

[1]MyHari2.java

/*

ビュホンの針

Android 4.1 (Jelly Bean)

Copyright (C) K. Niwa 2021. 2. 11

*/

package jp.kiyo.wuena.myhari2;

import android.content.Context;

import android.graphics.Canvas;

import android.graphics.Color;

import android.graphics.Paint;

import android.graphics.Rect;

import android.graphics.RectF;

import android.util.AttributeSet;

import android.view.MotionEvent;

import android.view.View;

public class MyHari2 **extends** View { //View クラスを継承した Mybuffon クラス

//変数宣言と初期化

int flag=0; //針をまくか(1)、否か(2)、初期化するか(0)。

int N=9999; //まく針の最大本数

int i; //for ループに使用

int sum=0; //平行線に交わった針の本数

int px1; //針の端の x 座標を整数型にしたもの

int py1; //針の端の y 座標を整数型にしたもの

int px2; //針の端の x 座標を整数型にしたもの

int py2; //針の端の y 座標を整数型にしたもの

int yy; //平行線の作成に使用

int k=0; //まいた針の本数

double[] x1=new **double**[10001]; //針の端の x 座標

double[] y1=new **double**[10001]; //針の端の y 座標

```

double[] x2=new double[10001]; //針の端の x 座標
double[] y2=new double[10001]; //針の端の y 座標
double pai;    //πの近似値

public MyHari2(Context context, AttributeSet attrs, int defStyle) {    //コンストラクタ
    super(context, attrs, defStyle);
}

public MyHari2(Context context, AttributeSet attrs) {    //コンストラクタ
    super(context, attrs);
}

public MyHari2(Context context) {    //コンストラクタ
    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()/2-360)+10, (getHeight()/2-600)+10, (getWidth()/2-
360)+710, (getHeight()/2-600)+1190, paint);

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

    for (int i=0;i<2;i++) {
        canvas.drawLine((getWidth()/2-360)+10+i, (getHeight()/2-600)+10+i, (getWidth()/2-
360)+10+i, (getHeight()/2-600)+1190-i, paint);
        canvas.drawLine((getWidth()/2-360)+10+i, (getHeight()/2-600)+1190-i, (getWidth()/2-

```

```

360)+710-i, (getHeight()/2-600)+1190-i, paint);
        canvas.drawLine((getWidth()/2-360)+710-i, (getHeight()/2-600)+1190-i, (getWidth()/2-
360)+710-i, (getHeight()/2-600)+10+i, paint);
        canvas.drawLine((getWidth()/2-360)+710-i, (getHeight()/2-600)+10+i, (getWidth()/2-
360)+10+i, (getHeight()/2-600)+10+i, paint);
    }

    paint.setColor(Color.BLACK); //平行線枠の描画
    canvas.drawRect((getWidth()/2-360)+90+120, (getHeight()/2-600)+100, (getWidth()/2-
360)+390+120, (getHeight()/2-600)+400, paint);
    paint.setColor(Color.WHITE);
    canvas.drawRect((getWidth()/2-360)+91+120, (getHeight()/2-600)+101, (getWidth()/2-
360)+389+120, (getHeight()/2-600)+399, paint);

    paint.setColor(Color.BLACK); //平行線の描画
    for (yy=150;yy<=350;yy=yy+50) {
        canvas.drawLine((getWidth()/2-360)+90+120, (getHeight()/2-600)+yy, (getWidth()/2-
360)+390+120, (getHeight()/2-600)+yy, paint);
    }

    paint.setColor(Color.BLUE); //表題の表示
    paint.setTextSize(45.0f);
    canvas.drawText("【ビュホンの針】", (getWidth()/2-360)+150-24+20+41, (getHeight()/2-
600)+65, paint);

    paint.setColor(Color.BLACK); //表題の表示
    paint.setTextSize(18.0f);
    //canvas.drawText("平行線に交わった針は赤で表示します...", (getWidth()/2-240)+80,
(getHeight()/2-343)+90, paint);

    paint.setColor(Color.BLACK); //説明の表示
    paint.setTextSize(30.0f);
    canvas.drawText("※画面をタッチすると針を自動でまきます。", (getWidth()/2-360)+50,
(getHeight()/2-600)+950, paint);
    canvas.drawText("※もう一度タッチすると止まります。", (getWidth()/2-360)+50,
(getHeight()/2-600)+990, paint);
    canvas.drawText("※更にタッチすると初期化されます。", (getWidth()/2-360)+50,

