Computer Graphics 1

7 Texture

Summer Semester 2021
Ludwig-Maximilians-Universität München
Tutorial 7: Texture

● Texture Mapping
  ○ Barycentric Interpolation
  ○ MIP Map

● Texture Maps
  ○ Normal Map
  ○ Displacement Map
  ○ Environment Map
Texture Coordinates

Texture coordinates define a mapping from surface coordinates to a texture image space.

Texture mapping is a process in which an artist manually creates a texture in a baking process, and then UV coordinates are saved for each given vertex (recall UV coordinates from a .OBJ file).

Basic idea:

```javascript
triangle.project().pixels.forEach((x, y) => {
    [u, v] = getTextureCoord(x, y)
    color = sampleTexture(u, v)
    draw(x, y, color)
})
```

*We assume the uv coordinates are provided in a model’s vertex since textures are often drawn manually by artists (very time consuming).*
Graphics Pipeline (Revisited)

Input

Vertex Shader

Tessellation Shaders

Geometry Shader

Fragment Shader

Frame Buffer

Uniform (per Frame)

uv-Coordinates as attribute (per Vertex)

uv-Coordinates as out (interpolated, per fragment)

Extract information (color, normal, ...) from texture and use it to colorize the fragment
Interpolation Between Vertex and Fragment Shaders

Take the vertex normal as an example:

- Many attributes are interpolation between vertex and fragment shaders:
  - Colors and textures
  - UV coordinates and Normals
  - ...
  - Interpolated in fragment shader using Barycentric interpolation
Barycentric Interpolation

If \( P \) is inside the triangle, geometrically:

\[
\vec{AP} = w_2 \vec{AB} + w_3 \vec{AC}, \quad w_2, w_3 \in [0, 1]
\]

\[
\implies P - A = w_2 (B - A) + w_3 (C - A)
\]

\[
\implies P = (1 - w_2 - w_3)A + w_2 B + w_3 C
\]

Let \( w_1 = 1 - w_2 - w_3 \in [0, 1] \)

We have: \( P = w_1 A + w_2 B + w_3 C \)

This is how we interpolate the vertex attributes for \( P \) given the color of \( A, B, \) and \( C \). For example, color:

\[
\text{color}(P) = w_1 \text{color}(A) + w_2 \text{color}(B) + w_3 \text{color}(C)
\]

But what are \( w_1, w_2, w_3 ? \)
Barycentric Interpolation (cont.)

Because:

\[ \vec{A}P = w_2 \vec{AB} + w_3 \vec{AC}, \quad w_2, w_3 \in [0, 1] \]

We can write this linear equations:

\[ \begin{align*}
\vec{A}P_x &= w_2 \vec{AB}_x + w_3 \vec{AC}_x \\
\vec{A}P_y &= w_2 \vec{AB}_y + w_3 \vec{AC}_y \\
\end{align*} \]

\[ w_1 + w_2 + w_3 = 1 \]

Solving them, we have:

\[ \Rightarrow \quad w_3 = \frac{\vec{A}P_x \vec{AB}_y - \vec{A}P_y \vec{AB}_x}{\vec{AC}_x \vec{AB}_y - \vec{AC}_y \vec{AB}_x} = \frac{\vec{A}P \times \vec{AB}}{\vec{AC} \times \vec{AB}} = \frac{S_{ABP}}{S_{ABC}} \]

\[ \begin{align*}
w_2 &= \frac{S_{APC}}{S_{ABC}} \\
w_1 &= \frac{S_{BCP}}{S_{ABC}} \end{align*} \]

This is how we compute the barycentric coordinates.
Example: Edge Cases

Meaning when barycentric coordinates have different values (also sign):

If \( w_1 = 1, w_2 = w_3 = 0 \Rightarrow P = A \)

If \( w_2 = 1, w_1 = w_3 = 0 \Rightarrow P = B \)

If \( w_3 = 1, w_1 = w_2 = 0 \Rightarrow P = C \)

...

