

[ 1 ] MyRandomwalks.java

/\*

```
-----  
                ランダムウォーク  
                Android 4.4 (Kit Kat)  
                Copyright(C) K.Niwa 2019.12.12  
-----
```

\*/

```
package jp.kiyo.wuena.myrandomwalks;
```

```
import android.content.Context;  
import android.graphics.Canvas;  
import android.graphics.Color;  
import android.graphics.Paint;  
import android.graphics.Rect;  
import android.util.AttributeSet;  
import android.view.View;  
import android.content.res.Resources; //画像用  
import android.graphics.*;  
import android.view.*;
```

```
public class MyRandomwalks extends View {
```

```
    private Bitmap bitmap1 = null; //酔っぱらいの画像を宣言し初期化する  
    int x=220,y=100;//,oldx,oldy; //酔っぱらいの位置  
    int xx; //座標軸に使用  
    int k=0; //ふらつき回数  
    int flag=0; //停止識別子  
    int ct=1; //実験回数  
    int[] cy = new int[61]; //ふらつき 30 回目の位置のカウン  
ター (配列)  
    double r; //右にふれるか左にふれる  
    かの識別子 (乱数)  
    int i; //ループカウンター  
    int first=0; //最初識別子  
    int pause=0; //終了識別子  
    int syoki=0; //初期化識別子  
    int width;  
    int height;
```

```

public MyRandomwalks (Context context) {
    super (context);
    init (context);
}

public MyRandomwalks (Context context, AttributeSet attrs) {
    super (context,attrs);
    init (context);
}

public MyRandomwalks (Context context, AttributeSet attrs,int defStyle) {
    super (context,attrs,defStyle);
    init (context);
}

private void init (Context context) {
    Resources res = context.getResources ();           //画像用
    bitmap1 = BitmapFactory.decodeResource (res, R.drawable.hito1); //画像用
}

@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<2;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);
    }

    paint.setColor (Color.BLUE);
    paint.setTextSize (25.0f);
    canvas.drawText (" 【ランダムウォーク】 ", (getWidth () /2-240) +140-24, (getHeight ()
/2-343) +50, paint);

    //paint.setColor (Color.BLUE);
    //canvas.drawLine ( (getWidth () /2-240) +10, (getHeight () /2-343) +10, (getWidth ()
/2-240) +470, (getHeight () /2-343) +10, paint);
    //canvas.drawLine ( (getWidth () /2-240) +10, (getHeight () /2-343) +10, (getWidth ()
/2-240) +10, (getHeight () /2-343) +675, paint);
    //canvas.drawLine ( (getWidth () /2-240) +10, (getHeight () /2-343) +675, (getWidth ()
/2-240) +470, (getHeight () /2-343) +675, paint);
    //canvas.drawLine ( (getWidth () /2-240) +470, (getHeight () /2-343) +675, (getWidth ()
/2-240) +470, (getHeight () /2-343) +10, paint);
    //canvas.drawLine ( (getWidth () /2-240) +11, (getHeight () /2-343) +11, (getWidth ()
/2-240) +469, (getHeight () /2-343) +11, paint);
    //canvas.drawLine ( (getWidth () /2-240) +11, (getHeight () /2-343) +11, (getWidth ()
/2-240) +11, (getHeight () /2-343) +674, paint);
    //canvas.drawLine ( (getWidth () /2-240) +11, (getHeight () /2-343) +674, (getWidth ()
/2-240) +469, (getHeight () /2-343) +674, paint);
    //canvas.drawLine ( (getWidth () /2-240) +469, (getHeight () /2-343) +674, (getWidth ()
/2-240) +469, (getHeight () /2-343) +11, paint);

    paint.setColor (Color.BLACK);
    canvas.drawLine ( (getWidth () /2-240) +90, (getHeight () /2-343) +150, (getWidth ()
/2-240) +390, (getHeight () /2-343) +150, paint); //数直線 (上)
    for (xx=100;xx<=380;xx=xx+10) {
        canvas.drawLine ( (getWidth () /2-240) +xx, (getHeight () /2-343) +148, (getWidth ()
/2-240) +xx, (getHeight () /2-343) +152, paint); //メモリ線
    }

