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

The Rotating Cube

- Ed Angel
- Professor Emeritus of Computer Science
- University of New Mexico

Objectives

- Put everything together to display rotating cube
- Two methods of display
  - by arrays
  - by elements

Modeling a Cube

Define global array for vertices

```
var vertices = [
        vec3( -0.5, -0.5,  0.5 ),
        vec3( -0.5,  0.5,  0.5 ),
        vec3(  0.5,  0.5,  0.5 ),
        vec3(  0.5, -0.5,  0.5 ),
        vec3( -0.5, -0.5, -0.5 ),
        vec3( -0.5,  0.5, -0.5 ),
        vec3(  0.5,  0.5, -0.5 ),
        vec3(  0.5, -0.5, -0.5 )
    ];
```

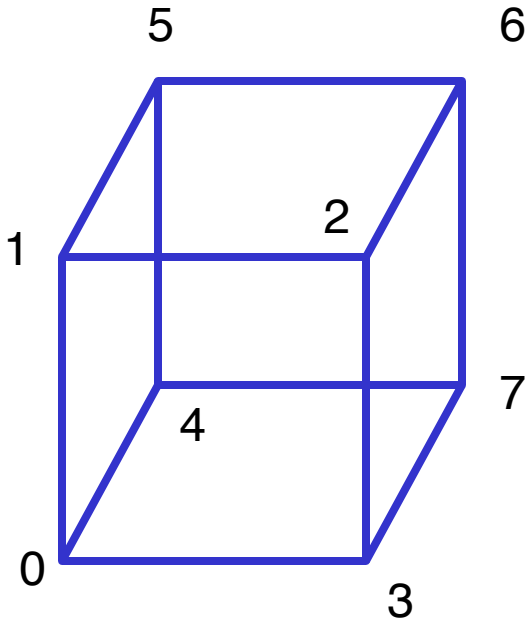
Colors

Define global array for colors

```
var vertexColors = [
        [ 0.0, 0.0, 0.0, 1.0 ],  // black
        [ 1.0, 0.0, 0.0, 1.0 ],  // red
        [ 1.0, 1.0, 0.0, 1.0 ],  // yellow
        [ 0.0, 1.0, 0.0, 1.0 ],  // green
        [ 0.0, 0.0, 1.0, 1.0 ],  // blue
        [ 1.0, 0.0, 1.0, 1.0 ],  // magenta
        [ 0.0, 1.0, 1.0, 1.0 ],  // cyan
        [ 1.0, 1.0, 1.0, 1.0 ]   // white
    ];
```

Draw cube from faces

```
function colorCube( )
{
    quad(1,0,3,2);
    quad(2,3,7,6);
    quad(3,0,4,7);
    quad(6,5,1,2);
    quad(4,5,6,7);
    quad(5,4,0,1);
}
```



Note that vertices are ordered so that
we obtain correct outward facing normals
Each quad generates two triangles

Initialization

```
var canvas, gl;
var numVertices  = 36;
var points = [];
var colors = [];

window.onload = function init(){
    canvas = document.getElementById( "gl-canvas" );
    gl = WebGLUtils.setupWebGL( canvas );

    colorCube();

   gl.viewport( 0, 0, canvas.width, canvas.height );
   gl.clearColor( 1.0, 1.0, 1.0, 1.0 );
   gl.enable(gl.DEPTH_TEST);
```

Needed for Hidden Surface Removal

The quad Function

Put position and color data for two triangles from a list of indices into the arrays **`vertices`**/ **`vertexColors`**

```
var quad(a, b, c, d)
{
   var indices = [ a, b, c, a, c, d ];
   for ( var i = 0; i < indices.length; ++i ) {

      points.push( vertices[indices[i]]);
      colors.push( vertexColors[indices[i]] );

      // for solid colored faces use
      //colors.push(vertexColors[a]);
 }
}
```

