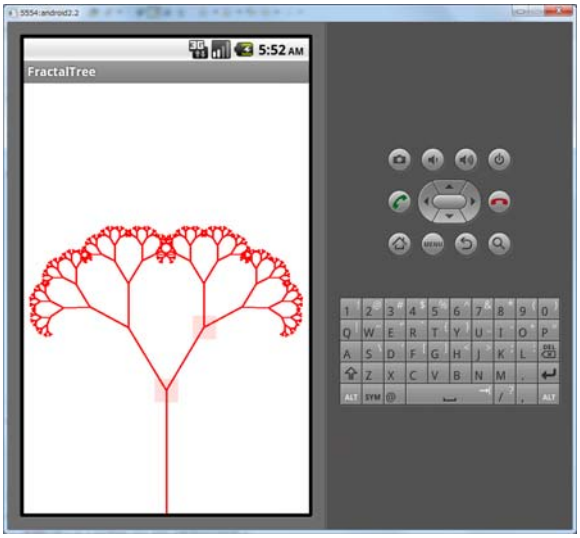
Android Fractal Tree

2012 年 1 月 20 日

徐　丙鉄




```
package so.kindai;
import android.app.Activity;
import android.os.Bundle;
import android.view.KeyEvent;

public class FractalTree extends Activity {

        FractalTreeSView fractalTreeView ;    //  描画 SurfaceView

        /** Called when the activity is first created. */
        @Override
        public void onCreate(Bundle savedInstanceState) {
           super.onCreate(savedInstanceState);
           fractalTreeView = new FractalTreeSView(this);
           setContentView(fractalTreeView);
        }
        // View のサイズは onCreate の時点では確定していないので、このタイミングで取得し、描画のために設定
        public void onWindowFocusChanged(boolean hasFocus){
                super.onWindowFocusChanged(hasFocus);
                fractalTreeView.setWH( fractalTreeView.getWidth(), fractalTreeView.getHeight() );
        }
       //数字キーで Fractal Tree の枝別れの回数を指定
        public boolean onKeyDown(int keyCode, KeyEvent ev) {
           switch(keyCode){
                case KeyEvent.KEYCODE_0:
                        fractalTreeView.setLevel(0);          return true;
                case KeyEvent.KEYCODE_1:
                        fractalTreeView.setLevel(1);          return true;
                case KeyEvent.KEYCODE_2:
                        fractalTreeView.setLevel(2);          return true;
                case KeyEvent.KEYCODE_3:
                        fractalTreeView.setLevel(3);          return true;
                case KeyEvent.KEYCODE_4:
                        fractalTreeView.setLevel(4);          return true;
                case KeyEvent.KEYCODE_5:
                        fractalTreeView.setLevel(5);          return true;
                case KeyEvent.KEYCODE_6:
                        fractalTreeView.setLevel(6);          return true;
                case KeyEvent.KEYCODE_7:
                        fractalTreeView.setLevel(7);          return true;
                case KeyEvent.KEYCODE_8:
                        fractalTreeView.setLevel(8);          return true;
                case KeyEvent.KEYCODE_9:
                        fractalTreeView.setLevel(9);          return true;
           }
           return true;
        }

}
```

