Manipulating Turtles

CS1316: Representing Structure and Behavior

Story
- Introduction to the Turtle
  - Historical: A Child’s First Object
  - Modern day: Traffic, Ants, and Termites
- A Java Turtle
  - On Worlds and Pictures
  - Moving and rotating and composing pictures
- Using the Java Turtle to create animations

The Logo Turtle
- A robot with pen
  - For children to program graphics, in a day before graphics terminals.
  - Literally, a pen would drop down (with the command `penDown`) and would draw on the paper below it, as the turtle moved with commands like `forward` and `right`.
- Nowadays, replaced with a graphical representation.

Turtles can go forward and turn; they know heading and position

```
> fred.forward(100);
> fred.turn(90);
> fred.getHeading();
90
> fred.getXPos();
320
> fred.getYPos();
140
```

Obviously: Turtles are objects
- Turtles can do:
  - `forward` (pixels)
  - `turn` (degrees)
- Turtles know:
  - Heading
  - Position
Drawing with Turtles

```java
> for (int sides=0; sides <= 4; sides++)
  {fred.forward(100);
   fred.turn(90);}
// Actually did five sides here...
```

Can we cascade?

- Will this work?
  ```java
turtle.forward(100).turn(90)
```
- Hint: Think about the returns!

Modern turtles: Turtle Geometry and StarLogo

- diSessa and Abelson’s *Turtle Geometry* showed that simple turtle geometry could explore complex math, including Einstein’s Theory of Relativity
- Mitchel Resnick’s StarLogo used thousands of turtles to explore behavior of traffic, ants, and termites.

Exploring ants with turtles

- Move randomly
- If you find food: Grab it, go home, dropping scent.
- If you find scent, turn towards the direction of the scent.

```java
public class LotsOfTurtles {
  public static void main(String[] args)
  {
    // Create a world
    World myWorld = new World();
    // A flotilla of turtles
    Turtle[] myTurtles = new Turtle[100];
    // Make a hundred turtles
    for (int i=0; i < 100; i++)
      myTurtles[i] = new Turtle(myWorld);
    // Tell them all what to do
    for (int i=0; i < 100; i++)
      { // Turn a random amount between 0 and 360
        myTurtles[i].turn((int) (360 * Math.random()));
        // Go 100 pixels
        myTurtles[i].forward(100);
      }
  }
}
```

Making a circle
Thought Experiment

- What’s the difference between this:
  ```java
  Turtle[] myTurtles = new Turtle[100];
  ```
- And this?
  ```java
  for (int i=0; i < 100; i++) {
    myTurtles[i] = new Turtle(myWorld);
  }
  ```
- What are each doing?

More than one Turtle at once

Putting Turtles on Pictures

- Picture canvas = new Picture(400,400);
- Turtle mabel = new Turtle(canvas);
- for (int sides=1; sides <= 4 ; sides++)
  {mabel.forward(100);
   mabel.turn(90);}
- canvas.show();

Using Turtles to compose Pictures

- Picture t = new Picture("D:/cs1316/MediaSources/Turtle.jpg");
- mabel.drop(t)
- canvas.repaint();

Adding new methods to Turtle

Testing our new method
Thought Experiment

- We can have two methods with the same name?
- How did Java know which one to use?

Making more complex pictures: Using main()

```java
public class MyTurtlePicture {
    public static void main(String[] args) {
        Picture canvas = new Picture(600, 600);
        Turtle jenny = new Turtle(canvas);
        Picture lilTurtle = new Picture(FileChooser.getMediaPath("Turtle.jpg"));
        for (int i = 0; i <= 40; i++) {
            if (i < 20) { jenny.turn(20); }
            else { jenny.turn(-20); }
            jenny.forward(40);
            jenny.drop(lilTurtle.scale(0.5));
        }
        canvas.show();
    }
}
```

Also: Note use of getMediaPath

Result:

Thought Experiments

- Is this myTurtlePicture a class? An object?
- Can we access variables from the Interactions Pane?
- Can we return values to the Interactions Pane?
- When is it useful to use a main()? 

