CSE4204

LAB-6 : Using library, Lighting

Mohammad Imrul Jubair

Using a library

```
<script src='./cuon-matrix.js'></script>
```

Using a library

```
var rotateMatX = new Matrix4();
rotateMatX.rotate(thetaX, 1, 0, 0);

var rotateMatY = new Matrix4();
rotateMatY.rotate(thetaY, 0, 1, 0);

var s = 0.25;
var scaleMat = new Matrix4();
scaleMat.scale(s, s, s);

var tx = 0.0, ty = 0.0, tz = -2.0;
var transMat = new Matrix4();
transMat.translate(tx, ty, tz);
```

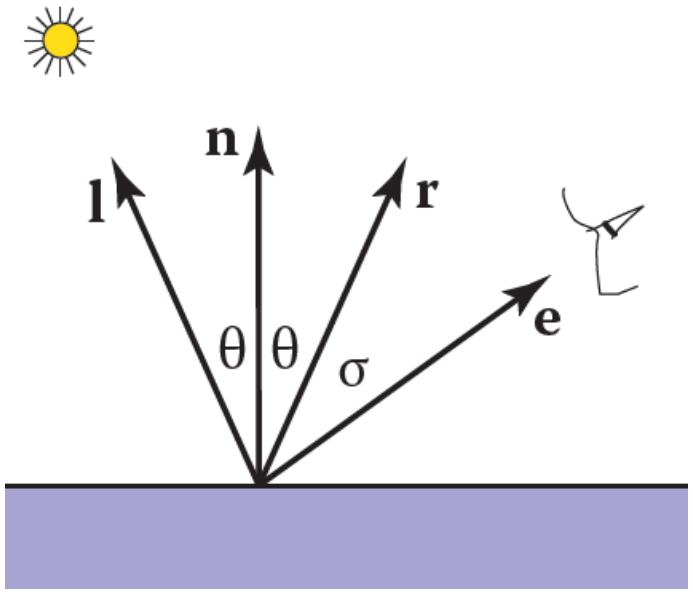
Using a library

```
var xe = 0.0, ye = 0.0, ze = 0.0;
var camMat = new Matrix4();
camMat.lookAt(xe, ye, ze, 0, 0, -2.0, 0, 1, 0);

var aspect = 1.0, fov = 75.0, far = 10.0, near = 1.0;
var persMat = new Matrix4();
persMat.perspective(fov, aspect, near, far);
```

Using a library

```
var mvpMat = new Matrix4();
mvpMat.multiply(persMat);
mvpMat.multiply(camMat);
mvpMat.multiply(transMat);
mvpMat.multiply(scaleMat);
mvpMat.multiply(rotateMatY);
mvpMat.multiply(rotateMatX);
```

