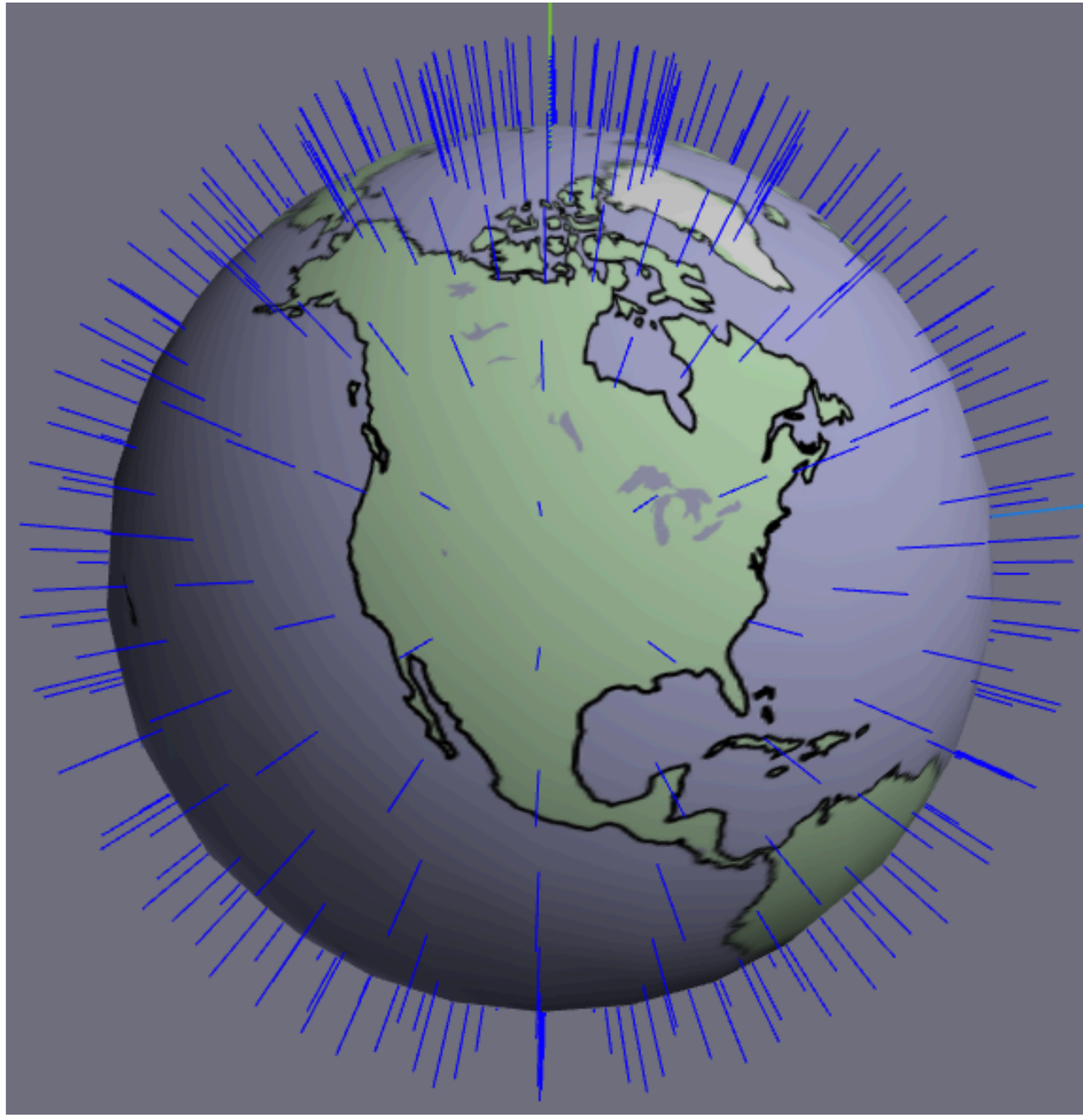
Computer Graphics



Lecture 4

Triangle Meshes: Texture Coordinates

Announcements

Announcements

- Slides, notes, demo files, etc are posted on the course webpage.