```

```

paint.setTextSize (16.0f);
canvas.drawText ("0", (getWidth ()/2-240)+236, (getHeight ()/2-343)+170, paint);
    //メモリ値
canvas.drawText ("8", (getWidth ()/2-240)+276, (getHeight ()/2-343)+170, paint);
canvas.drawText ("16", (getWidth ()/2-240)+312, (getHeight ()/2-343)+170, paint);
canvas.drawText ("24", (getWidth ()/2-240)+352, (getHeight ()/2-343)+170, paint);
canvas.drawText ("-8", (getWidth ()/2-240)+192, (getHeight ()/2-343)+170, paint);
canvas.drawText ("-16", (getWidth ()/2-240)+148, (getHeight ()/2-343)+170, paint);
canvas.drawText ("-24", (getWidth ()/2-240)+108, (getHeight ()/2-343)+170, paint);

paint.setColor (Color.BLACK);
    canvas.drawLine ((getWidth ()/2-240)+90, (getHeight ()/2-343)+390, (getWidth ()
/2-240)+390, (getHeight ()/2-343)+390, paint);        //数直線 (下)
    for (xx=100;xx<=380;xx=xx+10) {
        canvas.drawLine ((getWidth ()/2-240)+xx, (getHeight ()/2-343)+388, (getWidth ()
/2-240)+xx, (getHeight ()/2-343)+392, paint); //メモリ線
    }
paint.setTextSize (16.0f);
canvas.drawText ("0", (getWidth ()/2-240)+236, (getHeight ()/2-343)+410, paint);
    //メモリ値
canvas.drawText ("8", (getWidth ()/2-240)+276, (getHeight ()/2-343)+410, paint);
canvas.drawText ("16", (getWidth ()/2-240)+312, (getHeight ()/2-343)+410, paint);
canvas.drawText ("24", (getWidth ()/2-240)+352, (getHeight ()/2-343)+410, paint);
canvas.drawText ("-8", (getWidth ()/2-240)+192, (getHeight ()/2-343)+410, paint);
canvas.drawText ("-16", (getWidth ()/2-240)+148, (getHeight ()/2-343)+410, paint);
canvas.drawText ("-24", (getWidth ()/2-240)+108, (getHeight ()/2-343)+410, paint);

paint.setColor (Color.BLACK);
paint.setTextSize (19.0f);
    canvas.drawText ("※ 画面をタッチすると酔っぱらいが動きます。", (getWidth ()
/2-240)+30, (getHeight ()/2-343)+500, paint);
    canvas.drawText ("※ 画面をタッチすると酔っぱらいが止まります。", (getWidth ()
/2-240)+30, (getHeight ()/2-343)+525, paint);
    canvas.drawText ("※ 5回目の画面タッチで初期化されます。", (getWidth ()/2-240)
+30, (getHeight ()/2-343)+550, paint);
    canvas.drawText ("※ 画面が暗くなったらタイトルバーをタッチ!", (getWidth ()
/2-240)+30, (getHeight ()/2-343)+575, paint);

paint.setColor (Color.BLACK);
paint.setTextSize (18.0f);

