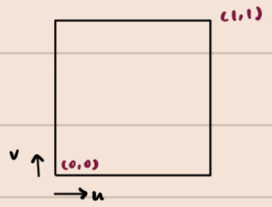
I. Texture Mapping.

1. Texture is (usually) an image of $n \times n$. We also define $u, v \in [0,1]$ as its horizontal & vertical span.



<u>Def</u> (Texel) a 1x1 region on a texture is called a texel.

2. Usually, each vertex corresponds to a coordinate (u,v) in the texture space.

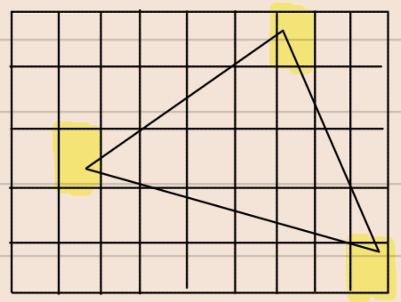
⇒ Textures encodes information not only colors but any encodable info, such as normals.

3. Sampling Algorithm.

<u>Algo</u> (Per-pixel)

For each pixel in the image:

- Interpolate (u,v) across triangle.
- Sample texture at interpolated (u,v).
- Set color of fragment to sampled texture value.



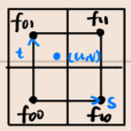
↑ only the marked 3 pixels have a vertex inside. All the other pixels requires interpolations.

II. Minification & Magnifications.

1. Magnification.

① When camera is <u>very close</u> to scene object. single pixel maps to tiny region of texture.

② Bilinear Interpolation.



- When (u,v) is a non-integer location.
- Compute:
  - $i = \lfloor u - ½ \rfloor$. $j = \lfloor v - ½ \rfloor$.
  - $s = u - (i + ½)$ $t = v - (j + ½)$. easily check that $s, t \in [0,1]$.
- Return $(1-t)\left((1-s) f_{00} + s f_{10}\right) + t\left((1-s) f_{01} + s f_{11}\right)$.

③ Nearest : simply return the color of nearest texel.

2. Minification.

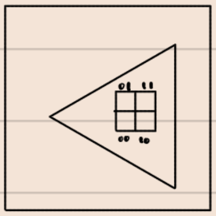
① When camera is far away. such that a single pixel covers many texels.

② Basic Idea: pre-compute "smaller" textures. and look up in these images.

③ MIP map.

Store textures of size $2^n \times 2^n$. look up a single pixel from MIP map of appropriate size.

④ Compute MIP Map level.



$\frac{\Delta u}{\Delta x} = u_{10} - u_{00}$. $\frac{\Delta u}{\Delta y} = u_{01} - u_{00}$. $L_x^2 = \left(\frac{\Delta u}{\Delta x}\right)^2 + \left(\frac{\Delta v}{\Delta x}\right)^2$.

$\frac{\Delta v}{\Delta x} = v_{10} - v_{00}$ $\frac{\Delta v}{\Delta y} = v_{01} - v_{00}$. $L_y^2 = \left(\frac{\Delta u}{\Delta y}\right)^2 + \left(\frac{\Delta v}{\Delta y}\right)^2$.

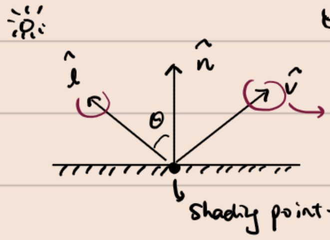
$L = \sqrt{\max(L_x^2, L_y^2)}$. ⇒ mipmap level $D = \log_2 L$.

⑤ Trilinear Interpolation for interpolating b/t levels.

(Roughly). - Bilinearly interpolation at 2 nearest levels to $D \in \mathbb{R}$.

- Interpolate b/t 2 bilinear values using $w = d - \lfloor d \rfloor$.

II. Blinn-Phong Model.



Model Color = Highlights (Specular) + Diffuse + Ambient

1. **Diffuse**: light gets reflected to multiple directions uniformly.

① Lambert's Law. $I_{light} \propto \cos\theta = \hat{l} \cdot \hat{n}$

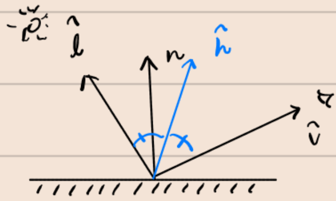
② $L_d = k_d (I/r^2) \max(0, \hat{n} \cdot \hat{l})$

- k_d: color
- (I/r^2) → energy arrived at shading point
- $\max(0, \hat{n} \cdot \hat{l})$ ↪ energy received by the shading point.
- L_d ↑ diffuse light.

⚠ By intuition, should have nothing to do with $\hat{v}$.

Black white.

k_d : $[0, 1]$



2. **Specular**: lights gets reflected to the eyes.

① Def (half vector) $\hat{h} = \text{bisector}(\hat{v}, \hat{l}) = \frac{\hat{v} + \hat{l}}{\|\hat{v} + \hat{l}\|}$.

② $L_s = k_s (I/r^2) \max(0, \hat{n} \cdot \hat{h})^p$ → power for limiting the specular region.

- L_s ↑ specular light
- k_s ↑ mirror factor.
- $\hat{h}$ ↓ half vector

3. **Ambient**: Environmental indirect lights.

① $L_a = k_a I_a$

- L_a ↑ ambient light
- k_a ↖ ambient factor.

⚠ By intuition, should stay nonrelated with $\hat{n}$ and $\hat{l}$ and $\hat{v}$.

Finally,

Blinn-Phong Reflection Model

$$L = L_a + L_d + L_s$$

$$= k_a I_a + k_d (I/r^2) \max(0, \hat{n} \cdot \hat{l}) + k_s (I/r^2) \max(0, \hat{n} \cdot \hat{h})^p.$$

III. Shading Schemes / Shading Frequency

1. **Per-Triangle** a.k.a. Flat Shading.

Presumptions: All triangles are "flat" or planar. i.e. has only one $\hat{n}$.

Shade a triangle to a single shading.

Output: is fairly acceptible. as it is generally not smooth.

2. **Per-vertex** a.k.a. **Gouraud Shading**   Quality are constrained by the size of primitives.

Stage: Vertex Shader.

From Application: ① Normal vector at the vertices.

② Positions of the lights ⇒ direction to the lights $\hat{l}$ can then be computed.

③ Color of the lights.

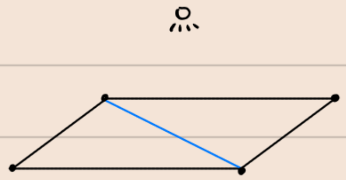
Vertex Shader: ① Direction to the lights $\hat{l}$ (Light Dir)

② Direction to the camera $\hat{v}$ (View Dir)

Disadvantages: ① Do not produce any details in the shading that are smaller than the primitives used to draw the surface.

② Also, curved surfaces must be drawn using primitives small enough for specular area.

Ex. for ①.



shading will only be evaluated at the corners.

the central part are interpolated. making it too dark.

3. **Per-fragment** a.k.a. **Phong Shading**.

Stage: Fragment Shader.

From Application: The same as per-vertex.

From Vertex Shader: Prepare values for interpolation.

Fragment shader: Each pixel has a interpolated vector set.

Compute the color on each pixel.

Additionally, how to compute the averaged normal for a vertex?

Sol: We average the normals of the surrounding surfaces, weighted by the area.

$$\hat{n}_v = \frac{\sum_i \hat{n}_i}{\|\sum_i \hat{n}_i\|}$$