The Modelview Stack
Lecture 8
Sections 4.10.6, 10.1 - 10.4

Robb T. Koether

Hampden-Sydney College

Mon, Sep 14, 2009
Outline

1. The Modelview Stack
2. Manipulating the Stack
3. A Hierarchical Model
4. Other Transformations
5. Assignment
The Modelview Stack

- OpenGL maintains a stack of matrices.
- The matrix on top of the stack is the current matrix.
- The function `glPushMatrix()` will push a copy of the current matrix onto the stack.
- The function `glPopMatrix()` will pop the top matrix off the stack.
Pushing and popping are used to “remember” previous transformations.

The basic pattern is

- Push the current matrix onto the stack (to remember it).
- Perform a series of geometric transformations and draw an object.
- Pop the current matrix off the stack, thereby restoring the former “current matrix.”
Manipulating the Stack

Example (Manipulating the Stack)

```c
glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
drawCylinder();  // Along pos. z-axis

glPushMatrix();
glRotatef(90.0, 0.0, 1.0, 0.0);
drawCylinder();

glTranslatef(0.0, 0.0, 2.0);
drawCylinder();

glPopMatrix();
glRotatef(90.0, 1.0, 0.0, 0.0);
drawCylinder();
```
## Manipulating the Stack

<table>
<thead>
<tr>
<th>Modelview Stack</th>
<th>Action</th>
<th>Drawing</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>Load identity</td>
<td></td>
</tr>
<tr>
<td>I</td>
<td>Draw cylinder</td>
<td>Along $z$-axis, 0 to 1</td>
</tr>
<tr>
<td>I, I</td>
<td>Push matrix</td>
<td></td>
</tr>
<tr>
<td>I, $R_y$</td>
<td>Rotate</td>
<td></td>
</tr>
<tr>
<td>I, $R_y$</td>
<td>Draw cylinder</td>
<td>Along $x$-axis, 0 to 1</td>
</tr>
<tr>
<td>I, $R_yT$</td>
<td>Translate</td>
<td></td>
</tr>
<tr>
<td>I, $R_yT$</td>
<td>Draw cylinder</td>
<td>Along $x$-axis, 2 to 3</td>
</tr>
<tr>
<td>I</td>
<td>Pop matrix</td>
<td></td>
</tr>
<tr>
<td>$R_x$</td>
<td>Rotate</td>
<td></td>
</tr>
<tr>
<td>$R_x$</td>
<td>Draw cylinder</td>
<td>Along $y$-axis, 0 to $-1$</td>
</tr>
</tbody>
</table>
Manipulating the Stack

- The identity matrix is placed on the stack.
- The first cylinder is drawn along the positive $z$-axis.
- The current matrix is pushed onto the stack.
- The next cylinder is drawn along positive $z$-axis and then rotated into the positive $x$-axis.
Manipulating the Stack

- The third cylinder is drawn along the positive $z$-axis, shifted forward along the $z$-axis, then rotated into the $x$-axis.
- The current matrix is popped off the stack,
- The fourth cylinder is drawn along the positive $z$-axis and the rotated into the negative $y$-axis.
Using the Stack to Draw a Person

Our person
Using the Stack to Draw a Person

Our person
We see that the person is constructed of rectangles and ovals (including circles).

Therefore, we will write two basic functions:
- `drawCircle()` - draws unit circle at the origin.
- `drawSquare()` - draws unit square at the origin in the first quadrant.
The hand is a circle, 6 inches in diameter.
The head is a circle of diameter 1 foot.
Drawing the Head

Example (Drawing the Head)

```c
void drawHead()
{
    glPushMatrix();
    glScalef(0.5, 0.5, 1.0);
    drawCircle();
    glPopMatrix();
    return;
}
```
The torso is 2.5 feet high and 1.25 feet wide.
**Draw the Torso**

```c
void drawTorso()
{
    glPushMatrix();
    glScalef(1.25, 2.5, 1.0);
    drawSquare();
    glPopMatrix();
    return;
}
```
Example (Drawing the Hand)

```c
void drawHand()
{
    glPushMatrix();
    glScalef(0.25, 0.25, 0.0);
    drawCircle();
    glPopMatrix();
    return;
}
```
The arm is a rectangle 1 foot, 9 inches long and 3 inches wide.
Drawing the Arm

