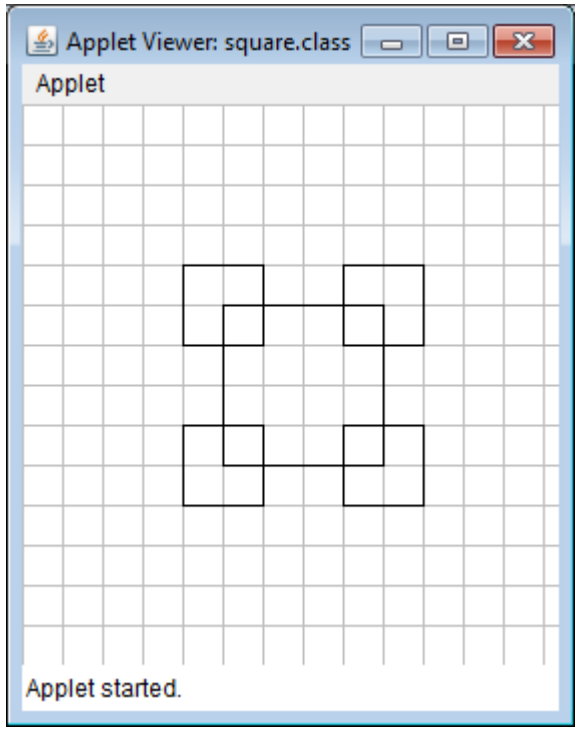
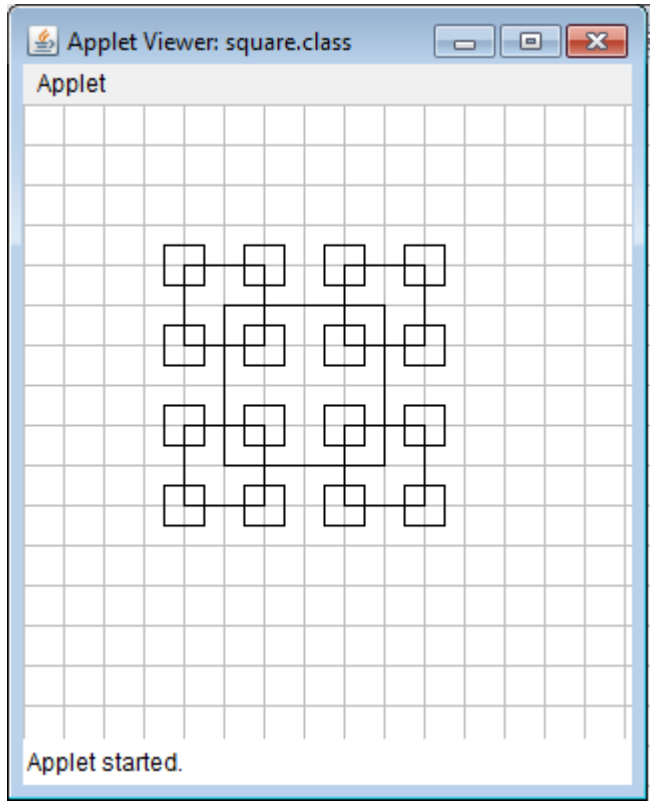
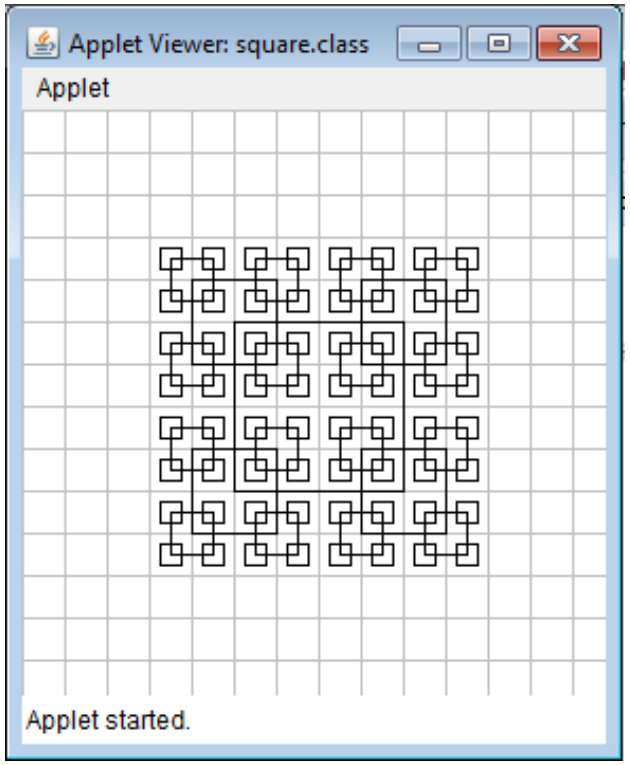
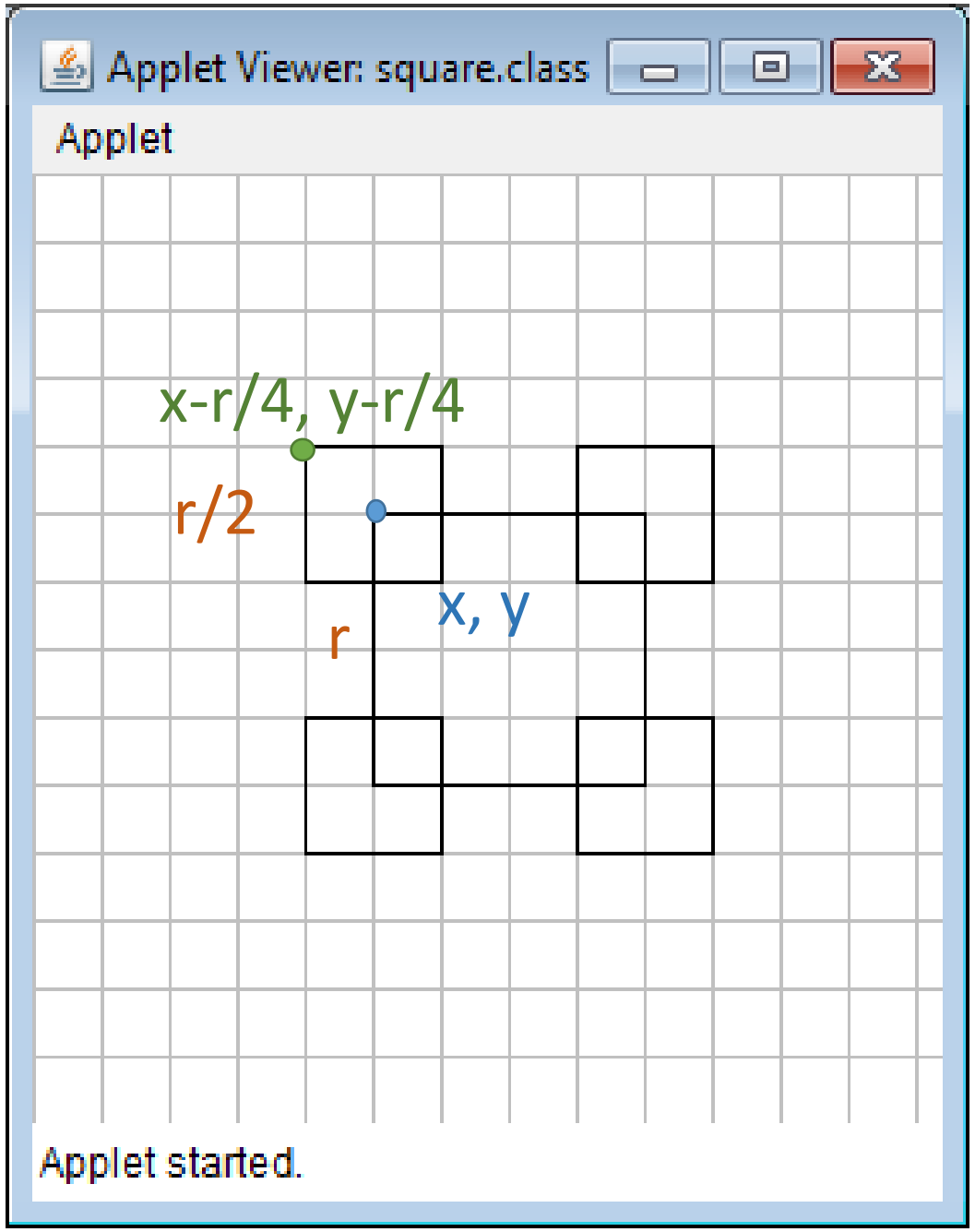
Recursive Pictures


Applet Viewer: square.class
Applet
Applet started.