Announcements

- Slides, notes, demo files, etc are posted on the course webpage.

- Also readings

Announcements

- Slides, notes, demo files, etc are posted on the course webpage.

- Also readings

- A1 - how's it going?

Announcements

- Slides, notes, demo files, etc are posted on the course webpage.

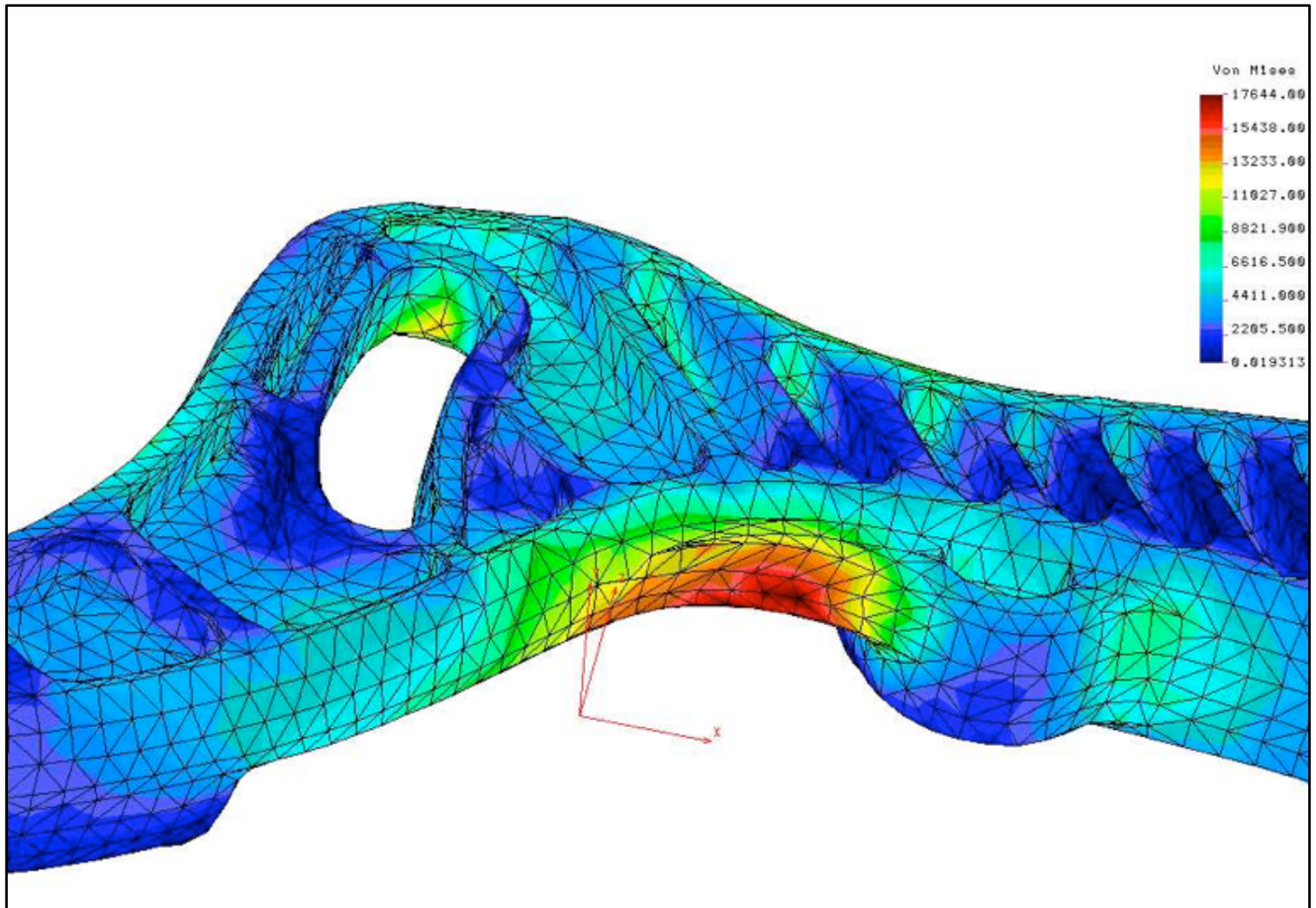
- Also readings

- A1 - how's it going?

