第8章　CG 入門

【学習内容とねらい】

本章では、Java 言語を用いた CG（コンピュータグラフィックス）の描き方を学習します。Java 言語では、Graphics クラスに CG 作成に必要なメソッドが用意されており、それらを利用するだけで簡単に CG を作成することができます。その、”簡単に CG を描ける”という体験を（課題プログラムの作成を通じて）してもらう、ということが本章のねらいです。ですから、少し突っ込んだ CG の理論に関する内容には触れていません。それらについては、３年次に開講される、皆川先生の基礎コンピュータグラフィックス論に譲ります。

以上のような理由で、本章で扱う題材は基本的なものに限定しています。少し物足りないと思った人は、ぜひ、市販のテキスト等でより本格的な知識・技術を身につけてください。Java 言語を用いてかなり本格的な CG あるいは CG アニメーションを作成できる事に気づくはずです。少し努力すれば、皆をあっと驚かせる CG を作成することができますよ。

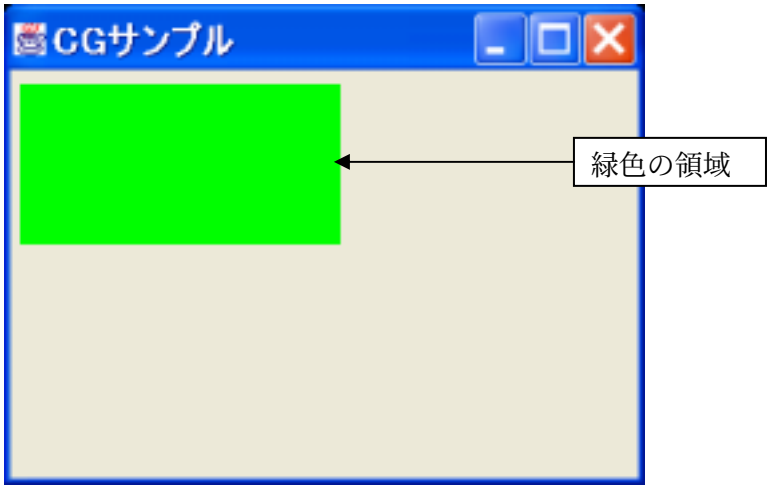
それでは、前置きはこの位にしてさっそく学習に取りかかりましょう。ともかく本章では、自分のプログラムで CG を描く楽しさを味わって下さい。

＜本章の構成＞

8-1 CG の描き方①ー標準的な方法 (paint メソッドを使用)

【応用課題 8-1-A】

まず手始めに、次のような CG を描いてみましょう。それは、実行すると、次のようにフレームの左上の矩形領域が緑色に塗られるというプログラムです。



市販のテキスト等で説明されている最も標準的な CG 作成方法は、コンポーネントに備わっている **paint** メソッド（内）に、CG 作成処理を記述することです。paint メソッドは主立ったコンポーネントに用意されており、そのコンポーネントが生成された瞬間に実行されます。また、別の Windows が重なり、一部分（あるいは全部）が隠れてしまった後に、再描画する際に自動的に呼び出されます。ですから、paint メソッドを用いると、プログラム実行と同時に（そこに記述された処理が）実行されるようになっています。

では、アプリケーションを新規作成し、ソースビューを開いて下さい。そして、次ページのように、プログラム上方の「フレームのビルド」とコメントがある部分（Frame1 クラスのコンストラクタ）を見つけ、その下に、同じく次ページのように paint メソッドを書き加えて下さい。枠内が新たに記述した部分です。

さて、プログラムの解説に進む前に、少し paint メソッドについて補足しておきましょう。今定義している Frame1 クラスは JFrame クラスを継承して作られたものです。そして JFrame クラスには paint メソッドが定義されており、その処理内容は、フレーム全体を background カラーで塗りつぶすというものです。そこで、それ以外の描画処理が必要になったときには、次ページのように、paint メソッドを書き換えるのです。すると、今度は書き直された paint メソッドが有効になります。このように、継承したサブクラスで（スーパクラスの）メソッドの内容を書き換えることをメソッドの**オーバーライド**（書き換え）と言います。以上を念頭に置いて、プログラムの作成に進んで下さい。

```
  //フレームのビルド
  public Frame1() {
    enableEvents(AWTEvent.WINDOW_EVENT_MASK);
    . . .
}
```

新たに挿入した部分

```
  public void paint(Graphics g) {
    g.setColor(Color.green);
    g.fillRect(10,50,200,100);
  }
```

```
  //コンポーネントの初期化
  private void jbInit() throws Exception  {
    . . .
```

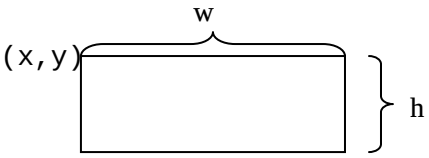
＜解説＞

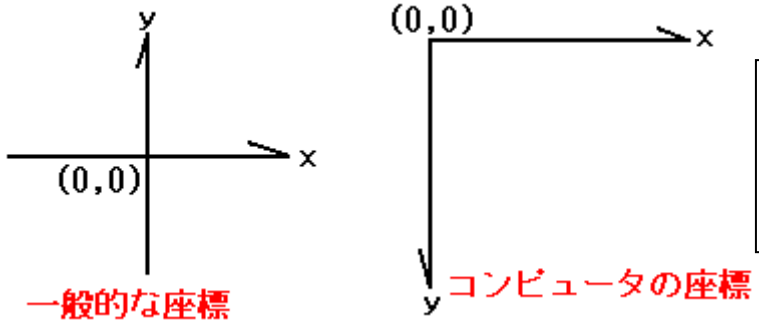
① paint メソッドの引数は Graphics クラスの変数（オブジェクト）です。上の例では g と言う名前にしています。これで、対象とするコンポーネント（今の場合フレーム）の Graphics オブジェクトを取得できます。