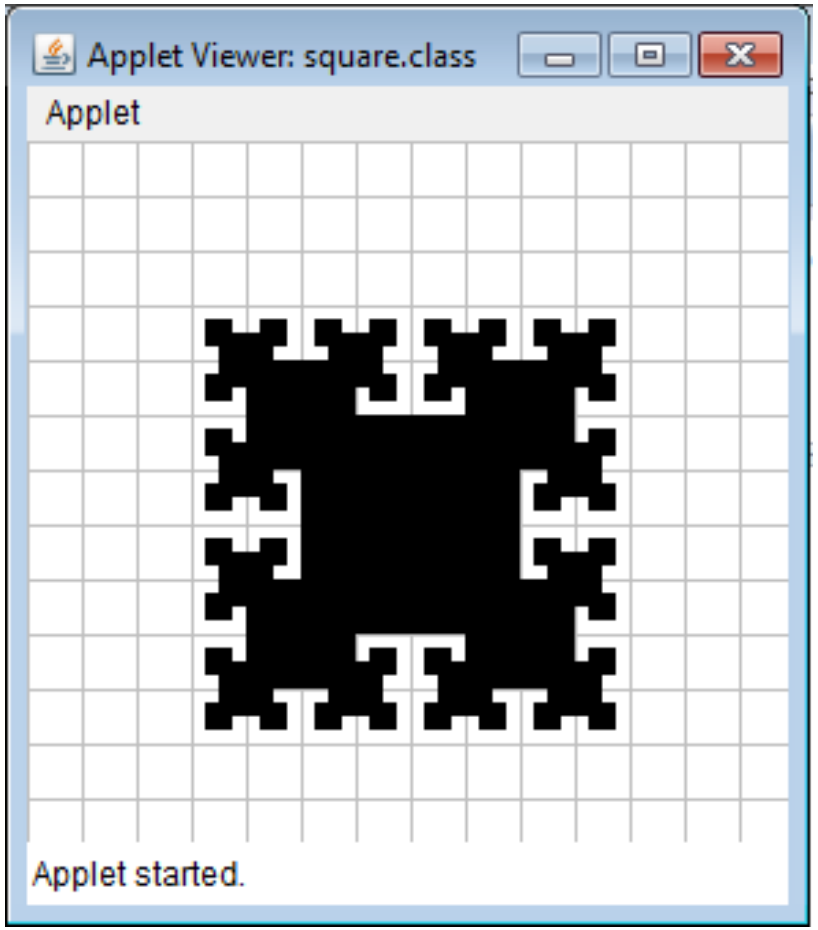

Applet Viewer: square.class
Applet
Applet started.

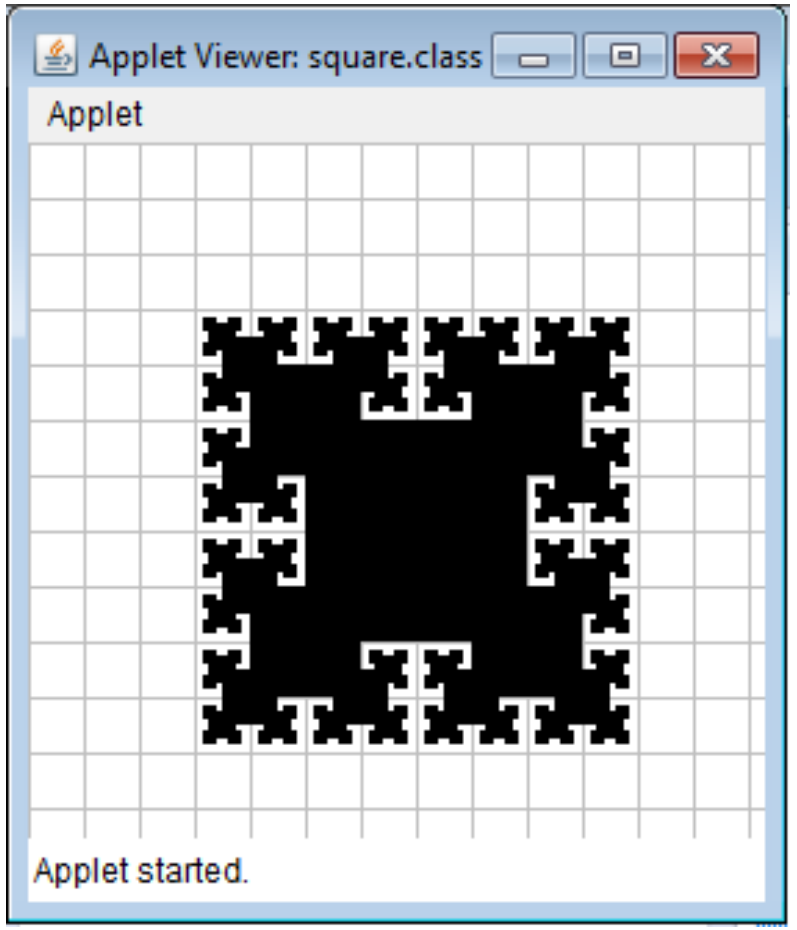

Applet Viewer: square.class
Applet
Applet started.

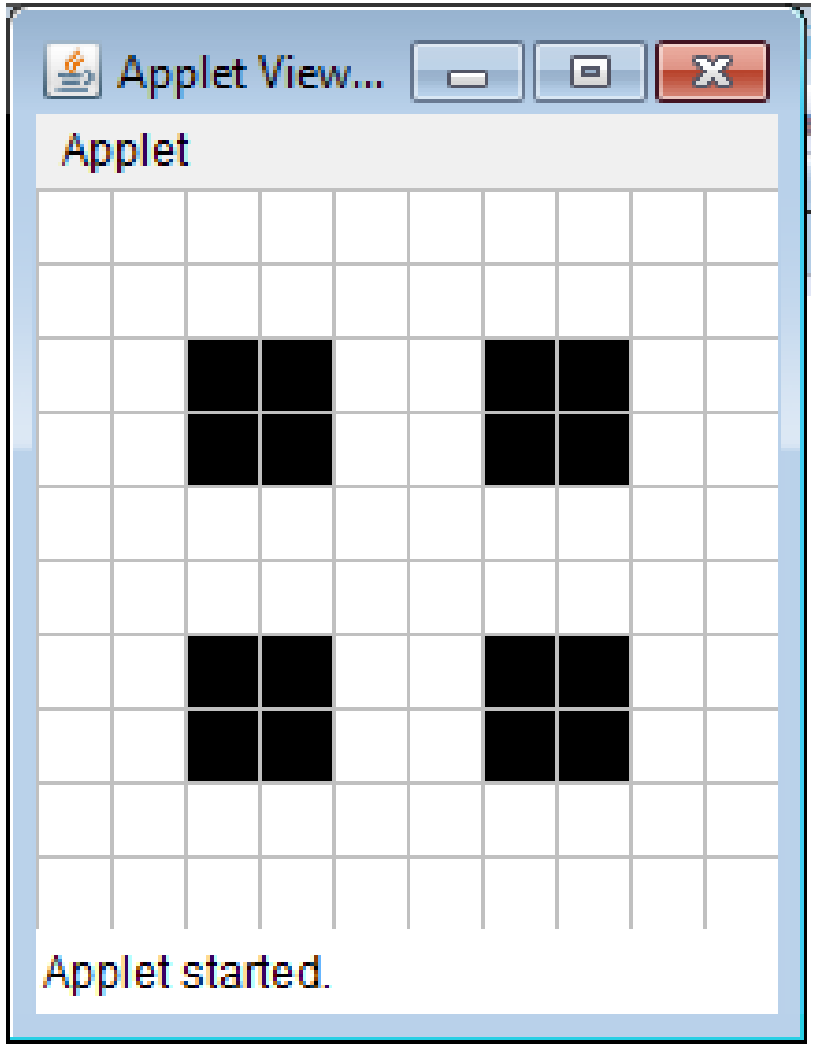



```
public void sq (int x, int y, int r) {
  Graphics g = getGraphics ();
  if (r <5)
     return;
  else {
   g.fillRect (x, y, r, r);
   sq (x-r/4 , y-r/4, r / 2);
   }
}
```

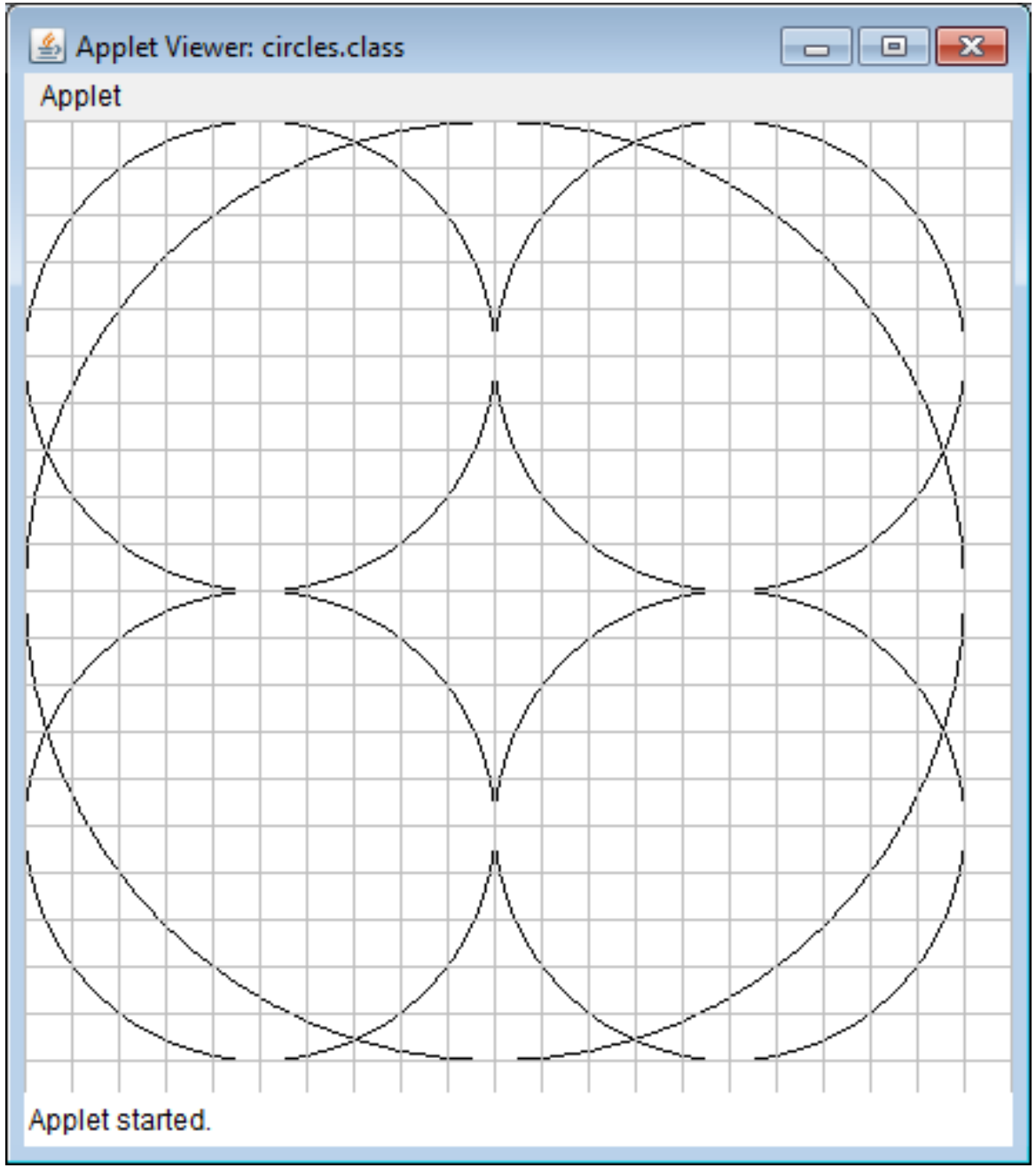
If you fill it, you can see where it’s name… T-square fractal… comes from.

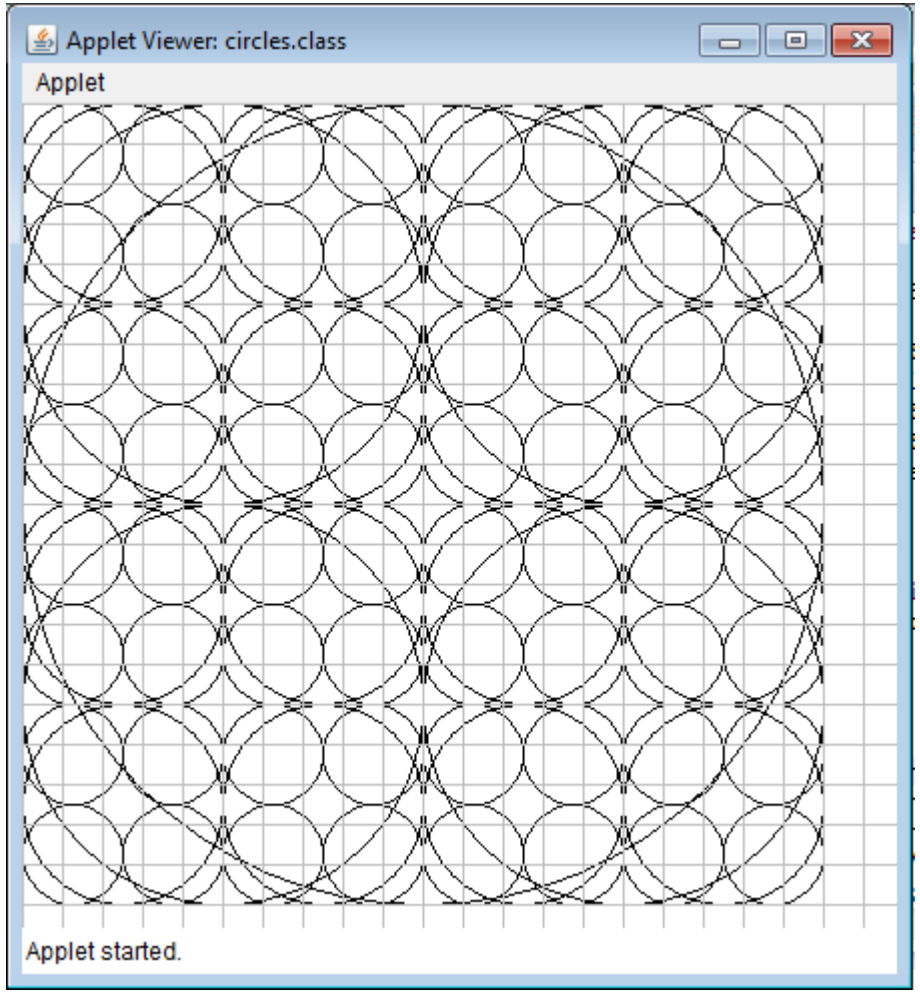


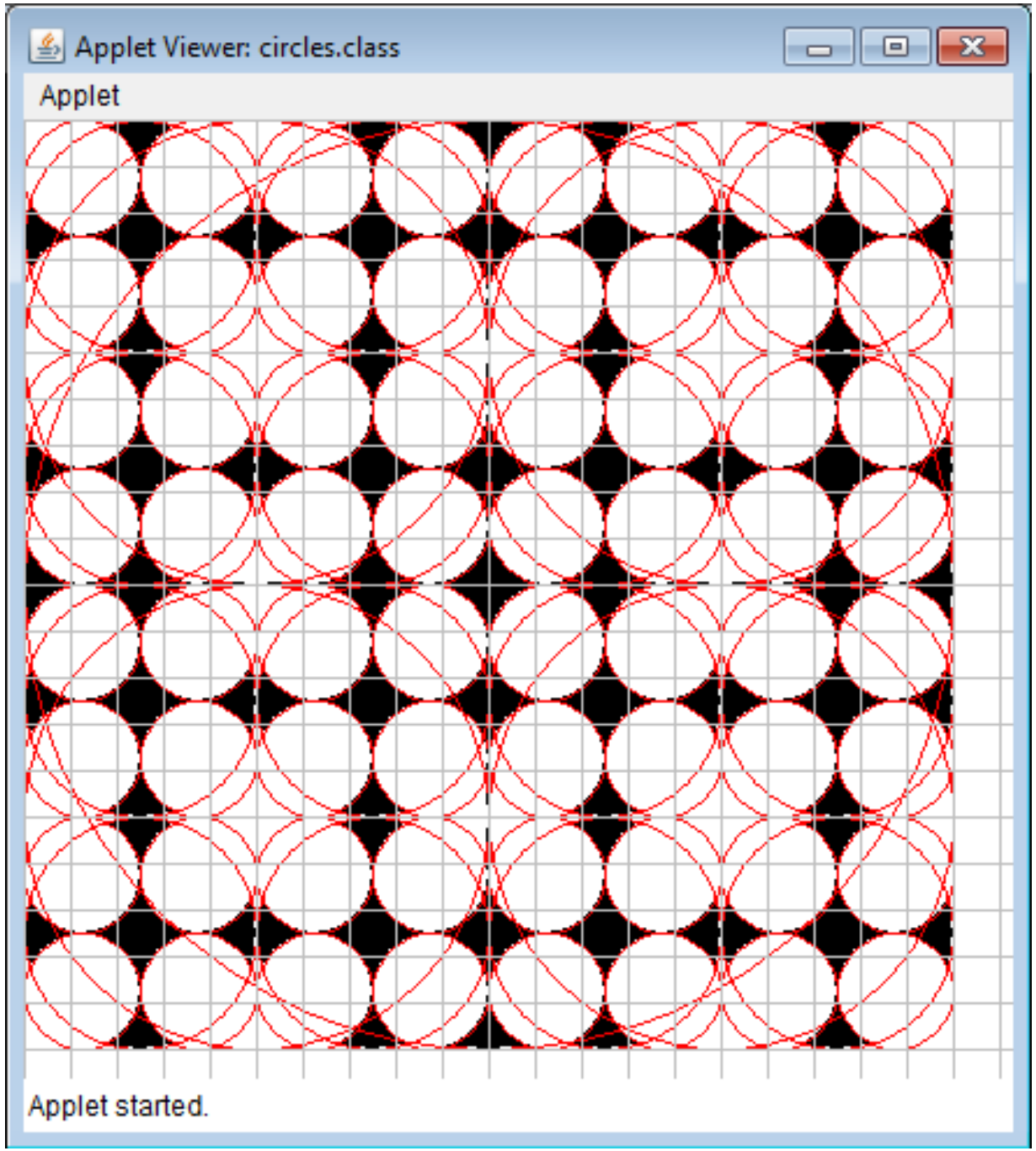





```
public void cantorDust (int x, int y, int r) {
Graphics g = getGraphics ();
if (r <20)
return;
else {
g.setColor(Color.white);
g.fillRect (x, y, r, r);
g.setColor(Color.black);
g.fillRect(x,y,r/3,r/3);
g.fillRect(x+2*r/3,y,r/3,r/3);
g.fillRect(x,y+2*r/3,r/3,r/3);
g.fillRect(x+2*r/3,y+2*r/3,r/3,r/3);

cantorDust (x , y, r / 3);
… and so on… for the 4 corners
}
}
```

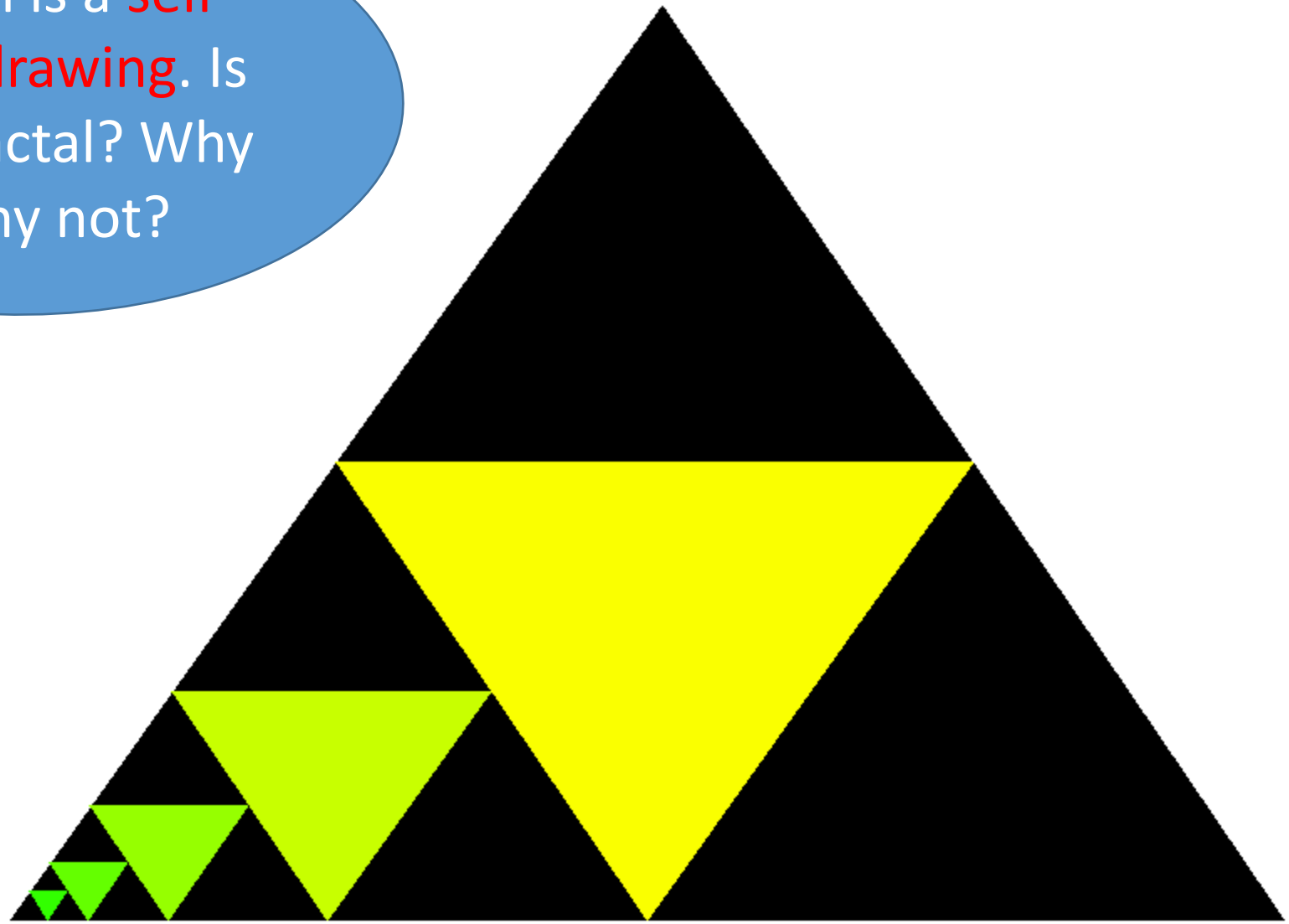

Applet Viewer: circles.class
Applet
Applet started.


Applet Viewer: circles.class
Applet
Applet started.



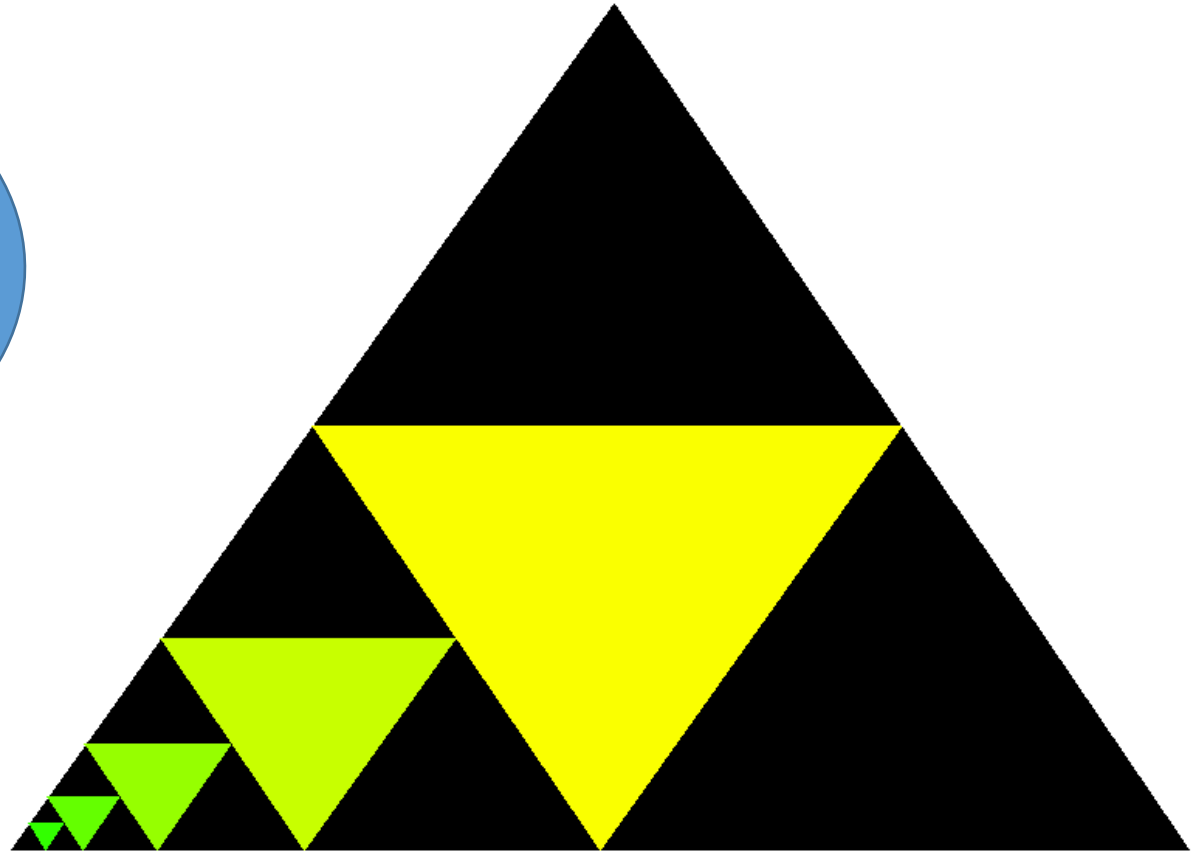

```
public void pattern (int x, int y, int r, int level) {
Graphics g = getGraphics ();
if (r < 30)
  return;
else {
  if(level%2==1)
    g.setColor(Color.black);
  else
    g.setColor(Color.white);

  g.fillOval (x, y, r, r);

  pattern(x+r/2, y, r/2, level+1);
  pattern(x, y+r/2, r/2, level+1);
  pattern(x+r/2, y+r/2, r/2, level+1);
  pattern(x, y, r/2, level+1);

  g.setColor(Color.red);
  g.drawOval (x, y, r, r);
}
}
```

A fractal is a self-similar drawing. Is this a fractal? Why or why not?



```
public void tri (int ax, int ay, int bx, int by, int cx, int cy, int i)
tri (487,   44,   859,  591,   97,   591,   5);
```

ax, ay



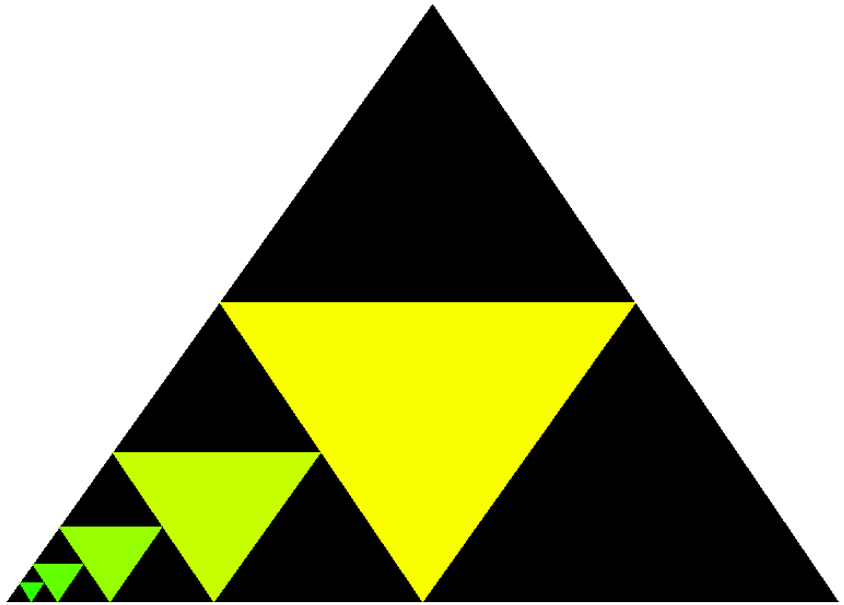
cx, cy

bx, by

Make sure you understand the method signature.