- Office hours 12-1 today to replace yesterday's.

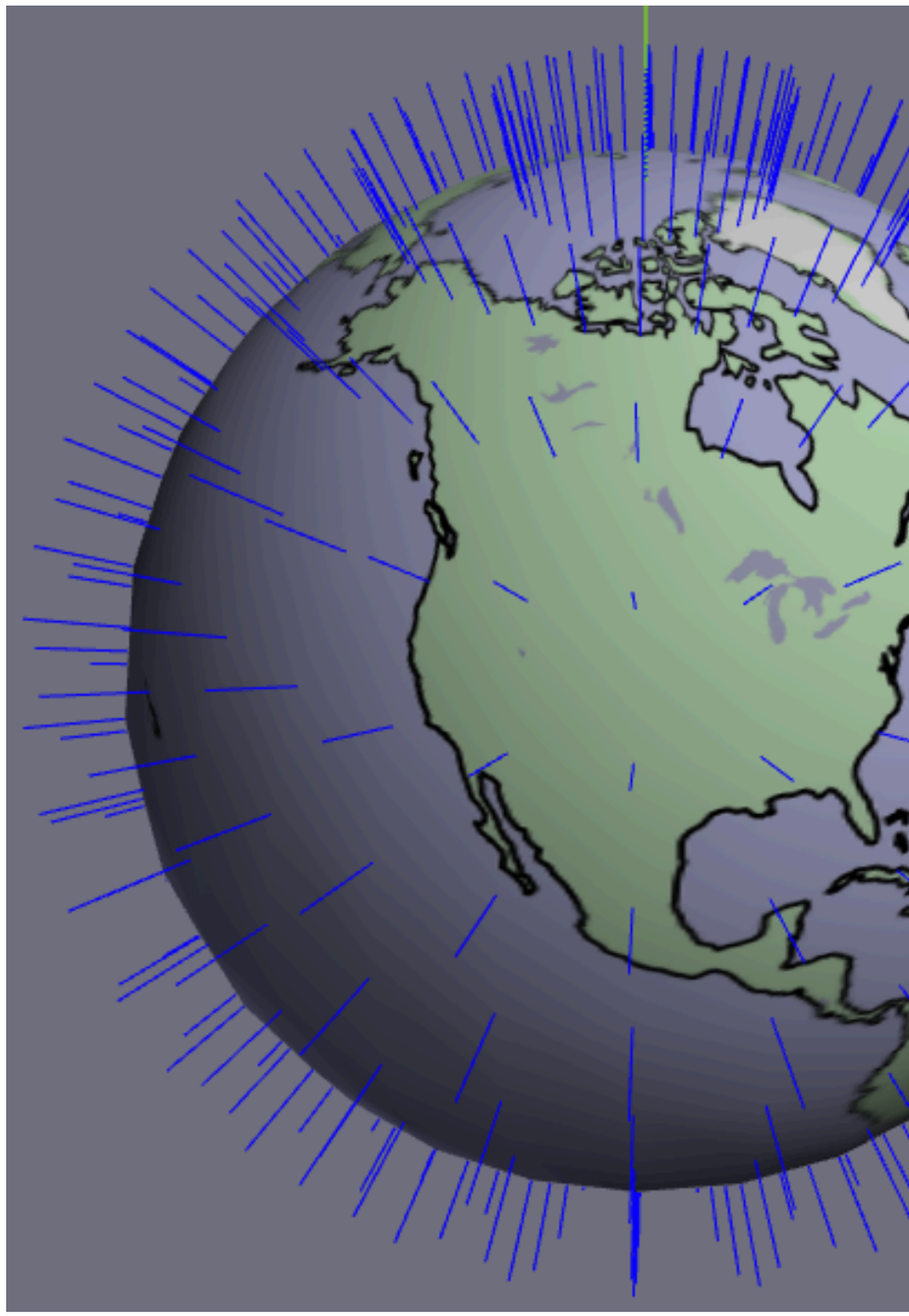
Last time: data on Meshes

- Often we need more than just geometry.
- Many properties vary continuously over a smooth surface.