```

```

        canvas.drawText("◎ ふらつき 30 回目の位置をグラフ表示します。", (getWidth()
/2-240)+30, (getHeight()/2-343)+600, paint);

        paint.setColor(Color.BLUE);
        paint.setTextSize(18.0f);
        canvas.drawText("Copyright (C) K.Niwa 2019.11.16", (getWidth()/2-240)+100,
(getHeight()/2-343)+640, paint);

        if (k<29) {           //ふらつき 29 回目以下のとき
            if (first>0) {
                k++;
            }

            paint.setColor(Color.BLUE);
            paint.setTextSize(15.0f);
            if (first==0) {
                canvas.drawText("ふらつき回数 ", (getWidth()/2-240)+190, (getHeight()
/2-343)+200, paint);
            }
            else if (first>0) {
                canvas.drawText("ふらつき回数 "+(k+1), (getWidth()/2-240)+190,
(getHeight()/2-343)+200, paint);
            }

            paint.setColor(Color.BLUE);
            paint.setTextSize(20.0f);
            if (first==0) {
                canvas.drawText("実験 回目 ", (getWidth()/2-240)+190, (getHeight()
/2-343)+450, paint);
            }
            else if (first>0) {
                canvas.drawText("実験 "+ct+" 回目", (getWidth()/2-240)+190, (getHeight
()/2-343)+450, paint);
            }

            r = Math.random();
            if (r<0.5) {           //左に動く
                x=x-5;
            }
            else if (r>=0.5) {     //右に動く

```

```

        x=x+5;
    }

    if (MainActivity.ritsu != 0) {
        a=(float) 0.7*320/MainActivity.ritsu; //----- < 画像の拡大・縮小の横の倍率を指定する >
        b=(float) 0.7*320/MainActivity.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 (); //-----***
    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)+x-4, (getHeight()/2-343)
+y, paint); //酔っぱらいを描写
    }
}

else if (k==29) { //ふらつき 30 回目のとき
    switch (x) {
        case 240: //パチンコ玉の落下した位置の x 座標
            cy[0]=cy[0]+1; //この位置に落下したパチンコ玉のカウンタ
            if (cy[0]>10) {
                pause=1; //自動描画を止める識別子
            }
            break;
        case 245: //パチンコ玉の落下した位置の x 座標
            cy[1]=cy[1]+1; //この位置に落下したパチンコ玉のカウンタ

```

```

if (cy[1]>10) {
    pause=1; //自動描画を止める識別子
}
break;
case 250: //パチンコ玉の落下した位置の x 座標
    cy[2]=cy[2]+1; //この位置に落下したパチンコ玉のカウンタ

if (cy[2]>10) {
    pause=1; //自動描画を止める識別子
}
break;
case 255: //パチンコ玉の落下した位置の x 座標
    cy[3]=cy[3]+1; //この位置に落下したパチンコ玉のカウンタ

if (cy[3]>10) {
    pause=1; //自動描画を止める識別子
}
break;
case 260: //パチンコ玉の落下した位置の x 座標
    cy[4]=cy[4]+1; //この位置に落下したパチンコ玉のカウンタ

if (cy[4]>10) {
    pause=1; //自動描画を止める識別子
}
break;
case 265: //パチンコ玉の落下した位置の x 座標
    cy[5]=cy[5]+1; //この位置に落下したパチンコ玉のカウンタ

if (cy[5]>10) {
    pause=1; //自動描画を止める識別子
}
break;
case 270: //パチンコ玉の落下した位置の x 座標
    cy[6]=cy[6]+1; //この位置に落下したパチンコ玉のカウンタ

if (cy[6]>10) {
    pause=1; //自動描画を止める識別子
}
break;

```

```

case 275:                //パチンコ玉の落下した位置の x 座標
    cy[7]=cy[7]+1;      //この位置に落下したパチンコ玉のカウンタ

    if (cy[7]>10) {
        pause=1; //自動描画を止める識別子
    }
    break;

case 280:                //パチンコ玉の落下した位置の x 座標
    cy[8]=cy[8]+1;      //この位置に落下したパチンコ玉のカウンタ

    if (cy[8]>10) {
        pause=1; //自動描画を止める識別子
    }
    break;

case 285:                //パチンコ玉の落下した位置の x 座標
    cy[9]=cy[9]+1;      //この位置に落下したパチンコ玉のカウンタ

    if (cy[9]>10) {
        pause=1; //自動描画を止める識別子
    }
    break;

case 290:                //パチンコ玉の落下した位置の x 座標
    cy[10]=cy[10]+1;    //この位置に落下したパチンコ玉のカウンタ

    if (cy[10]>10) {
        pause=1; //自動描画を止める識別子
    }
    break;

case 295:                //パチンコ玉の落下した位置の x 座標
    cy[11]=cy[11]+1;    //この位置に落下したパチンコ玉のカウンタ

    if (cy[11]>10) {
        pause=1; //自動描画を止める識別子
    }
    break;

case 300:                //パチンコ玉の落下した位置の x 座標
    cy[12]=cy[12]+1;    //この位置に落下したパチンコ玉のカウンタ

    if (cy[12]>10) {
        pause=1; //自動描画を止める識別子
    }

