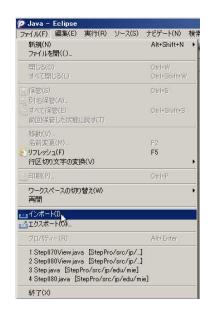
# V Android プログラミング【JUMP】

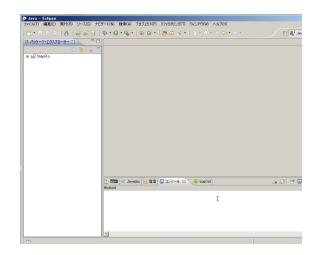
- 1 Jump の標準授業時数 時間 ( 月中旬まで)
- 2 実習用サーバーの「R:\footnotesia をの他\footnotesia JumpPro\_init」フォルダを「z:\footnotesia workspace」へインポートしてください。その方法は次のとおりです。
  - (1) Eclipse を起動して「ファイル」-「インポート」を選択してください。
  - (2) 「既存プロジェクトをワークスペースへ」を選択してください。





(3) 「ルートディレクトリの選択」の「参照」ボタンをクリックし、「R:¥その他 ¥Java¥JumpPro\_init」フォルダを選択してください。その際、「プロジェクト をワークスペースにコピー」をチェックしてください。





3 次のアプリケーションを新規作成し、①の箇所へ、それぞれの実習課題で指定されたソースを入力して、コンパイルし、実行結果を確認してください。

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

```
席 名前
/*
     年
           組
* Jump
      画面を横にする ctr+F11
*/
package jp.edu.mie;
import android.app.Activity;
import android.os.Bundle;
import android.view.Window;
public class Jump extends Activity
   public void onCreate(Bundle bundle)
       super.onCreate(bundle);
       requestWindowFeature(Window.FEATURE_NO_TITLE);//タイトルの非表示
       setContentView(new _______(this));//実画面に表示するビューの指定
   }
```

# 【実習1】 背景画像の表示と画像のアニメーション

# 1 ソースコード

- (1) テンプレート Jumpの①の箇所に Jump010View 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。

プロジェクト名: JumpPro????(年組席) アプリケーション名: Jump010View

```
年
          組
                席 名前
* Jump010View
     画面の背景に猫画像を表示
     ボールと壁との跳ね返り(衝突処理)
package jp.edu.mie;
import android.content.*;
import android.graphics.*;
import android.view.SurfaceHolder;
import android.view.SurfaceView;
import java.util.concurrent.Executors;
import java.util.concurrent.ScheduledExecutorService;
import java.util.concurrent.TimeUnit;
public class Jump010View extends SurfaceView
                implements SurfaceHolder. Callback
   private SurfaceHolder holder;
   private Canvas canvas;
   private Paint paint;
   private Bitmap[] bmp=new Bitmap[4];//画像保存用配列
   private int ballSPEED=10;//ボールスピード
   private int ballX=0;//ボール X 座標
   private int ballY=0;//ボール Y 座標
   private int ballVX=ballSPEED;//ボール X 速度
   private int ballVY=-ballSPEED;//ボールY速度
   private int ballWIDTH;//ボールの幅
   private int ballHEIGHT;//ボールの高さ
   private ScheduledExecutorService executor;
   public Jump010View(Context context)
       super(context);
       //ビットマップの読み込み
       // r0···背景 1280×800 r1···ブロック 96×32
          r2・・・ボール 32×32
                                 r3・・・バー 120×30
       for (int i=0; i < 4; i++) {bmp[i]=readBitmap(context, "r"+i);}
       ballWIDTH=bmp[2].getWidth();
       ballHEIGHT=bmp[2].getHeight();
       holder=getHolder();//サーフェイスフォルダの取得
       holder. addCallback(this);//サーフェイスフォルダの通知先の指定
       holder.setFixedSize(getWidth(),getHeight());//サイズの指定
```

```
//サーフェイス生成時に呼ばれる
public void surfaceCreated(SurfaceHolder holder)
   //描画の設定
   paint=new Paint();
   paint. setAntiAlias(true);//文字やラインを滑らかに見せる。
   ballX=getWidth()/2-ballWIDTH/2;//ボールの X 座標
   ballY=getHeight()/2-ballHEIGHT/2;//ボールのY座標
   executor=Executors.newSingleThreadScheduledExecutor();
   executor.scheduleAtFixedRate(
       new Runnable()
       {
           public void run()
              moveBall();//ボールの移動
               draw(canvas);
       }, 0, 1, TimeUnit.MILLISECONDS);
public void draw(Canvas canvas)
   canvas=holder. lockCanvas();//Canvas オブジェクトをロックして取得
   canvas. drawBitmap(bmp[0], 0, 0, null);//背景の描画
   canvas. drawBitmap(bmp[2], ballX, ballY, null);//ボールの描画
   holder.unlockCanvasAndPost(canvas);//実画面に反映
//ボールの移動
private void moveBall()
   ballX += ballVX;
   ballY += ballVY;
   //周囲との衝突判定
                      || 論理和
   if(ballX<0){ballX=0;ballVX=-ballVX;}//単項-演算子、符号の反転
   if(getWidth()-ballWIDTH<ballX)
       ballX=getWidth()-ballWIDTH;
       ballVX=-ballVX;
   if (ballY<0) {ballY=0;ballVY=-ballVY;}</pre>
   if(ballY>getHeight()-ballHEIGHT)
       ballY=getHeight()-ballHEIGHT;
       ballVY=-ballVY;
//ビットマップの読み込み
private static Bitmap readBitmap(Context context, String name)
   int resID=context.getResources().getIdentifier
                             (name, "drawable", context.getPackageName());
   return BitmapFactory.decodeResource(context.getResources(), resID);
//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){executor.shutdown();}
}
```

