Computer Graphics 1

6 Texture

Summer Semester 2022

Ludwig-Maximilians-Universität München

Tutorial 6: Texture

- Texture Mapping
  - Barycentric Interpolation
  - MIP Map
- Texture Maps
  - Normal Map
  - Displacement Map
  - Environment Map

Texture Coordinates

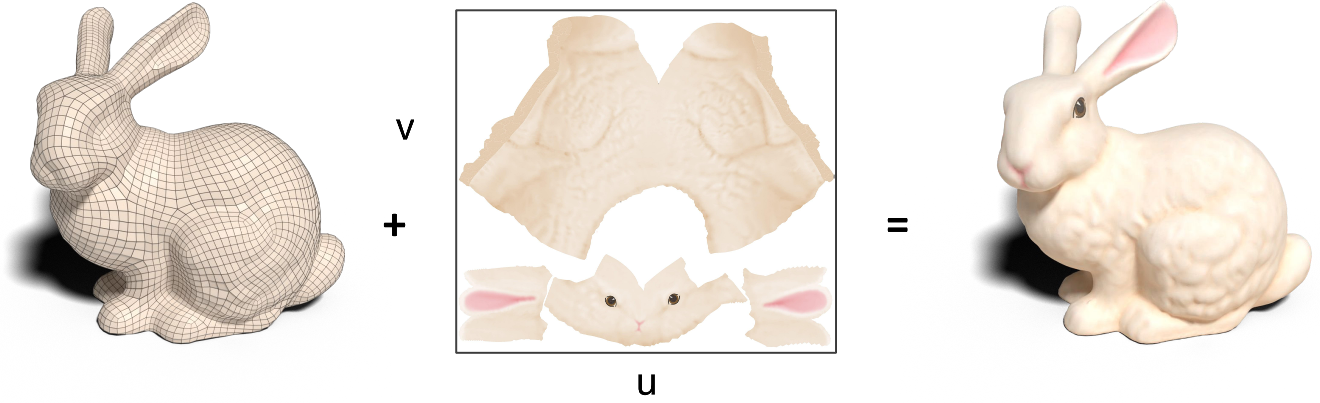
Texture coordinates define a mapping from surface coordinates to a texture image space

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:

