MAT 494 – Computer Graphics

Review Sheet #2

- Lighting

  1. Here, $\vec{v}$ is the unit vector pointing from the spotlight to the vertex. $\vec{d}$ is GL_SPOT_DIRECTION. $\vec{n}$ is the unit normal vector at a particular vertex. $\vec{L}$ is the unit vector that points from the vertex to the light position.
  2. glLight, know what each of the parameters does (p189)
  3. vertex color (emission (+) global ambient scaled by the material (+) ambient diffuse and specular of all lights scaled by the material accounting for attenuation) (p215)
  4. The global ambient component is: ambient(global) * ambient(material) in a component wise multiplication. (p216)
  5. contribution = attenuation factor * spotlight effect * (ambient term + diffuse term + specular term)
  6. attenuation factor = $\frac{1}{k_c + k_l \cdot d + k_q \cdot d^2}$ (p216)
  7. Spotlight effect
     - 1 if GL_SPOT_CUTOFF is 180.0.
     - 0 if the vertex is outside the cone of illumination (use $\vec{v} \cdot \vec{d}$ to make this determination)
     - max($\vec{v} \cdot \vec{d}$, 0)$^{GL\_SPOT\_EXPONENT}$
  8. Ambient Term = ambient light · ambient material (p217)
  9. Diffuse Term = max ($\vec{L} \cdot \vec{n}$, 0) · diffuse light · diffuse material (p217)
  10. Specular Term (p218)
      - If $\vec{L} \cdot \vec{n}$ is 0, then Specular Term is 0.
      - $\vec{s}$ is the sum of the two unit vectors which point between the vertex and the light position and the vertex and the viewpoint. This vector is then normalized.
      - Specular Term = max ($\vec{s} \cdot \vec{n}$,0)$^{GL\_SHININESS}$ · specular light · specular material

- Using Alpha (blending) (p 227) glBlendFunc(GL_SRC_COLOR, GL_DST_COLOR), glEnable(GL_BLEND)

- Fog (p 255–259), glFog know what each of the parameters does (p259)

  - $f = e^{-(density \cdot z)}$
  - $f = e^{-(density \cdot z)^2}$

  - $f - \frac{end-z}{end-start}$

- Polygon Offset (p268–270)
  - Why is it needed?
  - When is it used?
  - What does a function call like glPolygonOffset(1,1) do compared to glPolygonOffset(-1,-1)?
- Display Lists (p271–285)
  - Why are they used?
  - What do they do?
  - How are they used? (glNewList,glEndList,glCallList)
- Texture Mapping (p359–370)
  - What is a texture map?
  - Texture Coordinates (p414–416)
  - Clamp/Repeat (p 417–420)
  - Automatic Texture Generation (p422)
- Framebuffers (p451)
  - Color Buffer(p455)
  - Depth Buffer (Depth Func) p455,469–470
  - Stencil Buffer (Stencil Func and Stencil Op) p464–468
- Wire Frame Images (p269–270)
- Hidden Line Removal (p604–605)
- Ray Tracing
  - Ray Tracing vs. Z-buffer (visible surfaces at the pixel level)
  - More powerful routine for: shadows, reflection, refraction
  - Viewing (u,v,w)
  - The Ray ($p(t) = \vec{origin} + t \cdot \vec{direction}$)
  - i and j are pixel indices. n_x and n_y are the number of pixels in the screen. To find
  - $u_s = left + (right - left) * \frac{i+0.5}{n_x}$
  - $v_s = bottom + (top - bottom) * \frac{j+0.5}{n_y}$
  - $w_s = near$

- $gaze = \frac{at-eye}{|at-eye|}$, $w = -\frac{g}{|g|}$, $u = \frac{up \times w}{|up \times w|}$, $v = \frac{w \times u}{|w \times u|}$,
- Given u, v, w, u_s, and v_s, the world coordinate position on the screen is: $s = eye + u_s * u + v_s * v + w_s * w$
- The ray is then $p(t) = eye + t * (s - eye)$
- We use this ray to intersect every object in the scene. This is done by solving the position of the ray against a position equation for a particular piece of geometry. A circle has the formula $(p - center) \cdot (p - center) - radius^2 = 0$. Solving the ray equation in for p gives a quadratic equation for t. Solving this, we can determine the points of intersection (if any) the ray has with the sphere. If there are two roots, the smallest positive root is the intersection point we require. Given the value of t, we can solve for the position $p(t)$ if needed.