Lab Goals

- Review the primitive shapes discussed during lecture
- Construct a basic scene in Processing
- Add some form of animation to that scene

Assignment Details

Thus far in lecture, we have discussed the primitive shapes supported by Processing (detailed in Table 1 below, as well as on pages 16-17 of the *Getting Started with Processing* book). In this lab, you will practice using these primitive shapes in order to construct a basic scene.

<table>
<thead>
<tr>
<th>Shape</th>
<th>Function Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>point</td>
<td>point(x, y)</td>
</tr>
<tr>
<td>line</td>
<td>line(x1, y1, x2, y2)</td>
</tr>
<tr>
<td>triangle</td>
<td>triangle(x1, y1, x2, y2, x3, y3)</td>
</tr>
<tr>
<td>quad</td>
<td>quad(x1, y1, x2, y2, x3, y3, x4, y4)</td>
</tr>
<tr>
<td>rect</td>
<td>rect(x, y, width, height) (x,y of top-left)</td>
</tr>
<tr>
<td>ellipse</td>
<td>ellipse(x, y, width, height) (x,y of center)</td>
</tr>
<tr>
<td>arc</td>
<td>arc(x, y, width, height, start, stop) (x,y of center)</td>
</tr>
</tbody>
</table>

Table 1: Primitive shapes supported by Processing

Creating a Basic Scene (25 points)

For this component of the lab, you will create a new Processing sketch. You should then use your creativity to come up with some scene that you wish to draw in a Processing window (of size at least 640 pixels across and 480 pixels high) using the primitive shapes listed in Table 1. For example, you could use *lines* and *rects* to draw the view of Alden Hall as seen from the front of the building. For further creativity, you could use *quads* to show an oblique view of Alden as seen from the campus center, or a floorplan of Alden as seen from above.
In your scene, you may wish to make use of more complicated shapes than the primitives allow. In this case, you can define a custom shape (an arrow, for example, in code taken from Example 3-18 on page 30 of *Getting Started with Processing*) as seen here:

```java
beginShape();
  vertex(180, 82);
  vertex(207, 36);
  vertex(214, 63);
  vertex(407, 11);
  vertex(412, 30);
  vertex(219, 82);
  vertex(226, 109);
endShape();
```

Note that a sequence of *vertex* objects are listed between *beginShape()* and *endShape()* commands. To close the gap remaining in the shape, you can add the **CLOSE** parameter to the *endShape()* function, as seen in Example 3-19:

```java
endShape(CLOSE);
```

For maximum points (remember, creativity is worth 20% of your lab grade), consider using the various color modes that we discussed during lecture, as well as changing things like smoothing (page 23-24), stroke weight (page 24), and stroke corner/cap attributes (page 25).

You may also want to consider sketching your scene out on paper before beginning to plug (x,y) coordinates into Processing.

**Adding Motion/Animation (25 points)**

Once you have created a static scene, it is time to add some animation. Recall that there are two functions provided by the Processing system, *setup()* and *draw()*. The *setup()* function runs exactly once at the beginning of the sketch, while the *draw()* function continues to run on every screen refresh. The *draw()* function is what allows us to create animations.

Continuing the Alden Hall example from the previous section, you could have clouds in motion above the static building, or students walking along the sidewalk/road in front of the building. In the floorplan example, you could similarly show a view of students walking around the building from an overhead view.

(Note that if you plan to have multiple objects moving at the same time, you will likely need separate variables for each of them. If you are interested in reading about classes and objects in Processing, that material is covered in Chapter 9 of *Getting Started with Processing*, and we will discuss it in class on Wednesday, September 2.)
Submission Details

For this assignment, please submit the following items which you have followed while completing this lab in paper form. Also, please upload these same items to your cs382f2015-<your username> repository. Your submission should include the following:

1. A description of the scene you attempted to create, both with and without the animations
2. A commented version of your source code
3. An Assignment Information Sheet

Additional Group Requirements

You are welcome to submit this assignment on your own, or in a group with one other person. In cases of groups, I would prefer one experienced programmer and one inexperienced programmer in the group, but this is not an absolute requirement. Additionally, each group need only submit one paper copy of their work, but each member of the group should push all items to their own repositories.

Finally, each group must create a short document (0.5-1 pages) detailing the work breakdown of the group members: who worked on which components, which lines of code belong to each group member, etc. Comments in the code will help here too.