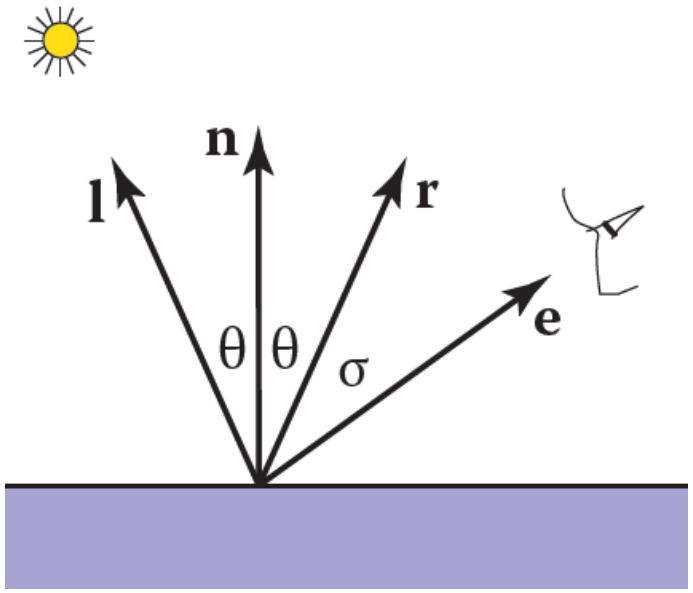
Phong Shading

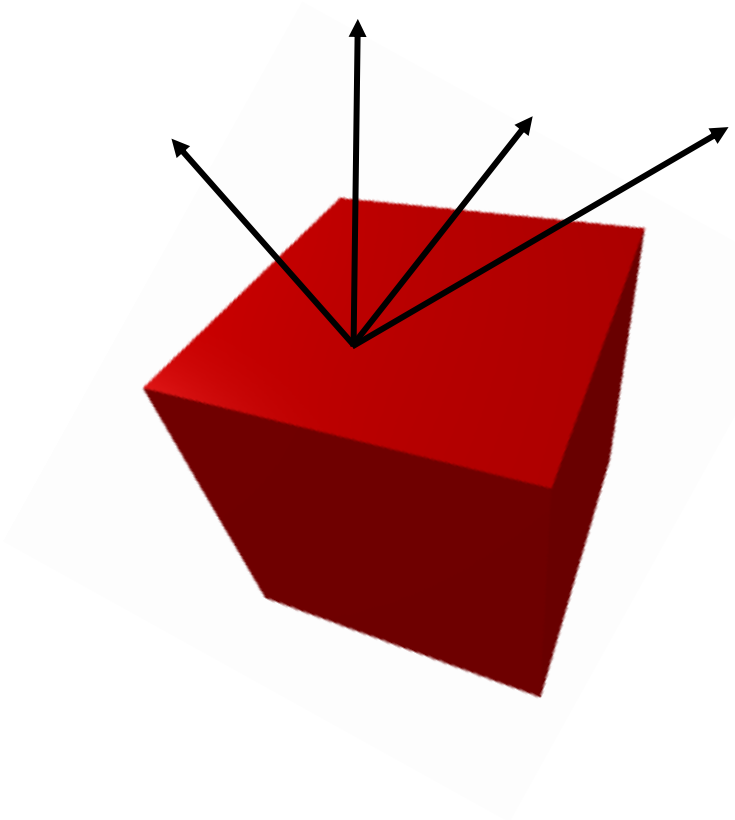


Color = Ambient + Diffuse + Highlights

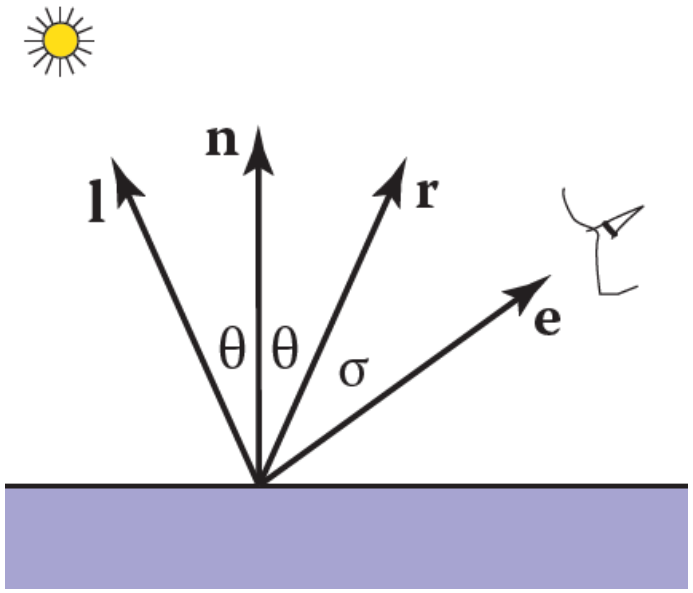
$$c = c_r \left(c_a + c_l \max\left(0, \mathbf{n} \cdot \mathbf{l}\right) \right) + c_l c_p \max(0, \mathbf{e} \cdot \mathbf{r})^p$$