Data on Meshes

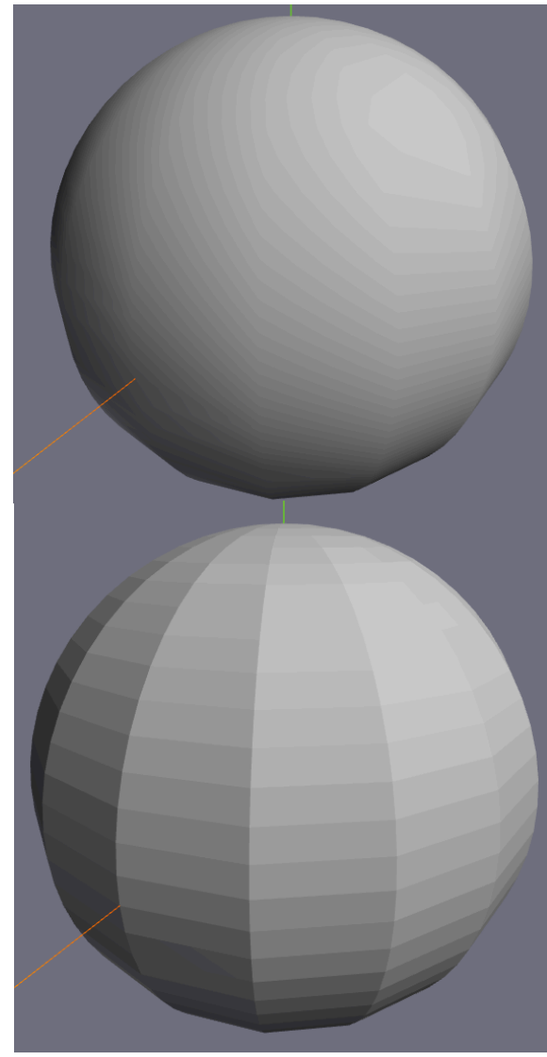
- What do we need to store at vertices?
  - **Positions**
    just another piece of per-vertex data
  - **Surface Normals** (last time)
    to more accurately portray geometry
  - **Texture Coordinates** (today)
    to paste image data onto surfaces

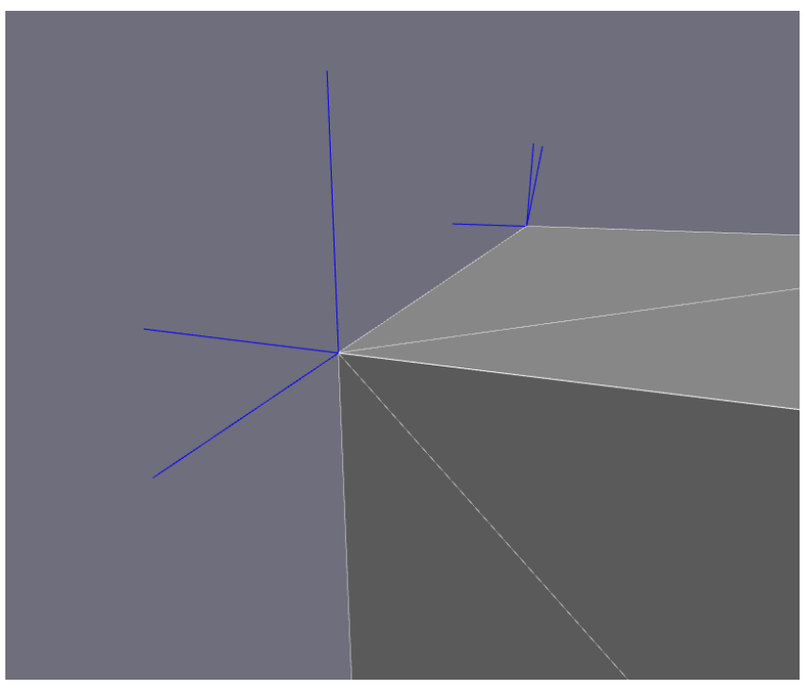


Surface Normals

- Most frequently we store data at vertices

- Last time: surface normals
  - allow for more accurate lighting of smooth geometry
  - smooth vs faceted: normal at a location depends on the triangle in question