What is ‘5’ on the picture?

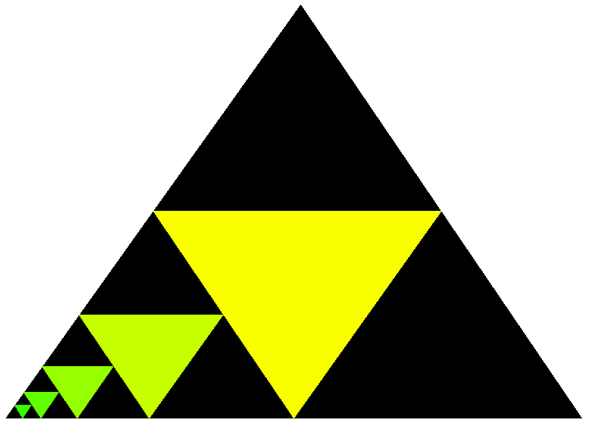
Every recursive method needs TWO pieces. What are they?



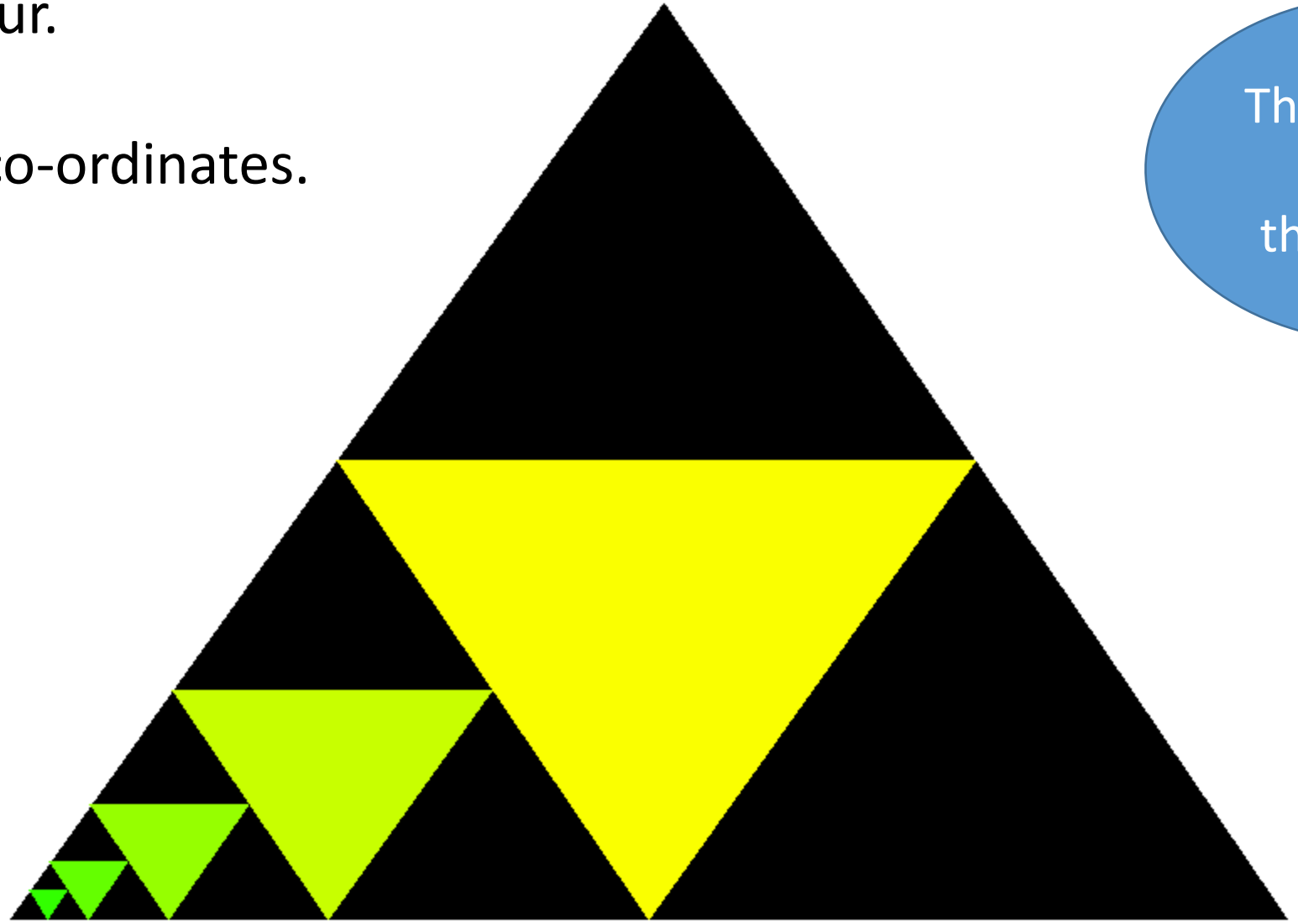
Assume that i slowly works
it's way down to 1.
What is the base case?

```
tri (  487,   44,   859,   591,   97,   591,   5);
```

```
public void tri (int ax, int ay, int bx, int by, int cx, int cy, int i)
{     Graphics g = getGraphics ();
      if (i < 1)
         return;
      else
      { //recursive case
      }
}
```



Find new triangle.
Find new colour.
Draw it.
Pass on your co-ordinates.



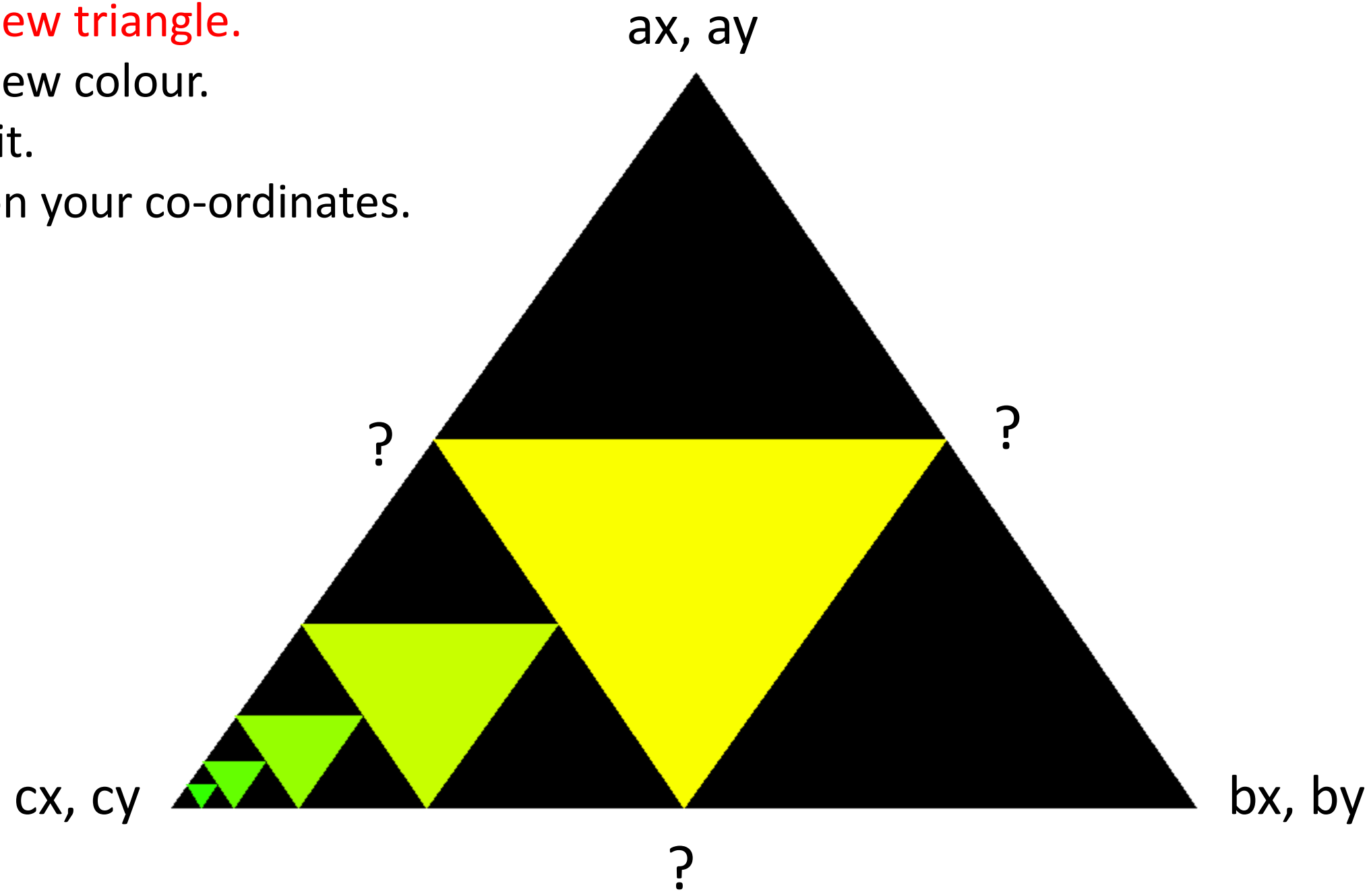
The recursive case has these steps.

