;i++) {
			canvas.drawLine((getWidth()/2-240)+10+i,(getHeight()/2-343)+10+i,(getWidth()/2-24
0)+10+i,(getHeight()/2-343)+675-i, paint);
			canvas.drawLine((getWidth()/2-240)+10+i,(getHeight()/2-343)+675-i,(getWidth()/2-2
40)+470-i,(getHeight()/2-343)+675-i, paint);
			canvas.drawLine((getWidth()/2-240)+470-i,(getHeight()/2-343)+675-i,(getWidth()/2-
240)+470-i,(getHeight()/2-343)+10+i, paint);
			canvas.drawLine((getWidth()/2-240)+470-i,(getHeight()/2-343)+10+i,(getWidth()/2-2
40)+10+i,(getHeight()/2-343)+10+i, paint);
		}

		if (Kinjiofp11Activity.ritsu != 0) {
			a=(float) (0.9*320/Kinjiofp11Activity.ritsu);	//----------------------------
＜画像の拡大・縮小の横の倍率を指定する＞
			b=(float) (0.9*320/Kinjiofp11Activity.ritsu);	//----------------------------
＜画像の拡大・縮小の縦の倍率を指定する＞
		}
		else {
			a=(float) 1.0;
			b=(float) 1.0;
		}

		Matrix Mat = new Matrix();		//------------------------------------ ＜画像を拡大・縮
小する＞
		Mat.postScale(a, b);	//----------------------------------------
		Bitmap bitmap2 = Bitmap.createBitmap(	//-------------------------
				bitmap1,0,0,	//---------------------------------------
				bitmap1.getWidth(),	//-----------------------------------
				bitmap1.getHeight(),	//--------------------------------
				Mat,true //-------------------------------------------
				);	//-------------------------------------------------

		if (bitmap2 != null) {
```

```
                        canvas.drawBitmap(bitmap2, (getWidth()/2-240)+180,(getHeight()/2-343)+130, paint)
;
                }


                paint.setTextSize(25.0f);
                canvas.drawText("【無限級数によるπの近似11】", (getWidth()/2-240)+90-20, (getHeight()/2-
343)+80, paint);
                paint.setTextSize(25.0f);
                canvas.drawText("（πの近似値を求める）", (getWidth()/2-240)+100, (getHeight()/2-343)+110,
 paint);

                paint.setColor(Color.BLUE);
                paint.setTextSize(19.0f);
                canvas.drawText("Copyright(C) K.Niwa 2014.08.03", (getWidth()/2-240)+100, (getHeight()/2-
343)+600, paint);


//---------- 計算部始まり ----------

                ct++;
                s=s*(double)((4*ct)*(4*ct))/((4*ct-1)*(4*ct+1));

                pai=(double)s*2*Math.sqrt(2);

//---------- 計算部終わり ----------


                paint.setColor(Color.BLACK);
                paint.setTextSize(23.0f);
                canvas.drawText("項数 ＝ "+ct+" のとき",(getWidth()/2-240)+40, (getHeight()/2-343)+310,
paint);

                canvas.drawText("円周率π", (getWidth()/2-240)+40, (getHeight()/2-343)+340, paint);
                canvas.drawText("＝2√2{(4･4)/(3･5)･(8･8)/(7･9)", (getWidth()/2-240)+50-10, (getHeight()/
2-343)+370, paint);
                canvas.drawLine((getWidth()/2-240)+101-10,(getHeight()/2-343)+355, (getWidth()/2-240)+110
-10,(getHeight()/2-343)+355 ,paint);
                canvas.drawLine((getWidth()/2-240)+101-10,(getHeight()/2-343)+355, (getWidth()/2-240)+98-
10,(getHeight()/2-343)+363 ,paint);
                canvas.drawText("･(12･12)/(11･13)･(16･16)/(15･17)･･･}", (getWidth()/2-240)+50-10, (getHei
ght()/2-343)+400, paint);

                paint.setColor(Color.BLUE);
                canvas.drawText("＝"+pai ,(getWidth()/2-240)+50-10, (getHeight()/2-343)+430, paint);

                paint.setColor(Color.BLACK);
                paint.setTextSize(18.0f);
                canvas.drawText("※ 画面をタッチすると自動になります。",(getWidth()/2-240)+30, (getHeigh
t()/2-343)+470, paint);
                canvas.drawText("※ 画面をタッチすると自動が止まります。",(getWidth()/2-240)+30, (getHei
ght()/2-343)+500, paint);
                canvas.drawText("※ 更に画面をタッチすると初期化されます。",(getWidth()/2-240)+30, (getH
eight()/2-343)+530, paint);
                canvas.drawText("※ 画面が暗くなったらステータスバーをタッチ！",(getWidth()/2-240)+30,
(getHeight()/2-343)+560, paint);

                if (flag==1) {          //flag=1で自動になる  flag=2で自動が止まる  flag=0で初期化する
                        invalidate();   //表示を更新する
                }

        }//protected void onDraw(Canvas canvas)

        @Override
        public boolean onTouchEvent(MotionEvent event) {
                flag++;
                flag = flag % 3;
                if (flag==0) {
```

```
                ct=0;           //項数
                s=1;            //πを求める過程で使用
            }

            invalidate();    //表示を更新する
            return false;

        }//public boolean onTouchEvent(MotionEvent event)

}//public class MyPai11 extends View
```

[２] main.xml

```
<?xml version="1.0" encoding="utf-8"?>

<LinearLayout
        xmlns:android="http://schemas.android.com/apk/res/android"
        android:orientation="vertical"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:weightSum="1" >

        <jp.seitoku.kinjiofp11.Kinjiofp11
                android:layout_height="match_parent"
                android:layout_width="match_parent"
                android:id="@+id/myview1">
        </jp.seitoku.kinjiofp11.Kinjiofp11>
</LinearLayout>
```

[３] Kinjiofp11Activity.java

```
package jp.seitoku.kinjiofp11;

import android.os.Bundle;
import android.app.Activity;
import android.view.Menu;

import android.util.DisplayMetrics;       //------------------------------〈画像の拡大・縮小に必要なライブラリ〉

public class Kinjiofp11Activity extends Activity {

    static int ritsu;

        @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        DisplayMetrics metrics = new DisplayMetrics();     //--------------------------〈端末の情報を取得する〉
                getWindowManager().getDefaultDisplay().getMetrics(metrics);//-----------------------------------
                StringBuilder buffer = new StringBuilder();        //-------------------------------------------
        buffer.append("densityDpi（ドット数／インチ）:" + String.valueOf(metrics.densityDpi) + "¥n");
//------
        ritsu=metrics.densityDpi;// ------------------------------------------------これで値が取り出せた!
        }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
```

```
        getMenuInflater().inflate(R.menu.main, menu);
        return true;
    }
}
```