Purpose of this Lecture

- OpenGL Transformations & Hierarchical Modeling
- A few demos.
Coming Assignment

- Hierarchical Modeling
- Last year’s assignment: to model a mobile.
- Inherently Hierarchical.
OpenGL Transformations

- Working with two important matrices: The Projection Matrix and The Model View Matrix
  - gl.glMatrixMode(GL.GL_MODELVIEW) or gl.glMatrixMode(GL.GL_PROJECTION)
- Projection Matrix: maps points in the world to points on the camera.
- Model View Matrix: Can be thought of as setting a 3D cursor in the world.
A Silly Example

- So let’s think conceptually through an example.
- Suppose P the point (1, 0, 0)’. Let R be a rotation of 90 degrees about the Z axis and let T be a translation of 1 unit along the X axis.
- What is the effect of $P' = T \circ R \circ P$?
A Silly Example

- $T(RP) = T(0, 1, 0)' = (1, 1, 0)$
- What if we flip the order? $R TP = R(TP) =$ ?
- $R TP = R(TP) = R(2, 0, 0) = (0, 2, 0)$
Model View Matrix

- OpenGL handles these sorts of transformations by keeping track of the model view matrix $M_{model}$.
- Any point OpenGL processes get’s multiplied with $M_{model}$.
- Instead of modifying points directly we modify the Model View matrix.
Modifying the Model View

- `glTranslate(X, Y, Z)`: Translates the Model View Matrix by \((X, Y, Z)\)’
- `glRotate(T, X, Y, Z)`: Rotates the Model View Matrix by \(T\) degrees around the vector given by \((X, Y, Z)\)’
- `glScale(X, Y, Z)`: Scales the Model View by \((X, Y, Z)\).
So we initially set the model view to the identity matrix by calling `glLoadIdentity();`

We can then use the `glTranslate`, `glRotate` and `glScale` functions to modify the Model View.

The Model View then gets applied to anything we subsequently draw.
Matrix Stack

- One other helpful thing: The matrix stack.
- OpenGL keeps a stack of the Model View matrix which you can interact with via `glPushMatrix()` and `glPopMatrix();`
- `glPushMatrix()` pushes a copy of the current matrix onto the stack.
- `glPopMatrix()` copies the top of the stack into the current matrix and then deletes it from the stack.
What is the Point?

- Say you have a teapot (we like teapots in graphics, never knew why...) that is rotating around some point in space and it has two other teapots orbiting it...

- You could glLoadIdentity() for every teapot, and then do all the rotations/translations/scaling each time OR you could do the common transformations once, and push the matrix onto the stack and then make the local modifications.

- Okay, maybe code would help...
Comparison

//The Blue One:
  glColor3f(0.f, 0.f, 1.f);
  glLoadIdentity();
  glRotatef(rotX, 0, 0, 1);
  glTranslatef(0.5f, 0.f, 0.f);
  glutSolidTeapot(0.15f);

//The Red One
  glPushMatrix();
  glColor3f(1.f, 0.f, 0.f);
  glRotatef(-6*rotX, 0, 0, 1);
  glTranslatef(0.65f, 0.f, 0.f);
  glRotatef(rotX, 0, 0, 1);
  glScalef(0.4f, 0.4f, 0.4f);
  glutSolidTeapot(0.1f);
  glPopMatrix();

//The Green One
  glPushMatrix();
  glColor3f(0.f, 1.f, 0.f);
  glRotatef(4*rotX, 0, 0, 1);
  glTranslatef(0.42f, 0.f, 0.f);
  glRotatef(10*rotX, 0, 0, 1);
  glScalef(0.5f, 0.5f, 0.5f);
  glutSolidTeapot(0.11f);
  glPopMatrix();
Example: More Rotating Teapots!

```c
-(void) drawRect: (NSRect) bounds {
    glClearColor(0, 0, 0, 0);
    glClear(GL_COLOR_BUFFER_BIT);
    //The Gold One:
    glColor3f(1.0f, 0.85f, 0.35f);
    glLoadIdentity();
    glRotatef(rotX, 0, 0, 1);
    glutSolidTeapot(0.1f);
    glEnd();

    //The Red One:
    glColor3f(1.0f, 0.f, 0.f);
    glLoadIdentity();
    glTranslatef(0.5f, 0.f, 0.f);
    glRotatef(rotX, 0, 0, 1);
    glutSolidTeapot(0.1f);

    //The Blue One:
    glColor3f(0.f, 0.f, 1.f);
    glLoadIdentity();
    glRotatef(rotX, 0, 0, 1);
    glutSolidTeapot(0.1f);

    //The Light Blue One
    glPushMatrix();
    glColor3f(0.f, 1.f, 1.f);
    glRotatef(-6*rotX, 0, 0, 1);
    glTranslatef(0.3f, 0.f, 0.f);
    glRotatef(rotX, 0, 0, 1);
    glScalef(0.3f, 0.3f, 0.3f);
    glutSolidTeapot(0.1f);
    glPopMatrix();

    //Update some variables
    rotX += 1.0f;
}
```
Managing Hierarchical Models

- Think object oriented.
- Can have a class that stores a list of children objects.
- The Draw function for each class will glPushMatrix(), then do its drawing, then call Draw() on all the children, then glPopMatrix().
- Nice method for dealing with hierarchical modeling.
And now for something... 
completely different.

www.espressosoft.com
john bowers::