The world is layed out in front, because that’s the coordinate we define view frustrum.

In eye coordinates its where your eye position is.

Apply world to eye by viewMatrix

Should use 3 matrices, ModelViewMatrix, ProjMatrix – maps to CVV, normal Matrix – inverse transpose of ModelViewMatrix.

In eye coordinates, in vertex shader take v\_position and turn into varying variable, so it is available to fragment shader. Eventually put into rasterizer (don’t worry). Take Normal Vector in eye coordinates and turn to varying variable. V\_Normal, will also be available in fragment shader. We get interpolated values. Normalize in the fragment shader to adjust, Use that normal to calculate for phong\_lighting and use v\_Position. Fragment shader must now compute 2 dots, ndotL , normal vec and light vector (take position – light position in camera coordinates, normalize). Getting half vector dotted with normal, the half vector is normalize average of view. Simply take position vector of negative to reverse object to person, normalize. Add light vectors divide by 2, use dot and that’s specular lighting. Should always do calculations in camera coordinates.

// Make unit sphere of r = 1

Then setLookAt with y position and point at position of sphere

Transofrm v\_Position into eyeSpace by multiplying by ModelView Matrix

Half vector is EyelOookAT – position, and average with something