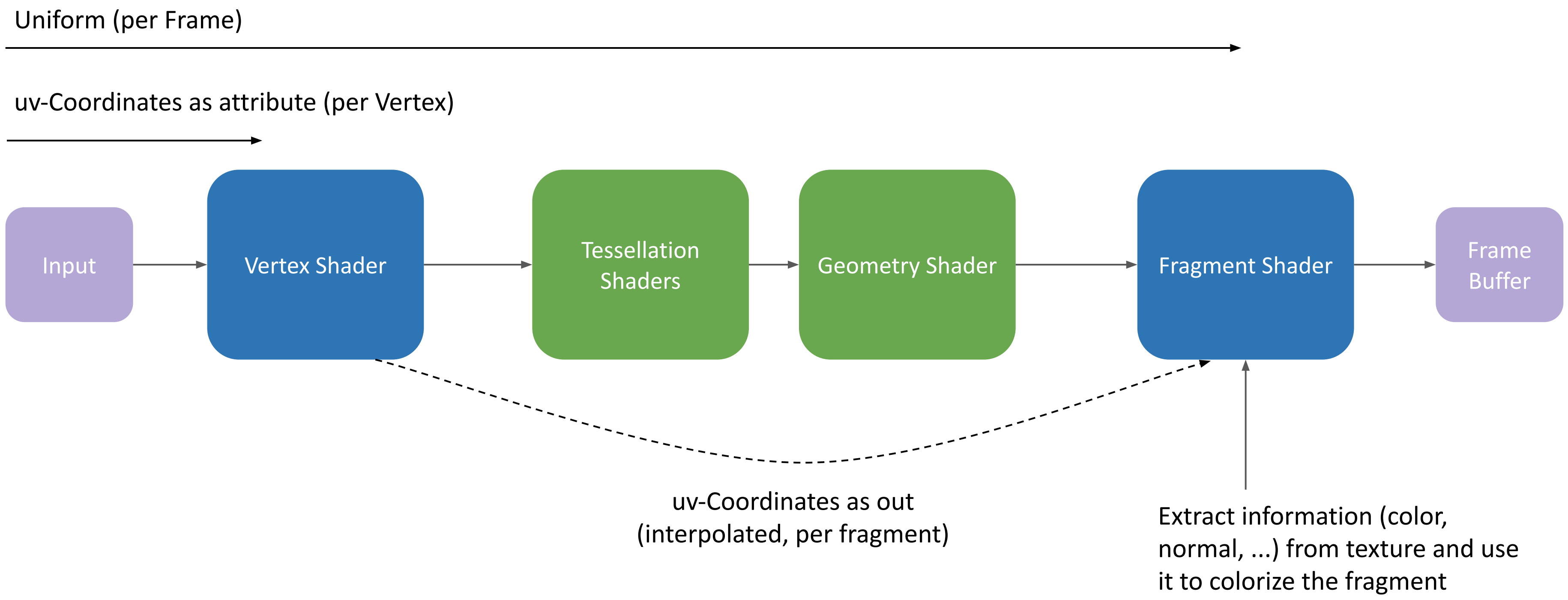
vertexshader

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



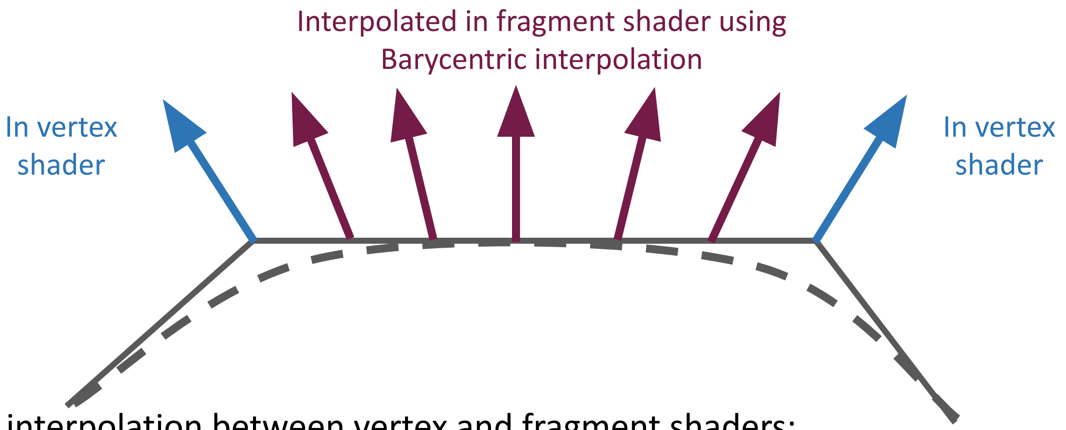
*We assume the uv coordinates are provided in a model's vertex since textures are often drawn manually by artists (very time consuming).

Graphics Pipeline (Revisited)



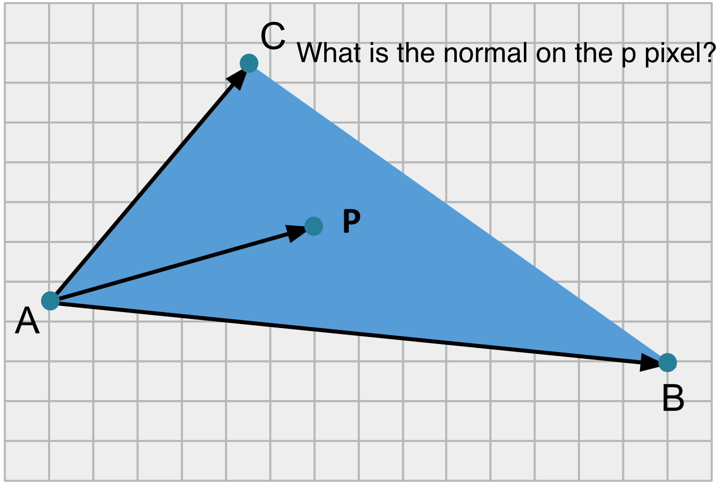
Interpolation Between Vertex and Fragment Shaders

Take the vertex normal as an example:



- Many attributes are interpolation between vertex and fragment shaders:
  - Colors and textures
  - UV coordinates and Normals
  - ...

Barycentric Interpolation



If P is inside the triangle, geometrically:

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

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

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

Let $w_1 = 1 - w_2 - w_3 \in [0,1]$ sum of all three = 1

We have: $P = w_1 A + w_2 B + w_3 C$

or normals, uv, colors

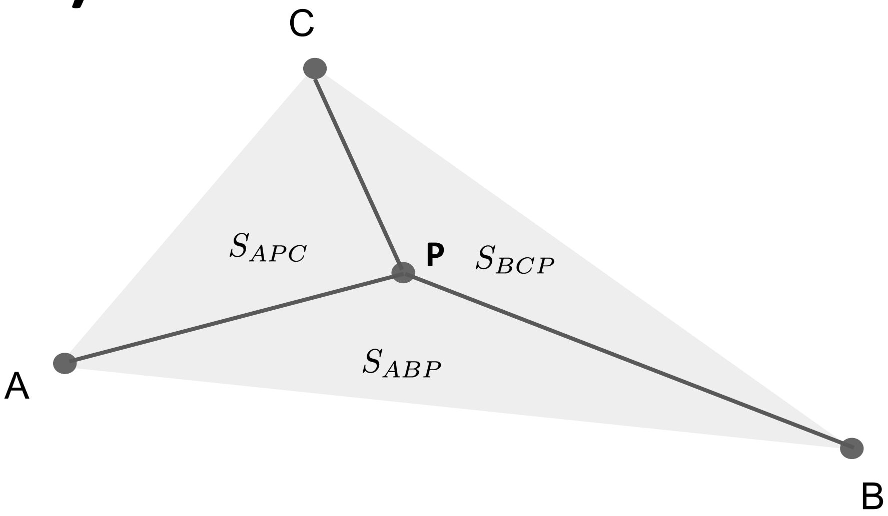
This is how we interpolate the vertex attributes for P given the color of A, B, and C. For example, color:

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

But what are w_1, w_2, w_3?

Barycentric Interpolation (cont.)

immer fläche gegenüber/ total fläche



Because:

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

We can write this linear equations:

$$\vec{AP}_x = w_2\vec{AB}_x + w_3\vec{AC}_x$$

$$\vec{AP}_y = w_2\vec{AB}_y + w_3\vec{AC}_y$$

$$w_1 + w_2 + w_3 = 1$$

Solving them, we have:

$$\Rightarrow \quad w_3 = \frac{\vec{AP}_x\vec{AB}_y - \vec{AP}_y\vec{AB}_x}{\vec{AC}_x\vec{AB}_y - \vec{AC}_y\vec{AB}_x} = \frac{\vec{AP} \times \vec{AB}}{\vec{AC} \times \vec{AB}} = \frac{S_{ABP}}{S_{ABC}} \quad w_2 = \frac{S_{APC}}{S_{ABC}} \quad w_1 = \frac{S_{BCP}}{S_{ABC}}$$

This is how we compute the barycentric coordinates.

Example: Edge Cases

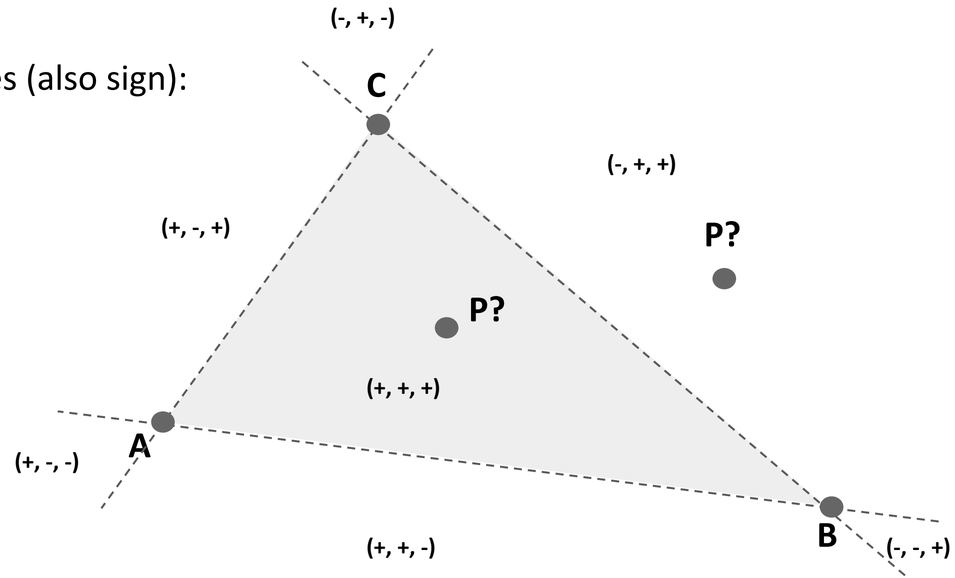
Meaning when barycentric coordinates have different values (also sign):

If $w_1 = 1, w_2 = w_3 = 0$ ⇒ P = A

If $w_2 = 1, w_1 = w_3 = 0$ ⇒ P = B

If $w_3 = 1, w_1 = w_2 = 0$ ⇒ 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

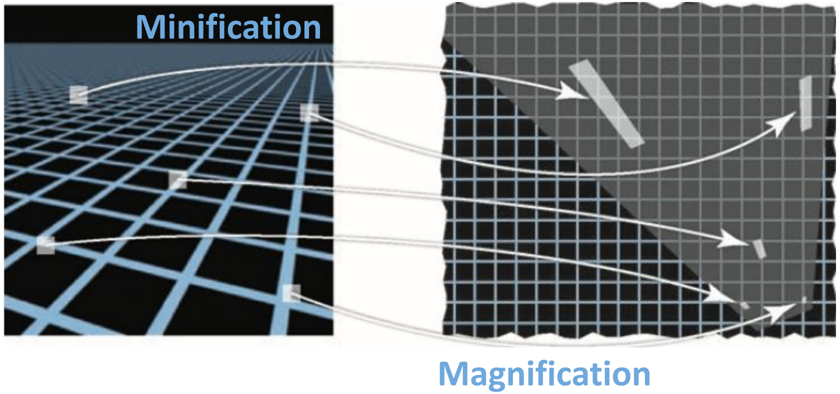
Aside: We just found a new approach to replace the point-in-triangle assertion.