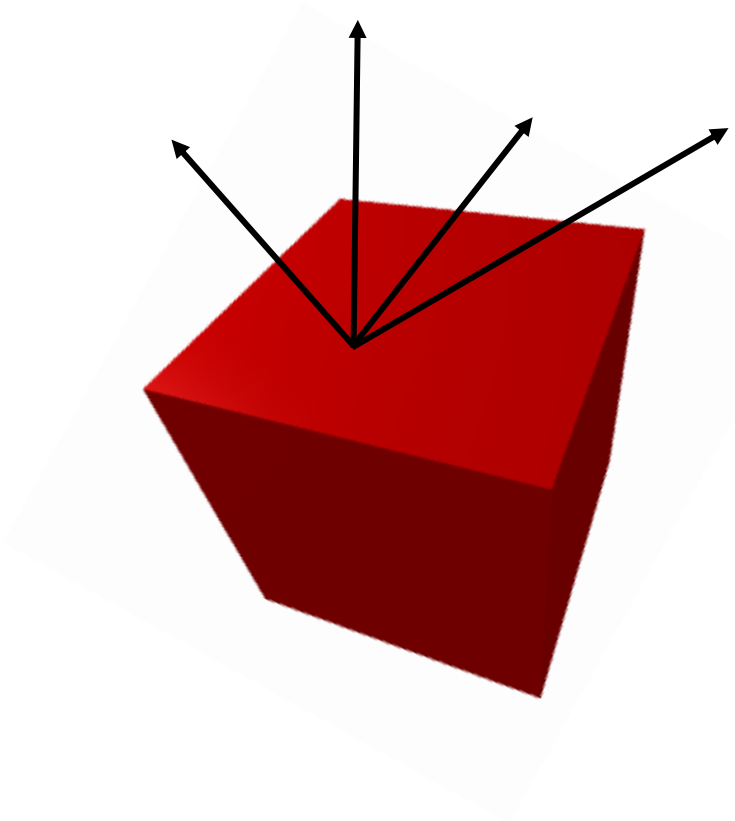
Phong Shading

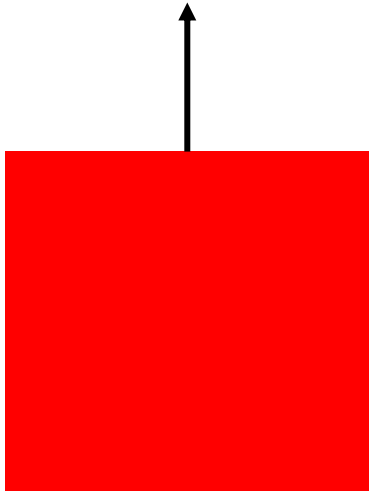




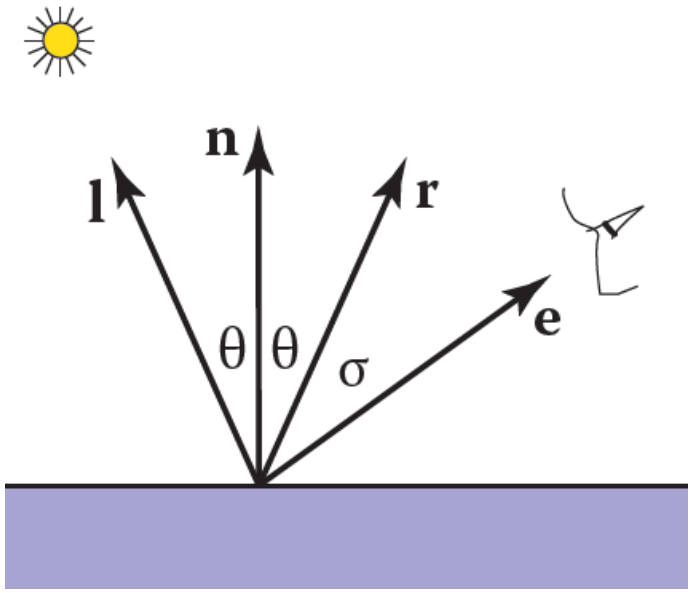
Normals

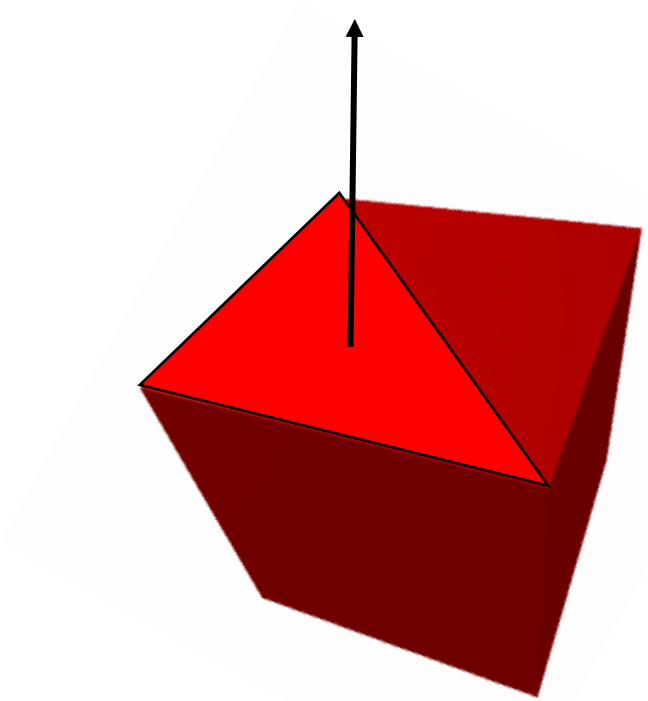


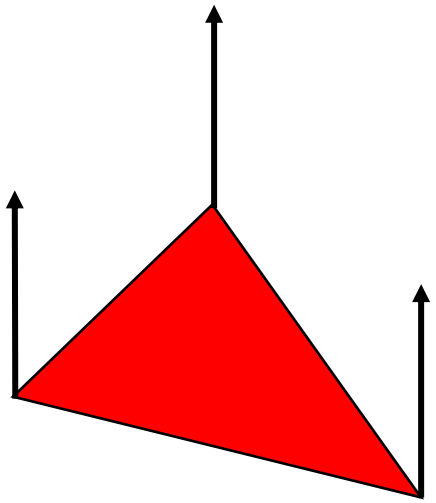




Vertex normal vs Face normal







Defining matrices in CPU

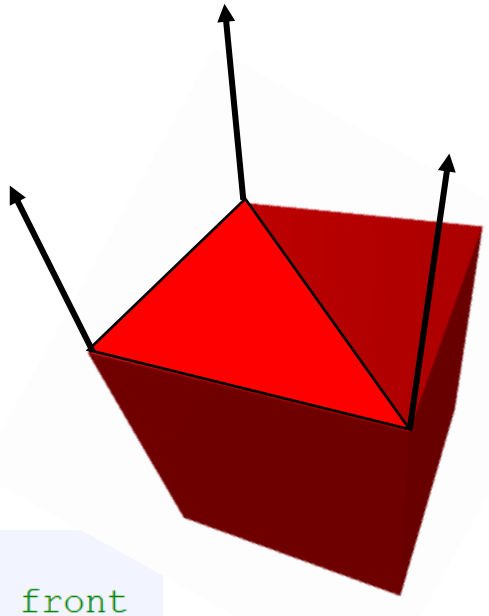
Model view matrix →

```
var modelViewMat = new Matrix4();
modelViewMat.multiply(camMat);
modelViewMat.multiply(transMat);
modelViewMat.multiply(scaleMat);
modelViewMat.multiply(rotateMatY);
modelViewMat.multiply(rotateMatX);
```

Model view projection (MVP) matrix →

```
var mvpMat = new Matrix4();
mvpMat.multiply(persMat);
mvpMat.multiply(modelViewMat);
```

Defining Normals



```
attribute vec3 a_coords;
attribute vec3 a_normals;
```

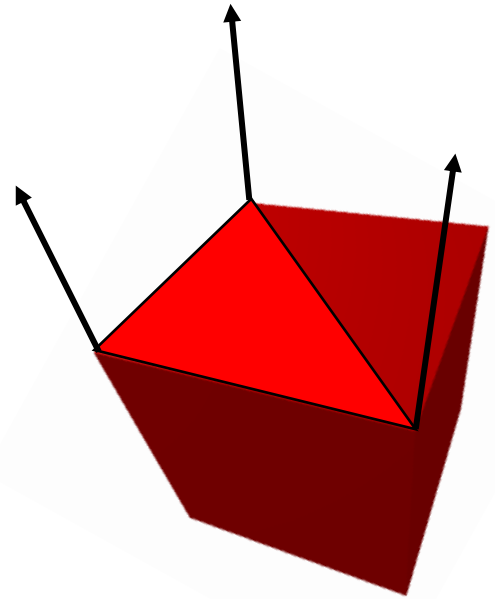
```
coords = new Float32Array([
     2.0, 2.0, 2.0,  -2.0, 2.0, 2.0,  -2.0,-2.0, 2.0,   2.0,-2.0, 2.0, // v0-v1-v2-v3 front