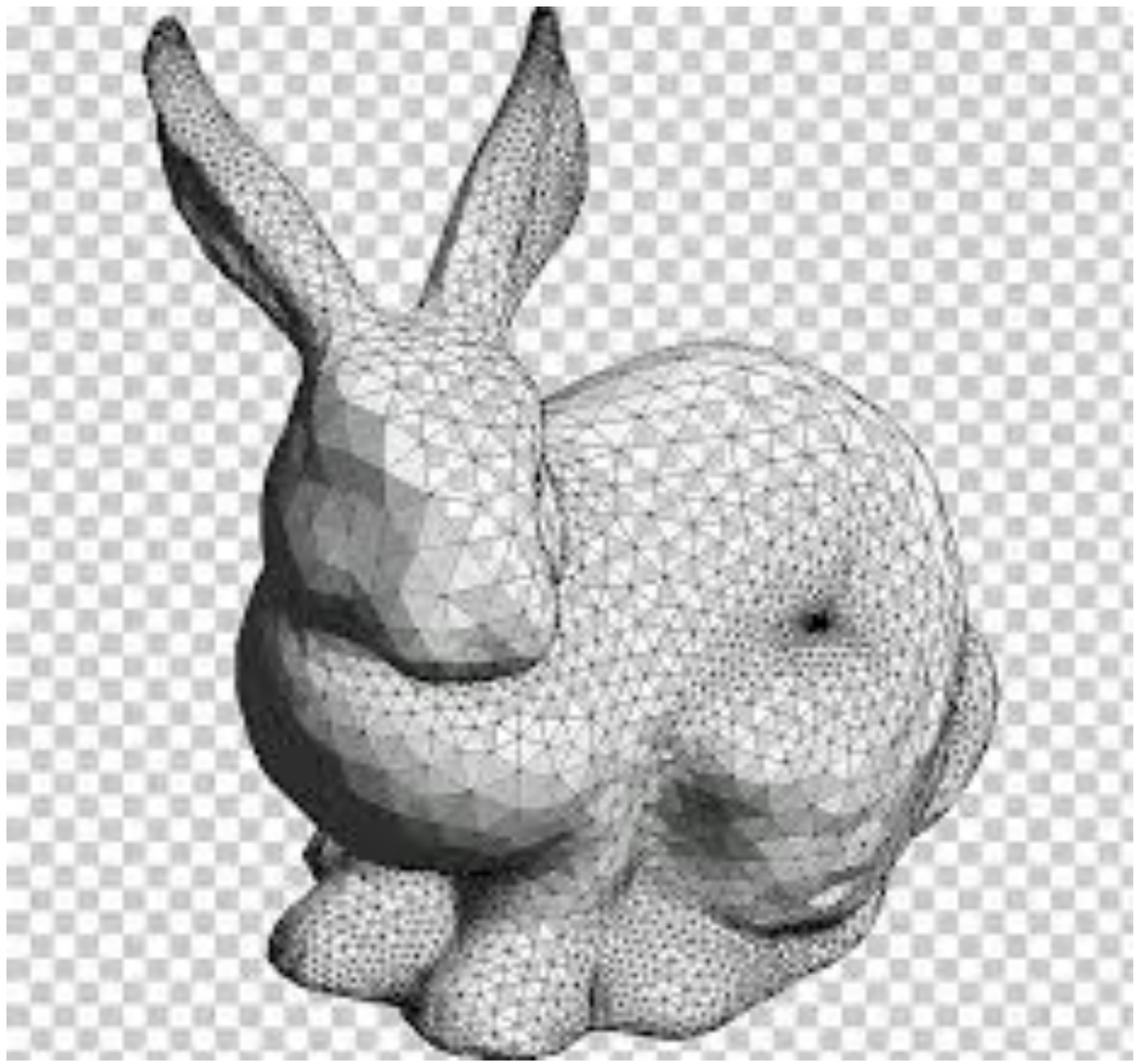




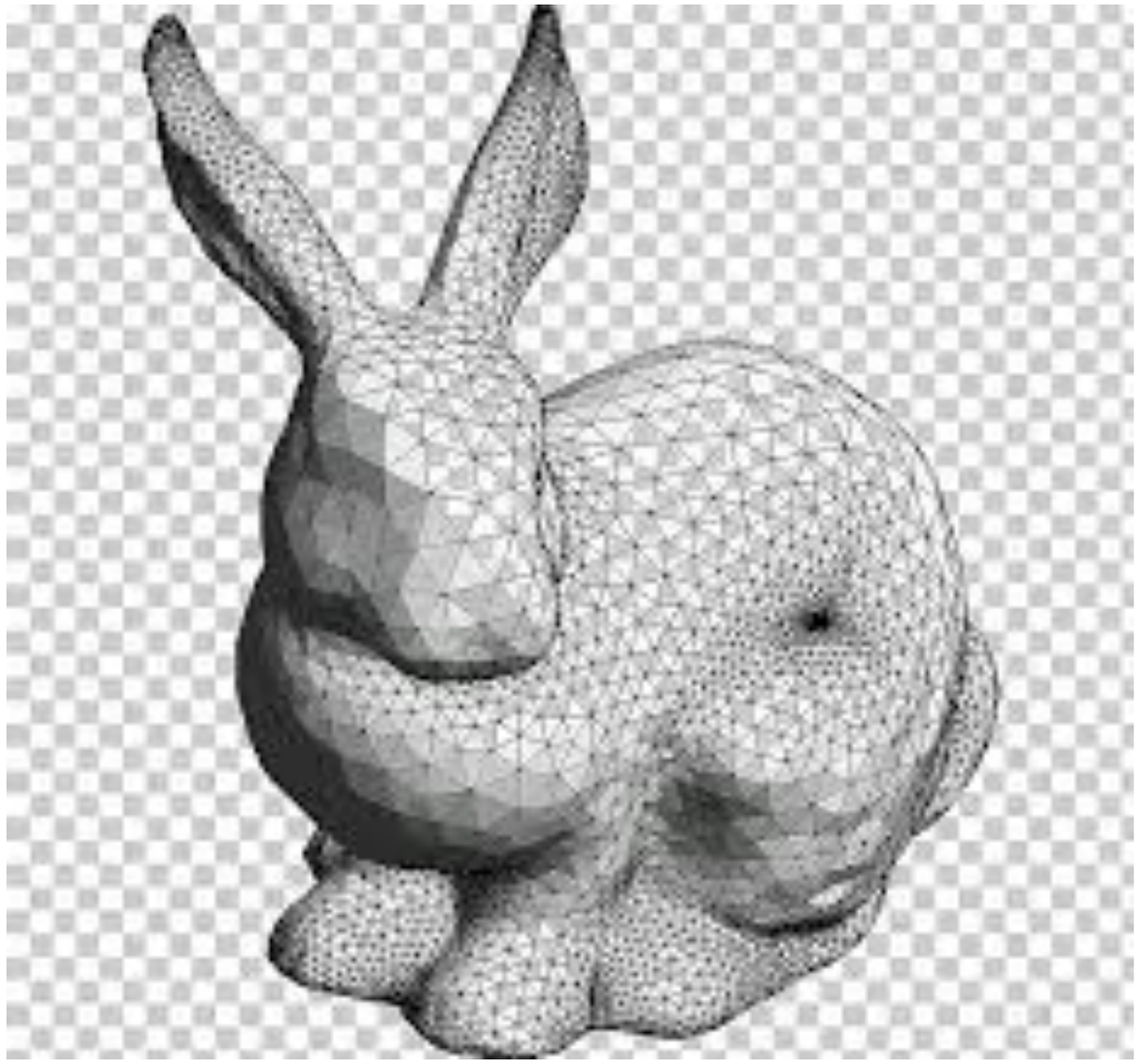
Estimating Surface Normals

- In shapes like a sphere and a cube, the normal is easy to calculate.