Initialization (colors)

the enlarged array with 36 entries, produced by quad()

```
// color array atrribute buffer

    var cBuffer = gl.createBuffer();
    gl.bindBuffer( gl.ARRAY_BUFFER, cBuffer );
    gl.bufferData( gl.ARRAY_BUFFER, flatten(colors), gl.STATIC_DRAW );

    var vColor = gl.getAttribLocation( program, "vColor" );
    gl.vertexAttribPointer( vColor, 4, gl.FLOAT, false, 0, 0 );
    gl.enableVertexAttribArray( vColor );
```

Initialization (vertices)

the enlarged array with 36 entries, produced by quad()

```
// vertex array attribute buffer

    var vBuffer = gl.createBuffer();
    gl.bindBuffer( gl.ARRAY_BUFFER, vBuffer );
    gl.bufferData( gl.ARRAY_BUFFER, flatten(points), gl.STATIC_DRAW );

    var vPosition = gl.getAttribLocation( program, "vPosition" );
    gl.vertexAttribPointer( vPosition, 3, gl.FLOAT, false, 0, 0 );
    gl.enableVertexAttribArray( vPosition );
```

Render Function

Needed for Hidden Surface Removal

```
function render(){
    gl.clear( gl.COLOR_BUFFER_BIT |gl.DEPTH_BUFFER_BIT);
    gl.drawArrays( gl.TRIANGLES, 0, numVertices );
    requestAnimFrame( render );
}
```

Mapping indices to faces

```
var indices = [
    1,0,3,
    3,2,1,
    2,3,7,
    7,6,2,
    3,0,4,
    4,7,3,
    6,5,1,
    1,2,6,
    4,5,6,
    6,7,4,
    5,4,0,
    0,1,5
];
```

All 12 triangular faces that will make up the cube are stored as indices to a vertex list.

The indices need to be sent to the GPU and the drawing call changed to work with indices (one indirection) instead of vertex arrays

Rendering by Elements

- Send indices to GPU

```
var iBuffer = gl.createBuffer();
gl.bindBuffer(gl.ELEMENT_ARRAY_BUFFER, iBuffer);
gl.bufferData(gl.ELEMENT_ARRAY_BUFFER,
       new Uint8Array(indices), gl.STATIC_DRAW);
```

- Render by elements

```
gl.drawElements( gl.TRIANGLES, numVertices, gl.UNSIGNED_BYTE, 0 );
```

- Even more efficient if we use triangle strips or triangle fans

Initialization (colors)

the original array with the 8 colours

```
// color array atrribute buffer

    var cBuffer = gl.createBuffer();
    gl.bindBuffer( gl.ARRAY_BUFFER, cBuffer );
    gl.bufferData( gl.ARRAY_BUFFER, flatten(vertexColors), gl.STATIC_DRAW );

    var vColor = gl.getAttribLocation( program, "vColor" );
    gl.vertexAttribPointer( vColor, 4, gl.FLOAT, false, 0, 0 );
    gl.enableVertexAttribArray( vColor );
```

Initialization (vertices)

the original array with the 8 vertices

```
// vertex array attribute buffer

    var vBuffer = gl.createBuffer();
    gl.bindBuffer( gl.ARRAY_BUFFER, vBuffer );
    gl.bufferData( gl.ARRAY_BUFFER, flatten(vertices), gl.STATIC_DRAW );

    var vPosition = gl.getAttribLocation( program, "vPosition" );
    gl.vertexAttribPointer( vPosition, 3, gl.FLOAT, false, 0, 0 );
    gl.enableVertexAttribArray( vPosition );
```

Adding Buttons for Rotation

```
var xAxis = 0;
var yAxis = 1;
var zAxis = 2;
var axis = 0;
var theta = [ 0, 0, 0 ];
var thetaLoc;

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

Render Function

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

Our
cube had only 8 vertices,
so a byte is enough to index it.
For larger models (vertex table with
more than 256 entries), use
gl.UNSIGNED_SHORT