```

```

(getHeight()/2-600)+1030, paint);
    canvas.drawText("※画面が暗くなったらタイトルバーをタッチ！", (getWidth()/2-360)+50,
(getHeight()/2-600)+1070, paint);

    //if (k==0) {
        canvas.drawText("■平行線に交わった針は赤で、交わらなかった", (getWidth()/2-
360)+50, (getHeight()/2-600)+690, paint);
        canvas.drawText(" 針は緑で表示します。", (getWidth()/2-360)+50, (getHeight()/2-
600)+720, paint);
        canvas.drawText("■「ばらまいた針の総数」を「平行線に交わっ", (getWidth()/2-
360)+50, (getHeight()/2-600)+760, paint);
        canvas.drawText(" た針の本数」で割ると、円周率 $\pi$ になってい", (getWidth()/2-
360)+50, (getHeight()/2-600)+790, paint);
        canvas.drawText(" ることを観察してみましょう。", (getWidth()/2-360)+50,
(getHeight()/2-600)+820, paint);
        canvas.drawText("■ただし、平行線の間隔は針の長さの2倍にし", (getWidth()/2-
360)+50, (getHeight()/2-600)+860, paint);
        canvas.drawText(" ています。", (getWidth()/2-360)+50, (getHeight()/2-600)+890,
paint);
    //}

    paint.setColor(Color.BLUE); //作者・作成年月の表示
    paint.setTextSize(30.0f);
    canvas.drawText("Copyright(C) K.Niwa 2021.2.11", (getWidth()/2-360)+150+10,
(getHeight()/2-600)+1130, paint);

    k=k+1; //まいた針の本数を1本増やす

    x1[k]=115+250*Math.random()+120; //k番目の針の両端の位置の座標
    (x1, y1), (x2, y2) を乱数で決める
    y1[k]=125+250*Math.random();
    x2[k]=x1[k]+25*Math.cos(2*Math.PI*Math.random());
    y2[k]=y1[k]+25*Math.sin(2*Math.PI*Math.random());

    if (y1[k]>y2[k]) { //k番目の針が平行線と交わっているか否かの

```

判断

```
if (y1[k]>150 && y2[k]<150) {
    sum++;
}
else if (y1[k]>200 && y2[k]<200) {
    sum++;
}
else if (y1[k]>250 && y2[k]<250) {
    sum++;
}
else if (y1[k]>300 && y2[k]<300) {
    sum++;
}
else if (y1[k]>350 && y2[k]<350) {
    sum++;
}
else if (y1[k]==150 || y2[k]==150) {
    sum++;
}
else if (y1[k]==200 || y2[k]==200) {
    sum++;
}
else if (y1[k]==250 || y2[k]==250) {
    sum++;
}
else if (y1[k]==300 || y2[k]==300) {
    sum++;
}
else if (y1[k]==350 || y2[k]==350) {
    sum++;
}
}
else if (y1[k]<y2[k]) {
    if (y1[k]<150 && y2[k]>150) {
        sum++;
    }
}
```

```

else if (y1[k]<200 && y2[k]>200) {
    sum++;
}
else if (y1[k]<250 && y2[k]>250) {
    sum++;
}
else if (y1[k]<300 && y2[k]>300) {
    sum++;
}
else if (y1[k]<350 && y2[k]>350) {
    sum++;
}
else if (y1[k]==150 || y2[k]==150) {
    sum++;
}
else if (y1[k]==200 || y2[k]==200) {
    sum++;
}
else if (y1[k]==250 || y2[k]==250) {
    sum++;
}
else if (y1[k]==300 || y2[k]==300) {
    sum++;
}
else if (y1[k]==350 || y2[k]==350) {
    sum++;
}
}
else if (y1[k]==y2[k]) {
    if (y1[k]==150 || y1[k]==200 || y1[k]==250 || y1[k]==300 || y1[k]==350) {
        sum++;
    }
}
}

for (i=1;i<=k;i++) {    //k本目の針をまく