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**

get mean, max_light, min_light

The camera projection causes the problem,
world coordniates to texture space is different (get disorted)



More Interpolation(s)

Linear interpolation between numbers:

$$\mathrm{lerp}(x, y, t) = x + t(y - x)$$

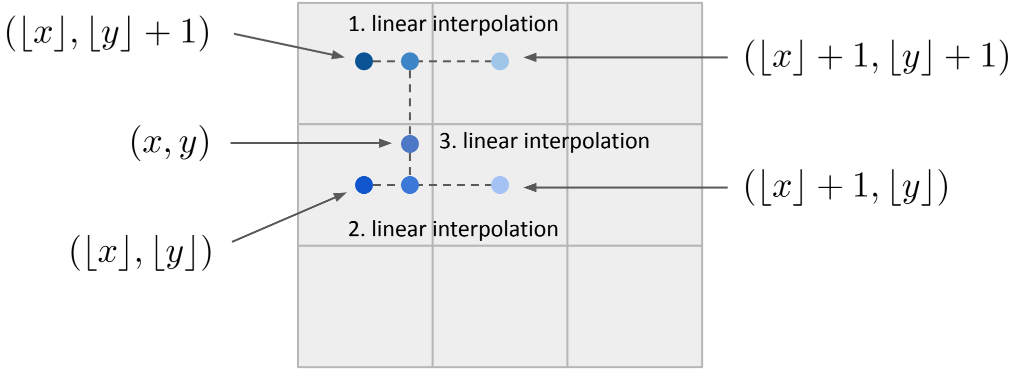
Linear interpolation between positions:

$$\mathrm{lerp}(\mathbf{u}, \mathbf{v}, t) = \mathbf{u} + t(\mathbf{v} - \mathbf{u})$$

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

Bilinear interpolation is just a 3x linear interpolation:

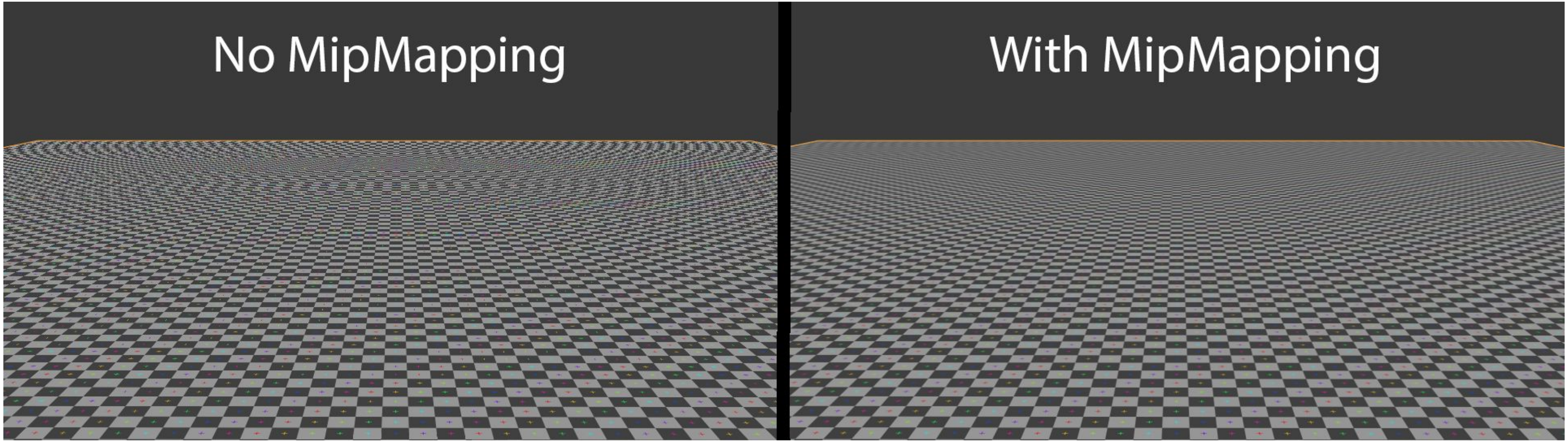
linear interpolation of surrounding 4 pixels



