The University of New Mexico

Introduction to Computer Graphics with WebGL

Ed Angel

Professor Emeritus of Computer Science

Founding Director, Arts, Research, Technology and Science Laboratory

University of New Mexico

Applying Transformations

Ed Angel
Professor Emeritus of Computer Science
University of New Mexico

Using Transformations

- Example: Begin with a cube rotating
- Use mouse or button listener to change direction of rotation
- Start with a program that draws a cube in a standard way
  - Centered at origin
  - Sides aligned with axes
  - Will discuss modeling in next lecture

Where do we apply transformation?

- Same issue as with rotating square
  - in application to vertices
  - in vertex shader: send MV matrix
  - in vertex shader: send angles
- Choice between second and third unclear
- Do we do trigonometry once in CPU or for every vertex in shader
  - GPUs have trig functions hardwired in silicon

Rotation Event Listeners

```
document.getElementById( "xButton" ).onclick =
  function () {        axis = xAxis;    };
document.getElementById( "yButton" ).onclick =
  function () {        axis = yAxis;    };
document.getElementById( "zButton" ).onclick =
  function () {        axis = zAxis;    };


function render(){
  gl.clear( gl.COLOR_BUFFER_BIT | gl.DEPTH_BUFFER_BIT );
  theta[axis] += 2.0;
  gl.uniform3fv(thetaLoc, theta);
  gl.drawArrays( gl.TRIANGLES, 0, NumVertices );
  requestAnimFrame( render );
}
```

Rotation Shader

```
Attribute vec4 vPosition;
Attribute vec4 vColor;
varying   vec4 fColor;
uniform   vec3 theta;

void main() {
  vec3 angles = radians( theta );
  vec3 c = cos( angles );
  vec3 s = sin( angles );
  // Remember: these matrices are column-major
  mat4 rx = mat4( 1.0,  0.0,  0.0, 0.0,
                  0.0,  c.x,  s.x, 0.0,
                  0.0, -s.x,  c.x, 0.0,
                  0.0,  0.0,  0.0, 1.0 );
```

Rotation Shader (cont)

```
mat4 ry = mat4( c.y, 0.0, -s.y, 0.0,
                0.0, 1.0,  0.0, 0.0,
                s.y, 0.0,  c.y, 0.0,
                0.0, 0.0,  0.0, 1.0 );

 mat4 rz = mat4( c.z, -s.z, 0.0, 0.0,
                 s.z,  c.z, 0.0, 0.0,
                 0.0,  0.0, 1.0, 0.0,
                 0.0,  0.0, 0.0, 1.0 );

 fColor = vColor;
 gl_Position = rz * ry * rx * vPosition;
}
```

Smooth Rotation

- From a practical standpoint, we are often want to use transformations to move and reorient an object smoothly
    - Problem: find a sequence of model-view matrices $\mathbf{M}_0,\mathbf{M}_1,\ldots..,\mathbf{M}_n$ so that when they are applied successively to one or more objects we see a smooth transition
- For orientating an object, we can use the fact that every rotation corresponds to part of a great circle on a sphere
    - Find the axis of rotation and angle
    - Virtual trackball (see text)

Incremental Rotation

- Consider the two approaches
  - For a sequence of rotation matrices $\mathbf{R}_0,\mathbf{R}_1, \ldots..,\mathbf{R}_n$, find the Euler angles for each and use $\mathbf{R}_i = \mathbf{R}_{iz}\,\mathbf{R}_{iy}\,\mathbf{R}_{ix}$
    - Not very efficient
  - Use the final positions to determine the axis and angle of rotation, then increment only the angle
- Quaternions can be more efficient than either

Quaternions

- Extension of imaginary numbers from two to three dimensions
- Requires one real and three imaginary components **i**, **j**, **k**

$$q = q_0 + q_1\mathbf{i} + q_2\mathbf{j} + q_3\mathbf{k}$$

- Quaternions can express rotations on sphere smoothly and efficiently. Process:
  - Model-view matrix → quaternion
  - Carry out operations with quaternions
  - Quaternion → Model-view matrix

Interfaces

- One of the major problems in interactive computer graphics is how to use a two-dimensional device such as a mouse to interface with three dimensional objects
- Example: how to form an instance matrix?
- 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