② `setColor()`メソッドは、Graphics クラスに用意されているメソッドで、CG を作成する際の色を指定します。今の場合、緑色です。Color オブジェクトについては、第３章（p.60）でも説明しました。

③ `fillRect(x,y,w,h)`メソッドは、点（x,y）を左上頂点とし、横幅 w、縦の高さ h の四角形を描き、その内部を指定色で塗りつぶす処理を行います。

座標の取り方については、下を参照して下さい。





CG の世界では、左上を原点(0,0,)とし、y 座標が下向きに伸びていることに注意して下さい。

作成したら、プログラムを実行し動作を確認して下さい。

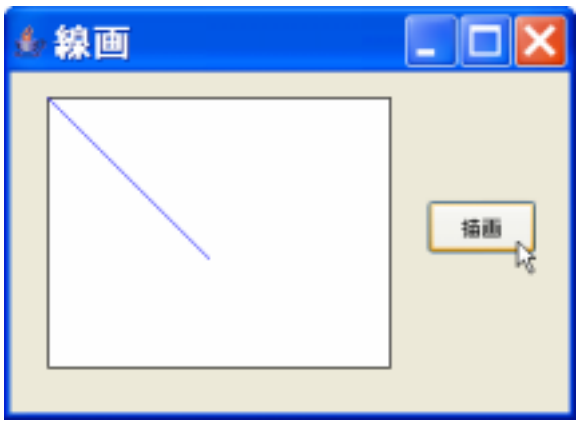
8-2 CG の描き方②－getGraphics()メソッドを用いる方法

前節の paint メソッドを用いる方法は簡単なのですが、プログラムの実行と同時に描画されてしまいます。そこで、例えばボタンを押したときに CG 描画を開始させたい場合など、CG 作成の動作制御を行う際には少し不便です。そこで、以下では、対象とするコンポーネントの Graphics オブジェクトを直接取得する **`getGraphics()`**メソッドを用いる方法を用いることにします。

【応用課題 8-2-A】

次のようなプログラムを作りましょう。

プログラムを起動すると、次の画面が表れる。



［描画］ボタンをクリックすると、左のように直線を青色で描画する。

ここに、パネルコンポーネントの background プロパティを白色にして下さい。

この［描画］ボタンのイベントハンドラは次のようになります。

```
void jButtonDraw_actionPerformed(ActionEvent e) {
  Graphics g=jPanel1.getGraphics(); //Graphics オブジェクトの取得
  g.setColor(Color.blue);
  g.drawLine(0,0,100,100);
}
```

＜解説＞

① コンポーネントの Graphics オブジェクトを取得するには、getGraphics()メソッドを用います。１行目では、パネルコンポーネントの Graphics オブジェクトを、Graphics 型変数（オブジェクト）g に初期値として与えています。

② ２行目で使用する色を青色にしています。

③ ３行目の drawLine(x1,x2,y1,y2)は、２点(x1,y1)－(x2,y2)を結ぶ直線を描くメソッドです。

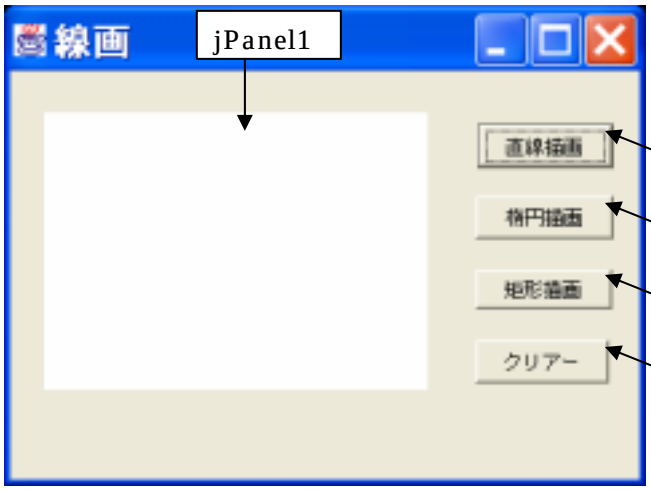
④ 今の場合、Graphics オブジェクト g が（フレームではなく）パネルコンポーネントのオブジェクトなので、CG 作成はパネル上で行うことに注意して下さい。このように getGraphics()メソッドを用いれば、ボタンやラベルの上など、様々なコンポーネントの Graphics オブジェクトを取得でき、したがって当該コンポーネント上での CG 作成を容易に行えます。

8-3 定型図形の描画

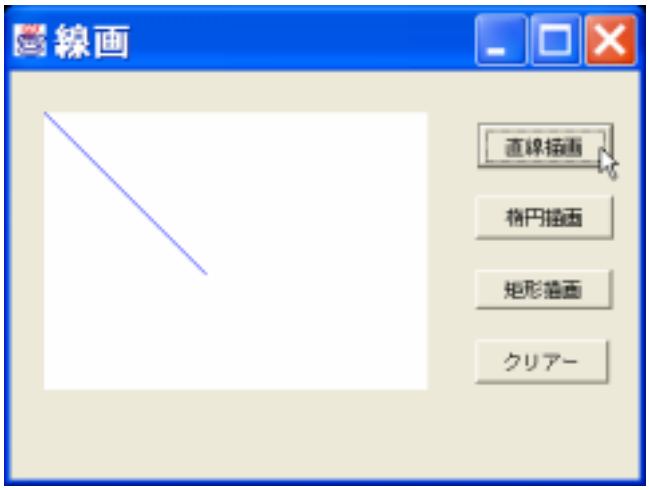
矩形（長方形や正方形）や円（楕円）については、それらを描画するメソッドが Java 言語には備わっています。その使い方を本節で練習しましょう。

