Things we know how to do right now.
Recap on how we have been painting our models.

\[ I = c_0 k_m + c k_m \max \left( 0, \tilde{n} \cdot \tilde{l} \right) + c_l k_a \max(0, \tilde{v} \cdot \tilde{n})^p \]

```
if (h > 0.8)
    show rock
else:
    show forest
```

color = \(a - \text{color}\)

pass \(h\) to FS

```
if -else directly in FS
    \(K_m\)
```
Other places we have seen this? Think about week 3.

$$I = c_0k_m + c_1k_m \max(0, \vec{n} \cdot \vec{l}) + c_2k_s \max(0, \vec{v} \cdot \vec{n})^p$$
Warmup: can we use a **Uint8Array** instead? (event # 9000136)

```javascript
gl.bufferData(gl.ELEMENT_ARRAY_BUFFER, new Uint8Array(mesh.triangles), gl.STATIC_DRAW);
```

For our cube (8 vertices) from last class, can we buffer triangles indices (using `gl.bufferData`) using a `Uint8Array`?

- Yes, but something else is missing. 48%
- No, 8 bits is not enough for this number of vertices. 22%
- Yes, we can just change this to `Uint8Array`. 22%
- I'm not sure. 9%

```javascript
gl.drawElements(gl.TRIANGLES, mesh.triangles.length, gl.UNSIGNED_SHORT, 0);
```
The main idea of texturing: sample an image to determine surface properties.

\[(s,t)\]

three ingredients:

1. image
2. texture coordinates \((s,t)\)
3. how we look up "texel"
Ingredients #1 and #3: which texel to sample?

For now, assume we use nearest texel (center)
Ingredient #2: we need texture coordinates.

(s₀, t₀) values might be different for different triangles.
A special case: texture coordinates on a sphere.

Why? approximate lots of things like spheres!

compute \( (s,t) \) using \( (\theta, \phi) \)

\[
\begin{align*}
z &= R \cos \phi \\
x &= R \sin \phi \cos \theta \\
y &= R \sin \phi \sin \theta \\
\phi &= \arccos(z/R) \text{ in range } [0, \pi] \\
\theta &= \arctan(y, x) \text{ in range } [-\pi, \pi] \\
(s, t) &\in [0, 1]^2
\end{align*}
\]
Doing it with **WebGL**.

In **JavaScript**:

```javascript
1 // retrieve the image
2 let image = document.getElementById("spot-texture");
3
4 // create the texture and activate it
5 let texture = gl.createTexture();
6 gl.getActiveTexture(gl.TEXTURE_0); // ------ important! make a note of N in gl.TEXTURE
7 gl.bindTexture(gl.TEXTURE_2D, texture);
8
9 // define the texture to be that of the requested image
10 gl.texImage2D(gl.TEXTURE_2D, 0, gl.RGB, gl.RGB, gl.UNSIGNED_BYTE, image);
11
12 // set filter parameters
13 gl.texParameteri(gl.TEXTURE_2D, gl.TEXTURE_MIN_FILTER, gl.NEAREST);
14 gl.texParameteri(gl.TEXTURE_2D, gl.TEXTURE_MAG_FILTER, gl.NEAREST);
15
16 // tell webgl which texture index to use for the uniform sampler2D in the shader
17 gl.uniform1i(gl.getUniformLocation(program, 'tex_image'), 0);
```

In shader:

```plaintext
1 precision highp float;
2 uniform sampler2D tex_image;
3
4 void main() {
5 float s = ...;
6 float t = ...;
7 vec3 kn = (texture2D(tex_image, vec2(s, t))).rgb;
8 gl_FragColor = vec4(kn, 1.0);
9 }
```
Other properties to modify with textures?

\[ I = c_\alpha k_m + c_i k_m \max (0, \vec{n} \cdot \vec{l}) + c_i k_s \max (0, \vec{v} \cdot \vec{r})^p \]