Find new triangle.

Find new colour.

Draw it.

Pass on your co-ordinates.

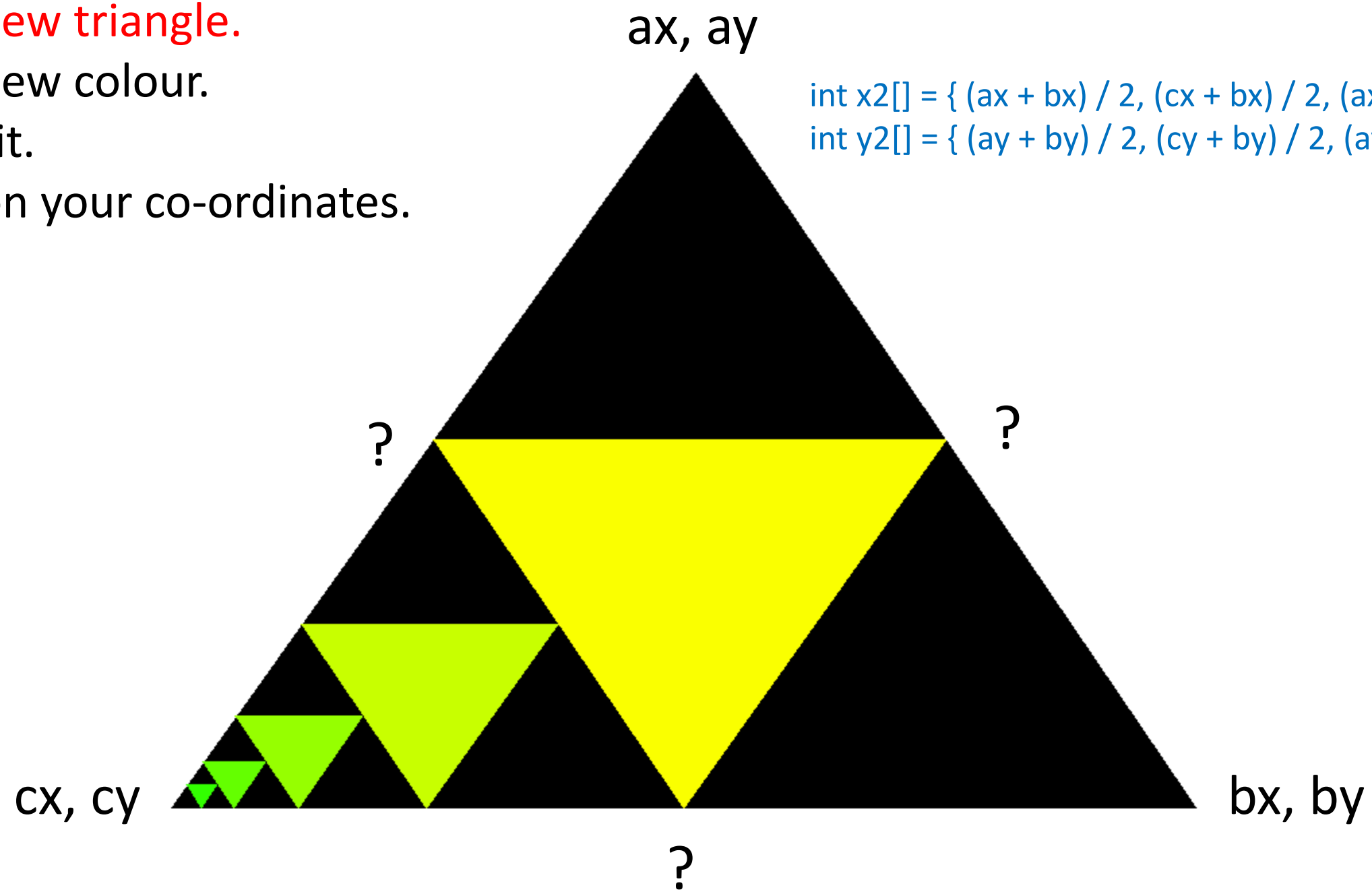


Find new triangle.

Find new colour.

Draw it.

Pass on your co-ordinates.




```
int x2[] = { (ax + bx) / 2, (cx + bx) / 2, (ax + cx) / 2};
int y2[] = { (ay + by) / 2, (cy + by) / 2, (ay + cy) / 2};
```

Find new triangle.

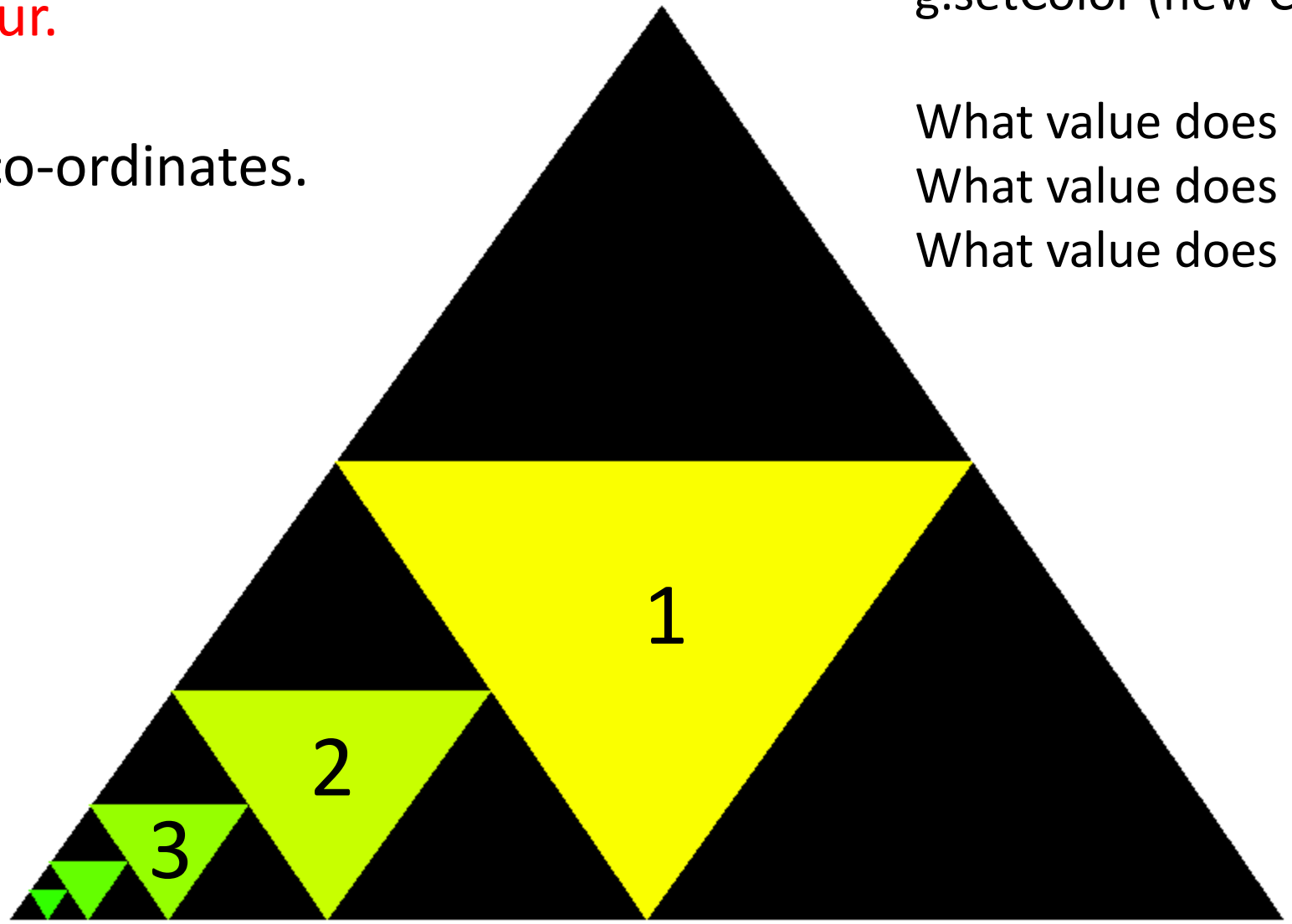
Find new colour.

Draw it.

Pass on your co-ordinates.

g.setColor (new Color (R, G, B));

What value does B start at? End?
What value does G start at? End?
What value does R start at? End?