```



```

    }
    break;
case 305:          //パチンコ玉の落下した位置の x 座標
    cy[13]=cy[13]+1;    //この位置に落下したパチンコ玉のカウンタ

    if (cy[13]>10) {
        pause=1; //自動描画を止める識別子
    }
    break;
case 310:          //パチンコ玉の落下した位置の x 座標
    cy[14]=cy[14]+1;    //この位置に落下したパチンコ玉のカウンタ

    if (cy[14]>10) {
        pause=1; //自動描画を止める識別子
    }
    break;
case 315:          //パチンコ玉の落下した位置の x 座標
    cy[15]=cy[15]+1;    //この位置に落下したパチンコ玉のカウンタ

    if (cy[15]>10) {
        pause=1; //自動描画を止める識別子
    }
    break;
case 320:          //パチンコ玉の落下した位置の x 座標
    cy[16]=cy[16]+1;    //この位置に落下したパチンコ玉のカウンタ

    if (cy[16]>10) {
        pause=1; //自動描画を止める識別子
    }
    break;
case 325:          //パチンコ玉の落下した位置の x 座標
    cy[17]=cy[17]+1;    //この位置に落下したパチンコ玉のカウンタ

    if (cy[17]>10) {
        pause=1; //自動描画を止める識別子
    }
    break;
case 330:          //パチンコ玉の落下した位置の x 座標
    cy[18]=cy[18]+1;    //この位置に落下したパチンコ玉のカウンタ

```

```

        if (cy[18]>10) {
            pause=1; //自動描画を止める識別子
        }
        break;
case 335:                //パチンコ玉の落下した位置の x 座標
    cy[19]=cy[19]+1;    //この位置に落下したパチンコ玉のカウンタ

        if (cy[19]>10) {
            pause=1; //自動描画を止める識別子
        }
        break;
case 340:                //パチンコ玉の落下した位置の x 座標
    cy[20]=cy[20]+1;    //この位置に落下したパチンコ玉のカウンタ

        if (cy[20]>10) {
            pause=1; //自動描画を止める識別子
        }
        break;
case 345:                //パチンコ玉の落下した位置の x 座標
    cy[21]=cy[21]+1;    //この位置に落下したパチンコ玉のカウンタ

        if (cy[21]>10) {
            pause=1; //自動描画を止める識別子
        }
        break;
case 350:                //パチンコ玉の落下した位置の x 座標
    cy[22]=cy[22]+1;    //この位置に落下したパチンコ玉のカウンタ

        if (cy[22]>10) {
            pause=1; //自動描画を止める識別子
        }
        break;
case 355:                //パチンコ玉の落下した位置の x 座標
    cy[23]=cy[23]+1;    //この位置に落下したパチンコ玉のカウンタ

        if (cy[23]>10) {
            pause=1; //自動描画を止める識別子
        }
        break;
case 360:                //パチンコ玉の落下した位置の x 座標