【練習課題】

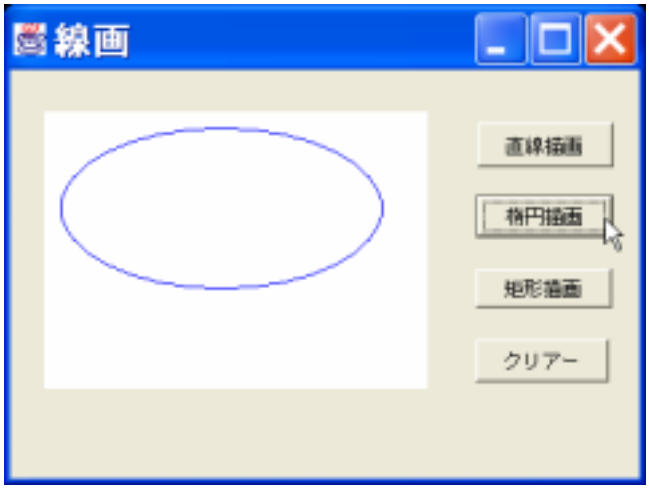
次のようなプログラムを作成しましょう。



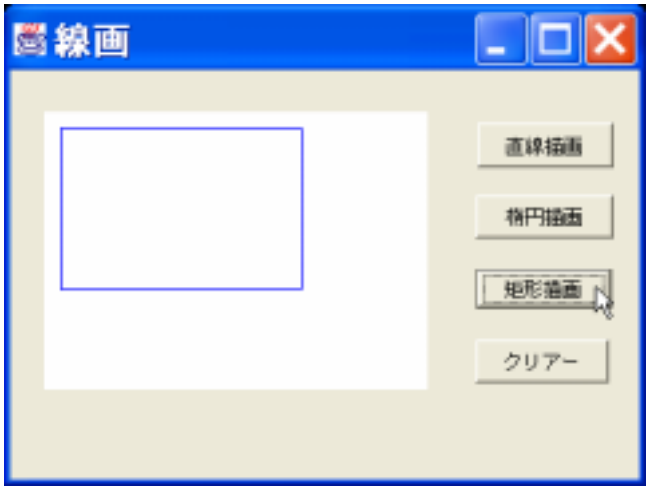
起動すると次のような画面が現れる。



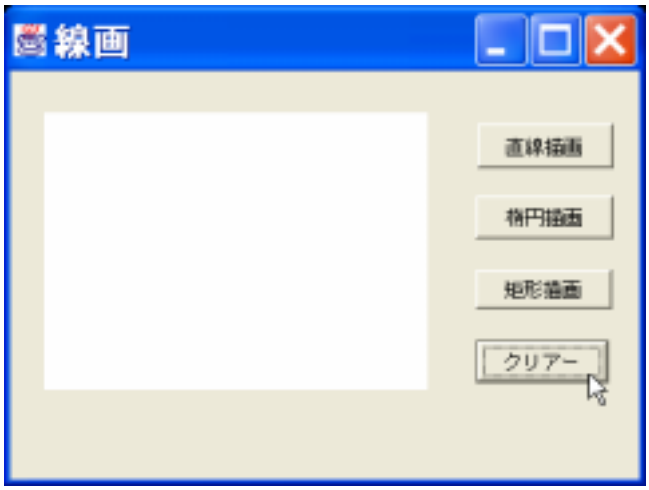
［直線描画］ボタンをクリックすると直線を描く。



［楕円描画］ボタンをクリックすると楕円を描く。



［矩形描画］ボタンをクリックすると、四角形を描く。



［クリアー］ボタンをクリックすると、全ての画像を消去する。

各ボタンのイベントハンドラは次の通りです。意味は大体分かると思いますが、正確な意味は＜解説＞で確認して下さい。

＜［直線描画］ボタン＞

```
void jButtonLine_actionPerformed(ActionEvent e) {
  Graphics g=jPanel1.getGraphics();
  g.setColor(Color.blue);
  g.drawLine(0,0,100,100);
}
```

＜［楕円描画］ボタン＞

```
void jButtonOval_actionPerformed(ActionEvent e) {
  Graphics g=jPanel1.getGraphics();
  g.setColor(Color.blue);
  g.drawOval(10,10,200,100);
}
```

＜［矩形描画］ボタン＞

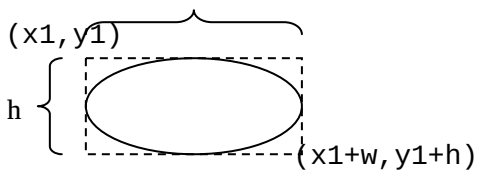
```
void jButtonRect_actionPerformed(ActionEvent e) {
  Graphics g=jPanel1.getGraphics();
  g.setColor(Color.blue);
  g.drawRect(10,10,150,100);
}
```

＜［クリアー］ボタン＞

```
void jButtonClear_actionPerformed(ActionEvent e) {
  int XMax=jPanel1.getWidth(); //パネルの幅の取得
  int YMax=jPanel1.getHeight(); //パネルの高さの取得
  Graphics g=jPanel1.getGraphics();
  g.setColor(Color.white);
  g.fillRect(0,0,XMax,YMax); //領域内を指定色で塗りつぶす
}
```