java.lang.Object
↳ android.view.**View**
↳ android.view.**SurfaceView**

```
package so.kindai;
import android.content.Context;     import android.graphics.Canvas;
import android.graphics.Color;      import android.graphics.Paint;
import android.view.MotionEvent;    import android.view.SurfaceHolder;
import android.view.SurfaceView;
/* 数字キーでFractal Treeの枝別れの回数を指定し，Fractal Tree上の赤いドラッグBOXをドラッグすると、
 * 幹の長さが変化し、幹と一次の枝の長さの比scaleと、枝の角度angleが変化する。
 */
public class FractalTreeSView extends SurfaceView implements SurfaceHolder.Callback{

    //--- fractal tree の初期値 ----------------------------------------//
    int angle = 30; double length = 128;        //枝の角度，幹の長さ
    double scale = 0.7; int level = 8;          //枝の縮小率，枝分かれの回数
    float BX1, BY1;           //  幹の終点座標,ここにドラッグBOX描画
    float BX2, BY2;           //  一次の枝の終点の座標,ここにドラッグBOX描画
    float TX,TY;              //  touch pointの座標
    float R = (float) 20.0;   //ドラッグBOXのサイズ，touch pointの誤差,
    float W, H;               //画面の幅と高さ

    double degree = Math.PI/180.0;
    SurfaceHolder holder;                       // SurfaceHolder経由でSurfaceViewにアクセスする
    Paint paint, paintBG, paintDragBox;         //Fractal Tree，背景，ドラッグBOXのpaint

  //コンストラクタ
  public FractalTreeSView(Context context) {
    super(context);
    holder = getHolder();    // SurfaceViewのSurfaceHolderを取得
    holder.addCallback(this);
    holder.setFixedSize(getWidth(),getHeight());
    setFocusable(true);

    //ペイント(絵具と刷毛)の準備
    paint = new Paint();     paintBG = new Paint();      paintDragBox = new Paint();
    paint.setStrokeWidth(3); paintBG.setStrokeWidth(1); paintDragBox.setStrokeWidth(1);
    paint.setStyle(Paint.Style.FILL);paintBG.setStyle(Paint.Style.FILL);
    paintDragBox.setStyle(Paint.Style.FILL);
    paint.setColor(Color.argb(255,255,0,0));
    paintBG.setColor(Color.argb(255,255,255,255));
    paintDragBox.setColor(Color.argb(32,255,0,0));
  }

  public void surfaceCreated(SurfaceHolder h){ doDraw(h); }

 //SurfaceHolder経由でCanvasに描画するメソッドを定義し、再描画の際はこれを呼び出す
 private void doDraw(SurfaceHolder h){
    Canvas c = h.lockCanvas();        onDraw(c);        h.unlockCanvasAndPost(c);
 }

  public boolean onTouchEvent(MotionEvent event){

    switch(event.getAction()){
      case MotionEvent.ACTION_MOVE:
           TX = event.getX(); TY = event.getY();
```

```
            //幹の終点がドラッグされたら
            if( TX > (BX1-R) && TX < (BX1+R) ){
                    if( TY > (BY1-R) && TY < (BY1+R) ){
                            length = H - TY;   break;
                    }
            }

            //一次の枝の終点がドラッグされたら
            if( TX > (BX2-R) && TX < (BX2+R) ){
                if( TY > (BY2-R) && TY < (BY2+R) ){
                      float x = TX-BX1; float y = BY1-TY;
                      scale = Math.sqrt( x*x + y*y )/length;
                      angle = (int) Math.toDegrees( Math.atan2(x, y) );
                      break;
                }
                break;
     }
   }

    doDraw( holder );          return true;
  }
```

// Canvas への描画

```
protected void onDraw(Canvas c){
    c.save();
    c.drawRect(0, 0, W, H, paintBG);  //背景色で塗りつぶして画面消去
    BX1=(float) (W*0.5);      BY1= (float) (H-length);    //幹の終点座標
    drawTree( c, level, W/2, H, length, 0);     //Fractal Tree
    c.drawRect(BX1-R, BY1+R, BX1+R, BY1-R, paintDragBox);          //幹のドラッグ BOX 描画
    c.drawRect(BX2-R, BY2+R, BX2+R, BY2-R, paintDragBox);          //枝のドラッグ BOX 描画
    c.restore();
}
```

//Fractal Tree を描く再帰アルゴリズム

```
public void drawTree(Canvas c,  int n, double x1, double y1, double len, double stand){

    double x2 = x1 + len*Math.sin( Math.toRadians(stand) );
    double y2 = y1 - len*Math.cos( Math.toRadians(stand) );

    if(n == level-1 ){ BX2=(float) x2; BY2=(float) y2; } // 一次の枝の終点座標

    c.drawLine( (int)x1, (int)y1, (int)x2, (int)y2, paint );

    if( n>=1 ){
          drawTree( c, n-1, x2, y2, len*scale, stand - angle );
          drawTree( c, n-1, x2, y2, len*scale, stand + angle );
    }
}
```

```
 public void setLevel(int lev){ level = lev; doDraw(holder);}
 public void setWH(float w, float h){W = w; H = h; length=(int)h/3; doDraw(holder);}
 public void surfaceDestroyed(SurfaceHolder holder) { }
 public void surfaceChanged(SurfaceHolder arg0, int arg1, int arg2,int arg3) {}

}
```