Computer Graphics – Hands-on

Two-Dimensional OpenGL Transformations

Objectives

- To get hands-on experience manipulating the OpenGL current transformation (MODELVIEW) matrix to achieve desired effects
- To gain experience using the OpenGL matrix stack
- To see how new models can be created out of multiple, transformed instances of other models

Background

- OpenGL functions glTranslate*, glScale*, glRotate*, glPushMatrix, and glPopMatrix.
- A basic understanding of the current transformation (MODELVIEW) matrix and the matrix stack.

Software Tool

- Applet posted online

  http://www.csc.villanova.edu/~mdamian/graphics/applets/OpenGLTransformations.html

  Courtesy of Dr. Robin Flatland, Siena College.

Activities

- Start the OpenGLTransformations applet (you may need to view it in full screen mode to see it all). On the right side of the window are a series of text boxes in which you can enter a code fragment consisting of calls to OpenGL functions and functions that draw various objects. To see which functions are available, click the down arrow next to a text box. The buttons below the text boxes allow you to step through the execution of the code. At the bottom of the window the OpenGL stack of transformation matrices is displayed. The top matrix on this stack (pointed to by the green arrow) is the current transformation matrix that is applied to objects before they are drawn. If the arrow points to nothing, as it does when the tool first starts, it means the current transformation matrix is the identity matrix.

Part 1: Warm-up Activities

Activity 0: To master the controls of this tool, begin by typing (or selecting) the line of code below into the first text box.

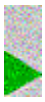
```
drawSmiley();
```

Now press the Reset button and then the Step button to execute the line and make the smiley face appear on the display. Notice that the green arrow of the matrix stack is pointing to nothing when the function drawSmiley executes, thus no transformations are applied to the face.

Activity 1: Add three more lines of code in the next three text boxes so that the fragment looks like that below:

```
drawSmiley();
glTranslated( 2, 2, 0 );
glScaled( 3, 0.5, 1 );
drawSmiley();
```

Press the Reset button and then press the Step button four times to execute the code fragment, observing what happens after each line executes. Notice that when the glTranslated function call executes, it modifies the current transformation shown in the matrix stack at the bottom of the window. After the glScaled function executes, the current transformation matrix is the product of a translation and a scale matrix. On the display this is shown as:

 T(2,2,0) S(3,0.5,1)

Notice that when the second face is drawn, each point making up the face was multiplied by the current transformation matrix before being displayed. Thus, the face was first scaled by a factor of 3 on the x axis and by a factor of 0.5 on the y axis. Then it was translated by 2 on both x and y axes.

The order in which these transformations were applied may seem backwards at first. The reason for the confusion is that in the code fragment you called glTranslated before glScaled, and yet the face was scaled first and then translated! To understand this apparent anomaly, you must remember that what the glTranslated function does (and similarly the glScaled function): it multiplies the current transformation matrix ***on the right*** by a translation (similarly scale) matrix. When the object is drawn, each point is multiplied by the current transformation matrix, as shown below.

Transformed Point = T(2,2,0) * S(3,0.5,1) * Point

Therefore the point's coordinates are first scaled and then translated before being drawn on the display.

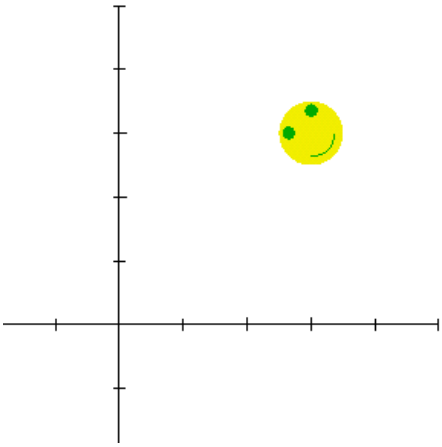
Activity 2: Now press the “Clear” button, and enter the code below in the first six text boxes:

```
glTranslated( 2, 2, 0 );
glPushMatrix();
glScaled( 3, 0.5, 1 );
drawSmiley();
glPopMatrix();
drawSmiley();
```

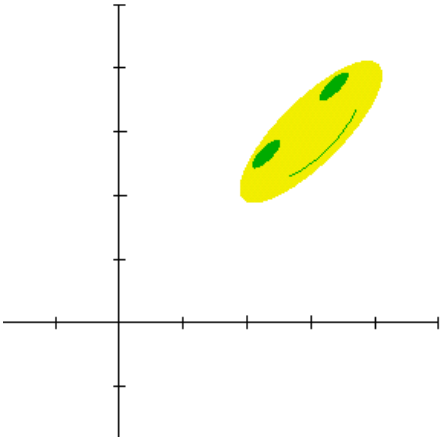
Press the Reset button and then step through each of line of the new code fragment, observing what happens after each line executes. Notice that when the glPushMatrix() is executed, a copy of the current transformation matrix is pushed onto the stack and this copy (now pointed to by the green arrow) becomes the new current transformation matrix. The glScaled function called next effects only the top matrix on the stack, making it equal to T(2,2,0)S(3,0.5,1). When the first face is drawn, the transformation T(2,2,0)S(3,0.5,1) is applied to it making it appear long and thin in the top right quadrant. Before drawing the second face, the matrix stack is popped, thus restoring the current transformation matrix to what it was just before the glPushMatrix was encountered. When the second face is drawn, the transformation applied to it is just T(2,2,0).