＜解説＞

① drawOval(x1,y1,w,h)は下のように、(x1,y1)を左上隅として、幅 w、高さ h の四角形に内接する楕円を描きます。



② `drawrect(x1,y1,w,h)`メソッドは、左上の頂点を`(x1,y1)`とし、幅w、高さｈの四角形を描きます。

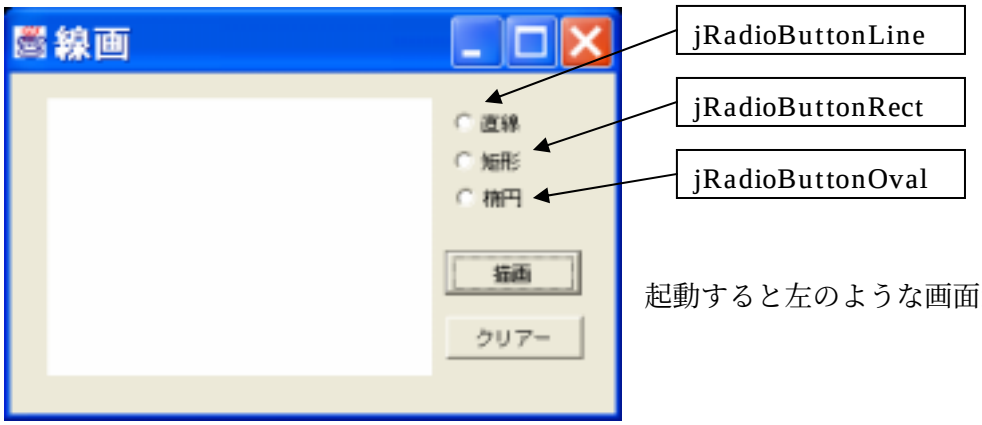
③ `getWidth()`メソッドは、対象とするオブジェクト（今の場合パネルコンポーネント）の横幅の値を与えます。

④ `getHeight()`メソッドは、対象とするオブジェクト（今の場合パネルコンポーネント）の（縦方向の）高さを与えます。

⑤ ［クリアー］ボタンのイベントハンドラの処理内容は、パネル領域を白色で塗りつぶすということです。上の例から分かる通り、「`draw･･･`」メソッドを「`fill･･･`」に変えると図形内の色を塗りつぶす処理を行います。

【応用課題 8-3-A】

上の【練習課題】のプログラムを改良して、次のようなプログラムを作成しましょう。作成に当たっては次ページの指示に従って下さい。



起動すると左のような画面が現れます。

「直線」欄をクリックして［描画］ボタンをクリックすると、直線を描きます。

「楕円」欄をチェックして［描画］ボタンをクリックすると、楕円を描きます。矩形についても同様です。
また、［クリアー］ボタンをクリックすると図形を消去します。

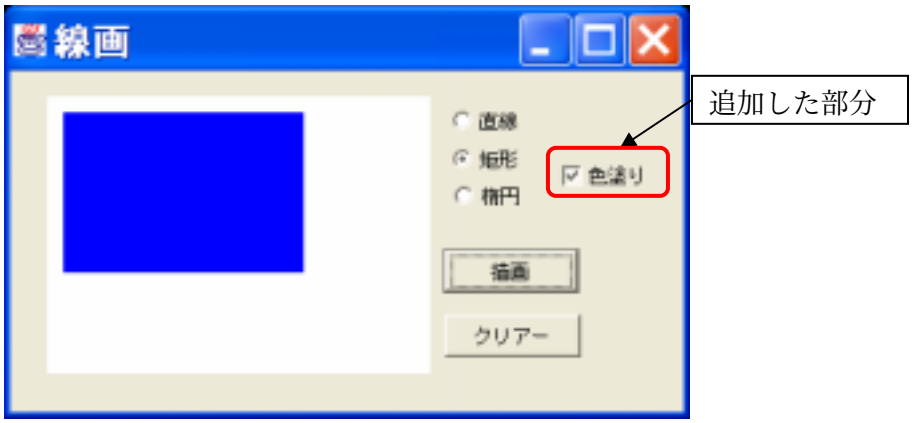
［描画］ボタンのイベントハンドラを次のように記述しました。

```
void jButtonDraw_actionPerformed(ActionEvent e) {
  Graphics g=jPanel1.getGraphics();
  if(jRadioButtonLine.isSelected()) {
    LineDraw(g);  //線を描くメソッド
  }
  else if(jRadioButtonRect.isSelected()) {
    RectDraw(g);  //四角形を描くメソッド
  }
  else {
    OvalDraw(g);  //楕円を描くメソッド
  }
}
```

上で示した処理を実現するよう、メソッド LineDraw(g)、RectDraw(g)、OvalDraw(g) を定義して下さい。矩形や楕円の大きさは、【練習課題】の通りで結構です。

【応用課題 8-3-B】

【応用課題 8-3-A】のプログラムに、次のように（図形内の）色を塗りつぶす選択欄を付け加えて下さい。例えば、「矩形」および「色塗り」欄をチェックして［描画］ボタンをクリックすると、色が塗りつぶされた四角形が描画されます。



※　直線に関しては、色塗り欄は無効（チェックの有無は関係ない）です。

【応用課題 8-3-C】

次のような、市松模様（隣り合う四角形領域の色が異なる模様）を描くプログラムを作成して下さい。下の例では、白色と赤色で塗り分けています。

8-4 任意の多角形の描画

本節では、任意の多角形を描く drawPolygon メソッドの用い方を学習しましょう。

【練習課題】

次のように、三角形を描くプログラムを考えましょう。

プログラムを起動し、［描画］ボタンをクリックすると、二つの三角形（下方は内部の色を塗りつぶしている）を描く。

このときの［描画］ボタンクリック時のイベントハンドラは、次のようになります。

```
void jButtonDraw_actionPerformed(ActionEvent e) {
  Graphics g=jPanel1.getGraphics();
  int xc=jPanel1.getWidth()/2;  //パネルのｘ座標の中点を求める
  int yc=jPanel1.getHeight()/2; //パネルのｙ座標の中点を求める
  int x[]=new int[3];  //大きさ３の配列ｘの宣言
  int y[]=new int[3];  //大きさ３の配列ｙの宣言
  x[0]=xc-50; x[1]=xc; x[2]=xc+50; //三角形の３点のｘ座標を配列に入れる
  y[0]=yc-10; y[1]=yc-40; y[2]=yc-10; //同じく３点のｙ座標を配列に入れる
  g.setColor(Color.blue); //描画色を青色に指定
  g.drawPolygon(x,y,3);  //多角形（今の場合三角形）の描画
  y[0]=yc+50; y[1]=yc+20; y[2]=yc+50;  //y 座標の更新（下に 60 だけずらす）
  g.fillPolygon(x,y,3);  //多角形（今の場合三角形）の描画および塗りつぶし
}
```

