Android ゲームプログラミング

○　第５回　画像の表示　○

画像の表示

・画像はBitmapFactoryクラスで読み込む
・画像の描画はCanvasクラスのdrawBitmapメソッド
・読み込める形式はPNG/BMP/GIF/JPGで、PNGが推奨されている
・透過色はアルファ付きPNGもしくは透過GIF
・大きな画像がぼやける場合は、"AndroidManifest.xml"に以下の項目を追加する
　<supports-screens android:anyDensity="true" />

画像の読み込み

画像は、BitmapFactoryクラスのdecodeから始まる名前のメソッドで読み込みます。対応している形式はPNG/BMP/GIF/JPGで、PNG形式が推奨されています。ファイル、リソース、配列、ネットワークなどから読み込むことができます。ただし、Androidはアプリケーションに割り当てられるメモリが少ないため、大きな画像を読み込もうとするとメモリ不足で強制終了する場合があります。

試BitmapFactory.decodeFile(String *pathName*)
ファイル名pathNameから画像を読み込みます

試BitmapFactory.decodeResource(Resources *res*, int *id*)
resで指定されるリソース内にある識別子idの画像を読み込みます。resはgetResources()で取得、idは"R.drawable.リソース名"で指定します

試BitmapFactory.decodeByteArray(byte[] data, int offset, int length)
バイト配列dateのoffsetバイト目からlengthバイトを画像として読み込みます

読み込みに成功すると、画像を管理するBitmapクラスの変数が返されます。Bitmapクラスには、画像のサイズを取得、画像のコピー、圧縮などのメソッドが備わっています。
ゲームで多く使用される透明色(キャラクターの周辺など、画像には存在するものの画面には描画しない色)は、アルファ情報付きのPNG形式もしくは透過GIFで行います。

Bitmapクラスが保持する画像データが不用になった場合は､Bitmap.recycleメソッドで解放できます。Android端末はメモリが少ない機種も多いので、不用な画像データは、このメソッドで明示的に解放した方がアプリケーションの動作が安定します。また、解放されているかどうかは、Bitmap.isRecycleメソッドで調べることができます(なので、解放後に変数にnullを代入する必要はありません)。

```
// Bitmapイメージの解放
bgBmp.recycle();          // bgBmpは生成済みのBitmapクラス型の変数
      ⋮
// イメージが解放されているかどうか調べる
if(bgBmp.isRecycle()){
    // recycleされている
}
```

画像はほとんどの場合、リソースへ登録します。リソースへ登録すると、実行ファイルとまとめて配布することができます。
リソースへの登録は、resフォルダのdrawableフォルダにファイルをドラッグ＆ドロップで追加し、プロジェクトをリフレッシュすれば、自動的にコンパイルされて認識されます。
drawableフォルダは、密度にあわせて３つのフォルダがあります。XPERIAやGALAXYなどの高密度端末は、"drawable-hdpi"フォルダに登録します。"drawable-mdpi"は中密度、"drawable-ldpi"は低密度用です。Android 1.6以上は複数の密度対応となっています。密度に合わせた画像を用意したり、プログラムを工夫することにより、あらゆる密度で同じように動作するアプリケーションを作成することができます。

※リソースフォルダに"Thumbs.db"があるとコンパイルエラーになります。削除しましょう。

画像の描画

Bitmapクラスの画像は、CanvasクラスのdrawBitmapメソッドで描画することができます。このメソッドは、オーバーロードされており、拡大縮小や変形ができるものもあります。

書Canvas.drawBitmap(Bitmap *bitmap*, float *left*, float *top*, Paint *paint*)

bitmap	描画するビットマップ
left	描画先のｘ座標
top	描画先のｙ座標
paint	描画時に使用するPaintクラス。使用しない場合はnull

－使用例－
(80, 124)にビットマップを描画(charaBmpとcanvasは生成済みとします)

```
canvas.drawBitmap(charaBmp, 80.0f, 124.0f, null);
```

書Canvas.drawBitmap(Bitmap *bitmap*, Rect *src*, Rect *dst*, Paint *paint*)

bitmap	描画するビットマップ
src	描画元の領域。nullにするとビットマップ全体が描画されます
dst	描画先の領域(nullは不可)
paint	描画時に使用するPaintクラス。使用しない場合はnull

－使用例－
キャンバスの(120, 240)から(360, 480)の領域にビットマップ全体を伸縮して描画

```
Rect  dst = new Rect(120, 240, 360, 480);
canvas.drawBitmap(charaBmp, null, dst, null);
```

書Canvas.drawBitmap(Bitmap *bitmap*, Rect *src*, RectF *dst*, Paint *paint*)

bitmap	描画するビットマップ
src	描画元の領域。nullにするとビットマップ全体が描画されます
dst	描画先の領域(nullは不可)。float型で指定できます
paint	描画時に使用するPaintクラス。使用しない場合はnull

書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

－使用例－
Matrixを用い、ビットマップ全体を変形して描画

```
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);
```

コーディング例

```
// フィールド変数
private Bitmap   mBgBmp, mCharaBmp;
                    :

// 初期化
private void Initialize() {
    // 画像の読み込み
    mBgBmp    = BitmapFactory.decodeResource(getResources(), R.drawable.bg);
    mCharaBmp = BitmapFactory.decodeResource(getResources(), R.drawable.chara);
}

// 描画
private void doDraw() {
    Canvas   canvas = mSurfaceHolder.lockCanvas();

    canvas.drawBitmap(mBgBmp,    0, 0, null);
    canvas.drawBitmap(mCharaBmp, 0, 0, null);

    mSurfaceHolder.unlockCanvasAndPost(canvas);
}
```

<supports-screens>

ソースファイルではなく、"androidmanifest.xml"というアプリケーションの情報を定義するファイルに記述されています。

<supports-screens>は、アプリケーションが対応する画面の解像度を指定します。XPERIAやGALAXYなど高密度の機種でも、デフォルトはノーマル画面・中密度という設定になっており、画像によっては0.75倍もしくは1.5倍に伸縮して描画されることがあります。そうなると、画像がくずれてしまいます。

そのような場合、"androidmanifest.xml"に以下のように、<supports-screens>という項目を追加すると、正しく描画できるようになります。

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
                    :
    <supports-screens android:anyDensity="true" />

</manifest>
```

anyDensityは、アプリケーションが中密度以外のスクリーン密度に対応できるかの指定です。trueは対応可能、falseは対応不可を示します。XPERIAやGALAXYなどは高密度対応なので、明示的にtrueにする必要があります(もしくは、プロジェクト作成時や上記ファイルで指定できる"minSdkVersion"を４以上にします。例：<uses-sdk android:minSdkVersion="4"></uses-sdk>)。

課　題

(１)画像を読み込み、画面に表示しましょう。

(２)背景とキャラクターを読み込み、正しく描画されるプログラムを作成しましょう。