MIP Map

used for minification

- (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

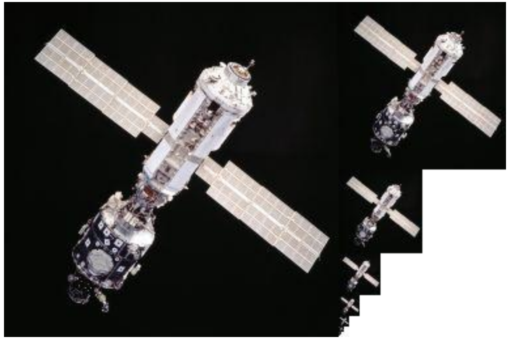


https://en.wikipedia.org/wiki/Mipmap#/media/File:Mipmap_Aliasing_Comparison.png

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: *⅓ more storage*

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



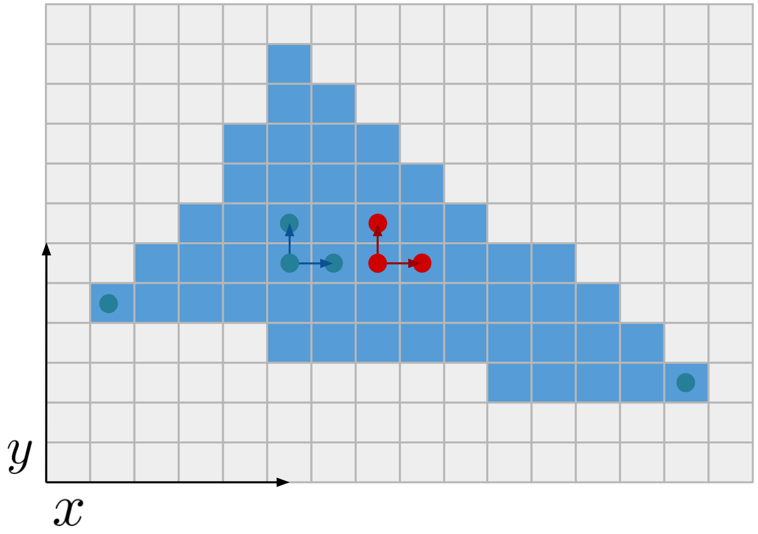
https://en.wikipedia.org/wiki/Mipmap#/media/File:MipMap_Example_STS101.jpg

Paul S. Heckbert, ''Texture Mapping Polygons in Perspective'', Technical Memo No. 13, NYIT. Computer Graphics Lab, April 1983.
Lance Williams. 1983. Pyramidal parametrics. In Proceedings of the 10th annual conference on Computer graphics and interactive techniques (SIGGRAPH ’83). Association for Computing Machinery, New York, NY, USA, 1–11. DOI:https://doi.org/10.1145/800059.801126

