Lab 22

We will modify the code in `Lab22Robot`, which is based on `09/figure`, where the slider bars have been removed, and initial angles have been hardcoded. Try some different values for these hardcoded angles and note the effect on the rendered robot.

Next, add lighting and light source. While there is some code to add (feel free to check out `shadedCube`), it should be fairly straightforward. In particular, you will define and calculate the polynomial normals, material and light properties, and the matrices in the shaders.

In the vertex shader, but outside `main()`:

```
in  vec3 aNormal;
out vec4 vColor;

uniform vec4 uAmbientProduct, uDiffuseProduct, uSpecularProduct;
uniform vec4 uLightPosition;
uniform float uShininess;
```

In the vertex shader, inside `main()`:

```
    vec3 pos = -(modelViewMatrix * aPosition).xyz;
    //fixed light postion
    vec3 light = uLightPosition.xyz;
    vec3 L = normalize(light - pos);
    vec3 E = normalize(-pos);
    vec3 H = normalize(L + E);
    vec4 NN = vec4(aNormal,0);
    // Transform vertex normal into eye coordinates
    vec3 N = normalize((modelViewMatrix*NN).xyz);
    // Compute terms in the illumination equation
    vec4 ambient = uAmbientProduct;
    float Kd = max(dot(L, N), 0.0);
    vec4  diffuse = Kd*uDiffuseProduct;
    float Ks = pow( max(dot(N, H), 0.0), uShininess );
    vec4  specular = Ks * uSpecularProduct;
    if( dot(L, N) < 0.0 ) {
       specular = vec4(0.0, 0.0, 0.0, 1.0);
    }
    vColor = ambient + diffuse +specular;
    vColor.a = 1.0;
```

In the fragment shader, but outside `main()`:

```
in vec4 vColor;
```

In the fragment shader and inside `main()` change the assignment so `fColor = vColor;`

In the Lab22Robot.js file, add the normals and light position after `pointsArray` is declared:

```
var normalsArray = [];
var lightPosition = vec4(5.0, -2.0, 4.0, 0.0);
var lightAmbient = vec4(0.2, 0.2, 0.2, 1.0);
var lightDiffuse = vec4(1.0, 1.0, 1.0, 1.0);
var lightSpecular = vec4(1.0, 1.0, 1.0, 1.0);
var materialAmbient = vec4(1.0, 0.0, 1.0, 1.0);
var materialDiffuse = vec4(1.0, 0.8, 0.0, 1.0);
var materialSpecular = vec4(1.0, 0.8, 0.0, 1.0);
var materialShininess = 100.0;
var ambientColor, diffuseColor, specularColor;
var viewerPos;
```

In `quad()`

```
var t1 = subtract(vertices[b], vertices[a]);
var t2 = subtract(vertices[c], vertices[b]);
var normal = cross(t1, t2);
normal = vec3(normal);
```

and

```
normalsArray.push(normal);
normalsArray.push(normal);
normalsArray.push(normal);
normalsArray.push(normal);
```

In `init()` after the call to `cube()`:

```
var nBuffer = gl.createBuffer();
gl.bindBuffer(gl.ARRAY_BUFFER, nBuffer);
gl.bufferData(gl.ARRAY_BUFFER, flatten(normalsArray),
gl.STATIC_DRAW);
var normalLoc = gl.getAttribLocation(program, "aNormal");
gl.vertexAttribPointer(normalLoc, 3, gl.FLOAT, false, 0, 0);
gl.enableVertexAttribArray(normalLoc);
    var ambientProduct = mult(lightAmbient, materialAmbient);

var diffuseProduct = mult(lightDiffuse, materialDiffuse);
var specularProduct = mult(lightSpecular, materialSpecular);
gl.uniform4fv(gl.getUniformLocation(program, "uAmbientProduct"),
ambientProduct);
gl.uniform4fv(gl.getUniformLocation(program, "uDiffuseProduct"),
diffuseProduct );
gl.uniform4fv(gl.getUniformLocation(program, "uSpecularProduct"),
specularProduct );
gl.uniform4fv(gl.getUniformLocation(program, "uLightPosition"),
lightPosition );
gl.uniform1f(gl.getUniformLocation(program,
   "uShininess"), materialShininess);
```

We've got lighting and shading, but no hidden-surface removal, so we need to enable the depth buffer,

Near the top of `init()`:

```
gl.enable(gl.DEPTH_TEST);
```

And then in `render()`:

```
gl.clear( gl.COLOR_BUFFER_BIT | gl.DEPTH_BUFFER_BIT);
```

Finally, let's make it animated.

Add the button to toggle the rotation in `Lab22Robot.html`:

```
<button id = "ButtonT">Toggle Rotation</button>
```

Add the flag to toggle animation on/off, in `Lab22Robot.js`, near the top:

```
var flag = true;
```

Add the button listener in `init()` at the bottom, just before the call to `render()`:

```
    document.getElementById("ButtonT").onclick = function(){flag =
!flag;};
```

Lastly, we will loop over the `initNodes` in `render()` :

```
        if (flag) {
           for(i=0; i<numNodes; i++)
           {
             theta[i] += i;
             initNodes(i);
           }
         }
```