＜解説＞

- プログラムの大まかな意味は、コメントから分かると思います。
- drawPolygon(x,y,n)メソッドは、ｘ座標配列、ｙ座標配列、そして点の個数（配列の大きさ）を引数として指定します。配列については、4-12 節を参照して下さい。
- そして、このメソッドが呼び出されると、点（x[0],y[0]）から点（x[1],y[1]）,・・・点(x[n-1],y[n-1])と順に直線で結び、最後に再び点（x[0],y[0]）に戻ってきます。これにより、n 点を結ぶ多角形が描画されます。
- fillPolygon(x,y,n)メソッドの場合は、線で結んだ多角形の内部を指定した色で塗りつぶします。

作成したら実行し、動作を確認して下さい。

【応用課題 8-4-A】

上の Polygon メソッドを用いて、次のように、任意の四角形を描いてその内部を赤色で塗りつぶすプログラムを作成して下さい。

４点のｘ、ｙ座標を入力して［描画］ボタンをクリックすると、４点を結び、その内部を赤色で塗りつぶした四角形を描く。

8-5 点画の描画

前節までの学習で、任意の多角形を描画することができるようになりました。しかし、多角形では表現しにくい、より一般的な図形はどのように描画すればよいのでしょうか？それは、画面の最小単位の点（**ピクセル**と言います）毎に指定色で色づけすればよいのです。と言うのは、どんなに複雑な図形でも点の集まりで表現できるからです。そこで、本節では、そのような点画を描く練習をしてみましょう。と言っても新しいメソッドを用いるわけではありません。実は、すでに学習した `drawLine(x1,y1,x2,y2)`メソッドを用いて、２点の座標を同じにすると、（当然ですが）その点に指定色で点を打つ命令になるのです。次の課題で、`drawLine` メソッドを用いて、少し面白い点画を描いてみましょう。

【応用課題 8-5-A】 (chaos)

今、ある点列$(X_1,Y_1),(X_2,Y_2),(X_3,Y_3),\cdots,(X_i,Y_i),\cdots,(X_n,Y_n)$を考えます。ここに、$(X_i,Y_i)$が決まったときにその次の（$X_{i+1},Y_{i+1}$）は次の式で決まるものとします。

$$X_{i+1}=Y_i-0.97\times X_i+5/(1+X_i^2)-5, \qquad Y_{i+1}=-0.995\times X_i$$

例えば、（X_1,Y_1）=(1,0)とすると、

$$X_2=Y_1-0.97\times X_1+5/(1+X_1^2)-5=-0.97+5/2-5=-3.47$$
$$Y_2=-0.995\times X_1=-0.995$$

と(X_2,Y_2)が求まり、同様に、$(X_3,Y_3),\cdots,(X_n,Y_n)$が次々と求まって行きます。

さて、最初の点を（X_1,Y_1）=(1,0)として上の要領で点列を求め、順に表示させて行くと次のように鳥が羽を広げたような不思議な図形が現れます。このプログラムを作成してください。

点の数を 1000 にして描画したところ
（点の色は青にしています。）

点の数を 30000 にして描画したところ

ヒント

① 点の数を N とすると、上の処理は、点(X_i,Y_i)を表示させるという処理を、i=1～N について N 回繰り返すことで実現できます。

② 上の（X_i，Y_i）の値は小さな数になるので見えにくくなります。そこで実際には何倍かに拡大する必要があります。上の例では（6*X_i，6*Y_i）と 6 倍に拡大して表示しています。

③ 座標の原点はパネルコンポーネントの中心となるようにしています。つまり、原点の位置をずらしています（座標の原点をパネルの左上隅のままにしておくと、マイナスの値を表示することができないので注意して下さい）。

④ パネルコンポーネントの幅と高さは 200 程度にしています。

⑤ （X_i，Y_i）が求まるたびにその点（実際には 6 倍に拡大した座標）に点を打てば良いのです。その際、（X_i，Y_i）は実数なので、`drawLine` メソッドの引数にするためには、整数型に型キャストする必要があります。

上の課題プログラムについては、プログラミングの HP

http://ext-web.edu.sgu.ac.jp/HIKO/Prog

の該当部分に、クラスファイルが入ったフォルダを chaos.exe という名前の自己解凍形式フォルダとして掲載しています。これをダウンロードしてダブルクリックして下さい。すると、自動的に解凍されフォルダ chaos が現れます。このフォルダ内のクラスファイルを用いれば、プログラムを実行し動作結果を確認する事ができます。各自、学習の際の参考にして下さい。クラスファイルからの実行の仕方は、巻末の「付録A　クラスファイルからの実行の仕方」を参照して下さい。

