

## SurfaceView のアニメーション

定期的に画面を書き換える例を示すために、「壁にあたって跳ね返る図形のアニメーション」を、SurfaceView クラスを使って実現してみます。

### ■コードのポイント解説

画面のサイズを取得するのに、SurfaceHolder.Callback インターフェースには surfaceChanged メソッドがあるので、これを利用しています。

また、定期的に画面を更新するためのタイマーとして、別スレッドから直接 GUI 操作ができるので、(SurfaceView クラスに Runnable インターフェースをインプリメントして) java で標準に使われている Thread クラスを使っています。

プログラム終了時にスレッドを終了させないと、終了後もスレッドが生き残ってしまうので、この処理を surfaceDestroyed メソッドに実装しています。

Log 出力は、プログラム開始時にスクリーンサイズを取得したり、プログラム終了時にスレッドが停止するかを確認するためのものです。

canvas#drawColor メソッドを追加して、背景を白色に塗りつぶしていますが、これは、View クラスでは前の画面がクリアされた後に画面が描画されるのに対して、SurfaceView では前の画面が残ったままになってしまうため、前の画面を消去するためにいれてあります。

### ●サンプルコード

```
package jp.surfaceView.animation;

import android.app.Activity;
import android.content.Context;
import android.graphics.Canvas;
import android.graphics.Color;
import android.graphics.Paint;
```

```

import android.os.Bundle;
import android.util.Log;
import android.view.SurfaceHolder;
import android.view.SurfaceView;

/**
 * 円が壁に跳ね返りながら移動するアニメーションサンプル
 *
 */
public class SurfaceViewAnimation01 extends Activity {

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        // 下記クラスを画面にセット
        setContentView(new CustomSurfaceView(this));
    }
}

// -----

// SurfaceView クラスを継承した描画処理担当クラス
class CustomSurfaceView extends SurfaceView implements SurfaceHolder.Callback,
    Runnable {
    Thread thread;
    boolean isAttached;

    private static final float circleR = 15; // 円の半径

    private float dx = 5, dy = 5; // 移動ピクセル数
    private float screenWidth, screenHeight;
    private float xOffset = 20, yOffset = 20; // 動作開始時の円の中心点座標

    public CustomSurfaceView(Context context) {
        super(context);
        getHolder().addCallback(this);
    }
}

```

```

    }

    public void surfaceChanged(SurfaceHolder holder, int format, int width,
                               int height) {
        Log.v("SurfaceViewAnimation01", "Changed");
        // スクリーン（画面）の縦・横サイズが引数から受け取れる
        screenWidth = width;
        screenHeight = height;
    }

    public void surfaceCreated(SurfaceHolder holder) {
        Log.v("SurfaceViewAnimation01", "Created");
        isAttached = true; // このクラスが生成時、下記、run() の処理が無限ループで動く

        thread = new Thread(this);
        thread.start();
    }

    public void surfaceDestroyed(SurfaceHolder holder) {
        isAttached = false; // このクラスが消滅時、下記、run() の処理が無限ループが while(false) となり、停止する
        thread = null; // スレッドオブジェクトを無化する
        Log.v("SurfaceViewAnimation01", "Destroyed& Thread stop!");
    }

    // 定期的に繰り返し更新処理を行う場合は、並行処理スレッドを使う
    public void run() {
        while (isAttached) {
            Log.v("SurfaceViewAnimation01", "run");

            // 周囲の境界線に到達したら、移動ピクセル数の正負を反転し、向きを変える
            if (xOffset - circleR < 0 || xOffset + circleR > screenWidth)
                dx = -dx;
            if (yOffset - circleR < 0 || yOffset + circleR > screenHeight)
                dy = -dy;
            // 描画座標点を変更

```

```

        xOffset += dx;
        yOffset += dy;

        doDraw(getHolder()); // 再描画によりアニメーションとなる
    }
}

// 描画処理記述
private void doDraw(SurfaceHolder holder) {
    Canvas canvas = holder.lockCanvas();
    Paint paint = new Paint();
    paint.setColor(Color.RED);
    canvas.drawColor(Color.WHITE);
    paint.setAntiAlias(true);
    //円を描画
    canvas.drawCircle(xOffset, yOffset, circleR, paint);
    holder.unlockCanvasAndPost(canvas);
}
}

```

---

以上