Example (Drawing the Arm)

```c
void drawArm()
{
    glPushMatrix();
    glPushMatrix();
    glScalef(0.25, 1.75, 1.0);
    drawSquare();
    glPopMatrix();
    glTranslatef(0.125, 1.75, 0.0);
    drawHand();
    glPopMatrix();
    return;
}
```
The foot is an oval, one foot long and 6 inches high.
Example (Drawing the Foot)

```c
void drawFoot()
{
    glPushMatrix();
    glScalef(0.5, 0.25, 0.0);
    drawCircle();
    glPopMatrix();
    return;
}
```
The leg is a rectangle 2 feet long and 6 inches wide.
Example (Drawing the Leg)

```c
void drawLeg()
{
    glPushMatrix();
        glPushMatrix();
            glScalef(0.5, 2.0, 1.0);
            drawSquare();
        glPopMatrix();
    glTranslatef(0.5, 0.0, 0.0);
    drawFoot();
    glPopMatrix();
    return;
}
```
Draw the Person
Draw the Person

Example (Drawing the Person)

```c
void drawPerson()
{
    glPushMatrix();
    glPushMatrix();
        glTranslatef(0.0, 2.25, 0.0);
        // Draw the torso
        glPushMatrix();
        glTranslatef(-0.625, 0.0, 0.0);
        drawTorso();
        glPopMatrix();
        // Draw the head
        glPushMatrix();
        glTranslatef(0.0, 3.0, 0.0);
        drawHead();
        glPopMatrix();
    glPopMatrix();
}
```
Example (Drawing the Person)

```c
: : 
// Draw right arm
glPushMatrix();
    glTranslatef(0.625, 2.5, 0.0);
    glRotatef(-160.0, 0.0, 0.0, 1.0);
    drawArm();
glPopMatrix();
// Draw left arm
glPushMatrix();
    glTranslatef(-0.625, 2.5, 0.0);
    glRotatef(160.0, 0.0, 0.0, 1.0);
    glScale(-1.0, 1.0, 1.0);
    drawArm();
glPopMatrix();
glPopMatrix();
: : 
```
Draw the Person

Example (Drawing the Person)

```c
:

// Draw right leg
glPushMatrix();
  glTranslatef(0.125, 0.25, 0.0);
  drawLeg();
  glPopMatrix();

// Draw left leg
glPushMatrix();
  glTranslatef(-0.125, 0.25, 0.0);
  glScalef(-1.0, 1.0, 1.0);
  drawLeg();
  glPopMatrix();

return;
```
Example (Draw a Person)

- The code.
- The executable.
Example (Draw a Crowd of People)

- The code.
- The executable.
Example (Draw a Orrery)

- The code.
- The executable.
Specifying a Transformation Matrix

- OpenGL allows the programmer to specify his own transformation matrix and multiply it by the current transformation.
Specifying a Transformation Matrix

- Create a $4 \times 4$ matrix as a 16-member one-dimensional array of \texttt{floats} or \texttt{doubles}, in column-major order.
- Use \texttt{glLoadMatrix*()} to make it the current transformation.
- Use \texttt{glMultMatrix*()} to multiply the current transformation by it.
Example: Creating a Shear

Suppose we want to create the shear transformation in 2D that transforms

- $(0, 0)$ to $(0, 0)$,
- $(1, 0)$ to $(1, 0)$,
- $(0, 1)$ to $(0.2, 1)$,
- $(1, 1)$ to $(1.2, 1)$.

In other words, it transforms $(x, y)$ to $(x + 0.2y, y)$.
Drawing the Foot

A shear transformation
Drawing the Foot

A shear transformation
Creating a Shear

The transformation matrix is

$$\begin{pmatrix}
1 & 0.2 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1
\end{pmatrix}.$$
Creating a Shear

- The following code will perform this transformation.

```c
// Store in column-major order
float mat[16] = {1.0, 0.0, 0.0, 0.0,
                 0.2, 1.0, 0.0, 0.0,
                 0.0, 0.0, 1.0, 0.0,
                 0.0, 0.0, 0.0, 1.0};

glMultMatrix(mat);
```
Example (Shearing the People)

- The code.
- The executable.
Homework

- Read Section 4.10.6 – pushing and popping matrices.
- Read Sections 10.1 - 10.4 – examples of hierarchical models.