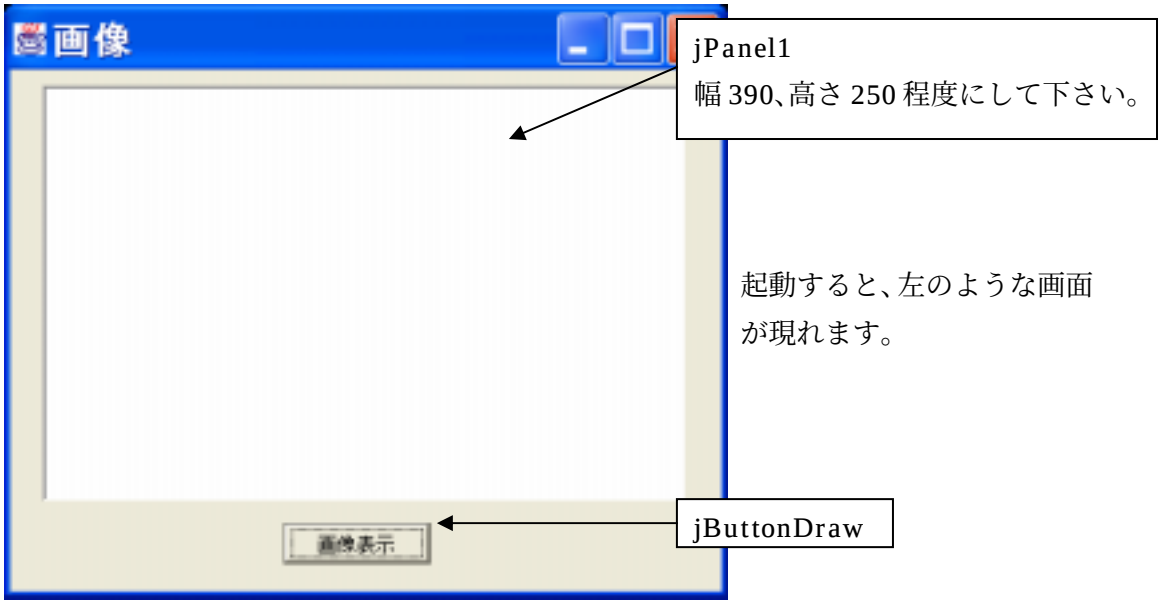
8-6 画像ファイルの読み込みと表示

前節までは、プログラムで描画を行う方法を学習しましたが、デジタルカメラの写真などファイルに保管されているデジタル画像については、画像ファイルから読み込んでそれを適当なコンポーネント（例えばパネルコンポーネント等）に表示させることができます。本章の最後に、その方法を学習しましょう。

Java 言語では、{gif,jpg,png} 形式の画像を読み込む機能があります。「プログラミング」の HP の該当部分に掲載している自己解凍形式のフォルダ Fig.exe をダウンロードし、それを解凍して下さい。フォルダ Fig の中には {Falcon.jpg, Grappa.jpg, Desk.jpg, Concert.jpg, Milano.jpg} の５つのサンプル画像を納めています。以下では、これを入力画像ファイルとして用います。以下の課題では、作成するプロジェクトのフォルダ内にフォルダ Fig を必ずコピーしておいて下さい。

【練習課題】

次のようなプログラムを作ってみましょう。



画像表示ボタンをクリックすると、左のような画像が現れます。

この［画像表示］ボタンのイベントハンドラは次のようになります。

```
void jButtonDraw_actionPerformed(ActionEvent e) {
   Graphics g=jPanel1.getGraphics();
   Image Fig1; //Image（画像）クラスのオブジェクトの宣言        ①
   Toolkit toolkit=getToolkit(); //Toolkit クラスのオブジェクトの宣言 ②
   Fig1=toolkit.getImage("Fig¥¥Falcon.jpg"); //指定したファイル内の画像を Image オブジェクト Fig1 に読み込む          ③
   g.drawImage(Fig1,0,0,this);     ④
 }
```

＜解説＞

① Java 言語では、画像データ（{gif,jpg,png} 形式のもの）は Image オブジェクトとして扱われます。そして、画像ファイルから読み込まれた画像データは、プログラム中では Image オブジェクトに代入して用います。上の例では、Fig1 という名前で Image オブジェクトを用意しています。

② 画像ファイルから画像を読み込むためには、getImage メソッドを用います。しかし getImage メソッドは Graphics クラスには用意されておらず、**Toolkit クラス**という別のクラスに用意されています。そこで、上のプログラムでは、

```
Toolkit toolkit=getToolkit();
```

によって、toolkit オブジェクトを用意し、そこに用意されている getImage メソッドを利用するようにしています。

③ ②で説明した getImage メソッドは getImage(ファイル名)の形で用います。上の例

では、ファイル名として"Fig¥¥Falcon.jpg"と指定していますが、これは、Fig フォルダ内にある Falcon.jpg ファイルという意味です。本来は¥記号は一つでフォルダの区切りを意味するのですが、これは特別な意味を持つ制御記号でもあるので、文字列定数の中で¥を表したい場合は¥¥の様に二つ続けて記述しなければなりません。1文字では、コマンドだと解釈されてしまうので注意して下さい。なお、使用しているフォントによっては、「¥」が「\」と表示されます。ですから「¥」とキーインしたのに「\」が表示されても、気にせず作業を続けて結構です。

④ そして Image オブジェクト **A** を表示させるためには、次のように drawImage メソッドを用います。

```
drawImage(A,x,y,this)
```

ここに、(x,y)は表示させる画像オブジェクト（の左上隅）の表示位置です。第４引数については、通常は **this** と指定すると覚えておいて下さい。

※ なお、用意した画像の大きさは、幅 390、高さ 250 程度なので、パネルの大きさもその程度の大きさにして下さい。パネルのサイズをこれより大きくすると、余白部分ができてしまいます。

