Other Techniques to Increase Realism

Intended Learning Outcomes

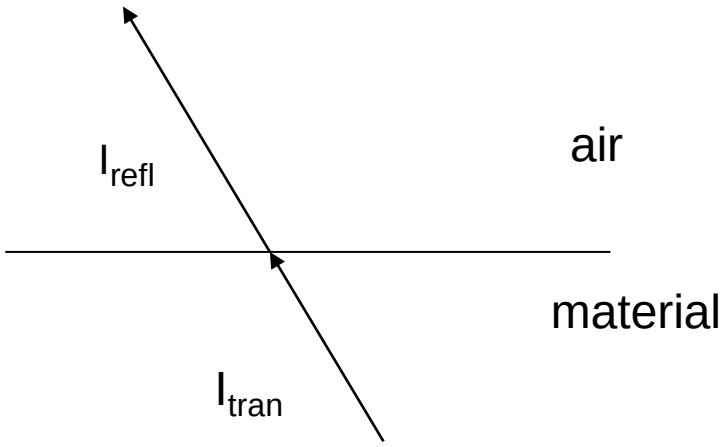
- Apply techniques to simulate transparency, fog and distance attenuation
- Able to model a kind of distributed light source

Simulating Transparency

- An approximate transparency model is

$$I = (1 - k_t) I_{refl} + k_t I_{tran}$$

- k_t transparency coefficent, $0 \le k_t \le 1$
- $k_t = 0 \Rightarrow$ opaque $k_t = 1 \Rightarrow$ transparent



It is an approximate model because refraction is not modelled

OpenGL implementation

- Set the colour or material of a surface to semi-transparent

glColor4f (R, G, B, A) *// A = k_t*
// A is called "alpha parameter

- Use colour blending

glEnable (GL_BLEND)
glBlendFunc (GL_ONE_MINUS_SRC_ALPHA, GL_SRC_ALPHA);

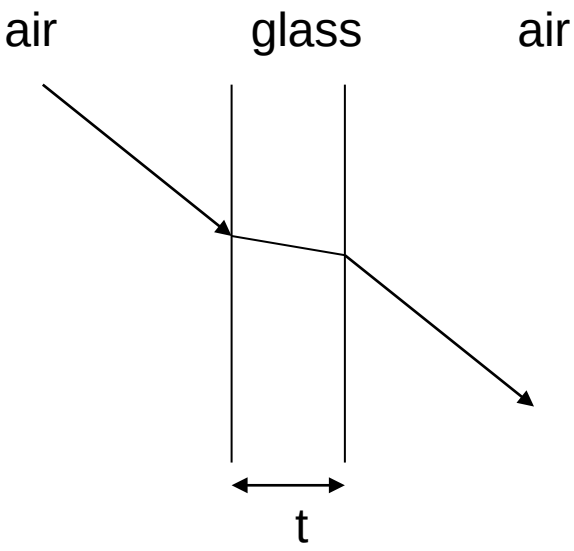
// combine the colour using A from current frame buffer value
// and (1-A) from the value of semi-transparent surface in front

- Correct visibility is achieved by not updating the Z buffer values when processing semi-transparent surfaces

```
glEnable (GL_DEPTH_TEST);
        :          // draw all opaque surfaces here
        :
glDepthMask (GL_FALSE);   // make Z buffer read only
glEnable  (GL_BLEND);      // blend surface colours from now on
glBlendFunc (GL_ONE_MINUS_SRC_ALPHA, GL_SRC_ALPHA);
        :          //  draw all semi-transparent surfaces here
        :
glDisable  (GL_BLEND);      //  disable blending
glDepthMask (GL_TRUE);    //  Z buffer can now both read and write
```

Better model

- models refraction for a slab of glass



The light ray is still parallel but only displaced by an amount which is a function of t

- Generally, can use ray tracing to calculate the refraction ray.
- No OpenGL implementation for the above exists.

Adding Atmospheric Effect

- Models haze, dust, smoke, smog etc in the atmosphere

$$f_{atmo}(d) = e^{-\rho d} \qquad \text{or}$$

$$f_{atmo}(d) = e^{-(\rho d)^2}$$

- d distance of object from viewing position
- To simulate atmosphere colour

$$I = f_{atmo}(d) I_{obj} + [1 - f_{atmo}(d)] I_{atmo}$$

Adding distance attenuation functions

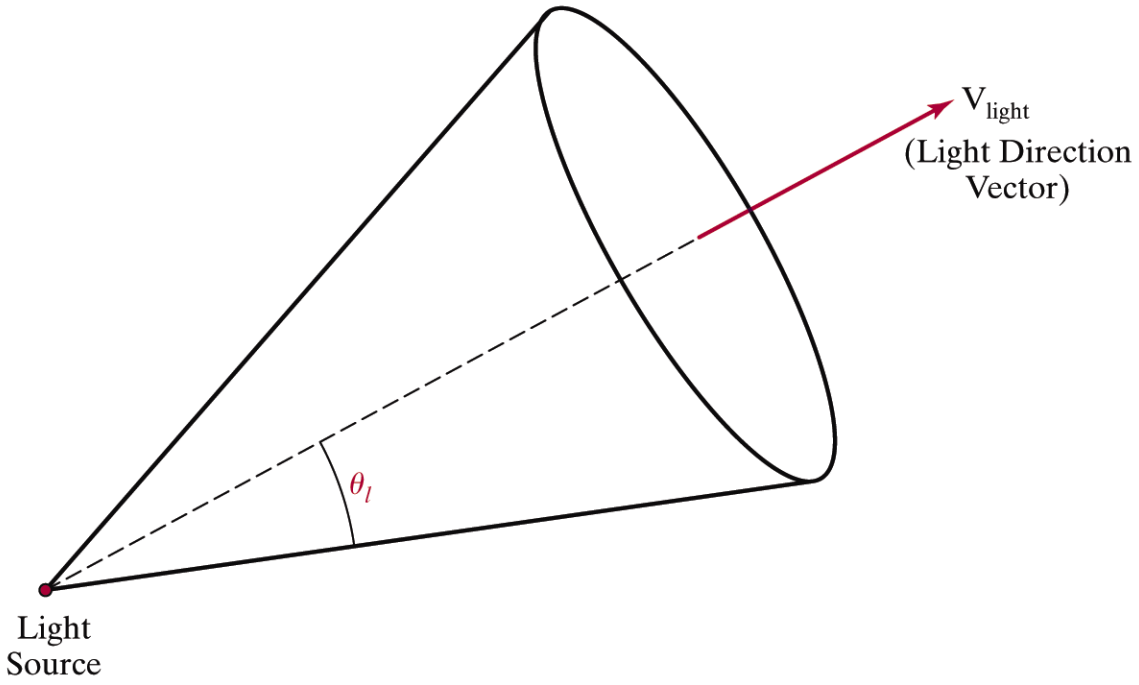
- Light attenuates according to inverse square law $1/d^2$
- Neglecting this effect, two overlapping surface that has the same normal will be indistinguishable
- For realistic light attenuation, use

$$f(d) = \begin{cases} 1.0 & \text{Light source at infinity} \\ \dfrac{1}{a_0 + a_1 d + a_2 d^2} & \text{otherwise} \end{cases}$$

- make the ray casting and tracing more realistic

Modelling spot light

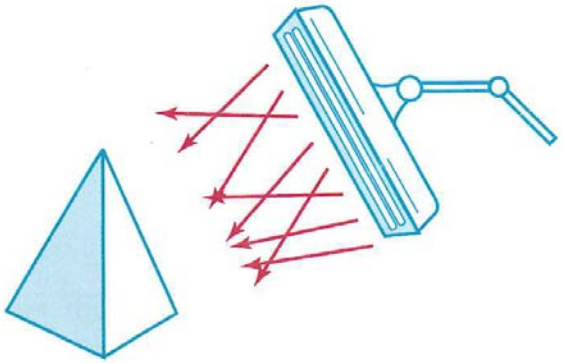
$$f_l = \begin{cases} 1.0 & \text{if source is not a spotlight} \\ 0.0 & \text{if } \mathbf{V}_{obj} \cdot \mathbf{V}_{light} = \cos\alpha < \cos\theta_l \\ (\mathbf{V}_{obj} \cdot \mathbf{V}_{light})^{n_s} & \text{otherwise} \end{cases}$$



A directional point light source. The unit light-direction vector defines the axis of a light cone, and angle θ_l defines the angular extent of the circular cone.

Warn model

- models distributed light source
- A light source is a mesh
- Each vertex is a spot light
- $\mathbf{V}_{\text{light}}$ is the normal at the vertex



OpenGL commands (1)

- Atmospheric effect

```
glEnable (GL_FOG);
GLfloat        atmoColor [4] = {0.8, 0.8, 1.0, 1.0};
glFogfv (GL_FOG_COLOR, atmoColor);

glFogi (GL_FOG_MODE, GL_EXP);
glFogf  (GL_FOG_DENSITY, rho);     //
```

$$f_{atmo}(d) = e^{-\rho d}$$

OpenGL commands (2)

- Specifying attenuation

```
glLightf (GL_LIGHT6, GL_CONSTANT_ATTENUATION, 1.5);
glLightf (GL_LIGHT6, GL_LINEAR_ATTENUATION,  0.75);
glLightf (GL_LIGHT6, GL_QUADRATIC_ATTENUATION, 0.4);
```

- Spotlight

```
GLfloat dirVector [ ] = {1.0, 0.0, 0.0};
glLightfv (GL_LIGHT3, GL_SPOT_DIRECTION, dirVector);
glLightf   (GL_LIGHT3, GL_SPOT_CUTOFF, 30.0);
glLightf   (GL_LIGHT3, GL_SPOT_EXPONENT, 2.5);
```

References

- Text: pp.135-36 for color blending
- Text: Ch. 17-4 for transparency, and also look up the *glDepthMask* command
- Text: Ch. 17.5 for light and fog modelling
- Demo: fog.exe in TUTORS program
- Text: Ch. 17.11 for OpenGL commands