```

```

    px1=(int)(x1[i]); //針の端のx座標を整数型にキャストする
    py1=(int)(y1[i]); //針の端のy座標を整数型にキャストする
    px2=(int)(x2[i]); //針の端のx座標を整数型にキャストする
    py2=(int)(y2[i]); //針の端のy座標を整数型にキャストする
    paint.setColor(Color.RED); //針の色を赤にする

    if (y1[i]>y2[i]) {
        if (y1[i]>150 && y2[i]<150) {
            canvas.drawLine((getWidth()/2-360)+px1, (getHeight()/2-
600)+py1, (getWidth()/2-360)+px2, (getHeight()/2-600)+py2, paint);
            canvas.drawLine((getWidth()/2-360)+px1-1, (getHeight()/2-
600)+py1, (getWidth()/2-360)+px2-1, (getHeight()/2-600)+py2, paint);
        }
        else if (y1[i]>200 && y2[i]<200) {
            canvas.drawLine((getWidth()/2-360)+px1, (getHeight()/2-
600)+py1, (getWidth()/2-360)+px2, (getHeight()/2-600)+py2, paint);
            canvas.drawLine((getWidth()/2-360)+px1-1, (getHeight()/2-
600)+py1, (getWidth()/2-360)+px2-1, (getHeight()/2-600)+py2, paint);
        }
        else if (y1[i]>250 && y2[i]<250) {
            canvas.drawLine((getWidth()/2-360)+px1, (getHeight()/2-
600)+py1, (getWidth()/2-360)+px2, (getHeight()/2-600)+py2, paint);
            canvas.drawLine((getWidth()/2-360)+px1-1, (getHeight()/2-
600)+py1, (getWidth()/2-360)+px2-1, (getHeight()/2-600)+py2, paint);
        }
        else if (y1[i]>300 && y2[i]<300) {
            canvas.drawLine((getWidth()/2-360)+px1, (getHeight()/2-
600)+py1, (getWidth()/2-360)+px2, (getHeight()/2-600)+py2, paint);
            canvas.drawLine((getWidth()/2-360)+px1-1, (getHeight()/2-
600)+py1, (getWidth()/2-360)+px2-1, (getHeight()/2-600)+py2, paint);
        }
        else if (y1[i]>350 && y2[i]<350) {
            canvas.drawLine((getWidth()/2-360)+px1, (getHeight()/2-
600)+py1, (getWidth()/2-360)+px2, (getHeight()/2-600)+py2, paint);
            canvas.drawLine((getWidth()/2-360)+px1-1, (getHeight()/2-
600)+py1, (getWidth()/2-360)+px2-1, (getHeight()/2-600)+py2, paint);
        }
    }

```

```

    }

    else if (y1[i]==150 || y2[i]==150) {
        canvas.drawLine((getWidth()/2-360)+px1, (getHeight()/2-
600)+py1, (getWidth()/2-360)+px2, (getHeight()/2-600)+py2, paint);
        canvas.drawLine((getWidth()/2-360)+px1-1, (getHeight()/2-
600)+py1, (getWidth()/2-360)+px2-1, (getHeight()/2-600)+py2, paint);
    }

    else if (y1[i]==200 || y2[i]==200) {
        canvas.drawLine((getWidth()/2-360)+px1, (getHeight()/2-
600)+py1, (getWidth()/2-360)+px2, (getHeight()/2-600)+py2, paint);
        canvas.drawLine((getWidth()/2-360)+px1-1, (getHeight()/2-
600)+py1, (getWidth()/2-360)+px2-1, (getHeight()/2-600)+py2, paint);
    }

    else if (y1[i]==250 || y2[i]==250) {
        canvas.drawLine((getWidth()/2-360)+px1, (getHeight()/2-
600)+py1, (getWidth()/2-360)+px2, (getHeight()/2-600)+py2, paint);
        canvas.drawLine((getWidth()/2-360)+px1-1, (getHeight()/2-
600)+py1, (getWidth()/2-360)+px2-1, (getHeight()/2-600)+py2, paint);
    }

    else if (y1[i]==300 || y2[i]==300) {
        canvas.drawLine((getWidth()/2-360)+px1, (getHeight()/2-
600)+py1, (getWidth()/2-360)+px2, (getHeight()/2-600)+py2, paint);
        canvas.drawLine((getWidth()/2-360)+px1-1, (getHeight()/2-
600)+py1, (getWidth()/2-360)+px2-1, (getHeight()/2-600)+py2, paint);
    }

    else if (y1[i]==350 || y2[i]==350) {
        canvas.drawLine((getWidth()/2-360)+px1, (getHeight()/2-
600)+py1, (getWidth()/2-360)+px2, (getHeight()/2-600)+py2, paint);
        canvas.drawLine((getWidth()/2-360)+px1-1, (getHeight()/2-
600)+py1, (getWidth()/2-360)+px2-1, (getHeight()/2-600)+py2, paint);
    }

    else {
        paint.setColor(Color. GREEN);           //針の色を緑にする
        canvas.drawLine((getWidth()/2-360)+px1, (getHeight()/2-
600)+py1, (getWidth()/2-360)+px2, (getHeight()/2-600)+py2, paint); //針を描く
        paint.setColor(Color. RED);             //針の色を赤にする
    }