```

```

cy[24]=cy[24]+1;    //この位置に落下したパチンコ玉のカウンタ
—

if (cy[24]>10) {
    pause=1; //自動描画を止める識別子
}
break;
case 365:           //パチンコ玉の落下した位置の x 座標
cy[25]=cy[25]+1;    //この位置に落下したパチンコ玉のカウンタ
—

if (cy[25]>10) {
    pause=1; //自動描画を止める識別子
}
break;
case 370:           //パチンコ玉の落下した位置の x 座標
cy[26]=cy[26]+1;    //この位置に落下したパチンコ玉のカウンタ
—

if (cy[26]>10) {
    pause=1; //自動描画を止める識別子
}
break;
case 375:           //パチンコ玉の落下した位置の x 座標
cy[27]=cy[27]+1;    //この位置に落下したパチンコ玉のカウンタ
—

if (cy[27]>10) {
    pause=1; //自動描画を止める識別子
}
break;
case 380:           //パチンコ玉の落下した位置の x 座標
cy[28]=cy[28]+1;    //この位置に落下したパチンコ玉のカウンタ
—

if (cy[28]>10) {
    pause=1; //自動描画を止める識別子
}
break;
case 385:           //パチンコ玉の落下した位置の x 座標
cy[29]=cy[29]+1;    //この位置に落下したパチンコ玉のカウンタ
—

if (cy[29]>10) {
    pause=1; //自動描画を止める識別子
}

```

```

        break;
case 235:                //パチンコ玉の落下した位置の x 座標
    cy[31]=cy[31]+1;    //この位置に落下したパチンコ玉のカウンタ

    if (cy[31]>10) {
        pause=1; //自動描画を止める識別子
    }
    break;
case 230:                //パチンコ玉の落下した位置の x 座標
    cy[32]=cy[32]+1;    //この位置に落下したパチンコ玉のカウンタ

    if (cy[32]>10) {
        pause=1; //自動描画を止める識別子
    }
    break;
case 225:                //パチンコ玉の落下した位置の x 座標
    cy[33]=cy[33]+1;    //この位置に落下したパチンコ玉のカウンタ

    if (cy[33]>10) {
        pause=1; //自動描画を止める識別子
    }
    break;
case 220:                //パチンコ玉の落下した位置の x 座標
    cy[34]=cy[34]+1;    //この位置に落下したパチンコ玉のカウンタ

    if (cy[34]>10) {
        pause=1; //自動描画を止める識別子
    }
    break;
case 215:                //パチンコ玉の落下した位置の x 座標
    cy[35]=cy[35]+1;    //この位置に落下したパチンコ玉のカウンタ

    if (cy[35]>10) {
        pause=1; //自動描画を止める識別子
    }
    break;
case 210:                //パチンコ玉の落下した位置の x 座標
    cy[36]=cy[36]+1;    //この位置に落下したパチンコ玉のカウンタ

    if (cy[36]>10) {

```

```

        pause=1; //自動描画を止める識別子
    }
    break;
case 205:                //パチンコ玉の落下した位置の x 座標
    cy[37]=cy[37]+1;    //この位置に落下したパチンコ玉のカウンタ

    if (cy[37]>10) {
        pause=1; //自動描画を止める識別子
    }
    break;
case 200:                //パチンコ玉の落下した位置の x 座標
    cy[38]=cy[38]+1;    //この位置に落下したパチンコ玉のカウンタ

    if (cy[38]>10) {
        pause=1; //自動描画を止める識別子
    }
    break;
case 195:                //パチンコ玉の落下した位置の x 座標
    cy[39]=cy[39]+1;    //この位置に落下したパチンコ玉のカウンタ

    if (cy[39]>10) {
        pause=1; //自動描画を止める識別子
    }
    break;
case 190:                //パチンコ玉の落下した位置の x 座標
    cy[40]=cy[40]+1;    //この位置に落下したパチンコ玉のカウンタ

    if (cy[40]>10) {
        pause=1; //自動描画を止める識別子
    }
    break;
case 185:                //パチンコ玉の落下した位置の x 座標
    cy[41]=cy[41]+1;    //この位置に落下したパチンコ玉のカウンタ

    if (cy[41]>10) {
        pause=1; //自動描画を止める識別子
    }
    break;
case 180:                //パチンコ玉の落下した位置の x 座標
    cy[42]=cy[42]+1;    //この位置に落下したパチンコ玉のカウンタ

