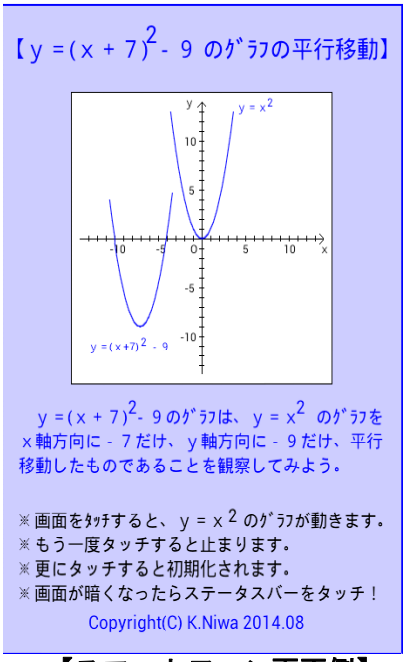
```
//-----------------------------------------------
//
//          ２次関数のグラフの平行移動（下に凸）
//              ｙ＝（ｘ＋７）^2－９　のグラフ
//                      Ver3
//          Copyright(C) K.Niwa 2014.08.10
//
//-----------------------------------------------
```

【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］

ｙ＝（ｘ＋７）²－９　のグラフは、ｙ＝ｘ²　のグラフをｘ軸方向に－７だけ、ｙ軸方向に－９だけ、平行移動したものであることを観察してみよう。

また、この２つのグラフは形も大きさも同じで、位置だけが異なったものであることに注意して見てみよう。

［１］Twokansuu1.java

```
package jp.seitoku.twokansuu1;     //パッケージを指定

import android.content.Context;    //ライブラリを読み込む
import android.graphics.*;
import android.util.AttributeSet;
import android.view.*;

public class Twokansuu1 extends View {     //Viewクラスを継承したTwokan1クラス

        int flag=0;       //グラフの移動(1)、グラフの停止(2)、グラフの初期化(0)初期化　識別子
        double x,y;       //グラフ描写に利用
        int px,py,oldpx,oldpy;    //グラフ描写に利用
        int fg; //グラフ描写に利用
        double a=0,b=0;  //グラフ描写に利用

        public Twokansuu1(Context context, AttributeSet attrs, int defStyle) {      //コンストラクタ（クラス名と同じメソッドで最初に読まれる）
                super(context, attrs, defStyle);
        }

        public Twokansuu1(Context context, AttributeSet attrs) {    //コンストラクタ（クラス名と同じメソッドで最初に読まれる）
                super(context, attrs);
        }
```

```
	public Twokansuu1(Context context) {	//コンストラクタ（クラス名と同じメソッドで最初に読まれる）
		super(context);
	}

//onDrawメソッド-----------------------------------------------------------------------------
	@Override
	protected void onDraw(Canvas canvas) {

		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<2;i++) {  //額縁を付ける
			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);
		}


		paint.setColor(Color.BLACK);	//実験枠の描画
		canvas.drawRect((getWidth()/2-240)+90, (getHeight()/2-343)+100,(getWidth()/2-240)+390,(getHeight()/2-343)+400,paint);
		paint.setColor(Color.WHITE);
		canvas.drawRect((getWidth()/2-240)+91, (getHeight()/2-343)+101,(getWidth()/2-240)+389,(getHeight()/2-343)+399,paint);

		paint.setColor(Color.BLACK);	//座標軸の描画
		canvas.drawLine((getWidth()/2-240)+100, (getHeight()/2-343)+250,(getWidth()/2-240)+380,(getHeight()/2-343)+250,paint);
		canvas.drawLine((getWidth()/2-240)+380, (getHeight()/2-343)+250,(getWidth()/2-240)+380-5,(getHeight()/2-343)+250-5,paint);
		canvas.drawLine((getWidth()/2-240)+380, (getHeight()/2-343)+250,(getWidth()/2-240)+380-5,(getHeight()/2-343)+250+5,paint);
		canvas.drawLine((getWidth()/2-240)+240, (getHeight()/2-343)+110,(getWidth()/2-240)+240,(getHeight()/2-343)+390,paint);
		canvas.drawLine((getWidth()/2-240)+240, (getHeight()/2-343)+110,(getWidth()/2-240)+240+5,(getHeight()/2-343)+110+5,paint);
		canvas.drawLine((getWidth()/2-240)+240, (getHeight()/2-343)+110,(getWidth()/2-240)+240-5,(getHeight()/2-343)+110+5,paint);

		for (int xx=110;xx<380;xx=xx+10) {		//ｘ軸メモリの描写
			canvas.drawLine((getWidth()/2-240)+xx, (getHeight()/2-343)+250-3,(getWidth()/2-240)+xx,(getHeight()/2-343)+250+3,paint);
		}
		for (int yy=120;yy<390;yy=yy+10) {		//ｙ軸メモリの描写
			canvas.drawLine((getWidth()/2-240)+240-3, (getHeight()/2-343)+yy,(getWidth()/2-240)+240+3,(getHeight()/2-343)+yy,paint);
		}

		//ｘ軸メモリの描写
		canvas.drawText("5", (getWidth()/2-240)+290-3, (getHeight()/2-343)+260+5, paint);
		canvas.drawText("10", (getWidth()/2-240)+340-6, (getHeight()/2-343)+260+5, paint);
		canvas.drawText("ｘ", (getWidth()/2-240)+375, (getHeight()/2-343)+260+5, paint);
		canvas.drawText("0", (getWidth()/2-240)+230-3, (getHeight()/2-343)+260+5, paint);
		canvas.drawText("-5", (getWidth()/2-240)+190-3, (getHeight()/2-343)+260+5, paint);
		canvas.drawText("-10", (getWidth()/2-240)+140-6, (getHeight()/2-343)+260+5, paint);
```