     2.0, 2.0, 2.0,   2.0,-2.0, 2.0,   2.0,-2.0,-2.0,   2.0, 2.0,-2.0, // v0-v3-v4-v5 right
     2.0, 2.0, 2.0,   2.0, 2.0,-2.0,  -2.0, 2.0,-2.0,  -2.0, 2.0, 2.0, // v0-v5-v6-v1 up
    -2.0, 2.0, 2.0,  -2.0, 2.0,-2.0,  -2.0,-2.0,-2.0,  -2.0,-2.0, 2.0, // v1-v6-v7-v2 left
    -2.0,-2.0,-2.0,   2.0,-2.0,-2.0,   2.0,-2.0, 2.0,  -2.0,-2.0, 2.0, // v7-v4-v3-v2 down
     2.0,-2.0,-2.0,  -2.0,-2.0,-2.0,  -2.0, 2.0,-2.0,   2.0, 2.0,-2.0  // v4-v7-v6-v5 back
  ]);
```

```
normals = new Float32Array([
  0.0, 0.0, 1.0,   0.0, 0.0, 1.0,   0.0, 0.0, 1.0,   0.0, 0.0, 1.0,  // v0-v1-v2-v3 front
  1.0, 0.0, 0.0,   1.0, 0.0, 0.0,   1.0, 0.0, 0.0,   1.0, 0.0, 0.0,  // v0-v3-v4-v5 right
  0.0, 1.0, 0.0,   0.0, 1.0, 0.0,   0.0, 1.0, 0.0,   0.0, 1.0, 0.0,  // v0-v5-v6-v1 up
 -1.0, 0.0, 0.0,  -1.0, 0.0, 0.0,  -1.0, 0.0, 0.0,  -1.0, 0.0, 0.0,  // v1-v6-v7-v2 left
  0.0,-1.0, 0.0,   0.0,-1.0, 0.0,   0.0,-1.0, 0.0,   0.0,-1.0, 0.0,  // v7-v4-v3-v2 down
  0.0, 0.0,-1.0,   0.0, 0.0,-1.0,   0.0, 0.0,-1.0,   0.0, 0.0,-1.0   // v4-v7-v6-v5 back
]);
```

