Turtle graphics

TurtleWorld as an object example

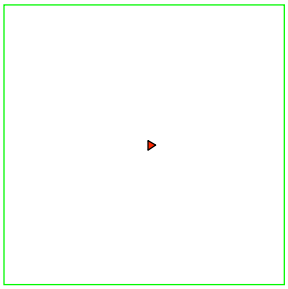
Friday, December 1, 2006

CS111 Computer Programming

Department of Computer Science
Wellesley College

TurtleWorld revisited

- Buggles' smaller, more carefree cousins return to teach an object lesson.
- Recall that Turtles are born centered, pointing EAST, brushDown (red).
- In this lecture, we design and implement a world for Turtles.



A world sophisticate enough to spiral

```
public class SpiralWorld extends TurtleWorld
{
     public void run()
     {
          SpiralMaker spiro = new SpiralMaker();
          spiro.spiral(4, 90, 0, 3)
     }
}

class SpiralMaker extends Turtle
{
     public void spiral(int steps, int angle, int length, int increment)
     {
          if (steps > 0) {
               fd(length);
               lt(angle);
               spiral(steps-1, angle, length+increment, increment);
          }
     }
}
```

Overrides the run() method in TurtleWorld

Or grow trees

```
public class TreeWorld extends TurtleWorld
{
     public void run()
     {
          TreeMaker tina = new TreeMaker();
          tina.tree(4, 100, 45, 0.6);
     }
}
public class TreeMaker extends Turtle
{
     public void tree(int levels, double length, double angle, double shrink)
     {
          if (levels > 0) {
               fd(length);
               rt(angle);
               tree(levels-1, length*shrink, angle, shrink);
               lt(2*angle);
               tree(levels-1, length*shrink, angle, shrink);
               rt(angle);
               bd(length);
          }
     }
}
```

Also overrides run() method in TurtleWorld

In other words, we implement the Turtle contract

`public void fd(double n)`
Moves this turtle forward by length `n` in the direction of its current heading.

`public void bd (double n)`
Moves this turtle backward by length `n` in the direction of its current heading.

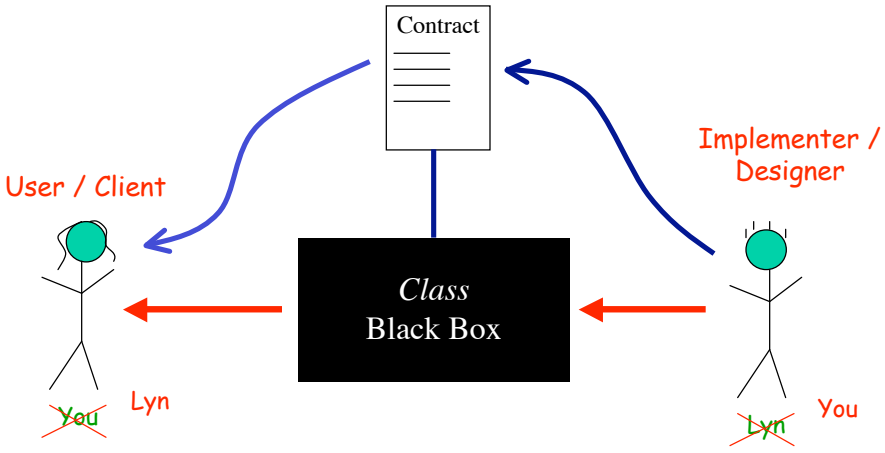
`public void lt (double angle)`
Turns this turtle left by `angle` degrees.

`public void rt (double angle)`
Turns this turtle right by `angle` degrees.

`public void pu ()`
Raises this turtle's pen. When the pen is raised, the turtle leaves no trail when it moves.

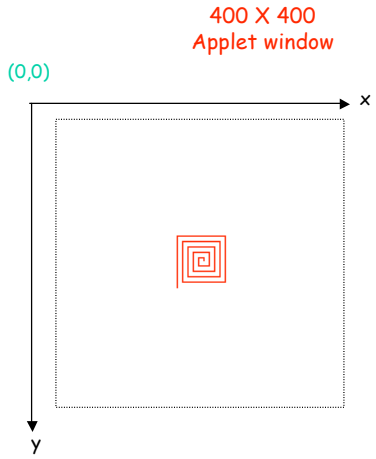
`public void pd ()`
Lowers this turtle's pen. When the pen is lowered, the turtle leaves a trail when it moves.