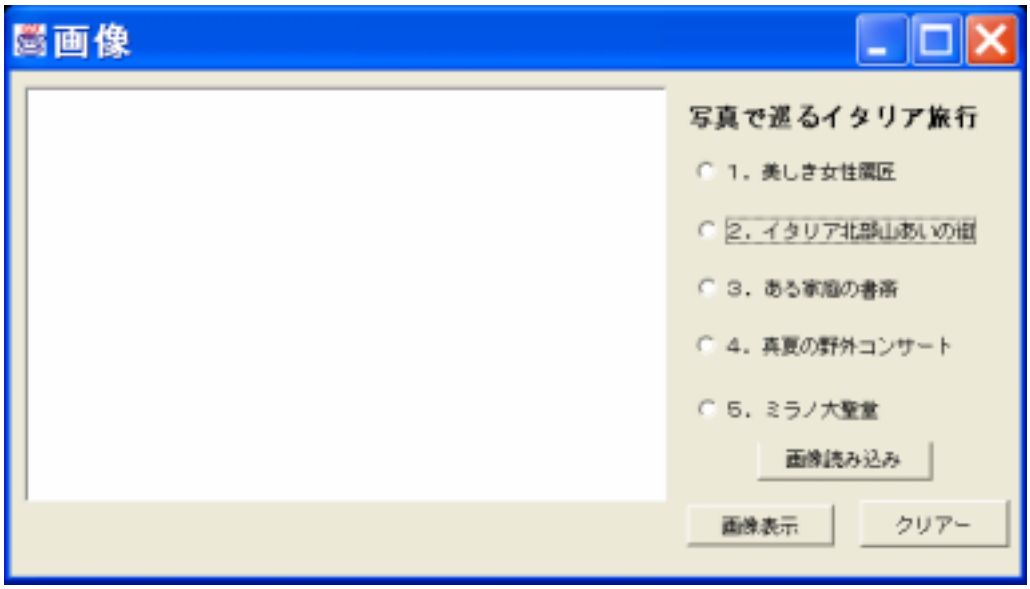
プログラムを作成したら動作を確かめて下さい。

確かめてみると分かりますが、ボタンを１回押しただけでは画像は表示されず、２回目（以降）で表示されると思います。動作を見る限り、最初の drawImage メソッドが呼び出された段階では画像をバッファ（一時的作業領域）に格納するだけでまだ（パネルに）表示せず、2回目に drawImage メソッドが呼び出されて始めて表示されるようになっているようです。この動作が Java 言語の仕様通りなのかどうか現時点では不明ですが、不都合な事に変わりありません。誰か、原因を解明してくれた人がいたら、次年度のテキストに氏名と共にその調査内容を掲載して、その功績を称えたいと思います。

【応用課題 8-6-A】 写真で巡るイタリア旅行

画像表示の技術を使って、次のように、指定した画像を表示させるプログラムを作成して下さい（作成するプロジェクトのフォルダ内に、フォルダ `Fig` をコピーすることを忘れないで下さい）。

まず、プログラムを起動すると下の画面が現れます。



ここで、［画像読み込み］ボタンをクリックし、画像ファイルの画像データをバッファへ格納します。続いて画像リストの適当な欄を選択した後、［画像表示］ボタンをクリックすると、該当する画像が表示されます。

下は、１を選択したところ。

下は、５を選択したところ。

このプログラムでは、５つの画像を５つの Image オブジェクトに格納するようにします。そこで、下のように、Image オブジェクト（の配列）Fig[]をインスタンス変数として宣言して下さい。点線枠内が入力部分です。

```
public class Frame1 extends JFrame {
  private JPanel contentPane;
  private JPanel jPanel1 = new JPanel();
  . . .
  Image Fig[]=new Image[5];  //Image オブジェクトを配列型で宣言
  //フレームのビルド
```

次に、ボタン［画像読み込み］のイベントハンドラを次のように記述して下さい。

```
void jButtonRead_actionPerformed(ActionEvent e) {
  Graphics g=jPanel1.getGraphics();
  int Xmax=jPanel1.getWidth();  //パネルの横幅を取得
  int Ymax=jPanel1.getHeight(); //パネルの高さを取得
  Toolkit toolkit=getToolkit();
  Fig[0]=toolkit.getImage("Fig¥¥Falcon.jpg");
  Fig[1]=toolkit.getImage("Fig¥¥Grappa.jpg");
  Fig[2]=toolkit.getImage("Fig¥¥Desk.jpg");
  Fig[3]=toolkit.getImage("Fig¥¥Concert.jpg");
  Fig[4]=toolkit.getImage("Fig¥¥Milano.jpg");
  for (int i=0;i<=4;i++) {
    g.drawImage(Fig[i],0,0,Xmax,Ymax,this);
  }
}
```

ここで、１回目の表示を行っておく。→バッファへの読み込みで表示はされない。

＜解説＞

① プログラムを見れば分かるように、このプログラムで行っている事は以下の２点です。

ｉ）Image オブジェクト Fig[0]～Fig[4]の定義（画像データの代入）

ii）drawImage メソッドの呼び出しによる画像オブジェクトのバッファへの保管

② その際、drawImage メソッドの引数が先程と変わっている事に気づいたでしょう。実は、drawImage メソッドは

drawImage(画像オブジェクト、x,y,w,h,this)

という形式で、指定した画像オブジェクトを、幅ｗ、高さｈの領域に収まるように（縮尺して）表示させる、という事もできます。これは大変便利なのですが、本来の画像の縦横サイズ比と指定領域のそれが大きくずれてしまうと、画像がひずんでしまうので注意が必要です。