Vertex Shader

```
attribute vec3 a_coords;
attribute vec3 a_normals;
```

```
varying vec3 v_coords;
varying vec3 v_normal;
```



Vertex Shader

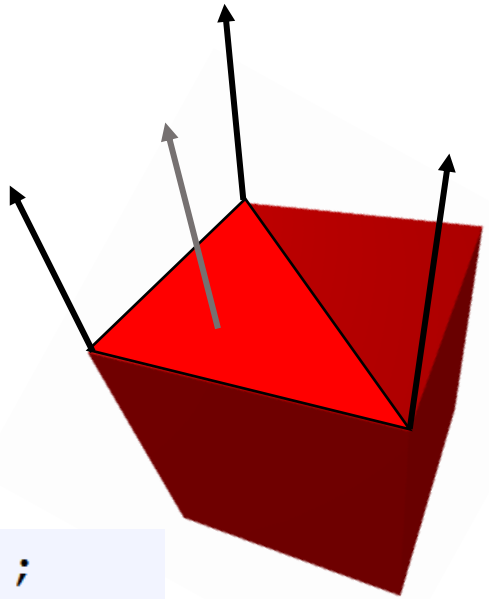
```
attribute vec3 a_coords;
attribute vec3 a_normals;
```

```
varying vec3 v_coords;
varying vec3 v_normal;
```

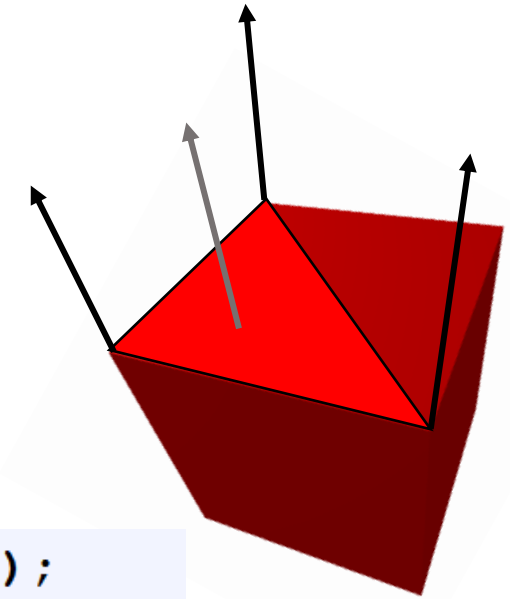
Multiplying with **model-view-proj** matrix

```
gl_Position = u_ModelViewProjection_matrix * vec4(a_coords,1.0);
vec4 position = u_ModelView_matrix * vec4(a_coords,1.0);
```

Multiplying with **model-view** matrix. we need the interpolated coordinates to calculate vectors in F.S

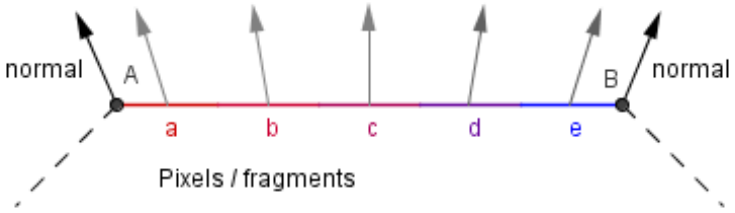


Vertex Shader

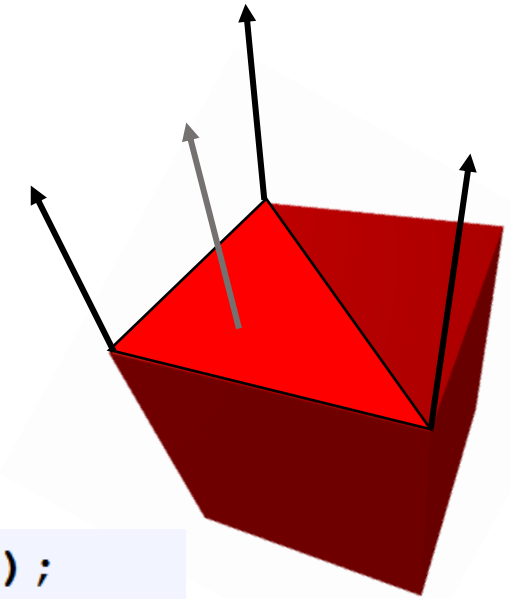


```
attribute vec3 a_coords;
attribute vec3 a_normals;

varying vec3 v_coords;
varying vec3 v_normal;

gl_Position = u_ModelViewProjection_matrix * vec4(a_coords,1.0);
vec4 position = u_ModelView_matrix * vec4(a_coords,1.0);
v_coords = position.xyz;
```

