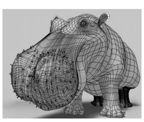
Texturing

Chapter 10

Texturing



Model

Model+shading

Model+shading+textures

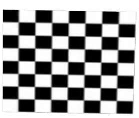
Texture mapping

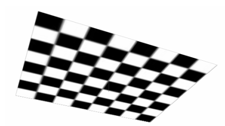
- Adding detail to polygon by using a image and mapping it onto the geometry
- Array of color values [RGBA]

Texturing

- After shading and rasterization we have
  - Per-pixel color value (fragment color) RGBA
  - Per-pixel depth value: z
- Given a texture, what to do with it?
  - We need to know <u>which part of the texture</u> that corresponds to the pixel. We need a mapping
  - Also, we need to know the <u>average texture color</u> contributing to the pixel. We need texture filtering.

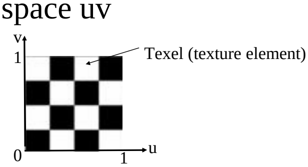
Parametric Texture Mapping





- Texture size and orientation are tied to the polygon
  - Separate texture space and screen space
  - Deform (render) the textured polygon into screen space

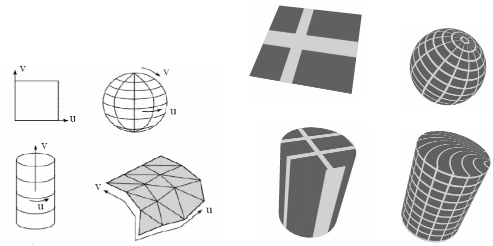
Texture space uv



- A texture lives in its own image coordinates, parameterized by (u,v) u? [0,1] and v? [0,1]
- If an image has the size w*h, then each texel is of size (1/w, 1/h) in the *uv* space.
- The texel value (RGBA(should locate at the center of the texel)
- w and h is usually powers of 2 for ease of computation.

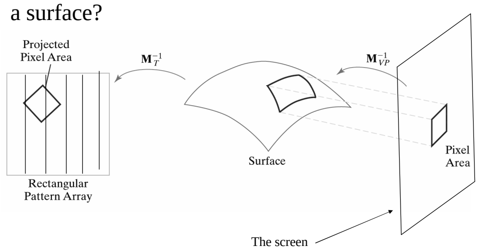
Texture mapping

? Textures can be wrapped around many different surfaces

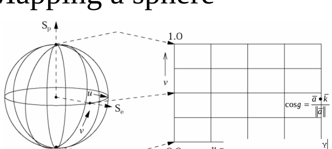


Pixel to Texture space

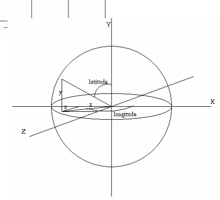
? From a given pixel, which texel should end up on a surface?



Mapping a sphere

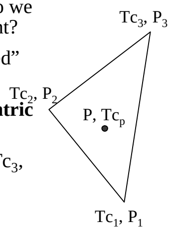


- F , lattitude = arctan2(z/x,z)
- T , longitude = arccos(y)



Mapping to a triangle

- Assume each vertex has a texture coordinate associated to it.
- Given a point on the triangle, how do we know what texel to apply to that point?
- We need to calculate the ”interpolated” texture coordinate.
- We can do this by using the **Barycentric coordinate**
- We have P, P_1, P_2, P_3 and Tc_1, Tc_2, Tc_3, We need Tc_x



Barycentric coordinates

- Given a line defined by p_1 and p_2
- Find: a point P on the line:
- Solution: $\mathbf{p} = (1 - t)\mathbf{p}_1 + t\mathbf{p}_2$.
- *(1-t)* and *t* is the barycentric coordinates, t might range [- 8 ,8], in the interval *t ?* [0, 1] we trace the line between p_1 and p_2
- For the barycentric coordinates u, v the following holds: u+v=1 (which we can see above!)
- How does this work for a triangle?

Barycentric coordinates

- **For a triangle we have 3 points:**
- $\mathbf{p} = u\mathbf{p}_1 + v\mathbf{p}_2 + w\mathbf{p}_3$, where $u + v + w = 1$.
- *[u,v,w]* are the barycentric coordinates
- We could also write:
  – $\mathbf{p} = u\mathbf{p}_1 + v\mathbf{p}_2 + (1 - u - v)\mathbf{p}_3$. *(1)*
- *(1) is a linear system:* $\begin{bmatrix} p_1 & p_2 & p_3 \end{bmatrix} \begin{bmatrix} u \\ v \\ w \end{bmatrix} = p$

Barycentric coordinate

- Now using cramers rule (checkout Mathworld!) give us the following determinants:

$$
\begin{aligned}
D &= \begin{vmatrix} p_1 & p_2 & p_3 \end{vmatrix} \\
A_1 &= \begin{vmatrix} p & p_2 & p_3 \end{vmatrix} \\
A_2 &= \begin{vmatrix} p_1 & p & p_3 \end{vmatrix} \\
A_3 &= \begin{vmatrix} p_1 & p_2 & p \end{vmatrix}
\end{aligned}
$$

- And finally: $\left[u = \frac{A_1}{D}, \quad v = \frac{A_2}{D}, \quad w = \frac{A_3}{D} \right]$
- Remember w = 1-u-v, so we can save some calculations here.

Mapping a triangle

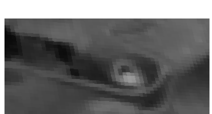
- Now, we wanted the texture coordinate at position p.

$$Tc_p = uTc_1 + vTc_2 + wTc_3$$

Filtering

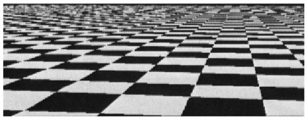
- What do we do when a texture sample lands between texel centers, *undersampling*.
- GL_NEAREST – Pick the closest one
- GL_LINEAR – Interpolate between the four closest, a BILINEAR approach.

GL_LINEAR



GL_NEAREST

Aliasing



- Reason?
  - Each pixel on the screen, maps to thousands of texels. Which one should we choose? We have oversampling

Mip Mapping

- One possible solution is to prefilter the texture.
- To reduce the high frequency
- Introduced by Lance Williams, 1983
- MIP = “multumin parvo”–“many things in a small place”
- Pre-process input textures by prefiltering it at multiple resolutions
- From the change in u/v (du, dv) we can pick a corresponding mip-level, and sample a texel from that level.
- Linear interpolate between the two nearest level
- Supported by OpenGL

Bump mapping

- Plain textures doesnt model rough surfaces well
- Illumination is still calculated on the underlying flat textured polygon
- What if we could model the illumination on the flag polygon using a texture?
- The bump texture is treated as a height function
- We calculate the partial derivatives and model the pixel normal using that. That would make the lighting effect such as if the surface were rough...
- Example?

Bump mapping

- Since only the normals of the surface is altered, the siluette is not affected.