```

```

if (cy[42]>10) {
    pause=1; //自動描画を止める識別子
}
break;
case 175:          //パチンコ玉の落下した位置の x 座標
cy[43]=cy[43]+1; //この位置に落下したパチンコ玉のカウンタ

if (cy[43]>10) {
    pause=1; //自動描画を止める識別子
}
break;
case 170:          //パチンコ玉の落下した位置の x 座標
cy[44]=cy[44]+1; //この位置に落下したパチンコ玉のカウンタ

if (cy[44]>10) {
    pause=1; //自動描画を止める識別子
}
break;
case 165:          //パチンコ玉の落下した位置の x 座標
cy[45]=cy[45]+1; //この位置に落下したパチンコ玉のカウンタ

if (cy[45]>10) {
    pause=1; //自動描画を止める識別子
}
break;
case 160:          //パチンコ玉の落下した位置の x 座標
cy[46]=cy[46]+1; //この位置に落下したパチンコ玉のカウンタ

if (cy[46]>10) {
    pause=1; //自動描画を止める識別子
}
break;
case 155:          //パチンコ玉の落下した位置の x 座標
cy[47]=cy[47]+1; //この位置に落下したパチンコ玉のカウンタ

if (cy[47]>10) {
    pause=1; //自動描画を止める識別子
}
break;

```

```

case 150:                //パチンコ玉の落下した位置の x 座標
    cy[48]=cy[48]+1;    //この位置に落下したパチンコ玉のカウンタ

    if (cy[48]>10) {
        pause=1; //自動描画を止める識別子
    }
    break;

case 145:                //パチンコ玉の落下した位置の x 座標
    cy[49]=cy[49]+1;    //この位置に落下したパチンコ玉のカウンタ

    if (cy[49]>10) {
        pause=1; //自動描画を止める識別子
    }
    break;

case 140:                //パチンコ玉の落下した位置の x 座標
    cy[50]=cy[50]+1;    //この位置に落下したパチンコ玉のカウンタ

    if (cy[50]>10) {
        pause=1; //自動描画を止める識別子
    }
    break;

case 135:                //パチンコ玉の落下した位置の x 座標
    cy[51]=cy[51]+1;    //この位置に落下したパチンコ玉のカウンタ

    if (cy[51]>10) {
        pause=1; //自動描画を止める識別子
    }
    break;

case 130:                //パチンコ玉の落下した位置の x 座標
    cy[52]=cy[52]+1;    //この位置に落下したパチンコ玉のカウンタ

    if (cy[52]>10) {
        pause=1; //自動描画を止める識別子
    }
    break;

case 125:                //パチンコ玉の落下した位置の x 座標
    cy[53]=cy[53]+1;    //この位置に落下したパチンコ玉のカウンタ

    if (cy[53]>10) {
        pause=1; //自動描画を止める識別子
    }

```

```

    }
    break;
case 120:          //パチンコ玉の落下した位置の x 座標
    cy[54]=cy[54]+1;    //この位置に落下したパチンコ玉のカウンタ

    if (cy[54]>10) {
        pause=1; //自動描画を止める識別子
    }
    break;
case 115:          //パチンコ玉の落下した位置の x 座標
    cy[55]=cy[55]+1;    //この位置に落下したパチンコ玉のカウンタ

    if (cy[55]>10) {
        pause=1; //自動描画を止める識別子
    }
    break;
case 110:          //パチンコ玉の落下した位置の x 座標
    cy[56]=cy[56]+1;    //この位置に落下したパチンコ玉のカウンタ

    if (cy[56]>10) {
        pause=1; //自動描画を止める識別子
    }
    break;
case 105:          //パチンコ玉の落下した位置の x 座標
    cy[57]=cy[57]+1;    //この位置に落下したパチンコ玉のカウンタ

    if (cy[57]>10) {
        pause=1; //自動描画を止める識別子
    }
    break;
case 100:          //パチンコ玉の落下した位置の x 座標
    cy[58]=cy[58]+1;    //この位置に落下したパチンコ玉のカウンタ

    if (cy[58]>10) {
        pause=1; //自動描画を止める識別子
    }
    break;
case 95:           //パチンコ玉の落下した位置の x 座標
    cy[59]=cy[59]+1;    //この位置に落下したパチンコ玉のカウンタ