Homogeneous coord is not important for normals



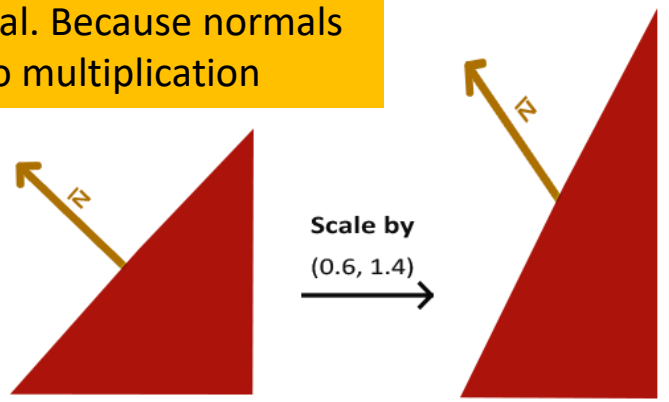
Vertex Shader



```
attribute vec3 a_coords;
attribute vec3 a_normals;

varying vec3 v_coords;
varying vec3 v_normal;

uniform mat4 u_Normal_matrix;

gl_Position = u_ModelViewProjection_matrix * vec4(a_coords,1.0);
vec4 position = u_ModelView_matrix * vec4(a_coords,1.0);
v_coords = position.xyz;
v_normal = normalize(vec3(u_Normal_matrix * vec4(a_normals, 1.0)));
```

Multiplying with **normal** matrix.

Normalizing the normal. Because normals can be changed due to multiplication

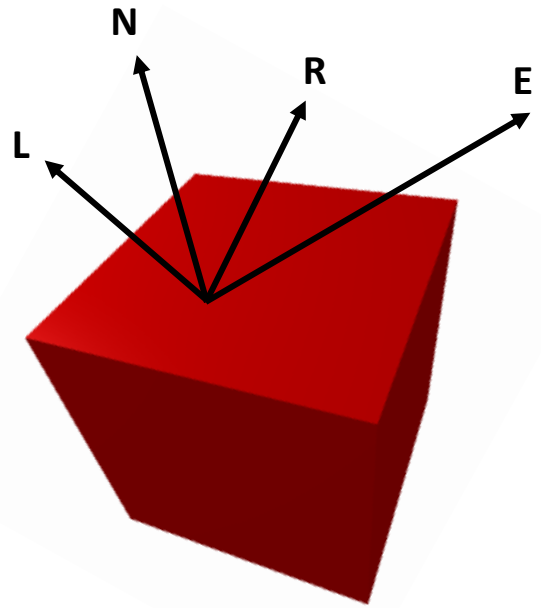



```
var normalMat = new Matrix4();
normalMat.setInverseOf(modelViewMat);
normalMat.transpose();
```

normal_matrix =trans(inv(model_view_matrix))

Fragment Shader

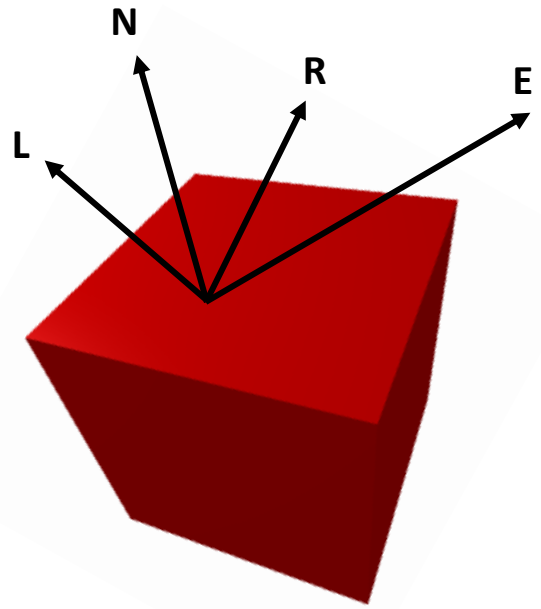
Color = Ambient + Diffuse + Highlights




```
vec3 C = Cr*(Ca + Cl*max(0.0, dot(L,N))) + Cl*Cp*pow(max(0.0, dot(R,E)), p);
gl_FragColor = vec4(C, 1.0);
```