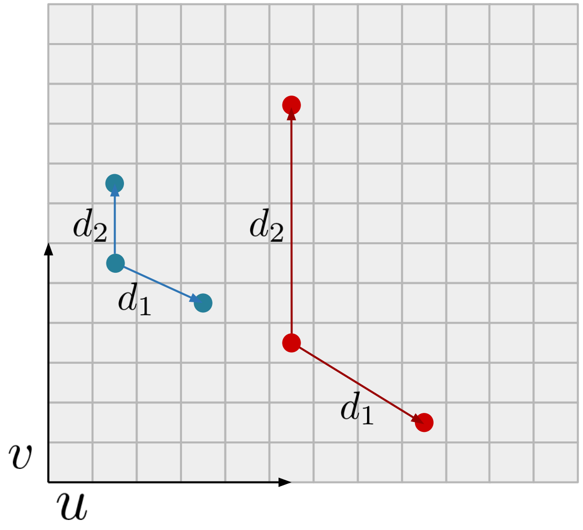
Computing MIP Map Level

- 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

red area is further away, higher level of mip map



screen space

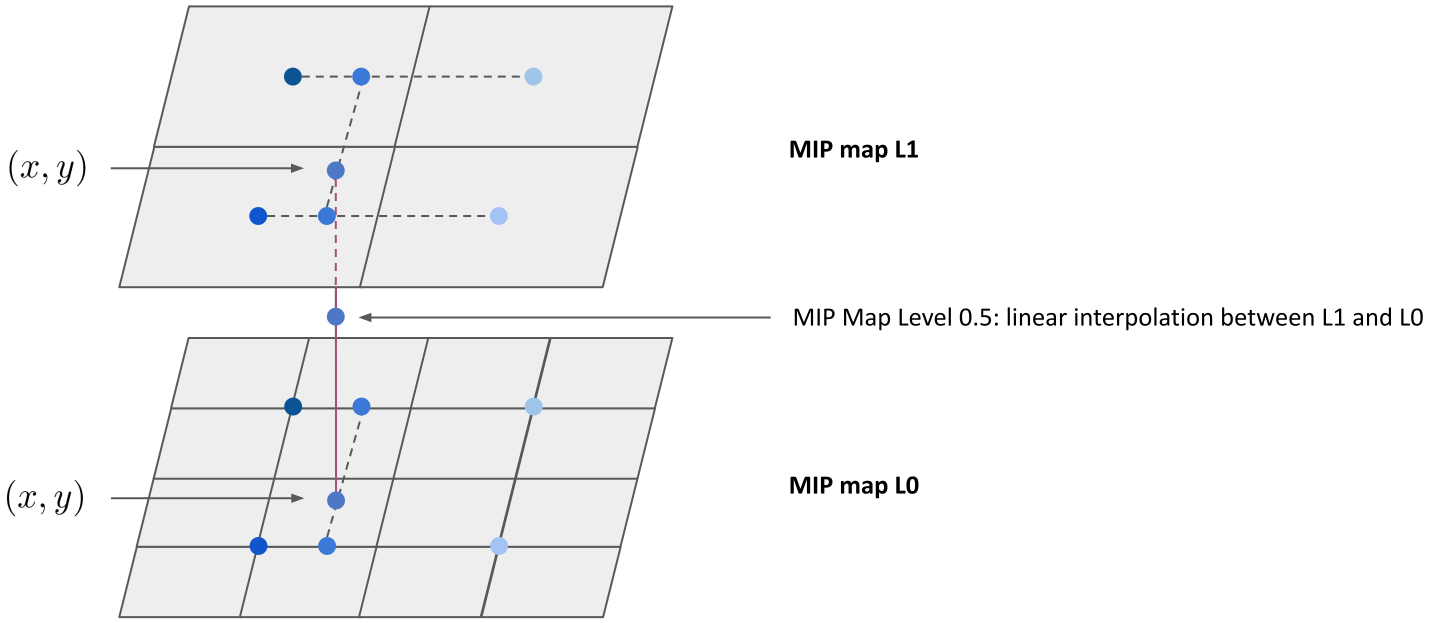


texture space

Trilinear Interpolation

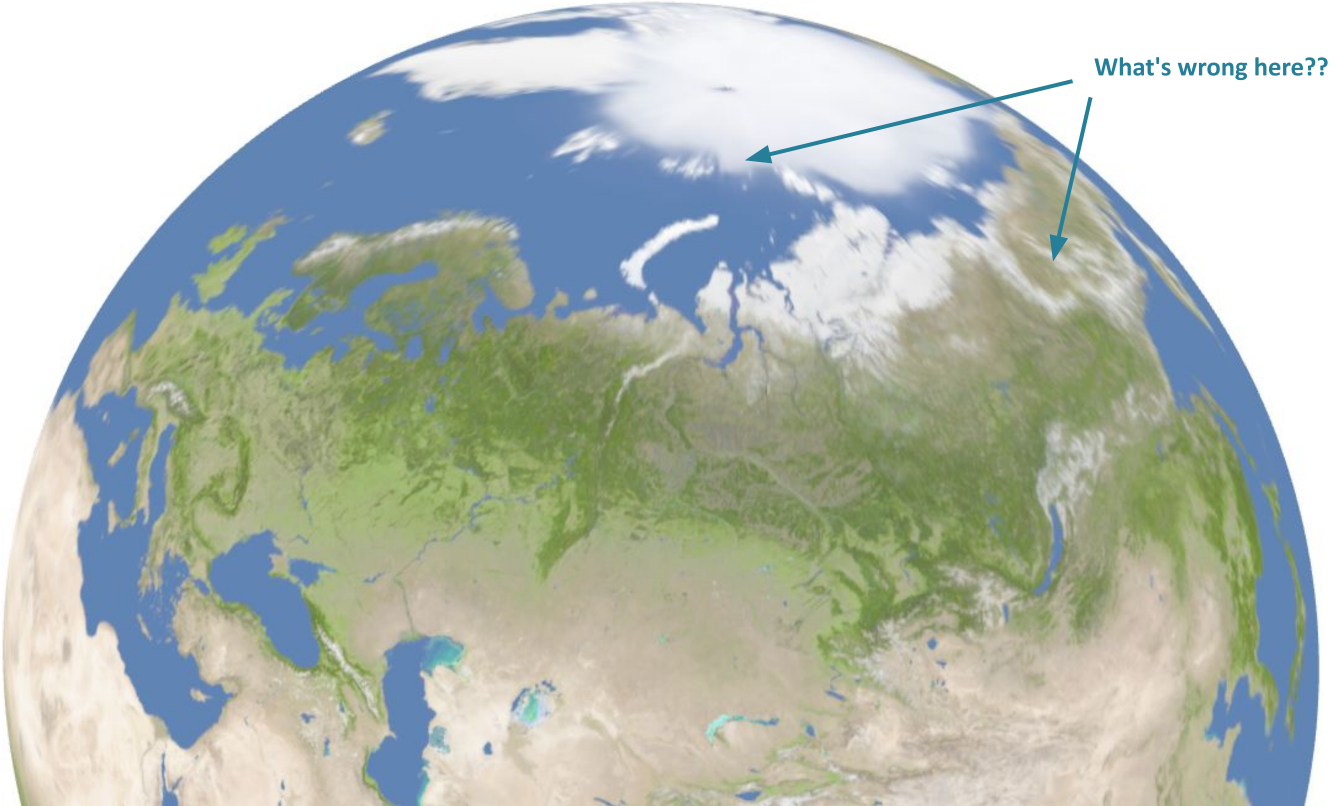
If the estimated MIP level is not an integer, we need to use *trilinear interpolation* between two discrete MIP map levels.