```



```

        if (cy[59]>10) {
            pause=1; //自動描画を止める識別子
        }
        break;
    }

    paint.setColor(Color.BLUE);
    paint.setTextSize(15.0f);
    canvas.drawText("ふらつき回数 "+(k+1), (getWidth()/2-240)+190, (getHeight()
/2-343)+200, paint);

    k=0;
    x=220;
    y=100;
    ct++;

    paint.setColor(Color.BLUE);
    paint.setTextSize(20.0f);
    canvas.drawText("実験 "+ct+" 回目", (getWidth()/2-240)+190, (getHeight()
/2-343)+450, paint);

    r = Math.random();
    if (r<0.5) { //左に移動
        x=x-5;
    }
    else if (r>=0.5) { //右に移動
        x=x+5;
    }

    if (MainActivity.ritsu != 0) {
        a=(float) 0.7*320/MainActivity.ritsu; //----- <画像の拡大・縮小の横の倍率を指定する>
        b=(float) 0.7*320/MainActivity.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 (); //-----***
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)+x-4, (getHeight () /2-343)
+y, paint); //酔っぱらい画像の描写
}
}

paint.setColor (Color.BLUE);
canvas.drawRect ( ( getWidth () /2-240)+238, ( getHeight () /2-343)+390-cy [ 0] *14,
(getWidth () /2-240)+242, (getHeight () /2-343)+390,paint);
canvas.drawRect ( ( getWidth () /2-240)+243, ( getHeight () /2-343)+390-cy [ 1] *14,
(getWidth () /2-240)+247, (getHeight () /2-343)+390,paint);
canvas.drawRect ( ( getWidth () /2-240)+248, ( getHeight () /2-343)+390-cy [ 2] *14,
(getWidth () /2-240)+252, (getHeight () /2-343)+390,paint);
canvas.drawRect ( ( getWidth () /2-240)+253, ( getHeight () /2-343)+390-cy [ 3] *14,
(getWidth () /2-240)+257, (getHeight () /2-343)+390,paint);
canvas.drawRect ( ( getWidth () /2-240)+258, ( getHeight () /2-343)+390-cy [ 4] *14,
(getWidth () /2-240)+262, (getHeight () /2-343)+390,paint);
canvas.drawRect ( ( getWidth () /2-240)+263, ( getHeight () /2-343)+390-cy [ 5] *14,
(getWidth () /2-240)+267, (getHeight () /2-343)+390,paint);
canvas.drawRect ( ( getWidth () /2-240)+268, ( getHeight () /2-343)+390-cy [ 6] *14,
(getWidth () /2-240)+272, (getHeight () /2-343)+390,paint);
canvas.drawRect ( ( getWidth () /2-240)+273, ( getHeight () /2-343)+390-cy [ 7] *14,
(getWidth () /2-240)+277, (getHeight () /2-343)+390,paint);
canvas.drawRect ( ( getWidth () /2-240)+278, ( getHeight () /2-343)+390-cy [ 8] *14,
(getWidth () /2-240)+282, (getHeight () /2-343)+390,paint);
canvas.drawRect ( ( getWidth () /2-240)+283, ( getHeight () /2-343)+390-cy [ 9] *14,
(getWidth () /2-240)+287, (getHeight () /2-343)+390,paint);
canvas.drawRect ( ( getWidth () /2-240)+288, ( getHeight () /2-343)+390-cy [ 10] *14,
(getWidth () /2-240)+292, (getHeight () /2-343)+390,paint);
canvas.drawRect ( ( getWidth () /2-240)+293, ( getHeight () /2-343)+390-cy [ 11] *14,
(getWidth () /2-240)+297, (getHeight () /2-343)+390,paint);

