Computer Graphics (CS 543) Lecture 6 (Part 1): Implementing Transformations

Prof Emmanuel Agu

Computer Science Dept.

Worcester Polytechnic Institute (WPI)

Arbitrary Matrices

- Can multiply by matrices from transformation commands (Translate, Rotate, Scale) into CTM
- Can also load arbitrary 4x4 matrices into CTM

Load into
CTM Matrix $\Leftarrow$

$$\begin{pmatrix} 1 & 0 & 15 & 3 \\ 0 & 2 & 0 & 12 \\ 34 & 0 & 3 & 12 \\ 0 & 24 & 0 & 1 \end{pmatrix}$$

Matrix Stacks

- Sometimes want to save transformation matrices for use later
- E.g: Traversing hierarchical data structures (Ch. 8)
- Pre 3.1 OpenGL maintained matrix stacks
- Right now just implement 1-level CTM
- Matrix stack later for hierarchical transforms

Reading Back State

- Can also access OpenGL variables (and other parts of the state) by *query* functions

```
glGetIntegerv
glGetFloatv
glGetBooleanv
glGetDoublev
glIsEnabled
```

- Example: to find out maximum number of texture units

```
glGetIntegerv(GL_MAX_TEXTURE_UNITS, &MaxTextureUnits);
```

Using Transformations

- **Example:** use idle function to rotate a cube and mouse function to change direction of rotation
- Start with program that draws cube as before
  - Centered at origin
  - Sides aligned with axes

main.c

```
void main(int argc, char **argv)
{
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB |
       GLUT_DEPTH);
    glutInitWindowSize(500, 500);
    glutCreateWindow("colorcube");
    glutReshapeFunc(myReshape);
    glutDisplayFunc(display);
    glutIdleFunc(spinCube);
    glutMouseFunc(mouse);
    glEnable(GL_DEPTH_TEST);
    glutMainLoop();
}
```

$\leftarrow$ Calls spinCube continuously
Whenever OpenGL program is idle

Idle and Mouse callbacks

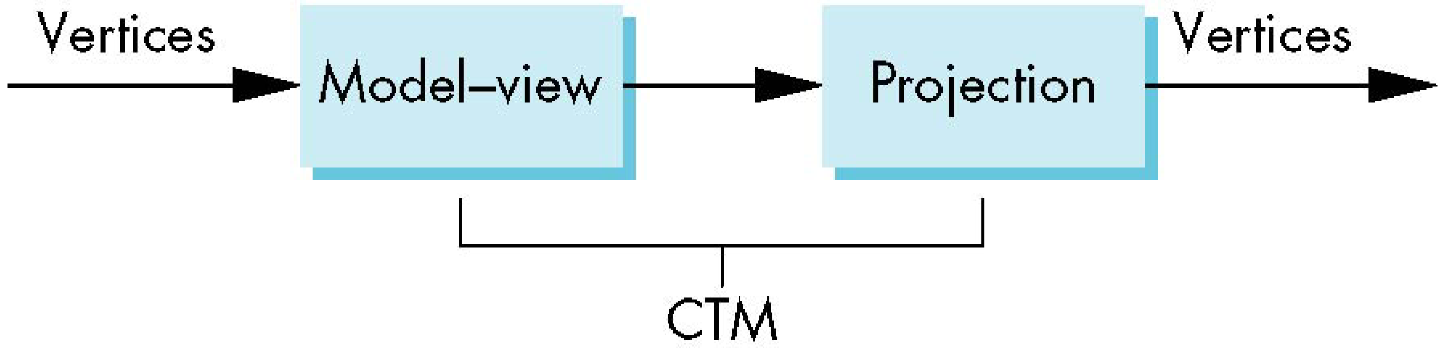
```
void spinCube()
{
   theta[axis] += 2.0;
   if( theta[axis] > 360.0 ) theta[axis] -= 360.0;
   glutPostRedisplay();
}

  void mouse(int button, int state, int x, int y)
  {
     if(button==GLUT_LEFT_BUTTON && state == GLUT_DOWN)
             axis = 0;
     if(button==GLUT_MIDDLE_BUTTON && state == GLUT_DOWN)
             axis = 1;
     if(button==GLUT_RIGHT_BUTTON && state == GLUT_DOWN)
             axis = 2;
  }
```