Trilinear interpolation is just one more linear interpolation between two bilinear interpolations.



Limitation of Isotropic Mipmap

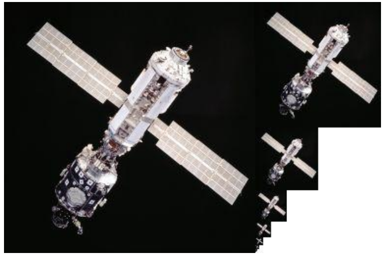
The texture is blurred!



Anisotropic Filtering

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

Isotropic
Mipmap



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

Anisotropic
(Filtering)
Mipmap

https://en.wikipedia.org/wiki/Anisotropic_filtering#/media/File:MipMap_Example_STS101_Anisotropic.png

J. McCormack, R. Perry, K. Farkas, and N. Jouppi, “Feline: Fast Elliptical Lines for Anisotropic Texture Mapping,” *SIGGRAPH 1999*.

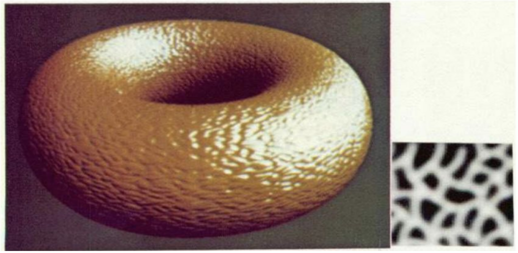
Example: Using Anisotropic Filtering

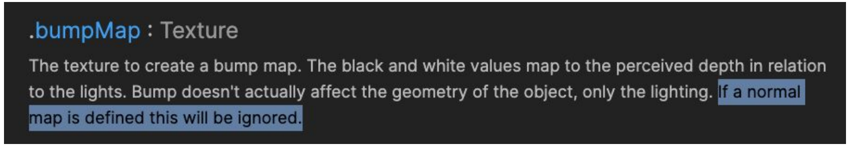
Tutorial 6: Texture

- 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
  - …

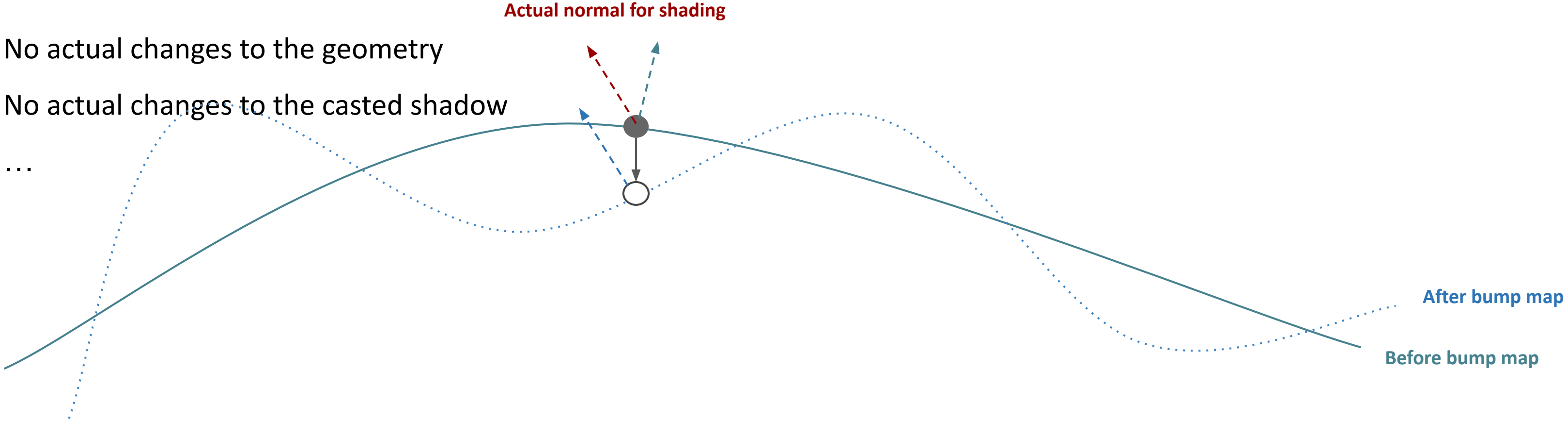




.bumpMap : Texture

The texture to create a bump map. The black and white values map to the perceived depth in relation to the lights. Bump doesn't actually affect the geometry of the object, only the lighting. If a normal map is defined this will be ignored.

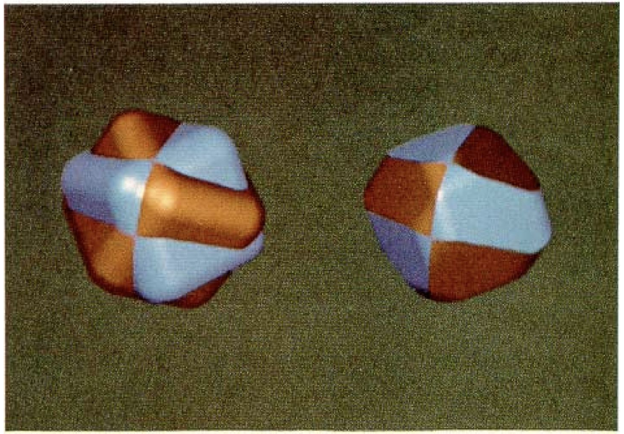
