The University of New Mexico

CS 251 — Intermediate Programming: Assignment 1

Pixel Tracing - Rudimentary programming skills

Andree Jacobson

Due: Thursday 2/3/2009 @ 11:59:00pm

Please note: This is an individual assignment. It is designed for you to get back into programming mode. The idea is that you will have to use most of the skills that you learned in cs152 (or its equivalent), and also get acquainted with Eclipse and Java in case you are not already.

Problem Specification

This program is a simple warmup exercise, in which the goal is get you used to: 1) Importing a jar file into eclipse, 2) Using methods in a predefined class (that you didn't write yourself), and 3) Writing your own small class to perform an operation of some sort.

The `Display` class

The `Display` class is a very simple little class. Its sole function is to display pixels. The window keeps track of the last n pixels that was displayed, and displays them as a trace with diminishing color.

- `Display()`
  A default constructor, creating an instance of the display with certain preset values.
- `Display ( int traceLength, int pixelSize )`
  A customized constructor that allows you to specify the number of trailing pixels, and the size of each pixel.
- `void drawNextPixel ( int x, int y )`
  A method that draws one pixel at coordinate (x, y) on the display window.

There are a few more useful methods that you can call on the Display class, and the two you might be interested in are: `getHeight()` and `getWidth()`, which will tell you how many pixels tall and wide the black display area is (as an `int`).

There are three parts to the assignment. Each part needs to be completed before the next part is attempted.

1. **Minimum required** – Make the display window draw a pixel that moves around the screen to create a square. This means that you will have to modify the indices of the pixel to move right, then down, then left, and then up. This is probably most easily accomplished by putting a number of loops in a method.
2. **Expected** – Is what was disucssed in class on Friday. The idea is to trace a point on the circumference of a circle that rotates around a point on the circumference of another circle. See some mathematical explanations below.
3. **Cool** - Come up with your own "cool" version of a moving point. The idea should be more complex than the movement on the circle - Can you come up with some nifty pattern?

Circle trigonometry

Calculating the x and y coordinates for a point p on a circle as described in the figure below, is not too complicated if we use a little trigonometry.

Bascially, referring to the figure, we can calculate:

$$\begin{aligned} \Delta x &= r \cos \alpha \\ \Delta y &= r \sin \alpha \end{aligned}$$

The trignometric methods in the java `Math` class, expects angles measured in radians, so if you are used to thinking of angles as degrees, you may want to convert those degrees to coordinates before applying the sin and cos methods. The conversion is simple: rads $= \frac{\pi \cdot \text{degs}}{180}$.

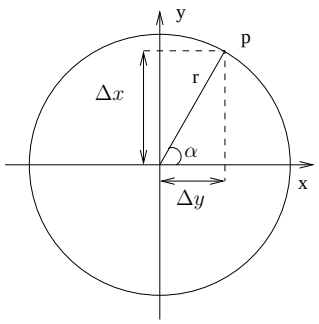


Figure 1: The measurements of a circle

What you have to do

Create a class named `Visualizer` that contains one method for each of the three parts as described above. Each method might well need one or more private helper methods if you so wish.

In order to use the `Display` class, you will need to import a package distributed with the assignment. Separate instructions on how to do this is available further down in this document.

Then, in your Visualizer class, you need an import statement looking like this:

```
import edu.unm.cs.cs251.andree.spring10.prog01.Display;
```

