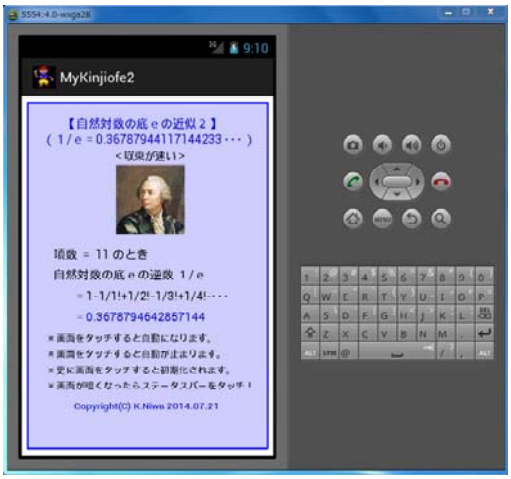
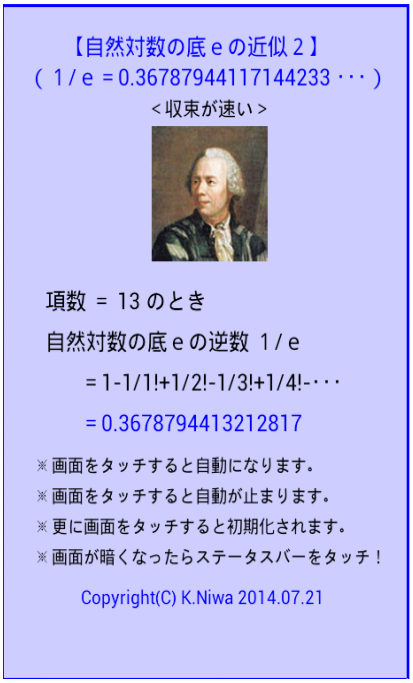
```
//----------------------------------------
//
//          自然対数の底e の近似2
//                Ver9
//    Copyright(C) K.Niwa 2014.07.21
//
//----------------------------------------
```



【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、自然対数の底ｅの逆数の近似値を求めます。

$$\frac{1}{e}=1-\frac{1}{1!}+\frac{1}{2!}-\frac{1}{3!}+\frac{1}{4!}-\cdots$$

項の数をどんどん多くしていったとき、自然対数の底ｅの逆数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] Kinjiofe2.java

```
package jp.seitoku.kinjiofe2;

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 Kinjiofe2 extends View {

        private Bitmap bitmap1 = null;

        int flag=0;                 //自動識別子
        int ct=0;                   //項数
        int count;                  //ループカウンター
        int n;
        double e=0;                 //eの近似値
        double s=1;                 //ｅの近似値を求める過程で使用
        double t=1;                 //ｅの近似値を求める過程で使用

        public Kinjiofe2(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 Kinjiofe2(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 Kinjiofe2(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.euler);
        }


        @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 (Kinjiofe2Activity.ritsu != 0) {
                        a=(float) (0.9*320/Kinjiofe2Activity.ritsu);       //----------------------------
 ＜画像の拡大・縮小の横の倍率を指定する＞
                        b=(float) (0.9*320/Kinjiofe2Activity.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(23.0f);
                canvas.drawText("【自然対数の底ｅの近似２】", (getWidth()/2-240)+105-24, (getHeight()/2-3
43)+60, paint);
                canvas.drawText("（１／ｅ＝"+1/Math.E+"・・・）" ,(getWidth()/2-240)+40, (getHeight()/2-343)
+90, paint);
                paint.setColor(Color.BLACK);
                paint.setTextSize(20.0f);
                canvas.drawText("＜収束が速い＞", (getWidth()/2-240)+175, (getHeight()/2-343)+120, paint)
;

                paint.setColor(Color.BLUE);
                paint.setTextSize(19.0f);
                canvas.drawText("Copyright(C) K.Niwa 2014.07.21", (getWidth()/2-240)+100, (getHeight()/2-
343)+600, paint);

//---------- 計算部始まり ----------

                ct=ct+1;
                double s=1;

                for (n=1;n<=ct;n++) {
                        s=s*(double)1/n;
                }

                if (ct%2==0) {
                        t=t+s;
                }
                else if (ct%2==1) {
                        t=t-s;
                }

                e=t;

//---------- 計算部終わり ----------

                paint.setColor(Color.BLACK);
                paint.setTextSize(23.0f);
                canvas.drawText("項数 ＝ "+(ct+1) +" のとき",(getWidth()/2-240)+60, (getHeight()/2-343)+3
10, paint);

                canvas.drawText("自然対数の底ｅの逆数 １／ｅ", (getWidth()/2-240)+60, (getHeight()/2-343)+
350, paint);
                canvas.drawText("＝1-1/1!+1/2!-1/3!+1/4!-・・・", (getWidth()/2-240)+100, (getHeight()/2-34
3)+390, paint);
                paint.setColor(Color.BLUE);
                canvas.drawText("＝"+e ,(getWidth()/2-240)+100, (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 && ct<16) {         //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
                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) {
                        e=0;
                        ct=0;            //項数
                        s=1;             //ｅを求める過程で使用
                        t=1;             //ｅ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 MyOirer1 extends View
```

［２］main.xml

[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.kinjiofe2.Kinjiofe2
                android:layout_height="match_parent"
                android:layout_width="match_parent"
                android:id="@+id/myview1">
        </jp.seitoku.kinjiofe2.Kinjiofe2>
</LinearLayout>
```

［３］Kinjiofe2Activity.java

```
package jp.seitoku.kinjiofe2;

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 Kinjiofe2Activity 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;
    }
}
```