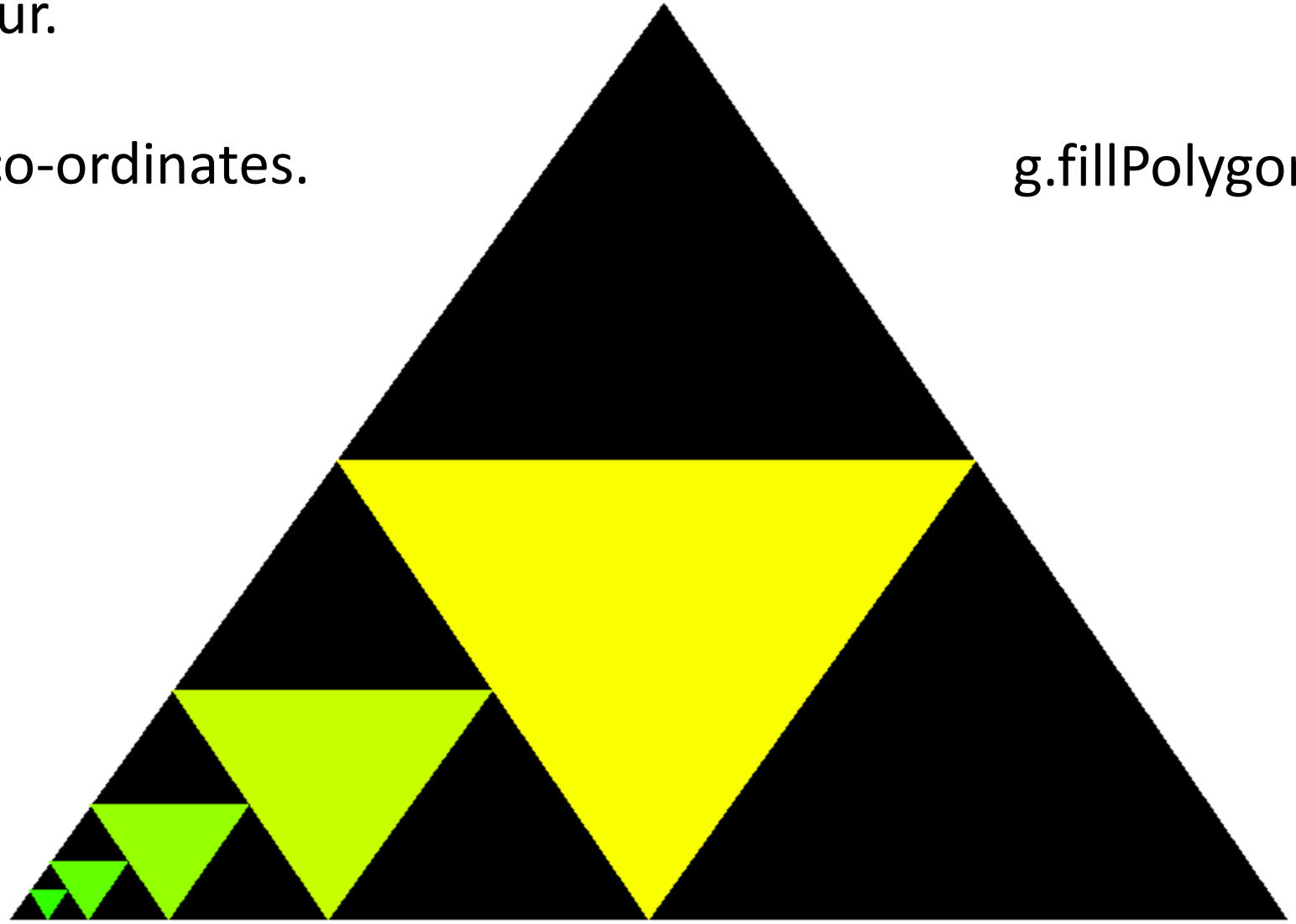
Find new triangle.

Find new colour.

Draw it.

Pass on your co-ordinates.

g.fillPolygon (x2, y2, 3);



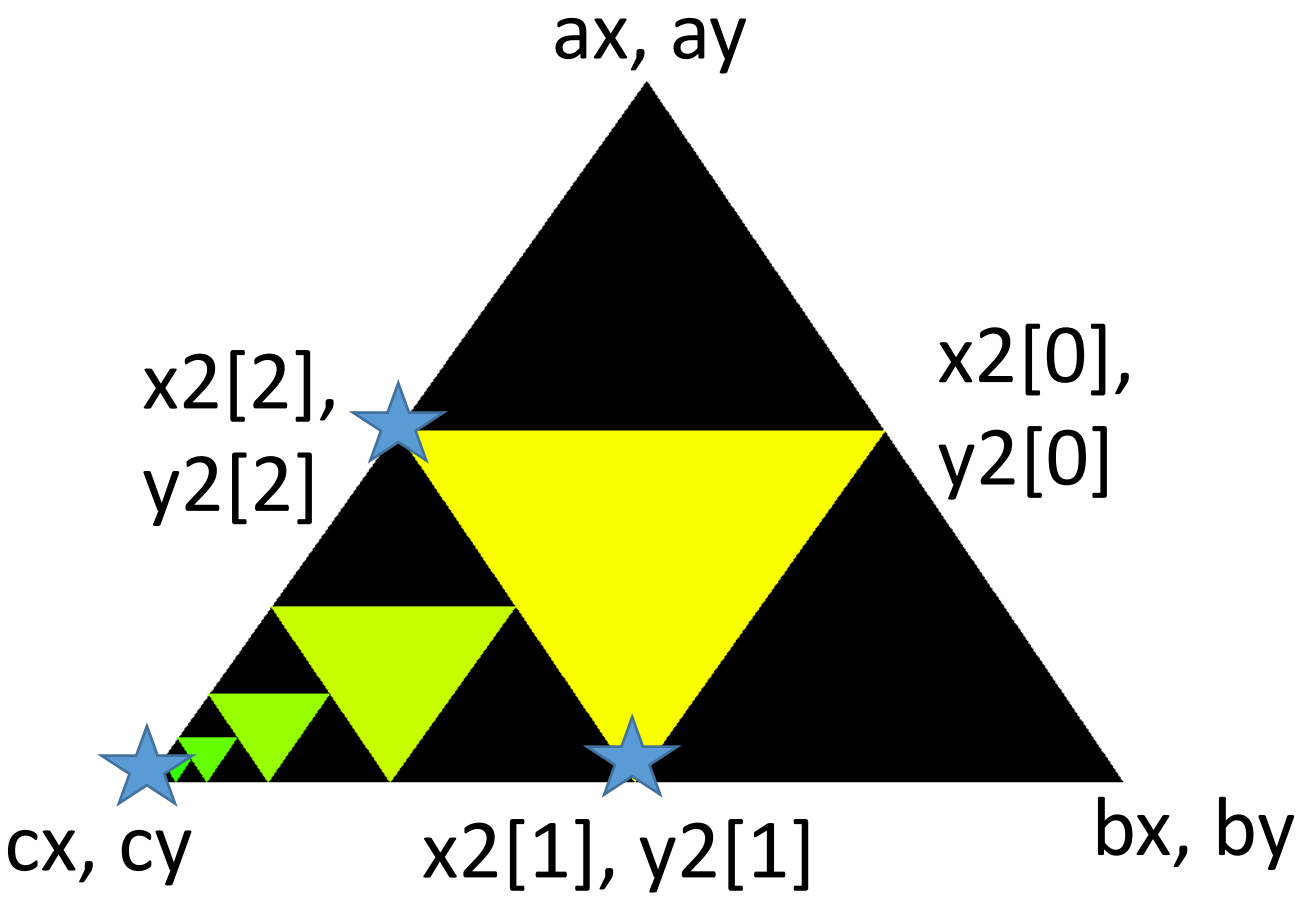
Find new triangle.

Find new colour.

Draw it.

Pass on your co-ordinates.

```
public void tri (int ax, int ay, int bx, int by, int cx, int cy, int i)
int x2[] = { (ax + bx) / 2, (cx + bx) / 2, (ax + cx) / 2};
int y2[] = { (ay + by) / 2, (cy + by) / 2, (ay + cy) / 2};
```



Find new triangle.

Find new colour.

Draw it.