Fragment Shader

Color = Ambient + Diffuse + Highlights



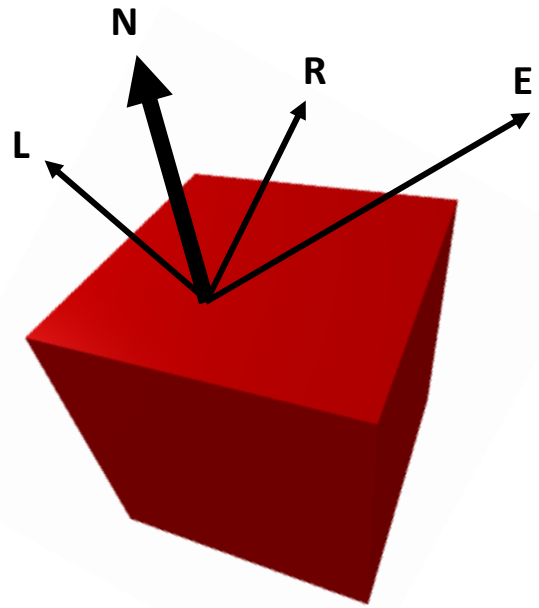

```
uniform vec3 u_LightColor;
uniform vec3 u_LightPosition;
uniform vec3 u_AmbientLight;
uniform vec3 u_Color;
varying vec3 v_coords;
varying vec3 v_normal;
```

```
vec3 C = Cr*(Ca + Cl*max(0.0, dot(L,N))) + Cl*Cp*pow(max(0.0, dot(R,E)), p);
gl_FragColor = vec4(C, 1.0);
```

Fragment Shader

Color = Ambient + Diffuse + Highlights

$$c = c_r \left(c_a + c_l \max\left(0, \mathbf{n} \cdot \mathbf{l}\right)\right) + c_l c_p \max(0, \mathbf{e} \cdot \mathbf{r})^p$$




```
vec3 N, L, R, E;
N = normalize(v_normal);
```

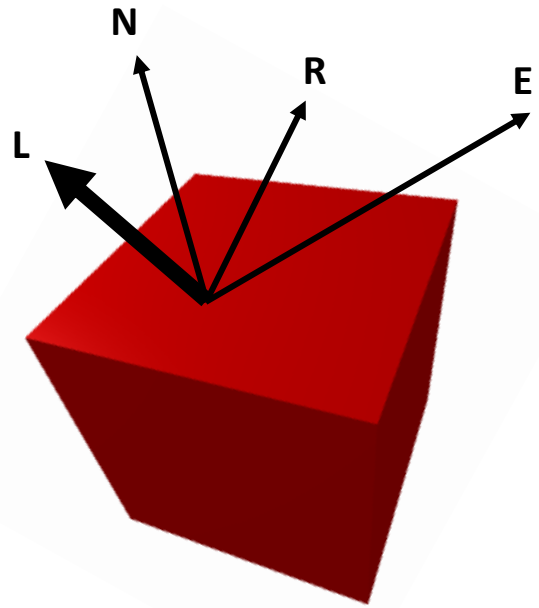
Normalizing the normal again. Because normals can be changed due to interpolation

```
vec3 C = Cr*(Ca + Cl*max(0.0, dot(L,N))) + Cl*Cp*pow(max(0.0, dot(R,E)), p);
gl_FragColor = vec4(C, 1.0);
```

Fragment Shader

Color = Ambient + Diffuse + Highlights

$$c = c_r \left(c_a + c_l \max\left(0, \mathbf{n} \cdot \mathbf{l}\right)\right) + c_l c_p \max(0, \mathbf{e} \cdot \mathbf{r})^p$$




```
vec3 N, L, R, E;
N = normalize(v_normal);
L = normalize(u_LightPosition - v_coords);
```

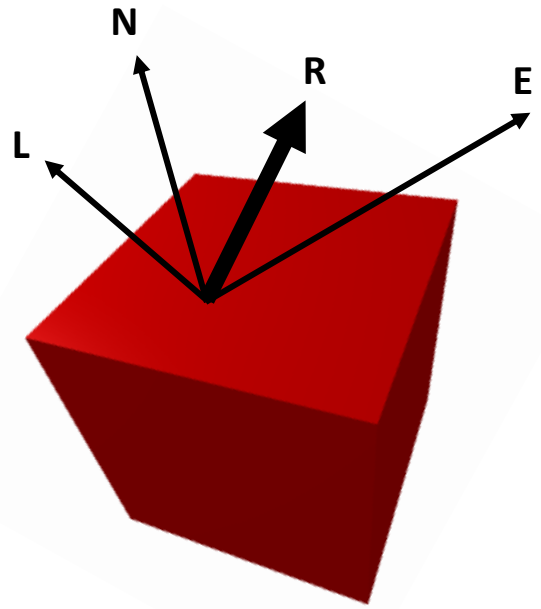
L → from coordinates to the light position

```
vec3 C = Cr*(Ca + Cl*max(0.0, dot(L,N))) + Cl*Cp*pow(max(0.0, dot(R,E)), p);
gl_FragColor = vec4(C, 1.0);
```

