今回は「黒板を作る」に続いて,「数学教材を作る」というテーマの演習とします.

メソッドを作ってみよう

今後，実用的なスケッチを作るとなると，コードがどんどん長くなってきます．その際，自分でメソッドを作るということをしないと，コードが非常に複雑になってしまいます．これまで使ってきたメソッドはあらかじめ用意されていたものだけでしたが，自分で作ることも可能です．ここで，その方法を紹介します.

メソッドの定義は

```
戻り値の型　メソッド名 ( パラメータ１の型　パラメータ１を受け取る変数名 , ・・・ )
{
    処理
}
```

とします．たとえば,

```
int add( int a, int b )
{
  int c;
  c = a + b;
  return c;
}
```

は, int 型のパラメータを二つ受け取り, int 型の値を生成するメソッド add を定義しています．このメソッドが呼ばれると，二つのパラメータがそれぞれ a, b に格納された後, 処理を実行します. 処理は a と b を加え，それを return 文で戻り値とします．このメソッドが生成する値はこの戻り値となります.

メソッドを呼び出すときには今までと同じように，たとえば,

```
int sum;
sum = add( 2, 4 );
```

とすると，sum には add(2, 4) が生成する値 6 が格納されます.

三角形の面積を表示するスケッチを作ろう

では，三角形の面積を表示するスケッチを作ってみましょう．これは，面積が変わらないように頂点を移動させると，その頂点が描く軌跡は底辺と並行であることを用いて，並行を教えるときに用いることができる教材です.

プログラム

座標系の変換を行う

三角形を描き, 面積を表示するとなると, スケッチの座標計そのままでは, 1 単位が小さすぎる

と思います．そこで，40pixel を 1 となるようにしてみます．さらに，原点を描画ウィンドウ内の任意の位置に指定でき，上方向を Y 軸正方向となる座標で，描画を行えるようにしてみます．この座標系をグラフ座標系と呼ぶことにします．なお，グラフ座標系では 3.5 のような座標も扱いたくなる可能性もありますので，float 型であらわすことにします．

描画ウィンドウの座標系とグラフ座標系の座標を相互に変換するメソッドを定義していきます．まず，座標変換の基礎となる，1 単位のピクセル数と，原点の座標を格納する変数を用意します．

```
int unit = 40;
int orgx = width/2;
int orgy = height/2;
```

なお，これらの値はコード全体で使うことになるので，グローバル変数として宣言します．しかし，初期値の設定は描画ウィンドウのサイズを用いているので，size メソッドを呼び出した後に行うようにします．したがって，上記のコードを宣言と初期値の代入の二つに分ける必要があります．

これを元に，描画ウィンドウの座標系における X 座標をグラフ座標系に変換するメソッドは次のようになります．

```
float WXtoGX( int wx )
{
  float gx = (float)(wx - orgx) / (float)unit;
  return gx;
}
```

この逆，グラフ座標系における X 座標を描画ウィンドウの座標系に変換するメソッドは次のようになります．

```
int GXtoWX( float gx )
{
  int wx = orgx + (int)( gx * unit );
  return wx;
}
```

同様にして Y 座標に関する変換メソッドを作ってみましょう．

グラフ用紙を描く

描画ウィンドウをグラフ用紙に見立てるように，線を描いてみます．まず setup メソッドの中で，黒板のときと同じように適当なサイズで背景色が深緑色になる設定を行います．

線を描くのは draw の中が好ましいです．ただ，せっかくメソッド定義を覚えたので，グラフ線を描くメソッドを作り，それを draw から呼び出すことにしましょう．次のメソッドでは X 軸と X 軸に平行なメモリ線を描く処理が入っています．

```
void drawOrgLine()
{
  stroke(127);
  strokeWeight(1);
  for( int y = orgy ; y >= 0 ; y -= unit )
```

```
  {
    line( 0, y, width-1, y );
  }
  for( int y = orgy ; y <= height-1 ; y += unit )
  {
    line( 0, y, width-1, y );
  }

  strokeWeight(3);
  line( 0, orgy, width-1, orgy );
}

void draw()
{
  drawOrgLine()
}
```

Y 軸と Y 軸に平行なメモリ線も引いてみましょう.

三角形を描く

三角形を描くために，三つの頂点の座標を格納する変数を用意します．格納する座標の座標系はもちろんグラフ座標系を使いましょう．三つの頂点を別々の変数で格納しても構いませんが, 配列に入れた方が便利なことが多いので，配列を使うことにします．この配列は，コード全体から利用しますのでグローバル変数として宣言します.

```
float[] posx = new float[3];
float[] posy = new float[3];
      :
      :
void setup()
{
      :
      :
  posx[0] =  0.0; posy[0] =  5.0;
  posx[1] = -4.0; posy[1] = -3.0;
  posx[2] =  4.0; posy[2] = -3.0;
}
```

三角形の描画はもちろん draw の中で行います．ただ, この描画もメソッドを作り, draw から呼び出すようにしてみます.

```
void drawTriangle( float[] px, float[] py )
{
  noFill();
  stroke(255);
  strokeWeight(3);
  strokeJoin(MITER);
  beginShape();
    for( int i = 0; i <= 3 ; i++ )
    {
      vertex( GXtoWX( px[i%3] ), GYtoWY( py[i%3]) );
    }
  endShape();
}
void draw()
{
      :
      :
  drawTriangle( posx, posy );
}
```

drawTriangle の中でも配列 posx, posy は利用できるので, わざわざパラメータで渡すことはありません．ただ，配列を渡す方法を示したかったために，今回はわざわざ三角形を構成する三点を渡すようにしてみました.