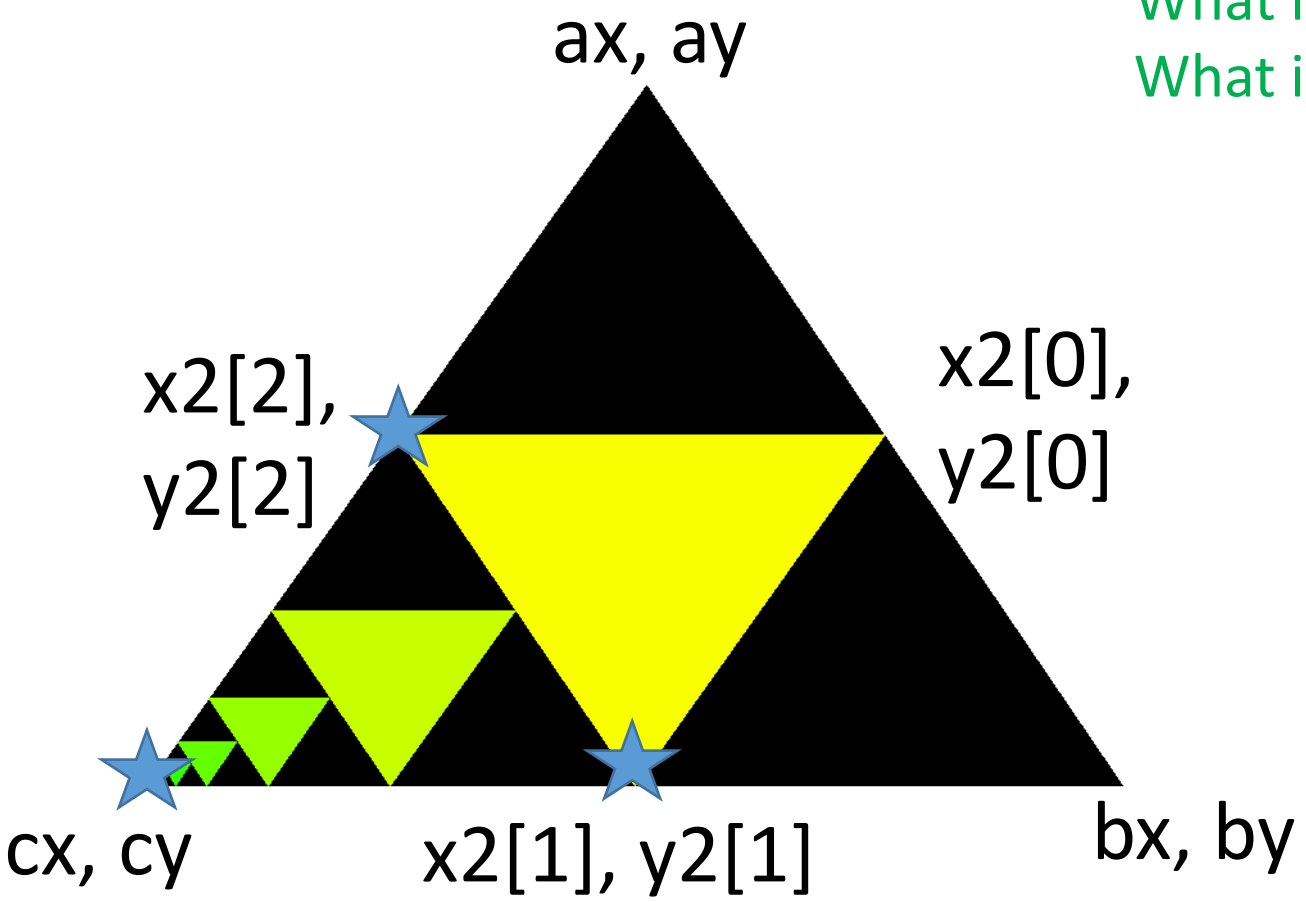
Pass on your co-ordinates.

```
public void tri (int ax, int ay, int bx, int by, int cx, int cy, int i)
int x2[] = { (ax + bx) / 2, (cx + bx) / 2, (ax + cx) / 2};
int y2[] = { (ay + by) / 2, (cy + by) / 2, (ay + cy) / 2};
```

What is needed for ax, ay in the next generation?

What is needed for bx, by in the next generation?

What is needed for cx, cy in the next generation?



Find new triangle.

Find new colour.

Draw it.

Pass on your co-ordinates.

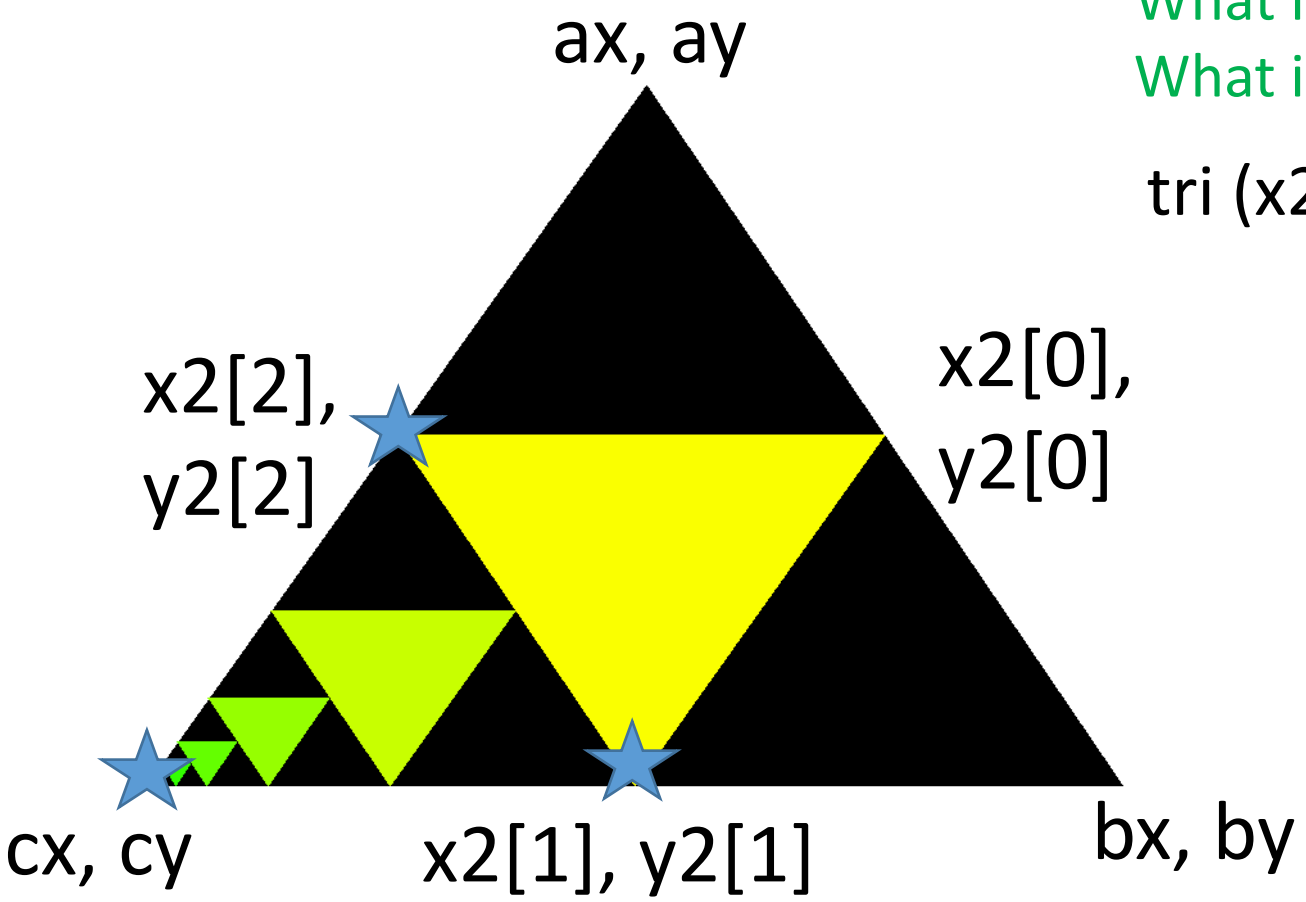
```
public void tri (int ax, int ay, int bx, int by, int cx, int cy, int i)
int x2[] = { (ax + bx) / 2, (cx + bx) / 2, (ax + cx) / 2};
int y2[] = { (ay + by) / 2, (cy + by) / 2, (ay + cy) / 2};
```

What is needed for ax, ay in the next generation?

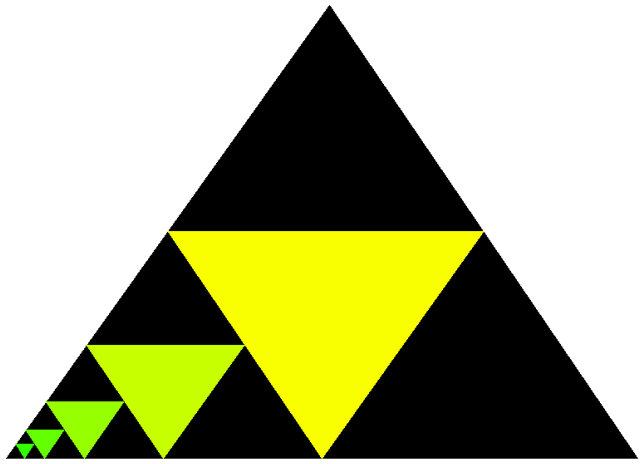
What is needed for bx, by in the next generation?

What is needed for cx, cy in the next generation?

```
tri (x2 [2], y2 [2], x2 [1], y2 [1], cx, cy, i - 1);
```



```
import java.applet.*; import java.awt.*;
public class exampleTri extends Applet
{
   public void paint (Graphics g)
   {
      int x[] = {487, 859, 97};
      int y[] = {44, 591, 591};
      g.fillPolygon (x, y, 3);
      g.setColor (Color.white);
      tri (487, 44, 859, 591, 97, 591, 5);
   }
```



```
public void tri (int ax, int ay, int bx, int by, int cx, int cy, int i)
{
   Graphics g = getGraphics ();
   if (i < 1)
      return;
   else
   {
      int x2[] = { (ax + bx) / 2, (cx + bx) / 2, (ax + cx) / 2};
      int y2[] = { (ay + by) / 2, (cy + by) / 2, (ay + cy) / 2};
      g.setColor (new Color (i * 50, 255, 0));
      g.fillPolygon (x2, y2, 3);
      tri (x2 [2], y2 [2], x2 [1], y2 [1], cx, cy, i - 1);
   }
}
```