# 2 実行方法と実行結果

#### 2.1 準備

「res」フォルダに画像ファイルとサウンドファイルを置きます。

※画像ファイルは、「drawble\_nodpi」「drawble\_mdpi」「drawble\_ldpi」のいずれかに保存してください。

#### 2.2 実行方法

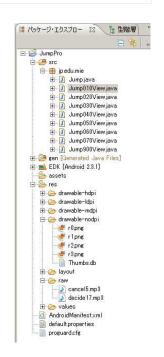
Jump. java をクリックして、メニューから「実行」又は実行アイコンをクリックしてください。

### 2.3 実行結果

ボールの画像が斜め方向に動きます。

## 3 画像ファイルの読み込み

この画像サイズは、実サイズ 1280×703 です。



3.1 画像保存用配列に次のように読み込んでいます。

```
Bitmap[0]=r0. jpg・・・ 背景 1280×800 ドット 色深度 24bpp
Bitmap[1]=r1. png・・・ブロック 96×32 ドット 色深度 32bpp
Bitmap[2]=r2. png・・・ボール 32×32 ドット 色深度 8bpp
Bitmap[3]=r3. png・・・反射板 120×30 ドット 色深度 32bpp
```

- 3.2 リソースオブジェクトは、Context クラスの getResources()メソッドで取得します。
- 3.3 リソースの画像ファイルを読み込むには、BitmapFactory クラスの decodeResource()メソッドを使います。このメソッドの引数は、リソースオブ ジェクトとリソース ID を指定します。

しかし、データを読み込んだだけでは表示はされません。表示させるには、Canvas クラスの drawBitmap()メソッドを使います。

canvas.drawBitmap(image, int 表示位置の x 座標, int 表示位置の y 座標, Paint インスタンス);

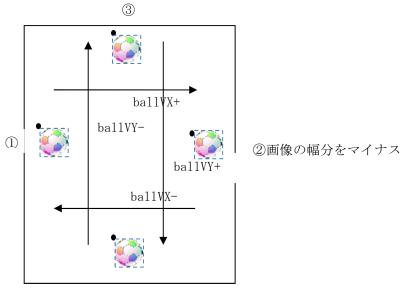
X 座標、Y 座標の基準点は画像の左上になります。画面のサイズは幅が 1280、高さが 703 です。画面の真ん中に画像を表示させる座標は (640, 351) になります。

canvas.drawBitmap(image, getWidth()/2, getHeight()/2, paint);

# 4 ボールの移動と衝突判定

```
canvas. drawBitmap(bmp[2], ballX, ballY, null);//ボールの描画
・・・
//ボールの移動
private void moveBall()
{
  ballX+=ballVX;
  ballY+=ballVY;
  //周囲との衝突判定 ||論理和
  ①if(ballX<0) {ballX=0; ballVX=-ballVX;}//単項一演算子、符号の反転
  ②if(getWidth()-BMP_2_WIDTH</br>
  ballX=getWidth()-BMP_2_WIDTH;
  ballVX=-ballVX;
  }
  ③if(ballY<0) {ballY=0; ballVY=-ballVY;}
  ④if(ballY>getHeight()-BMP_2_HEIGHT)
  {
  ballY=getHeight()-BMP_2_HEIGHT;
  ballVY=-ballVY;
  }
}
```

#### ■座標軸



④画像の高さ分をマイナス

ボールが斜め方向へ移動します。