Part 2: Manipulating the current transformation matrix

Activity 3: To gain experience manipulating the current transformation matrix to achieve a desired effect, write a code fragment that makes the smiley face appear as shown below. Next to the figure, write down the code fragment you used to achieve this effect.



Activity 4: Write a code fragment that makes the smiley face appear as shown below. Next to the figure, write down the code fragment you used to achieve this effect.

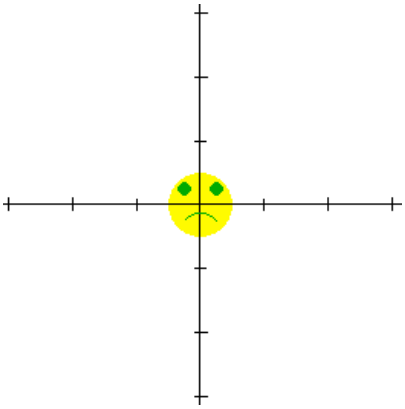


Part 3: Building a model out of parts

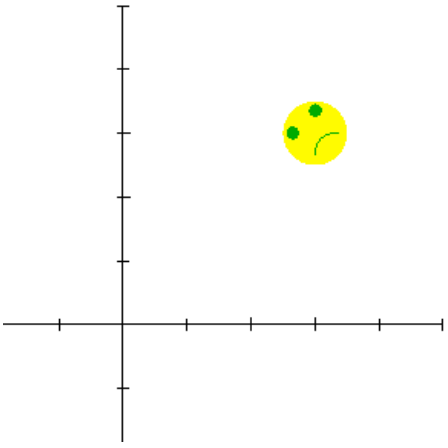
Often times it is useful to create a model out of other, simpler models. For example, the smiley face is created out of three parts – a circle, the eyes, and a smile. Using the software tool, you can draw these individual parts using the function calls drawCircle(), drawEyes(), and drawSmile().Try them now by writing a code fragment that draws the smiley face using these three function calls, instead of one call to drawSmiley().

Different models can be created out of the same set of parts by making the parts appear differently using the current transformation matrix. In this section of the lab you have the opportunity to create some new models out of parts.

Activity 5: Write a code fragment that draws a frowning face, as shown below. Next to the figure, write down the code fragment you used to achieve this effect.



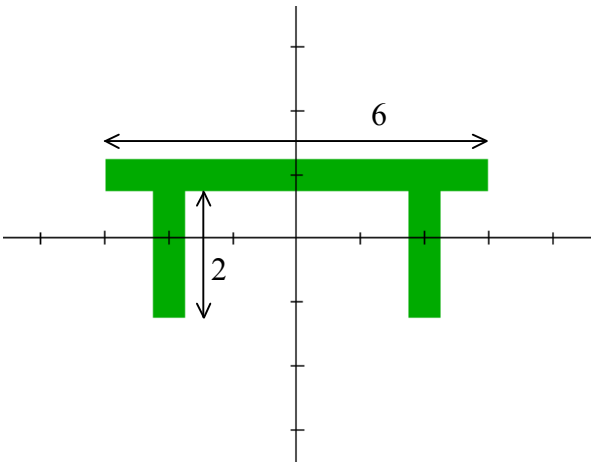
Activity 6: Write a code fragment that displays the frowning face in the top right quadrant, as shown below. Next to the figure, write down the code fragment you used to achieve this effect.



Activity 7: Another model you can draw using this software tool is a square centered at the origin having side lengths of 1. The function that draws this model is called drawSquare(). Test this function now by typing it into the first textbox and executing it.

Now use this function to create a model of table as shown below. Do it by calling drawSquare() three times, once for the table top and once for each leg. You will need to make sure the current transformation matrix is correctly set up before each call to drawSquare().Your code fragment should not include any calls to glLoadIdentity(). Instead, use glPushMatrix() and glPopMatrix() to save and restore the current transformation as necessary.

Write down the code fragment you used to achieve this (on a separate piece of paper, if necessary).



Activity 8: Be creative and write a code fragment that draws something of your choice. For example, you might try to create a person or a snowman. The only requirements are that it consist of three or more parts, that you use glTranslate*, glScale*, and glRotate* at least once each, and that you do not use glLoadIdentity().

Use Ctrl-Alt-Print Screen to get a picture of your creation and the code fragment that creates it. Print it out and attach a copy to the back of this handout.