```

```

    }
}
else if (y1[i]<y2[i]) {
    if (y1[i]<150 && y2[i]>150) {
        canvas.drawLine((getWidth()/2-360)+px1, (getHeight()/2-
600)+py1, (getWidth()/2-360)+px2, (getHeight()/2-600)+py2, paint);
        canvas.drawLine((getWidth()/2-360)+px1-1, (getHeight()/2-
600)+py1, (getWidth()/2-360)+px2-1, (getHeight()/2-600)+py2, paint);
    }
    else if (y1[i]<200 && y2[i]>200) {
        canvas.drawLine((getWidth()/2-360)+px1, (getHeight()/2-
600)+py1, (getWidth()/2-360)+px2, (getHeight()/2-600)+py2, paint);
        canvas.drawLine((getWidth()/2-360)+px1-1, (getHeight()/2-
600)+py1, (getWidth()/2-360)+px2-1, (getHeight()/2-600)+py2, paint);
    }
    else if (y1[i]<250 && y2[i]>250) {
        canvas.drawLine((getWidth()/2-360)+px1, (getHeight()/2-
600)+py1, (getWidth()/2-360)+px2, (getHeight()/2-600)+py2, paint);
        canvas.drawLine((getWidth()/2-360)+px1-1, (getHeight()/2-
600)+py1, (getWidth()/2-360)+px2-1, (getHeight()/2-600)+py2, paint);
    }
    else if (y1[i]<300 && y2[i]>300) {
        canvas.drawLine((getWidth()/2-360)+px1, (getHeight()/2-
600)+py1, (getWidth()/2-360)+px2, (getHeight()/2-600)+py2, paint);
        canvas.drawLine((getWidth()/2-360)+px1-1, (getHeight()/2-
600)+py1, (getWidth()/2-360)+px2-1, (getHeight()/2-600)+py2, paint);
    }
    else if (y1[i]<350 && y2[i]>350) {
        canvas.drawLine((getWidth()/2-360)+px1, (getHeight()/2-
600)+py1, (getWidth()/2-360)+px2, (getHeight()/2-600)+py2, paint);
        canvas.drawLine((getWidth()/2-360)+px1-1, (getHeight()/2-
600)+py1, (getWidth()/2-360)+px2-1, (getHeight()/2-600)+py2, paint);
    }
    else if (y1[i]==150 || y2[i]==150) {
        canvas.drawLine((getWidth()/2-360)+px1, (getHeight()/2-
600)+py1, (getWidth()/2-360)+px2, (getHeight()/2-600)+py2, paint);

```

```

        canvas.drawLine((getWidth()/2-360)+px1-1, (getHeight()/2-
600)+py1, (getWidth()/2-360)+px2-1, (getHeight()/2-600)+py2, paint);
    }
    else if (y1[i]==200 || y2[i]==200) {
        canvas.drawLine((getWidth()/2-360)+px1, (getHeight()/2-
600)+py1, (getWidth()/2-360)+px2, (getHeight()/2-600)+py2, paint);
        canvas.drawLine((getWidth()/2-360)+px1-1, (getHeight()/2-
600)+py1, (getWidth()/2-360)+px2-1, (getHeight()/2-600)+py2, paint);
    }
    else if (y1[i]==250 || y2[i]==250) {
        canvas.drawLine((getWidth()/2-360)+px1, (getHeight()/2-
600)+py1, (getWidth()/2-360)+px2, (getHeight()/2-600)+py2, paint);
        canvas.drawLine((getWidth()/2-360)+px1-1, (getHeight()/2-
600)+py1, (getWidth()/2-360)+px2-1, (getHeight()/2-600)+py2, paint);
    }
    else if (y1[i]==300 || y2[i]==300) {
        canvas.drawLine((getWidth()/2-360)+px1, (getHeight()/2-
600)+py1, (getWidth()/2-360)+px2, (getHeight()/2-600)+py2, paint);
        canvas.drawLine((getWidth()/2-360)+px1-1, (getHeight()/2-
600)+py1, (getWidth()/2-360)+px2-1, (getHeight()/2-600)+py2, paint);
    }
    else if (y1[i]==350 || y2[i]==350) {
        canvas.drawLine((getWidth()/2-360)+px1, (getHeight()/2-
600)+py1, (getWidth()/2-360)+px2, (getHeight()/2-600)+py2, paint);
        canvas.drawLine((getWidth()/2-360)+px1-1, (getHeight()/2-
600)+py1, (getWidth()/2-360)+px2-1, (getHeight()/2-600)+py2, paint);
    }
    else {
        paint.setColor(Color.GREEN);           //針の色を緑にする
        canvas.drawLine((getWidth()/2-360)+px1, (getHeight()/2-
600)+py1, (getWidth()/2-360)+px2, (getHeight()/2-600)+py2, paint); //針を描く
        paint.setColor(Color.RED);           //針の色を赤にする
    }
}

else if (y1[k]==y2[k]) {
    if (y1[k]==150 || y1[k]==200 || y1[k]==250 || y1[k]==300 || y1[k]==350) {

```

```

        canvas.drawLine((getWidth()/2-360)+px1, (getHeight()/2-
600)+py1, (getWidth()/2-360)+px2, (getHeight()/2-600)+py2, paint); //針を描く
        canvas.drawLine((getWidth()/2-360)+px1, (getHeight()/2-600)+py1-
1, (getWidth()/2-360)+px2, (getHeight()/2-600)+py2-1, paint); //針を描く
    }
    else {
        paint.setColor(Color. GREEN); //針の色を緑にする
        canvas.drawLine((getWidth()/2-360)+px1, (getHeight()/2-
600)+py1, (getWidth()/2-360)+px2, (getHeight()/2-600)+py2, paint); //針を描く
        paint.setColor(Color. RED); //針の色を赤にする
    }
}
} //for (i=1; i<=k; i++)

if (sum!=0) {
    pai=(double)k/sum; //実験結果から $\pi$ を計算し倍精度型にする
}
else if (sum==0) { //0で割ったときの例外処理
    pai=0;
}

paint.setColor(Color. BLUE); //実験結果の表示
paint.setTextSize(40.0f);
canvas.drawText("円周率 $\pi$ ="+pai, (getWidth()/2-360)+50, (getHeight()/2-600)+510-20,
paint);
paint.setColor(Color. BLACK);
//paint.setTextSize(20.0f);
canvas.drawText("平行線に交った針の本数="+sum+" 本", (getWidth()/2-360)+50,
(getHeight()/2-600)+570-20, paint);
canvas.drawText("ばらまいた針の総数="+k+" 本", (getWidth()/2-360)+50, (getHeight()/2-
600)+630-20, paint);

if (k<=N && flag==1) {
    invalidate(); //再描画、clear & goto onDraw
}

```

```

        if (k==N) {                //針を最大本数まいたとき
            flag=2;                //針をまくとのを止める
        }

    } //protected void onDraw(Canvas canvas) {

    //画面をタッチしたときのイベント処理-----
    -----

    @Override
    public boolean onTouchEvent(MotionEvent event) {

        flag=flag+1;                //針をまくか(1)、否か(2)、初期化するか(0)
        flag=flag % 3;              //flag には、1、2、0が入る

        if (flag==0) {              //初期化する
            sum=0;
            k=0;
        }

        invalidate();              //再描画、clear & goto onDraw
        return false;
    }

}

```

[2]activity_main.xml

```

<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" />
```

```
<?xml xmlns:android="http://schemas.android.com/apk/res/android"
    package="jp.kiyo.wuena.myhari2"
    android:layout_width="match_parent"
    android:layout_height="match_parent" />
```

```
</androidx.constraintlayout.widget.ConstraintLayout>
```

[3]MainActivity.java

```
/*
```

```
-----
        ビュホンの針
        Android 4.1 (Jelly Bean)
        Copyright (C) K. Niwa 2021. 2. 11
    -----
```

```
*/
```

```
package jp.kiyo.wuena.myhari2;
```

```
import androidx.appcompat.app.AppCompatActivity;
```

```
import android.os.Bundle;
```

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

```
    @Override
```

```
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
```

```
        setContentView(R.layout.activity_main);  
    }  
}
```