Explaining public, and static, and void, and main, and String [] args

```java
public static void main(String[] args) {
    // Public: This method can be accessed by any other class.
    // Static: This is a method that can be accessed through the class, even if no instances of the class exist.
    // Void: This method doesn’t return anything.
    // String[] args: If called from the Command Line (outside DrJava), inputs could be provided.
    // They’d show up as strings in this array.
```

Creating an animation with FrameSequence

```java
public class MyTurtlePicture {
    public static void main(String[] args) {
        Picture canvas = new Picture(600, 600);
        Turtle jenny = new Turtle(canvas);
        Picture lilTurtle = new Picture(FileChooser.getMediaPath("Turtle.jpg"));
        for (int i = 0; i <= 40; i++) {
            if (i < 20) { jenny.turn(20); }
            else { jenny.turn(-20); }
            jenny.forward(40);
            jenny.drop(lilTurtle.scale(0.5));
        }
        canvas.show();
    }
}
```

```java
public class FrameSequence {
    // FrameSequence stores out Pictures to a directory, and can show/replay the sequence.
    // new FrameSequence(dir): dir where the Pictures should be stored as JPEG frames
    // .addFrame(aPicture): Adds this Picture as a frame
    // .show(): Show the frames as they get added
    // .replay(wait): Replay the sequence, with wait milliseconds between frames.
```

```
```
Using FrameSequence

Welcome to DrJava.
> FrameSequence f = new FrameSequence("D:/Temp");
> f.show()
There are no frames to show yet. When you add a frame it will be shown
> Picture t = new Picture("D:/cs1316/MediaSources/Turtle.jpg");
> f.addFrame(t);
> Picture barb = new Picture("D:/cs1316/MediaSources/Barbara.jpg");
> f.addFrame(barb);
> Picture katie = new Picture("D:/cs1316/MediaSources/Katie.jpg");
> f.addFrame(katie);
> f.replay(1000);

Making a turtle drawing animate

Welcome to DrJava.
> MyTurtleAnimation anim = new MyTurtleAnimation();
> anim.next(20);
> anim.replay(500);

Declarations

public class MyTurtleAnimation {
    Picture canvas;
    Turtle jenny;
    FrameSequence f;
    
    public MyTurtleAnimation() {
        canvas = new Picture(600,600);
        jenny = new Turtle(canvas);
        f = new FrameSequence("D:/Temp");
    }

    public void next() {
        Picture lilTurtle = new Picture(FileChooser.getMediaPath("Turtle.jpg"));
        jenny.turn(-20);
        jenny.forward(30);
        jenny.turn(30);
        jenny.forward(-5);
        jenny.drop(lilTurtle.scale(0.5));
        f.addFrame(canvas.copy());
    }

    public void next(int numTimes) {
        for (int i = 0; i < numTimes; i++) {
            this.next();
        }
    }

    public void show() {
        f.show();
    }

    public void replay(int delay) {
        f.show();
        f.replay(delay);
    }
}

A constructor

public MyTurtleAnimation() {
    canvas = new Picture(600,600);
    jenny = new Turtle(canvas);
    f = new FrameSequence("D:/Temp");
}

- We’re going to need a canvas, a Turtle, and a FrameSequence for each instance of MyTurtleAnimation.
  * That’s what the instances know
  * These are called instance variables

Each step of the animation

public void next() {
    Picture iTurtle = new Picture(FileChooser.getMediaPath("Turtle.jpg"));
    jenny.turn(-20);
    jenny.forward(30);
    jenny.turn(30);
    jenny.forward(-5);
    jenny.drop(iTurtle.scale(0.5));
    f.addFrame(canvas.copy());
}

- Do one stage of the drawing.
Try it!

- Why do we call .copy on the canvas?
- Try it without it!
- What does the result suggest to you about how FrameSequence instances store their frames internally?

Being able to replay and see it

```java
public void next(int numTimes)
    for (int i=0; i < numTimes; i++)
        this.next();

public void show()
    f.show();

public void replay(int delay)
    f.show();
    f.replay(delay);
```

JavaDoc on SimpleTurtle

Thought Experiment

- Why SimpleTurtle (and SimplePicture)?
- Hint:
  - Think about information hiding

Other useful methods known to Turtles

- getPicture() – returns the picture that the turtle was opened on.
- turnToFace(aTurtle) – turns to face a particular turtle.
- getDistance(x,y) – returns the number of turtle steps (roughly, pixels) from this turtle to the (x,y) location.

We'll use these later, in simulations