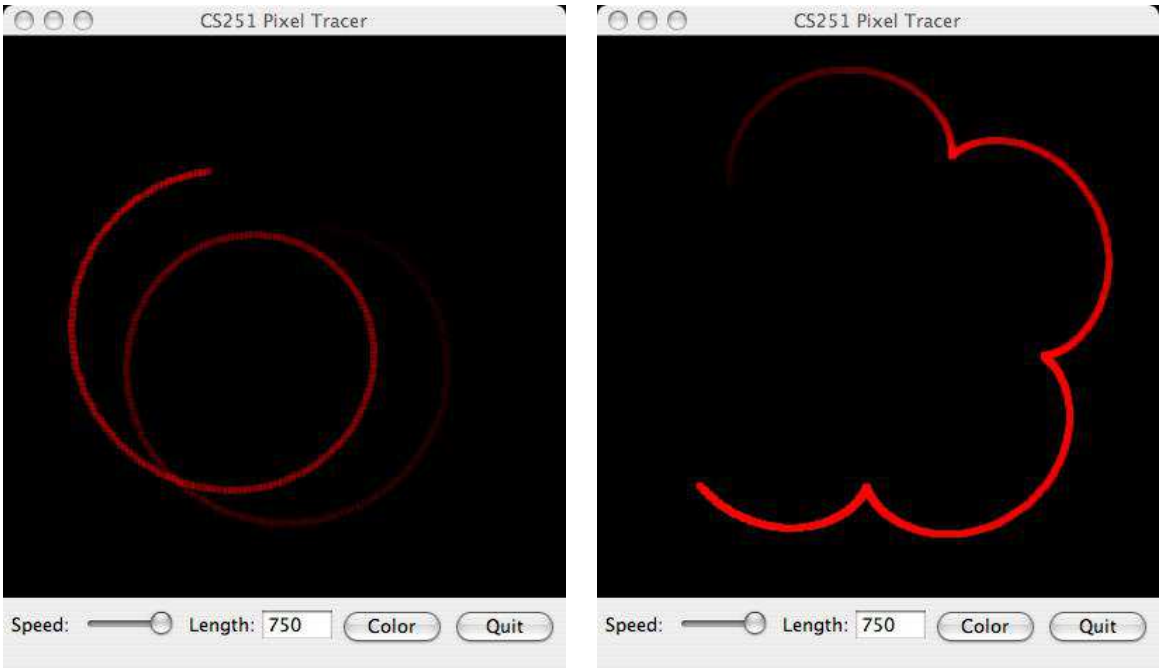
Once that's done, creating the display interface is now easy:

```
Display myDisplay = new Display ( );
```

Call the drawNextPixel method in the Display class with a coordinate of (200, 200), and a pixel should be drawn on the center of the screen.

Examples of what you could achieve

Below are two patterns that you might be able to form with the right constants in the epicycle code.



Installing and using subversion

The truth is that how this works depends a little on the machine that you are on, but some general instructions that work well in the computer pods at UNM will be described here. The reasons for starting out with subversion is that we we want to get you used to working with this tool early on in the semester, because it's good practise. With that disclaimer, let's get started! Have your usb drive, or laptop ready!

- Insert USB drive (or open laptop)
- Create a Workspace directory on the drive (or open one that you have from before)
- Start Eclipse, switch workspace to your workspace
- Go to help menu, Install new software
- Add the subclipse site name to the *Work with:* box
  `http://subclipse.tigris.org/update_1.6.x`
- Now in the boxes below, check off
  - "Subclipse"
  - "Subversion client adapter"
  - "Subversion JavaHL Native Library Adapter"
- Click next
- Click next after Accepting License Agreement.
- Install software (And answer yes when it says it's unsigned something)

- When installation is done, you are ready to go... Restart Eclipse!
  The subversion plugins are now installed in your workspace on your USB drive or on your laptop.

Next thing to do is to get a project up and working, and download the materials you need for the assignment:

- Right-click in your "Package Explorer", and choose Import, then ”SVN“, then ”Checkout Projects from SVN“
- Now Choose "Create New Repository location"
  - Type in: `svn://jacobson.cc/cs251-spring10` in the URL field, and click Finish.
- Click on the + next to the URL and you should see "Java_Classes", click on that, and then "Next"
- Next choose "Check out as project configured using New Project Wizard", then Click "Finish".
- Choose Java Project, click "Next"
- Give it a name "Andree’s Class Examples"
- Click Finish, then "OK" to confirm Overwrite.
- The repository should now be ready to use...

If you want to you can put your own packages in this structure, but it’s probably better to create a new project for your own files. So... Right click in the package explorer, click "New", then "Java Project". Give it a name, like "My Java Programs", hit Next. In the window that comes up, select the "Projects" tab, click "Add", then select the project you just created with my examples in it. Click, Ok, then Finish. And we are ready to rock!

Create a new package in your workspace, call it program1, then create a class in there. Call it Visualizer. In main you can type:

Display d = new Display();

you’ll get errors saying that you’ll have to import Display, which you can now do since we have andree’s class examples project on the build path.

And now the program will run.

Turning in your assignment

Once you are done with your assignment, use WebCT to turn in the `Visualizer.java` file that you have created. Note that each source code file should contain a header comment with your name, class, and other related information. Failure to properly comment your files may lead to point deductions.