https://threejs.org/docs/#api/en/materials/MeshPhongMaterial.bumpMap

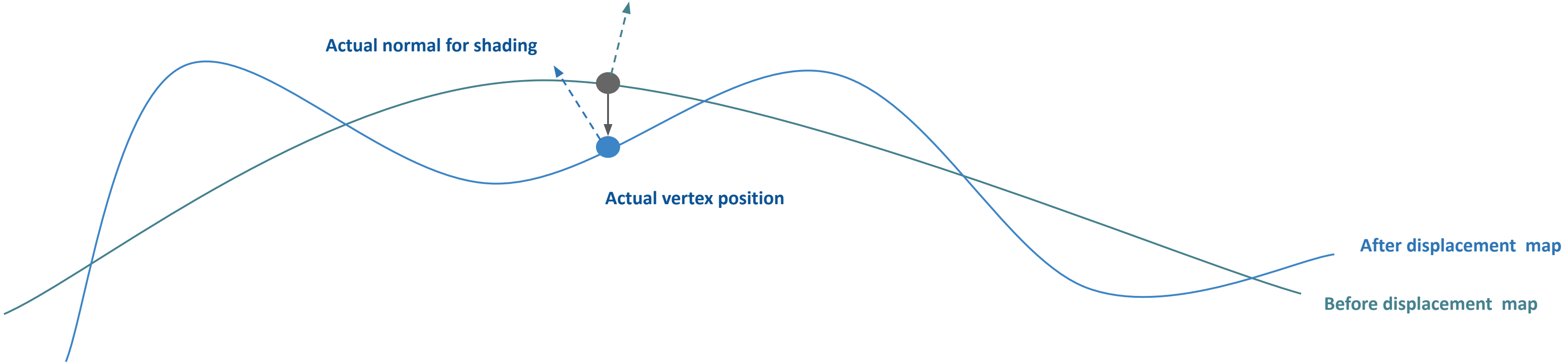


James F. Blinn. 1978. Simulation of wrinkled surfaces. In Proceedings of the 5th annual conference on Computer graphics and interactive techniques (SIGGRAPH '78). Association for Computing Machinery, New York, NY, USA, 286–292. DOI:https://doi.org/10.1145/800248.507101

Displacement Map

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





Robert L. Cook. 1984. Shade trees. In Proceedings of the 11th annual conference on Computer graphics and interactive techniques (SIGGRAPH '84). Association for Computing Machinery, New York, NY, USA, 223–231. DOI:https://doi.org/10.1145/800031.808602

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
- …

pixel buffer, that stores information

James F. Blinn and Martin E. Newell. 1976. Texture and reflection in computer generated images. Commun. ACM 19, 10 (Oct. 1976), 542–547. DOI:https://doi.org/10.1145/360349.360353

Breakout 1: Earth

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

Summary

- We discussed:
  - 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