```
                    //ｙ軸メモリの描写
                    canvas.drawText("5", (getWidth()/2-240)+220+5, (getHeight()/2-343)+200+5, paint);
                    canvas.drawText("10", (getWidth()/2-240)+220, (getHeight()/2-343)+150+5, paint);
                    canvas.drawText("ｙ", (getWidth()/2-240)+220, (getHeight()/2-343)+110+5, paint);
                    canvas.drawText("-5", (getWidth()/2-240)+220, (getHeight()/2-343)+300+5, paint);
                    canvas.drawText("-10", (getWidth()/2-240)+220-5, (getHeight()/2-343)+350+5, paint);

                    //y=x^2 のグラフの描画
                    paint.setColor(Color.BLUE);
                    canvas.drawText("ｙ＝ｘ ", (getWidth()/2-240)+300-20, (getHeight()/2-343)+120, paint);
                    canvas.drawText("2", (getWidth()/2-240)+315, (getHeight()/2-343)+120-5, paint);
                    fg=0;
                    for (x=-3.6+a;x<=3.7+a;x=x+0.1) {
                              y=(x-a)*(x-a)+b;
                              px=(int)(240+10*x);
                              py=(int)(250-10*y);
                              if (fg==0) {
                                        canvas.drawLine((int)(getWidth()/2-240)+px, (int)(getHeight()/2-343)+py,
(int)(getWidth()/2-240)+px,(int)(getHeight()/2-343)+py,paint);
                              }
                              else {
                                        canvas.drawLine((int)(getWidth()/2-240)+oldpx, (int)(getHeight()/2-343)+
oldpy,(int)(getWidth()/2-240)+px,(int)(getHeight()/2-343)+py,paint);
                              }
                              oldpx=px;oldpy=py;
                              fg++;
                    }

                    //y=(x+7)^2-9 のグラフの描画
                    paint.setColor(Color.BLUE);
                    canvas.drawText("ｙ＝(ｘ+7) －９", (getWidth()/2-240)+100+10, (getHeight()/2-343)+370-5,
paint);
                    canvas.drawText("2", (getWidth()/2-240)+160+10, (getHeight()/2-343)+370-5-5, paint);
                    fg=0;
                    for (x=-3.6-7;x<=3.7-7;x=x+0.1) {
                              y=(x+7)*(x+7)-9;
                              px=(int)(240+10*x);
                              py=(int)(250-10*y);
                              if (fg==0) {
                                        canvas.drawLine((int)(getWidth()/2-240)+px, (int)(getHeight()/2-343)+py,
(int)(getWidth()/2-240)+px,(int)(getHeight()/2-343)+py,paint);
                              }
                              else {
                                        canvas.drawLine((int)(getWidth()/2-240)+oldpx, (int)(getHeight()/2-343)+
oldpy,(int)(getWidth()/2-240)+px,(int)(getHeight()/2-343)+py,paint);
                              }
                              oldpx=px;oldpy=py;
                              fg++;
                    }

                    paint.setColor(Color.BLACK);       //実験枠の描画
                    canvas.drawLine((getWidth()/2-240)+90, (getHeight()/2-343)+100,(getWidth()/2-240)+90,(get
Height()/2-343)+400,paint);
                    canvas.drawLine((getWidth()/2-240)+90, (getHeight()/2-343)+100,(getWidth()/2-240)+390,(ge
tHeight()/2-343)+100,paint);

                    paint.setColor(Color.BLUE);        //表題の表示
                    paint.setTextSize(25.0f);
                    canvas.drawText("", (getWidth()/2-240)+60, (getHeight()/2-343)+65, paint);
                    canvas.drawText("【ｙ＝(ｘ＋７) －９ のｸﾞﾗﾌの平行移動】", (getWidth()/2-240)+60-30-17, (g
etHeight()/2-343)+65, paint);
                    canvas.drawText("2", (getWidth()/2-240)+170+5, (getHeight()/2-343)+50, paint);

                    paint.setColor(Color.BLUE);        //目標の提示
