２　イメージの回転・反転

2.1 ソースコード

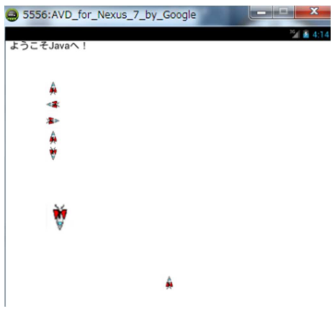
(1)テンプレート　Step の①の箇所に　Step020View　を入力してください。

(2)次のアプリケーションを新規作成してください。

Step010View をコピー＆ペーストして、ファイル名を「Step020View」に変更してください。

プロジェクト名：StepPro????(年組席)　アプリケーション名：Step020View

```
/*    年    組    席  名前
* Step020View
*   イメージの回転・反転
*/
・・・
import android.graphics.Matrix;
public class Step020View extends SurfaceView implements SurfaceHolder.Callback
{
    ・・・
    private final String TEXT1="ようこそ Java へ！";
    private float rotateX; //回転の中心の X
    private float rotateY; //回転の中心の y
    //コンストラクタ
    public Step020View(Context context)
    {
        ・・・
        this.rotateX = this.image.getWidth() / 2.0f;
        this.rotateY = this.image.getHeight() / 2.0f;
    }
    //サーフェイスの生成
    public void surfaceCreated(SurfaceHolder holder)
    {
        ・・・
        canvas.drawBitmap(image, getWidth()/2-16, getHeight()/2-16, paint);
        Matrix matrix= new Matrix();//①
        canvas.save();//②
        // Bitmap を 0 度回転
        matrix.postRotate(0.0f,rotateX,rotateY);//③
        image = Bitmap.createBitmap(image, 0, 0, image.getWidth(),
                                    image.getHeight(), matrix, true);
        canvas.drawBitmap(image, 100, 100, null);
        //Bitmap を反時計回りに 90 度回転
        matrix.postRotate(-90.0f,rotateX,rotateY);
        image = Bitmap.createBitmap(image, 0, 0, image.getWidth(),
                                    image.getHeight(), matrix, true);
        canvas.drawBitmap(image, 100, 140, null);
        //Bitmap の左右を逆に変更
        matrix = new Matrix();
        matrix.preScale(-1, 1);//④水平反転
        image = Bitmap.createBitmap(image, 0, 0, image.getWidth(),
```



```
                                    image.getHeight(), matrix, true);
        canvas.drawBitmap(image, 100, 180, null);
        //Bitmap を時計回りで 90 度回転
        matrix.postRotate(90.0f,rotateX,rotateY);
        image = Bitmap.createBitmap(image, 0, 0, image.getWidth(),
                                    image.getHeight(), matrix, true);
        canvas.drawBitmap(image, 100, 220, null);
        //Bitmap の上下を逆に変更
        matrix = new Matrix();
        matrix.preScale(1, -1);//垂直反転
        image = Bitmap.createBitmap(image, 0, 0, image.getWidth(),
                                    image.getHeight(), matrix, true);
        canvas.drawBitmap(image, 100, 260, null);
        //⑤Bitmap の縦横の長さを２倍に変更
        matrix = new Matrix();
        matrix.postScale(2.0f, 2.0f,rotateX, rotateY);
        image = Bitmap.createBitmap(image, 0, 0, image.getWidth(),
                                    image.getHeight(), matrix, true);
        canvas.drawBitmap(image, 100, 400, null);
        canvas.restore();//⑥
        //Canvas のロックを解除する。実画面に反映
        holder.unlockCanvasAndPost(canvas);
    }
    //サーフェイスの変更
    public void surfaceChanged(SurfaceHolder holder, int format, int w, int h){}
    //サーフェイスの破棄
    public void surfaceDestroyed(SurfaceHolder holder){ }
}
```

2.2　画像の表示

①Matrix matrix= new Matrix();

Matrix オブジェクトを操作すると、Bitmap 画像の回転やスケールの変更等の操作をおこなう事ができます。「Matrix に回転設定をして、Bitmap に適用する」という意味です。

②canvas.save();
⑥canvas.restore();

View#onDraw メソッドの表示処理では Canvas を用いて描画を行います。複数の Bitmap やオブジェクトを表示したい場合は描画ごとに表示位置を計算する必要があります。

Canvas の状態を保存する主な目的は描画処理の簡略化です。描画位置がわかりやすくなるほか、任意のタイミングで Canvas の状態を復帰させられるメリットがあります。

・Canvas#save() 状態を保存する

・Canvas#restore() 保存した状態へ復帰する

Canvas#save/restore メソッドで保存できるのは回転、平行移動、図形のゆがみなどを表現する matrix、表示範囲を指定する clip、の 2 種類の情報です。アプリケーションのスクリーンショットを取得するように 1 ピクセルの色情報単位で保存・復帰することはできませんので注意してください。

③matrix.postRotate(0.0f,rotateX,rotateY);

Matrix の postRotate メソッドを使うと、Bitmap オブジェクトを回転させることができます。

④matrix.preScale(-1, 1);

postScale メソッドを使うと、スケールを変更することができます。

⑤matrix = new Matrix();
matrix.postScale(2.0f, 2.0f,rotateX, rotateY);
image = Bitmap.createBitmap(image, 0, 0, image.getWidth(),
image.getHeight(), matrix, true);
canvas.drawBitmap(image, 100, 400, null);

Bitmap.createBitmap()に Matrix を渡すことで拡大縮小、Canvas.drawBitmap()で範囲を指定して描画します。

書式
Bitmap createBitmap (Bitmap source, int x, int y,
int width, int height, Matrix m, boolean filter)
source　・・・描画するビットマップ
x　・・・描画先の x 座標
y　・・・描画先の y 座標
width　・・・各行のピクセル数の幅
height　・・・高さの行の数
m　・・・Matrix をピクセルに適用する
filter　・・・true：ソースをフィルター処理する必要がある。

書式
Canvas.drawBitmap(Bitmap bitmap, Matrix matrix, Paint paint)
bitmap　・・・描画するビットマップ
matrix　・・・ビットマップを変形するための情報を格納した Matrix 型の変数
paint　・・・描画時に使用する Paint クラス。使用しない場合は null

Matrix を用い、ビットマップ全体を変形して描画する例

Matrix matrix = new Matrix(); //Matrix オブジェクト生成
matrix.postScale(2.0f, 1.5f); //横:2.0, 縦:1.5 倍ズーム(マイナスも OK)
matrix.postRotate(45.0f); //45 度回転
matrix.postTranslate(240.0f, 480.0f); //(240.0, 480.0)へ移動
canvas.drawBitmap(charaBmp, matrix, null);

書式
matrix.postScale(float 図形の横の倍率, float 図形の縦の倍率);
matrix.postRotate(回転角度, float 図形の中心の X 座標,
float 図形の中心の Y 座標);
matrix.preScale(-1, 1);//水平反転
matrix.preScale(1, -1);//垂直反転