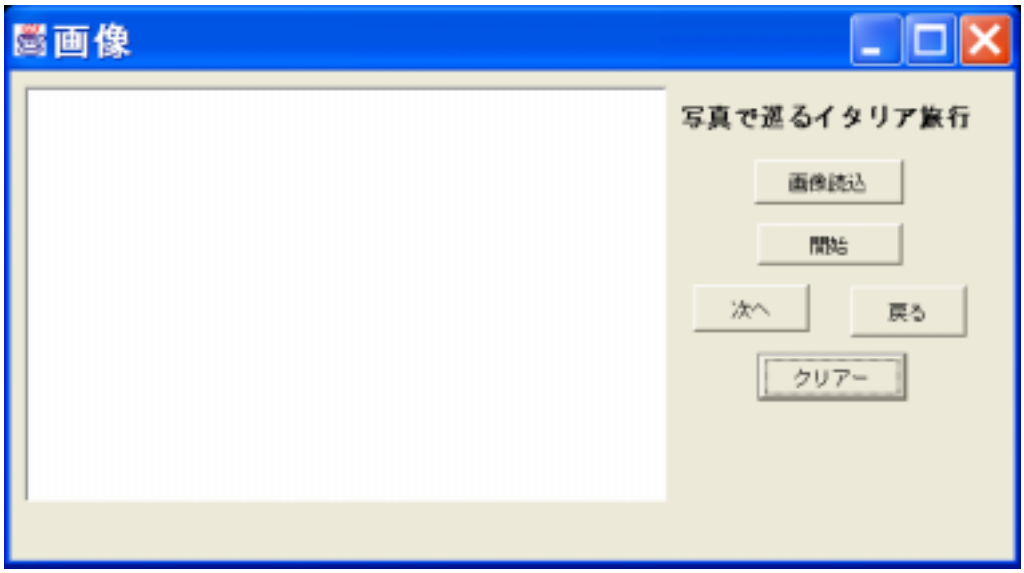
［画像表示］ボタンの処理を記述して、プログラムを完成させて下さい。

なお、このプログラムを実行する際には、プログジェクトのフォルダ内にフォルダ Fig をコピーしておくことを忘れないように注意して下さい。

【応用課題 8-6-B】 スライドショー

【応用課題 8-6-A】を改良して、次のように５枚の画像を（順番に）次々と表示するプログラムを作成して下さい。

プログラムを起動すると、下のような画面が現れます。



ここで、まず［画像読込］ボタンをクリックし、画像データをバッファに格納します。

続いて［開始］ボタンをクリックすると、最初の画像（【応用課題 8-6-A】の１の画像）が表示されます。

［次へ］ボタンをクリックすると、２番目の画像が表示されます。

以下、［次へ］ボタンをクリックすると、順番に次の画像が表示されます。

そして５番目の画像が表示された状態で［次へ］ボタンをクリックすると・・・

下のように最初の画像に戻ります。

このように、循環的に画像を表示できるようにします。[戻る]ボタンについても同様に、循環的に一つ前の画像を表示するようにして下さい。

ヒント

① 何番目の画像を表示させるのか、という添え字番号（例えば i とします）をインスタンス変数として宣言する必要があります。

② ［次へ］ボタンをクリックしたときには、添え字番号を一つ増やしてから、そして［戻る］の場合は、一つ減らしてから（当該番号の）画像を表示します。

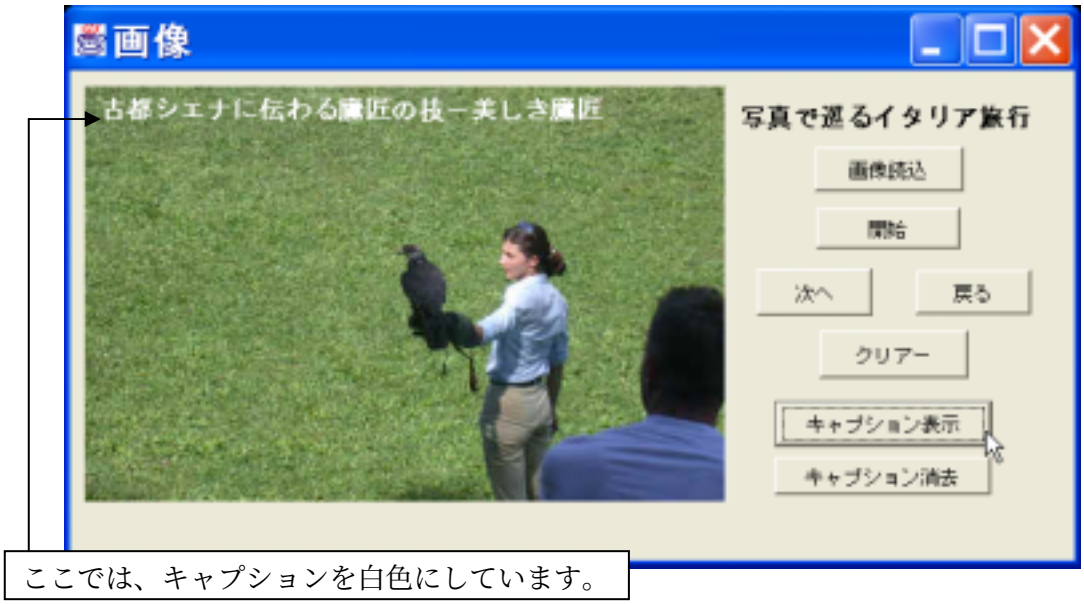
③ ②において、`i >4` になった時には、`i=1` に戻す必要があります。逆に `i<0` になった時には、`i=4` に戻す必要があります。

【応用課題 8-6-C】 キャプションの表示･消去

【応用課題 8-6-B】を改良して、次のように画像の説明（キャプション）を表示させるプログラムを作成して下さい。

前課題同様、プログラムを起動して［画像読込］→［開始ボタン］とクリックすることで最初の画像が現れます。

ここで、［キャプション表示］ボタンをクリックすると、画像の説明が表示されます。



また、［キャプション消去］ボタンをクリックすると、キャプションは消えます。

このように、キャプションの表示／消去を全ての画像について行うようにして下さい。各画像のキャプションは次の通りとします。

画像の順番	キャプション
1	古都シエナに伝わる鷹匠の技－美しき鷹匠
2	イタリア北部の街－バッサーノ・デル・グラッパ
3	書斎－イタリア人はアンティークな家具がお好き
4	真夏の夜のコンサート－古都ペルージャにて
5	ミラノ大聖堂－青空に映える大聖堂の尖塔

＜ヒント＞

① パネル上の Graphics クラスのオブジェクトを g とするとパネル上の座標(x,y)に文字列を表示させるには次のようにします（文字列"abc"を表示させる例です）。

```
String Moji="abc";
g.drawString(Moji,x,y);
```

ただし、この(x,y)は表示させる文字列の左下隅の座標です。

② 表示させる文字列のフォントを指定するには次のようにします。

```
int Fsize=14; //フォントサイズの指定
g.setFont(new Font("Dialog",Font.PLAIN,Fsize));
```

これは、フォントとして Dialog フォントを選び、通常の文字スタイルを選んだ場合です。フォントの種類と文字スタイルは次の通りです。

フォント名	Dialog、DialogInput、Monospaced、Serif、SansSerif
フォントスタイル	Font.PLAIN、Font.BOLD、Font.ITALIC

※　スタイルを組み合わせる場合は「Font.BOLD|Font.ITALIC」などとします。