Big idea number 1: Abstraction



First, Turtle's box

- TurtleWorld is an Applet whose contract contains one method:
  `public void run().`
- We use a Graphics object, to draw the Turtle's art.

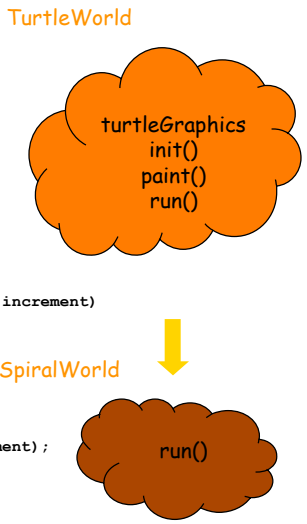


TurtleWorld

```
import javax.swing.*;
import java.awt.*;

public class TurtleWorld extends JApplet
{
    private static Container canvas;

    public void init()
    {
        canvas = getContentPane();
    }
    public void paint (Graphics g)
    {
        this.run();
    }
    public void run()
    {
        // Users will override this method
    }
    public static Graphics getTurtleGraphics()
    {
        return canvas.getGraphics();
    }
    public static int getCanvasHeight()
    {
        return canvas.getHeight();
    }
    …
}
```

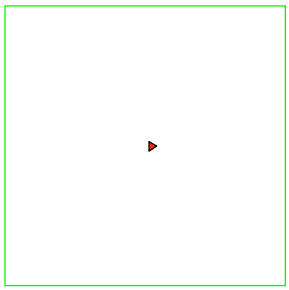
A SpiralWorld object

```
public class SpiralWorld extends TurtleWorld
{
     public void run()
     {
          SpiralMaker spiro = new SpiralMaker();
          spiro.spiral(4, 90, 0, 3)
     }
}
class SpiralMaker extends Turtle
{
     public void spiral(int steps, int angle, int length, int increment)
     {
          if (steps > 0) {
               fd(length);
               lt(angle);
               spiral(steps-1, angle, length+increment, increment);
          }
     }
}
```



Now for the main event

```
import java.awt.*;     // Import Abstract Window Toolkit

public class Turtle
{
     // Instance variables?

          private or public?

     // Constructor methods?



     // Instance methods?


}
```

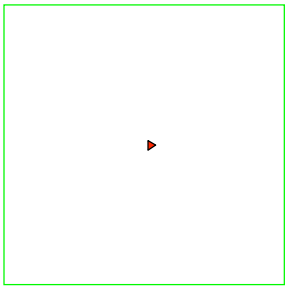


Now for the main event

```
import java.awt.*;       // Import Abstract Window Toolkit

public class Turtle
{
     // Instance variables
     private double x;          // x position of turtle
     private double y;          // y position of turtle
     private double heading;    // heading in degrees
     private boolean pendown;   // is turtle in drawing mode?
     private Color color;       // Red is the default

     // some more stuff
}
```


Constructor methods

```
import java.awt.*;    // Import Abstract Window Toolkit

public class Turtle
{
     private double x;
     private double y;
     private double heading;
     private boolean pendown;
     private Color color;

     // Constructor methods?
          private or public?

     // Instance methods?
}
```

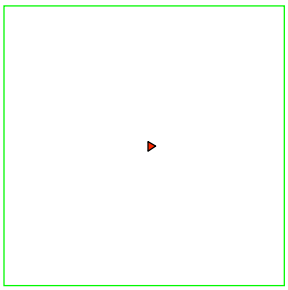


Birth of a Turtle

```
public class Turtle
{
      private double x;          // x position of turtle
      private double y;          // y position of turtle
      private double heading;    // heading in degrees
      private boolean pendown;   // is turtle in drawing mode?
      private Color color;       // Red is the default

      public Turtle()
      {
            this.color = Color.red;
            this.x = 0.0;
            this.y = 0.0;
            this.heading = 0.0;
            this.pendown = true;
      }
      // Instance methods?
}
```


Instance methods*

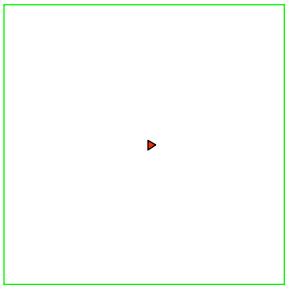
```
public class Turtle
{
      // Instance variables and
      // Constructor methods
      // (been there, done that)

      // Instance methods?

          private or public?




}
```



*How do you decide what these should be?