Display callback

```
void display()
{
   glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
   ctm = RotateX(theta[0])*RotateY(theta[1])
                                    *RotateZ(theta[2]);
    glUniformMatrix4fv(matrix_loc,1,GL_TRUE,ctm);
    glDrawArrays(GL_TRIANGLES, 0, N);
   glutSwapBuffers();
}
```

- Alternatively, we can send rotation angle and axis to vertex shader,
- Let shader form CTM then do rotation
- Inefficient to apply vertex transform data in application (CPU) and send data to GPU to render

Using the Model-view Matrix



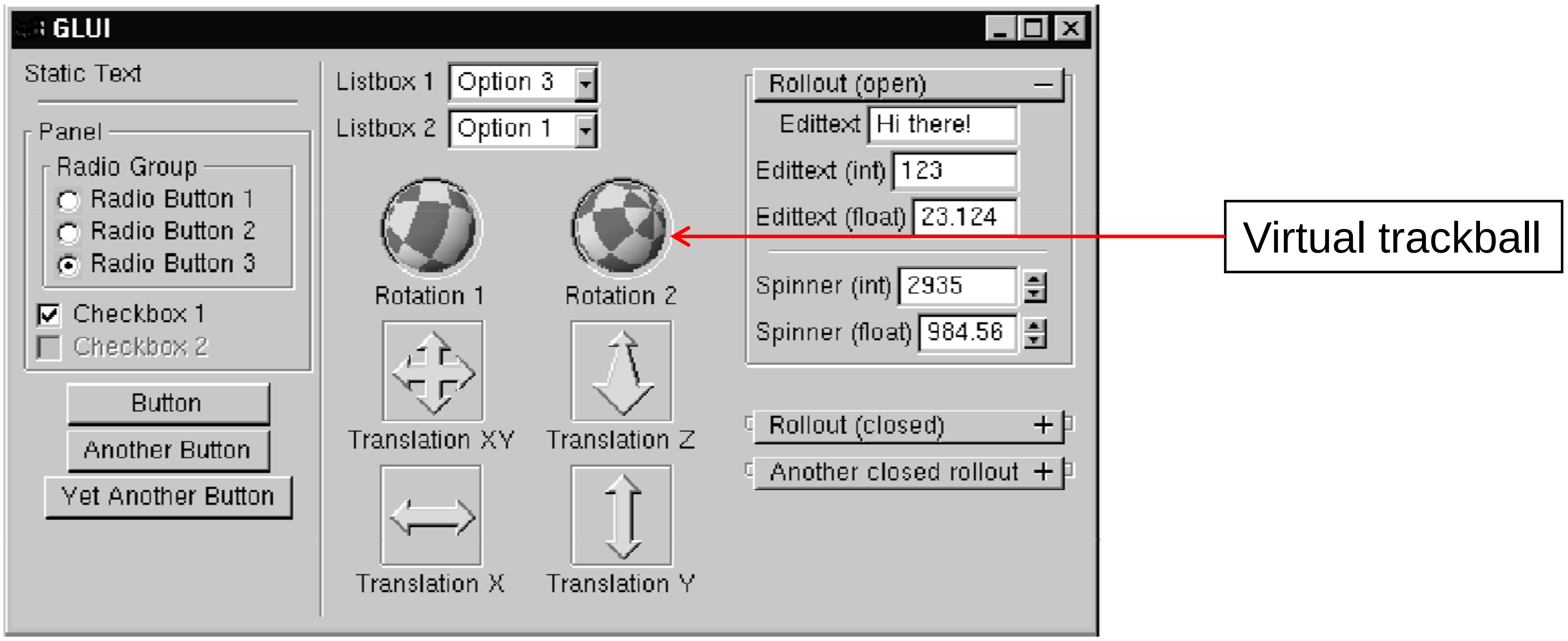
- In OpenGL the model-view matrix used to
  - Transform 3D models
  - Position camera (using LookAt function) (next)
- The projection matrix used to define view volume and select a camera lens (later)
- Although these matrices no longer part of OpenGL, good to create them in our applications (as CTM)

3D? Interfaces

- Major interactive graphics problem: how to use 2D devices (e.g. mouse) to control 3D objects
- Some alternatives
  - Virtual trackball
  - 3D input devices such as the spaceball
  - Use areas of the screen
    - Distance from center controls angle, position, scale depending on mouse button depressed

GLUI

- User Interface Library by Paul Rademacher
- Provides sophisticated controls and menus
- Not used in this class/optional



References

- Angel and Shreiner, Chapter 3
- Hill and Kelley, appendix 4