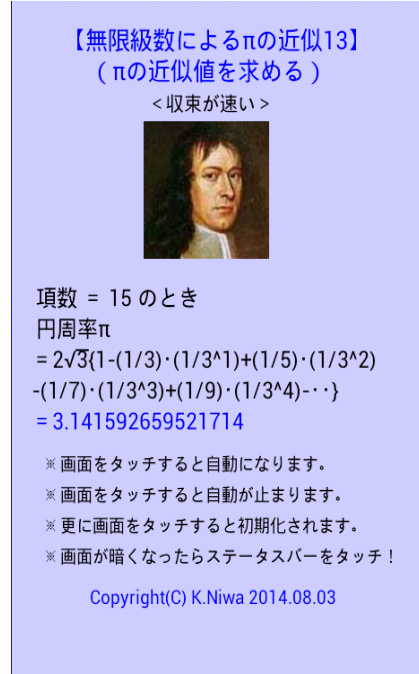


```
//-----
//
//          無限級数によるπの近似13
//          (πの近似値を求める)
//          Ver9
//          Copyright(C) K.Niwa 2014.08.03
//-----
```



【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。

$$\pi = 2\sqrt{3} \left( 1 - \frac{1}{3} \cdot \frac{1}{3^1} + \frac{1}{5} \cdot \frac{1}{3^2} - \frac{1}{7} \cdot \frac{1}{3^3} + \frac{1}{9} \cdot \frac{1}{3^4} - \dots \right)$$

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。収束の速度が大変速いことが分かります。

【1】 Kinjiofp13.java

```
package jp.seitoku.kinjiofp13;

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 Kinjiofp13 extends View {

    private Bitmap bitmap1 = null;

    int flag=0;                      //自動識別子
    double ct=0;                     //項の数 int型にするとうまく計算結果が得られない！
    int n=0;                          //項数
    double t=1;                      //πの近似値を求める過程で使用
    int i;                            //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 Kinjiofp13(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 Kinjiofp13(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 Kinjiofp13(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.gregory);
}

@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, (getHeight()/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()/2-240)+10+i, (getHeight()/2-343)+675-i, paint);
        canvas.drawLine((getWidth()/2-240)+10+i, (getHeight()/2-343)+675-i, (getWidth()/2-240)+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()/2-240)+10+i, (getHeight()/2-343)+10+i, paint);
    }

    if (Kinjiofp13Activity.ritsu != 0) {
        a=(float) (0.9*320/Kinjiofp13Activity.ritsu); //-----
        <画像の拡大・縮小の横の倍率を指定する>
        b=(float) (0.9*320/Kinjiofp13Activity.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)+160, (getHeight()/2-343)+130, paint)
;
    }

    paint.setTextSize(25.0f);
    canvas.drawText("【無限級数によるπの近似13】", (getWidth()/2-240)+90-20, (getHeight()/2-
343)+60, paint);
    paint.setTextSize(25.0f);
    canvas.drawText(" (πの近似値を求める) ", (getWidth()/2-240)+100, (getHeight()/2-343)+90,
paint);
    paint.setColor(Color. BLACK);
    paint.setTextSize(20.0f);
    canvas.drawText("<収束が速い>", (getWidth()/2-240)+165, (getHeight()/2-343)+120, 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++;
    n++;

    t=1;
    for (i=1;i<=ct;i++){
        t=(t/3);
    }

    if (ct % 2 == 1){
        s=s-(1/(2*ct+1))*t;
    }
    else {
        s=s+(1/(2*ct+1))*t;
    }
    pai=s*2*Math.sqrt(3);

//----- 計算部終わり -----

    paint.setColor(Color. BLACK);
    paint.setTextSize(23.0f);
    canvas.drawText("項数 = "+(n+1)+" のとき", (getWidth()/2-240)+40, (getHeight()/2-343)+31
0, paint);

    canvas.drawText("円周率π", (getWidth()/2-240)+40, (getHeight()/2-343)+340, paint);
    canvas.drawText("=2√3{1-(1/3)·(1/3^1)+(1/5)·(1/3^2)", (getWidth()/2-240)+50-15, (getHei
ght()/2-343)+370, paint);
    canvas.drawLine((getWidth()/2-240)+50+20+10+4, (getHeight()/2-343)+352, (getWidth()/2-240)+
98, (getHeight()/2-343)+352, paint);
    canvas.drawLine((getWidth()/2-240)+50+20+10+4, (getHeight()/2-343)+353, (getWidth()/2-240)+
98, (getHeight()/2-343)+353, paint);
    canvas.drawText("-(1/7)·(1/3^3)+(1/9)·(1/3^4)-...", (getWidth()/2-240)+50-15, (getHeight
()/2-343)+400, paint);

    paint.setColor(Color. BLUE);
    canvas.drawText("="+pai, (getWidth()/2-240)+50-15, (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)+50, (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)+50, (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)+50, (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)+50,
(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
            n=0;                //項数
        }

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

} //public boolean onTouchEvent(MotionEvent event)

} //public class MyPai13 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.kinjiofp13.Kinjiofp13
        android:layout_height="match_parent"
        android:layout_width="match_parent"
        android:id="@+id/myview1" >
    </jp.seitoku.kinjiofp13.Kinjiofp13>
</LinearLayout>

```

## [3] Kinjiofp13Activity.java

```

package jp.seitoku.kinjiofp13;

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 Kinjiofp13Activity 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;
    }
}

```