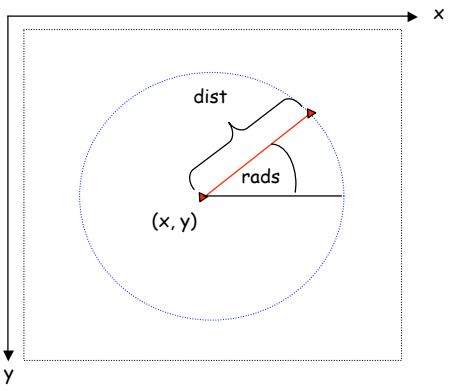
A few methods to implement

```
public class Turtle
{
     // Instance variables and constructor methods
     public void fd(double dist)
     {     …
     }
     public void bd(double dist)
     {     …
     }
     public void lt (double degrees)
     {     …
     }
     public void rt(double angle)
     {     …
     }
     public void pu()
     {     …
     }
     public void pd()
     {     …
     }
}
```

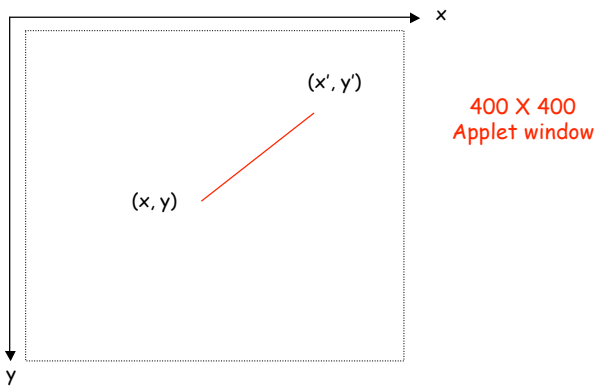


fd(double dist)

```
public void fd(double dist)
{
     double rads = degreesToRadians(this.heading);
     this.moveTo(x + dist*Math.cos(rads), y + dist*Math.sin(rads));
}
```



drawLine(int x1, int y1, int x2, int y2)



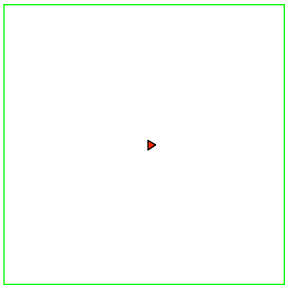

```
Graphics g = TurtleWorld.getTurtleGraphics();
g.setColor(Color.red);
g.drawLine(x, y, x', y');
```


moveTo(double new_x, double new_y)

```
public void moveTo(double new_x, double new_y)
{
    if (pendown) {
         Graphics g = TurtleWorld.getTurtleGraphics();
         g.setColor(color);
         g.drawLine(screenX(x), screenY(y),
                    screenX(new_x), screenY(new_y));
    }
    x = new_x;
    y = new_y;
}
// Assume origin is (0,0)
private int screenX(double xcoord)
{
    return (int) Math.round((TurtleWorld.getCanvasWidth()/2) + xcoord);
}
private int screenY(double ycoord)
{
    return (int) Math.round((TurtleWorld.getCanvasHeight()/2)
                            - ycoord);
}
```

The rest are a piece of cake

```
public class Turtle
{
     // Instance variables and constructor methods
     // fd() method as above

     public void bd(double dist)
     {
     }
     public void lt (double degrees)
     {
     }
     public void rt(double angle)
     {
     }
     public void pu()
     {
     }
     public void pd()
     {
     }
}
```



There is a problem

```
public class SpiralWorld extends TurtleWorld
{
     public void run()
     {
          SpiralMaker spiro = new SpiralMaker();
          spiro.spiral(4, 90, 0, 3)
     }
}
class SpiralMaker extends Turtle
{
     public void spiral(int steps, int angle, int length, int increment)
     {
          if (steps > 0) {
              fd(length);
              lt(angle);
              spiral(steps-1, angle, length+increment, increment);
          }
     }
}
```

Overloading methods


```
public class Turtle
{
     // Instance variables and constructor methods
     // fd() method as above

     // Backward
    public void bd(double dist)
    {
         this.fd(- dist);
     }
     public void bd (int dist)
     {
          bd((double) dist);
     }
     // Left
     public void lt(double degrees)
     {
          this.heading = heading + degrees;
     }
     public void lt (int degrees)
     {
          lt((double) degrees);
     }
}
```

casting ints into doubles