CS 201
Advanced Object-Oriented Programming
Lab 3 Preparation

To make better use of lab time, you should do the following before your lab meeting next week.

To get a better understanding of how translation and rotation work, download the TranslateRotateExample from the website and import it into Eclipse. When you run it, you should see:

Translate the rectangle by 200 horizontally and 200 vertically. You should now see something that looks like this.

Notice how the rectangle seems to be bigger now. It isn’t actually any bigger. It is just that the origin (0, 0) has been moved to (roughly) the middle of the window. The rectangle is being drawn centered at (0, 0). When (0,0) was at the top left corner of the window, as in the first image, only \( \frac{1}{4} \) of the rectangle was visible.
A full rotation is 2*PI. Enter .25 for the rotation. That is equivalent to 45°. Now you should see this.

Now, we have rotated the axes by .25 PI. We are still drawing the rectangle centered at (0, 0) and upright relative to the axes. This makes drawing easy for us. We don’t need to calculate where the corners of the rectangle should be to draw it rotated!

Now, look at the code that creates and draws the rectangle. Here is how the constructor is defined:

```java
private static final int POLYGON_HEIGHT = 200;
private static final int POLYGON_WIDTH = 100;
private Polygon shape;

public TranslateRotate() {
    shape = new Polygon();
    shape.addPoint(-POLYGON_WIDTH / 2, POLYGON_HEIGHT / 2);
    shape.addPoint(-POLYGON_WIDTH / 2, -POLYGON_HEIGHT / 2);
    shape.addPoint(POLYGON_WIDTH / 2, -POLYGON_HEIGHT / 2);
    shape.addPoint(POLYGON_WIDTH / 2, POLYGON_HEIGHT / 2);
}
```

Here, we have a shape instance variable, which we define by adding the points that correspond to the four corners of the rectangle. Substituting in the constant values in the addPoint calls, what are the coordinates for the corners of the rectangle?
Now, let’s look at the code that draws the rectangle:

```java
public void paintComponent(Graphics g) {
    Graphics2D g2d = (Graphics2D) g;

    // Set the translation and rotation values
    g2d.translate(x, y);
    g2d.rotate(radians);

    // Draw the axes
    g2d.setColor(Color.BLACK);
    g2d.drawLine(-getWidth(), 0, getWidth(), 0);
    g2d.drawLine(0, -getHeight(), 0, getHeight());
    g2d.drawString("(0, 0)", 5, 15);

    // Draw the shape
    g2d.setColor(Color.RED);
    g2d.drawPolygon(shape);

    // Undo the translation and rotation
    g2d.rotate(-radians);
    g2d.translate(-x, -y);
}
```

The rectangle itself is drawn by the one line that calls drawPolygon. This method simply draws a line between each consecutive pair of points that we added to our shape. Notice, in particular, that we do not calculate new screen coordinates to account for the desired translation and rotation. Instead, we translate and rotate the axes with the calls to the translate and rotate methods. Then, we draw the axes just to help us understand what is happening. Then we draw our shape. Finally, we re-orient the axes to their original position by undoing the translation and rotation. (This last step is not strictly needed for this program but will be needed when we have multiple shapes in Asteroids.)

As the final step for lab preparation, modify TranslateRotate so that it draws a triangle shape like what you will need in lab for your asteroid ship. Its center should be at (0, 0). See that it does what you want when you translate and rotate it. When done, you should get an effect like this: