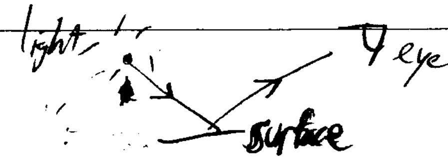


local illumination (one bounce)

Phong Lighting Model

$$I = \underbrace{I_a k_a}_{\text{ambient}} + \sum_L \Big[\underbrace{I_L k_d (\vec{N}\cdot\vec{L})}_{\text{diffuse}} + \underbrace{I_L k_s (\vec{R}\cdot\vec{V})^n}_{\text{specular}} \Big]$$

lights: I_a, I_L | surface: k_a, k_d, k_s] user

geom: N, L, R, V

Suppose we want: white light, shiny yellow material

$$\begin{bmatrix} I_r \\ I_g \\ I_b \end{bmatrix} = \begin{bmatrix} 0.2 \\ 0.2 \\ 0.2 \end{bmatrix} \begin{bmatrix} 0.6 \\ 0.6 \\ 0 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \begin{bmatrix} 0.6 \\ 0.6 \\ 0 \end{bmatrix} (\vec{N}\cdot\vec{L}) + \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \begin{bmatrix} 0.5 \\ 0.5 \\ 0 \end{bmatrix} (\vec{R}\cdot\vec{V})^{50}$$

→ ensure $\vec{N}\cdot\vec{L} \geq 0$, i.e., use $\max(0, \vec{N}\cdot\vec{L})$

ensure $\vec{R}\cdot\vec{V} \geq 0$, i.e., use $\max(0, \vec{R}\cdot\vec{V})$

+ basic WebGL demos of Phong illumination on a teapot.

- Q: What is the colour of the darkest pixel on a rendered sphere?

Occurs when $N\cdot L = 0$, $R\cdot V = 0$, only left with ambient light, $I = \begin{bmatrix} 0.12 \\ 0.12 \\ 0 \end{bmatrix}$

Q: What is the colour of the brightest pixel on a rendered sphere?

occurs when $N\cdot L = 1$, $R\cdot V = 1$

$$I = \begin{bmatrix} 0.12 \\ 0.12 \\ 0 \end{bmatrix} + \begin{bmatrix} 0.6 \\ 0.6 \\ 0 \end{bmatrix} + \begin{bmatrix} 0.5 \\ 0.5 \\ 0 \end{bmatrix} = \begin{bmatrix} 1.22 \\ 1.22 \\ 0 \end{bmatrix} \xrightarrow{\text{clamp to 1}} \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}$$

Q: What would we see when the yellow surface is illuminated with a blue light? → I_a and I_L

black

$$I = \begin{bmatrix}0\\0\\1\end{bmatrix}\begin{bmatrix}0.2\\0.2\\0\end{bmatrix} + \begin{bmatrix}0\\0\\1\end{bmatrix}\begin{bmatrix}0.6\\0.6\\0\end{bmatrix}(N \cdot L) + \begin{bmatrix}0\\0\\1\end{bmatrix}\begin{bmatrix}0.5\\0.5\\0\end{bmatrix}(R \cdot V)^{50} = \begin{bmatrix}0\\0\\0\end{bmatrix}$$

yellow surfaces absorb blue light

Q: What would be seen with a red light?

$$\begin{bmatrix}1\\0\\0\end{bmatrix}\begin{bmatrix}0.2\\0.2\\0\end{bmatrix} + \begin{bmatrix}1\\0\\0\end{bmatrix}\begin{bmatrix}0.6\\0.6\\0\end{bmatrix}(N \cdot L) + \begin{bmatrix}1\\0\\0\end{bmatrix}\begin{bmatrix}0.5\\0.5\\0\end{bmatrix}(R \cdot V)^{50} = \begin{bmatrix}0 \cdots 1\\0\\0\end{bmatrix}$$

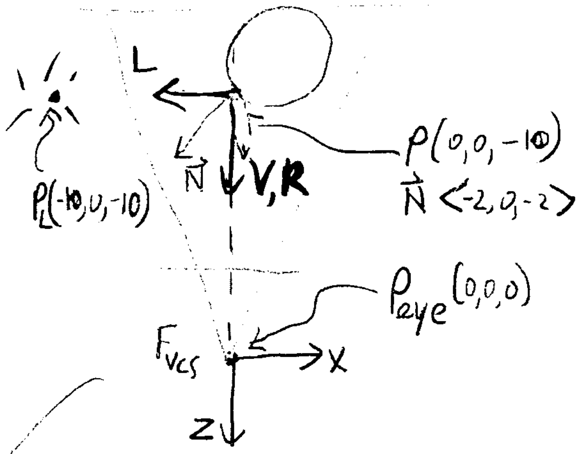
Q: Do specular highlights have the colour of the light or the surface?

⇒ Usually the colour of light

i.e., commonly use $k_s = \begin{bmatrix}1\\1\\1\end{bmatrix}$

Need N, L, R, V

Q: What is the rendered colour for point P, as shown below?



$$\hat{N} = \frac{N}{\|N\|} = \frac{1}{\sqrt{8}} \langle -2,0,-2\rangle$$

$$L = P_L - P = \langle -10,0,0\rangle$$

$$\hat{L} = \frac{L}{\|L\|} = \frac{1}{10}\langle -10,0,0\rangle$$

$$V = P_{eye} - P = \langle 0,0,10\rangle$$

$$\hat{V} = \langle 0,0,1\rangle$$

→ lighting computations done in VCS

$$I = f($$

$$R = 2\vec{N}(\vec{N}\cdot\vec{L}) - \vec{L} \quad \text{(see notes)}$$
$$= \langle 0,0,1\rangle$$

$$\vec{N}\cdot\vec{L} = \frac{1}{\sqrt{2}} \simeq 0.7$$

$$\vec{R}\cdot\vec{V} = 1$$

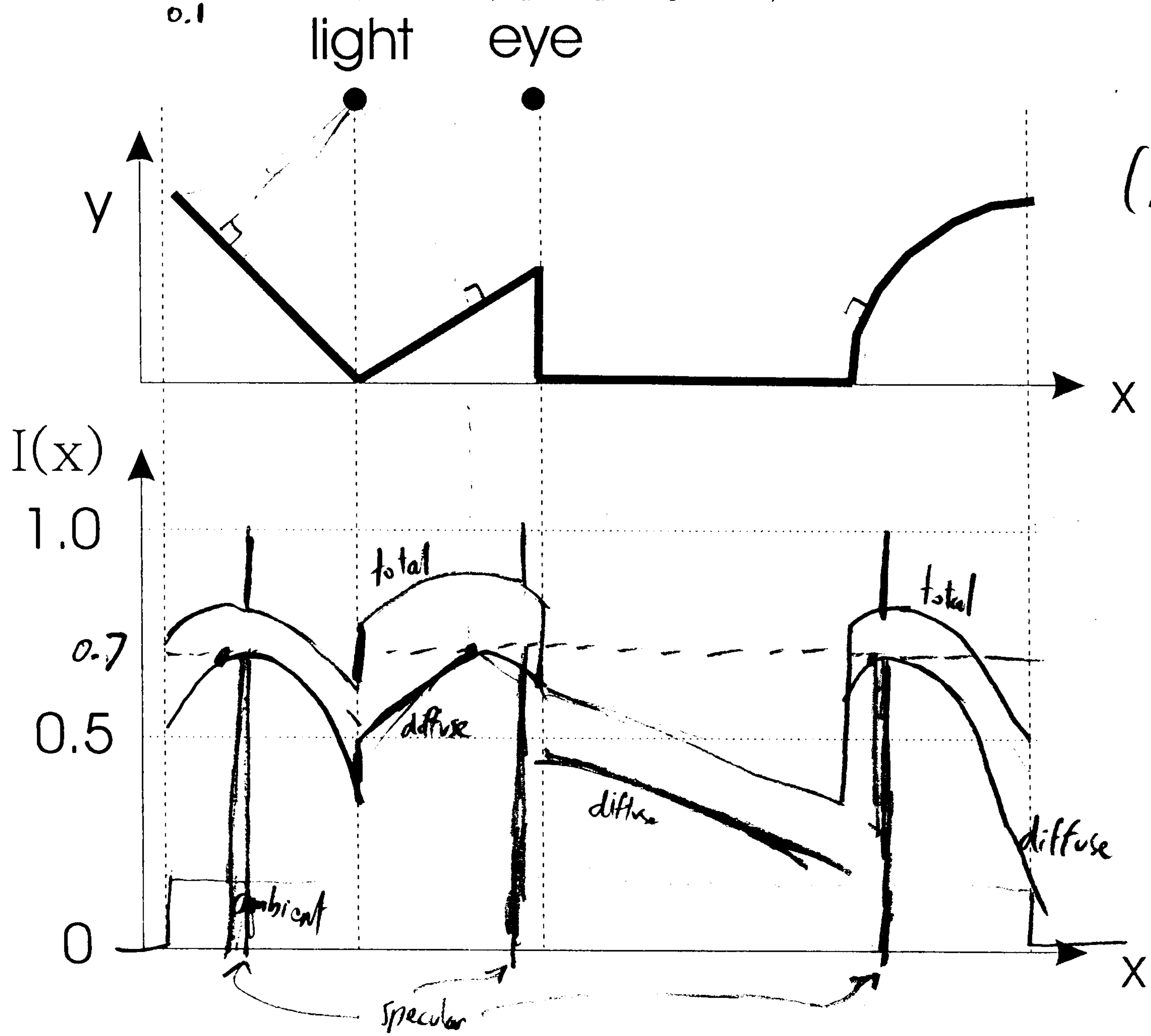
$$I = \begin{bmatrix}0.2\\0.2\\0.2\end{bmatrix}\begin{bmatrix}0.6\\0.6\\0\end{bmatrix} + \begin{bmatrix}1\\1\\1\end{bmatrix}\begin{bmatrix}0.6\\0.6\\0\end{bmatrix}(0.7) + \begin{bmatrix}1\\1\\1\end{bmatrix}\begin{bmatrix}0.5\\0.5\\0\end{bmatrix}(1)$$

$$= \begin{bmatrix}0.12\\0.12\\0\end{bmatrix} + \begin{bmatrix}0.42\\0.42\\0\end{bmatrix} + \begin{bmatrix}0.5\\0.5\\0\end{bmatrix} = \begin{bmatrix}1.04\\1.04\\0\end{bmatrix} \Rightarrow \begin{bmatrix}1\\1\\0\end{bmatrix}$$

Sketch the ambient, diffuse, specular, and total illumination for the following scene as a function of x. Assume the Phong illumination model, i.e., $I = k_a I_a + k_d I_d (N \cdot L) + k_s I_s (R \cdot V)^n$, where $k_a =$ ~~0.3~~ 0.1, $k_d = 0.7, k_s = 0.7, I_a = I_d = I_s = 1.0, n = 100$.

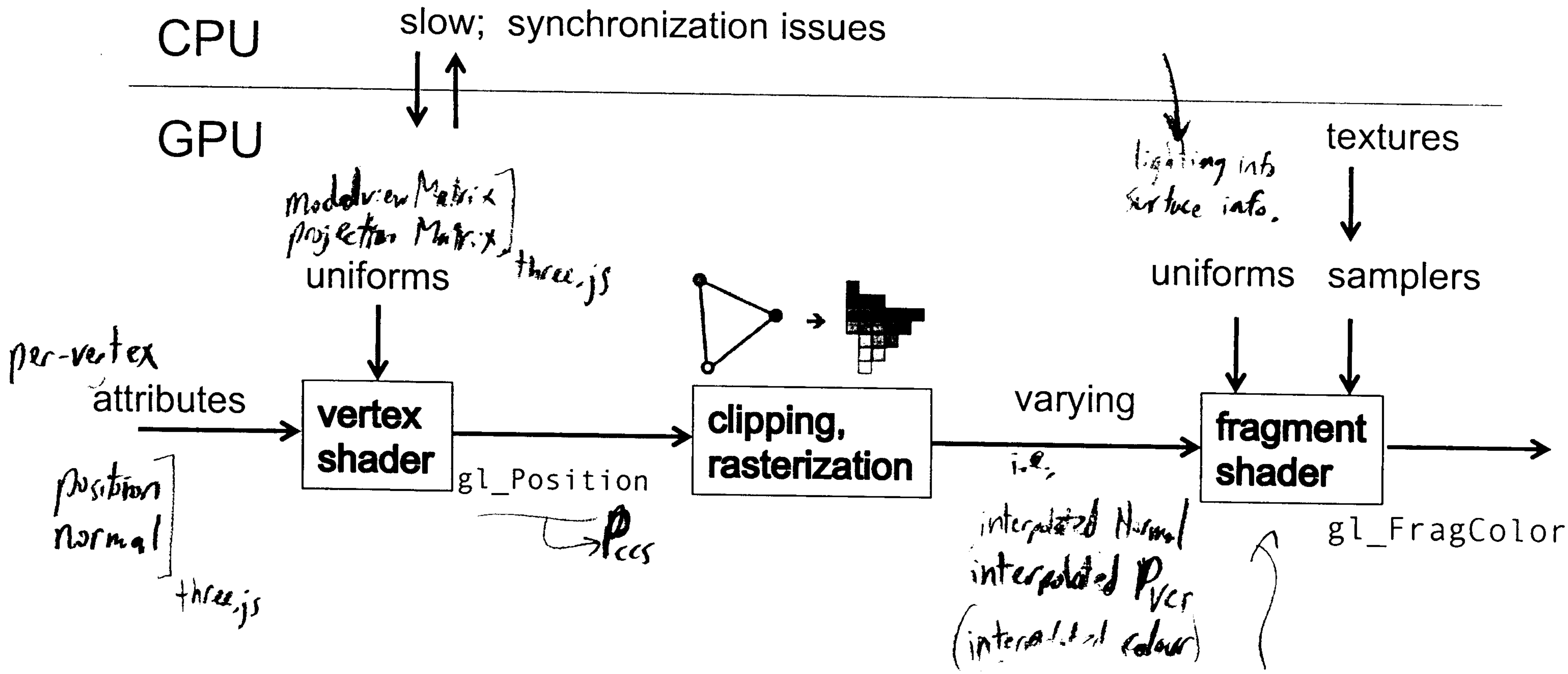
For a grey-scale world, i.e., $I_r = I_g = I_b$

(not covered in class – no time)



(began with a quick shadertoy demo)

Shader Overview



$I_a k_a + I_L k_d (N \cdot L) + I_L k_s (R \cdot$

\+ live coding demo of three.js, building a shader that renders N•L as a gray-scale colour, i.e. basic diffuse illumination.

THREE.JS Built-in uniforms and attributes for GL_SL shaders

VERTEX SHADER

```
uniform mat4 modelMatrix;          // = object.matrixWorld
uniform mat4 modelViewMatrix;      // = camera.matrixWorldInverse * object.matrixWorld
uniform mat4 projectionMatrix;     // = camera.projectionMatrix
uniform mat4 viewMatrix;           // = camera.matrixWorldInverse
uniform mat3 normalMatrix;         // = inverse transpose of modelViewMatrix
uniform vec3 cameraPosition;       // = camera position in world space

attribute vec3 position;           // vertex position
attribute vec3 normal;             // vertex normal
attribute vec2 uv;                 // texture coordinate
attribute vec2 uv2;                // texture coordinate
```

The vertex shader should always output:

- gl_Position   This is a vec4 that gives the vertex coord after projection
- any "varying" variables you wish to have interpolated and passed on to the fragment shader

Note that you can therefore calculate the position of a vertex in the vertex shader by either:

```
gl_Position = projectionMatrix * modelViewMatrix * vec4( position, 1.0 );
gl_Position = projectionMatrix * viewMatrix * modelMatrix * vec4( position, 1.0 );
```

FRAGMENT SHADER

```
uniform mat4 viewMatrix;           // = camera.matrixWorldInverse
uniform vec3 cameraPosition;       // = camera position in world space
```

The fragment shader should also declare any "varying" variables that it will receive as input

The fragment shader should always output:
- gl_FragColor   This is a vec4 that describes the desired "fragment" or pixel color