Conclusion:

If \( \forall w_i \in [0, 1], P \) is inside the triangle ABC

If \( \exists w_i < 0 \), P is outside the triangle ABC

Aside: We just found a new approach to replace the point-in-triangle assertion.
Texture Filtering (Sampling)

- Magnification
  - Area of screen pixel maps to tiny region of texture
  - Texture resolution is too low, we want an interpolated color of a given pixel ⇒ Interpolation, e.g. Linear interpolation

- Minification
  - Area of screen pixel maps to large region of texture
  - Texture resolution is too high, we want the average color of an area ⇒ Range query
More Interpolation(s)

Linear interpolation between numbers:

\[ \text{lerp}(x, y, t) = x + t(y - x) \]

Linear interpolation between positions:

\[ \text{lerp}(u, v, t) = u + t(v - u) \]

Barycentric interpolation is also a linear interpolation with respect to three positions

Bilinear interpolation is just a 3x linear interpolation:
**MIP Map**

- (Isotropic) MIP map is a fast approximation for a range query
- Basic idea: Pre-compute a texture version for the "LOD". Find the correct level (or levels in between) and get the color directly
MIP Map (cont.)

- MIP map levels of a $d \times d$ image, e.g: in a 1024x1024 size image
  - Level 0: 1024x1024
  - Level 1: 512x512
  - Level 2: 256x256
  - ... until we get 1x1 pixel
  - **There are $1 + \log_2 d$ levels in total.**

- Storage overhead: $\frac{1}{3}$ more storage

$$
\frac{d^2}{4} \left( 1 + \frac{1}{4} + \left( \frac{1}{4} \right)^2 + \left( \frac{1}{4} \right)^3 + \cdots \right) = d^2 \lim_{n \to \infty} \sum_{i=1}^{n} \frac{1}{4^i} = \frac{4}{3} d^2
$$

https://en.wikipedia.org/wiki/Mipmap#/media/File:MipMap_Example_STS101.jpg

Computing MIP Map Level

- How do we know which MIP map level to choose?
- The level of a MIP map hierarchy for a color query is estimated by: \( L = \log_2 \max(d_1, d_2) \)
  - \( d_1, d_2 \) are Euclidean distances between pixels
Trilinear Interpolation

If the estimated MIP level is not an integer, we need to use *trilinear interpolation* between two discrete MIP map levels. Trilinear interpolation is just one more linear interpolation between two bilinear interpolations.
Limitation of Isotropic Mipmap

The texture is blurred!

What's wrong here??
Anisotropic Filtering

- look up axis-aligned rectangular zones
- Diagonal range query is still an issue

Example: Using Anisotropic Filtering
Tutorial 7: Texture

- Texture Mapping
  - Barycentric Interpolation
  - MIP Map

- Texture Maps
  - Normal Map
  - Displacement Map
  - Environment Map
Bump Map

- Often referred as "normal map", although they are different
- A normal map primarily affects the normals of a surface
  - It can add surface detail without adding more triangles
  - Perturbs the surface normals per pixel (for shading)
  - The object’s geometry doesn’t change

- Limitations
  - No actual changes to the geometry
  - No actual changes to the casted shadow
  - …

Displacement Map

- A more advanced method
- Changes the geometry (moves vertices)

Environment Map

An efficient image-based lighting technique for approximating the appearance of a reflective surface by means of a precomputed texture image.

Limitations:

- No self reflections
- Some geometric objects cannot be correctly mapped to a sphere
- ...

Breakout: Earth

Open earth.blend. See how the earth with star background is created with its texture and an environment map.
Texture Mapping is Powerful!

Fog map from "Fabian Bauer, Creating the Atmospheric World of Red Dead Redemption 2: A Complete and Integrated Solution, SIGGRAPH 2019"
Summary

● We covered:
  ○ Texture mapping as a process of querying the corresponding color on a pixel
  ○ Barycentric interpolation as a linear interpolation to interpolate properties of a triangle interior
  ○ Magnification and minification in texture sampling and how to use bilinear interpolation and MIP maps for color queries
  ○ Different kinds of texture maps for creating fancy objects

● Modern texture mapping is done procedurally (or programmably). To learn more about procedural texturing,

  We encourage to check these books:
Next

Shading and Shadowing