Fragment Shader

Color = Ambient + Diffuse + Highlights

$$c = c_r\left(c_a + c_l \max\left(0, \mathbf{n}\cdot\mathbf{l}\right)\right) + c_l c_p \max(0, \mathbf{e}\cdot\mathbf{r})^p$$



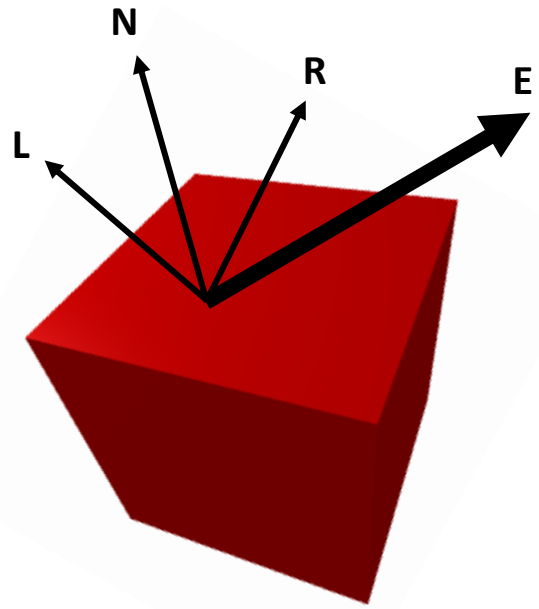

```
vec3 N, L, R, E;
N = normalize(v_normal);
L = normalize(u_LightPosition - v_coords);
R = -reflect(L,N);
```

```
vec3 C = Cr*(Ca + Cl*max(0.0, dot(L,N))) + Cl*Cp*pow(max(0.0, dot(R,E)), p);
gl_FragColor = vec4(C, 1.0);
```

Fragment Shader

Color = Ambient + Diffuse + Highlights

$$c = c_r \left(c_a + c_l \max\left(0, \mathbf{n} \cdot \mathbf{l}\right)\right) + c_l c_p \max(0, \mathbf{e} \cdot \mathbf{r})^p$$




```
vec3 N, L, R, E;
N = normalize(v_normal);
L = normalize(u_LightPosition - v_coords);
R = -reflect(L,N);
E = normalize(-v_coords);
```

v_coords --> from origin (eye) to the coordinate
E --> from coordinate to the origin (eye) --> -ve of v_coords

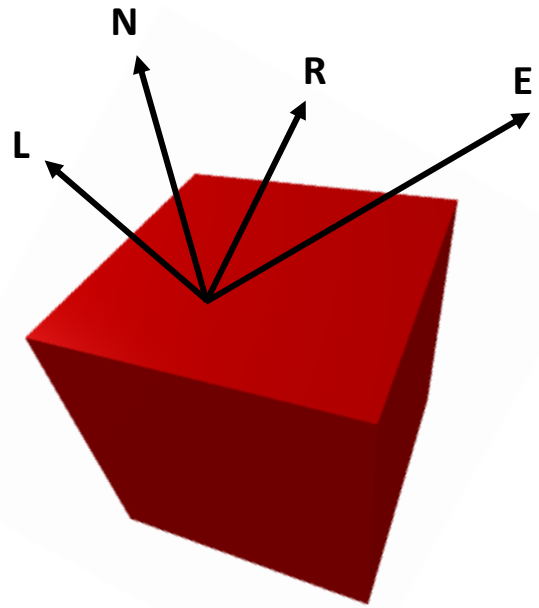
Normalizing the coordinates. Because they can be changed due to interpolation

```
vec3 C = Cr*(Ca + Cl*max(0.0, dot(L,N))) + Cl*Cp*pow(max(0.0, dot(R,E)), p);
gl_FragColor = vec4(C, 1.0);
```

Fragment Shader

Color = Ambient + Diffuse + Highlights

$$c = c_r\left(c_a + c_l \max\left(0, \mathbf{n}\cdot\mathbf{l}\right)\right) + c_l c_p \max(0, \mathbf{e}\cdot\mathbf{r})^p$$




```
vec3 N, L, R, E;
N = normalize(v_normal);
L = normalize(u_LightPosition - v_coords);
R = -reflect(L,N);
E = normalize(-v_coords);

vec3 Cr = u_Color;
vec3 Ca = u_AmbientLight;
vec3 Cl = u_LightColor;
float Cp = 0.7;
float p = 15.0;
```

```
vec3 C = Cr*(Ca + Cl*max(0.0, dot(L,N))) + Cl*Cp*pow(max(0.0, dot(R,E)), p);
gl_FragColor = vec4(C, 1.0);
```