このコードの中にはいくつか新しい組み込みメソッドがあるので，その説明をします．beginShape は endShape と対になって，その中で指定された点を結ぶ描画を行います．vertex は直線で結ぶ点を登録するメソッドですので，全体として登録した 3 点を結ぶ直線分を描きます．strokeJoin は線分と線分の接続の仕方を指定するメソッドで，MITER は接続部分を尖がらせる指定になります．これ以外に BEVEL と ROUND があります．どのように変わるかは実際に試してみてください．

頂点の付近にマウスカーソルが乗ったら効果を示す

マウスカーソルが頂点の付近に近付いたら，ドラッグで移動ができることを示すために，赤丸を表示してみましょう．

赤丸の表示は draw の中が適当なので, draw の中でマウスカーソルの位置を取得し, 頂点のそばであるかを判定して，そばであったら，頂点の座標を中心に赤丸を描くことにします．「そば」の定義は，頂点座標とマウスカーソルの座標の差が，X, Y 座標共に一定値以下であったらということにします．一定値については変数 bs に格納しておきます．i 番目の頂点についての処理は次のようになります．

```
int bs = 10;
   :
void draw()
{
  noStroke();
  fill(119,100,25,96);
  rect(0, 0, width, height);
    :
  int px, py;
  px = GXtoWX(posx[0]);
  py = GYtoWY(posy[0]);
  if (abs(mouseX-px)<bs && abs(mouseY-py)<bs )
  {
    stroke(#FF0000);
    fill(#FF0000);
    ellipse(px, py, bs*2, bs*2);
  }
}
```

頂点の座標を描画ウィンドウの座標系に変換した後，マウスカーソルの座標との差を求め，その差が bs 以下であったら丸を描いています．なお，差については絶対値を用いないと，マウスカーソルが頂点よりも小さな座標であるときに常に成り立つということになってしまいますので，絶対値を取って比較をしています．絶対値をとるメソッドは abs です．

また，マウスカーソルが頂点から離れたときにきちんと赤丸を消すために，draw の先頭に，背景色で描画ウィンドウ全体を塗りつぶす処理を入れてあります．

他の頂点についても赤丸が表示されるようにしてみましょう．もちろん for 文を使って，同じコードを何度も書かないようにしましょう．

ドラッグで頂点を動かせるようにしてみる

表示した赤い丸の上でマウスボタンを押すと，ドラッグで頂点を移動できるようにします．頂点の上でマウスボタンを押されたかどうかの判定は，今までと同じように mousePressed の中で行います．ただ，今まではドラッグしているか，していないかを保存すればよかったのですが，今回はどの頂点がドラッグされているかを保存しておく必要があるため，ドラッグ中のフラグは boolean 型ではなく，int 型にして，ドラッグしていないときは -1，i 番目の頂点をドラッグしているときは i を格納することにします．なお，赤い丸の上でマウスボタンが押されたかどうかの判定は，頂点のそばにマウスカーソルがあるかどうかの判定と同じ式になります．

```
int isdrag = -1;
void mousePressed()
{
  int px, py;
  px = GXtoWX(posx[0]);
  py = GYtoWY(posy[0]);
  if (abs(mouseX-px)<bs && abs(mouseY-py)<bs)
  {
    isdrag = 0;
  }
}
void mouseReleased()
{
  isdrag = -1;
}
```

このコードは 0 番目の頂点についての処理を行っていますので，他の頂点についても処理を行うようにしましょう．

ドラッグを行ったときの実際の頂点の移動は，三角形の頂点の座標が常に整数となる（頂点が目盛線の交点上になる）ように，グラフ座標系における 1 単位ごとにしてみます．頂点の移動処理はドラッグ中に呼び出される mouseDragged 内で行うのが適当です．draw の中でもかまいませんが，draw は描画を行うための場所と分けたほうがよいと思います。

isdrag フラグを見て，頂点をドラッグ中であれば頂点の座標を変更します．頂点の座標値を整数にするには切り上げ処理を行う ceil メソッドを用います．しかし，単純に切り上げると，0.1 から 0.9 までが 1 になってしまい，ある座標点からちょっとでも正方向にマウスカーソルを移動させると，ぴょん！と頂点が動いてしまいますので，整数点の間隔の半分以上マウスカーソルが動いたら頂点が動くように四捨五入処理を入れています．

```
void mouseDragged()
{
  if( isdrag >= 0 )
  {
    posx[isdrag] = ceil(WXtoGX(mouseX)-0.5);
    posy[isdrag] = ceil(WYtoGY(mouseY)-0.5);
  }
}
```

また，今回は頂点が整数の座標しか取れないため，底辺が水平でなくなると，同じ面積の三角形を作ることが難しくなるので，底辺を構成する 2 つの頂点は，Y 軸方向には移動できないようにしておくとよいと思います．

なお，ドラッグ中は常に赤い丸を表示するように draw 内のコードを変更しておかないと，ある程度マウスカーソルが頂点から離れたときに赤い丸が消えてしまうという現象が発生してしまいますので，変更しておきましょう．

面積を表示する

あとは適当な場所に面積を表示するだけです．面積を表示するコードを入れて完成させてみましょう．