```





```

(getWidth()/2-240)+132,(getHeight()/2-343)+390,paint);
        canvas.drawRect((getWidth()/2-240)+123, (getHeight()/2-343)+390-cy[53]*14,
(getWidth()/2-240)+127,(getHeight()/2-343)+390,paint);
        canvas.drawRect((getWidth()/2-240)+118, (getHeight()/2-343)+390-cy[54]*14,
(getWidth()/2-240)+122,(getHeight()/2-343)+390,paint);
        canvas.drawRect((getWidth()/2-240)+113, (getHeight()/2-343)+390-cy[55]*14,
(getWidth()/2-240)+117,(getHeight()/2-343)+390,paint);
        canvas.drawRect((getWidth()/2-240)+108, (getHeight()/2-343)+390-cy[56]*14,
(getWidth()/2-240)+112,(getHeight()/2-343)+390,paint);
        canvas.drawRect((getWidth()/2-240)+103, (getHeight()/2-343)+390-cy[57]*14,
(getWidth()/2-240)+107,(getHeight()/2-343)+390,paint);
        canvas.drawRect((getWidth()/2-240)+98, (getHeight()/2-343)+390-cy[58]*14,
(getWidth()/2-240)+102,(getHeight()/2-343)+390,paint);
        canvas.drawRect((getWidth()/2-240)+93, (getHeight()/2-343)+390-cy[59]*14,
(getWidth()/2-240)+97,(getHeight()/2-343)+390,paint);

        if (flag==1 && pause==0) {
            invalidate(); //自動描写
        }
    }

//private void init(Context context) {
    //Resources res = context.getResources();
    //bitmap1 = BitmapFactory.decodeResource(res, R.drawable.hito1);

    //WindowManager wm = (WindowManager) context.getSystemService
(Context.WINDOW_SERVICE);
    //Display disp = wm.getDefaultDisplay();
    //width = disp.getWidth();
    //height = disp.getHeight();
//}

@Override
public boolean onTouchEvent(MotionEvent event) {
    flag = flag + 1;
    flag = flag % 2;
    first++;

    syoki = syoki +1;
    if (syoki==5) {

```

```

        x=220;
        y=100;
        k=0;
        flag=0;
        ct=1;

        for (i=0;i<=59;i++) { //棒グラフの高さを0にする（配列を初期化する）
            cy[i]=0;
        }

        first=0;
        pause=0;
        syoki=0;
    }

    if (pause==0) {
        invalidate();
    }
    return false;
}
}

```

[ 2 ] activity\_main.xml

```

<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.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"
    tools:context=".MainActivity">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello World!"
        app:layout_constraintBottom_toBottomOf="parent"

```

```

        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

<jp.kiyo.wuena.myrandomwalks.MyRandomwalks
    android:id="@+id/myfview1"
    android:layout_height="match_parent"
    android:layout_width="match_parent"/>

</androidx.constraintlayout.widget.ConstraintLayout>

```

### [ 3 ] MainActivity.java

```
/*
```

```

-----
                ランダムウォーク
                Android 4.4 (Kit Kat)
                Copyright (C) K.Niwa 2019.12.12
-----

```

```
*/
```

```
package jp.kiyo.wuena.myrandomwalks;
```

```

import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
import android.util.DisplayMetrics;    //<画像の拡大・縮小に必要なライブラリ>
import android.app.Activity;
import android.view.Menu;

```

```
public class MainActivity extends AppCompatActivity {
```

```
    static int ritsu;
```

```
    @Override
```

```
    protected void onCreate (Bundle savedInstanceState) {
        super.onCreate (savedInstanceState);
        setContentView (R.layout.activity_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;
}
}
```