                    paint.setTextSize(19.0f);
                    canvas.drawText("ｙ＝(ｘ＋７) －９のｸﾞﾗﾌは、ｙ＝ｘ のｸﾞﾗﾌを", (getWidth()/2-240)+30+18, (g
etHeight()/2-343)+440, paint);
```

```
                canvas.drawText("2", (getWidth()/2-240)+140-3+18, (getHeight()/2-343)+440-5-5, paint);
                canvas.drawText("2", (getWidth()/2-240)+300+30+18, (getHeight()/2-343)+440-5-5, paint);
                canvas.drawText("ｘ軸方向に－７だけ、ｙ軸方向に－９だけ、平行", (getWidth()/2-240)+30, (getHeight()/2-343)+465, paint);
                canvas.drawText("移動したものであることを観察してみよう。", (getWidth()/2-240)+30, (getHeight()/2-343)+490, paint);

                paint.setColor(Color.BLACK);        //説明の表示
                paint.setTextSize(19.0f);
                canvas.drawText("※ 画面をﾀｯﾁすると、ｙ＝ｘ のｸﾞﾗﾌが動きます。", (getWidth()/2-240)+50-20, (getHeight()/2-343)+550-5, paint);
                canvas.drawText("2", (getWidth()/2-240)+290-20, (getHeight()/2-343)+545-5, paint);
                canvas.drawText("※ もう一度タッチすると止まります。", (getWidth()/2-240)+50-20, (getHeight()/2-343)+575-5, paint);
                canvas.drawText("※ 更にタッチすると初期化されます。", (getWidth()/2-240)+50-20, (getHeight()/2-343)+600-5, 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)+50-20, (getHeight()/2-343)+625-5, paint);
                paint.setColor(Color.BLUE);
                paint.setTextSize(19.0f);
                canvas.drawText("Copyright(C) K.Niwa 2014.08", (getWidth()/2-240)+110, (getHeight()/2-343)+650, paint);          //作者・作成年月の表示

                if (flag==1) {   //識別子が１のとき、グラフが平行移動する
                        if (a>=-6.9) {
                                a=a-0.1;
                        }
                        else {
                                if (b>=-8.9) {
                                        b=b-0.1;
                                }
                        }

                        invalidate();    //再描画する（clear & goto onDraw）　そして、この行へ戻ってくる。
                                                //invalidate()は、onDrawメソッドの中にも記述できる。

                                                //invalidate()は、繰り返し処理に利用できる。
                                                //もちろん、onTouchEventソッドの中にも記述できる。
                }
                else if (flag==2) {       //識別子が２のとき、グラフが停止する
                        //何も変化を加えない
                }
                else if (flag==0) {       //識別子が０のとき、グラフを初期化する（元の位置に戻す）
                        a=0;     //初期化する
                        b=0;     //初期化する
                        invalidate();    //再描画する（clear & goto onDraw）　そして、この行へ戻ってくる。
                }

        }//protected void onDraw(Canvas canvas)


//画面にタッチしたときのイベント処理----------------------------------------------------------------------------
        @Override
        public boolean onTouchEvent(MotionEvent event) {

                flag=flag+1;     //flagに１を加える
                flag=flag % 3;   //flagに１、２、０を代入する

                invalidate();                           //再描画する（clear & goto onDraw）
                return false;

        }
}
```

[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.twokansuu1.Twokansuu1
        android:layout_height="match_parent"
        android:layout_width="match_parent"
                android:id="@+id/myview1">
        </jp.seitoku.twokansuu1.Twokansuu1>

</LinearLayout>
```

[３] Twokansuu1Activity.java

```
package jp.seitoku.twokansuu1;

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

public class Twokansuu1Activity extends Activity {

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

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(R.menu.main, menu);
        return true;
    }
}
```