- What if the "true" surface isn't known?

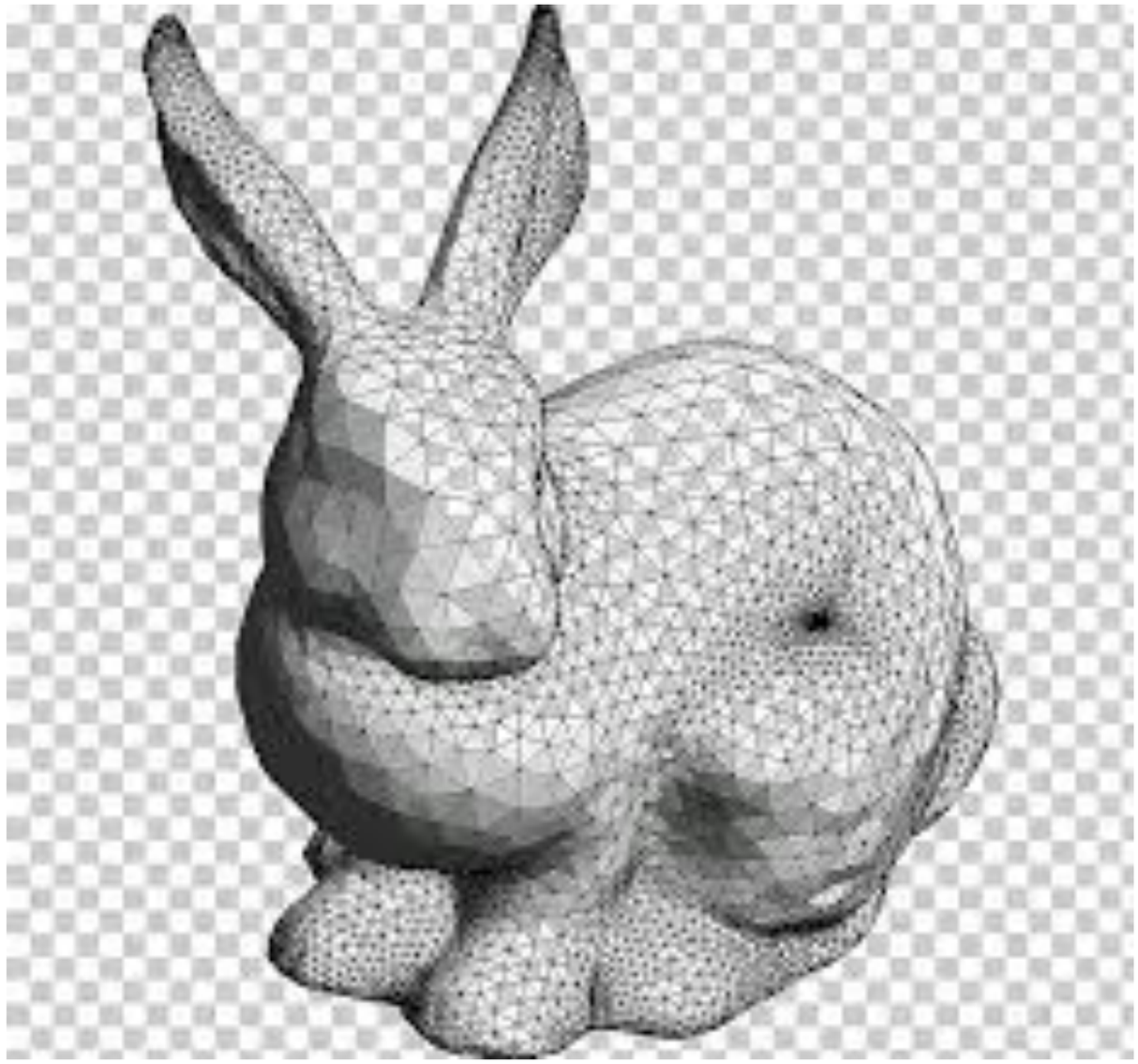


Estimating Surface Normals



Estimating Surface Normals

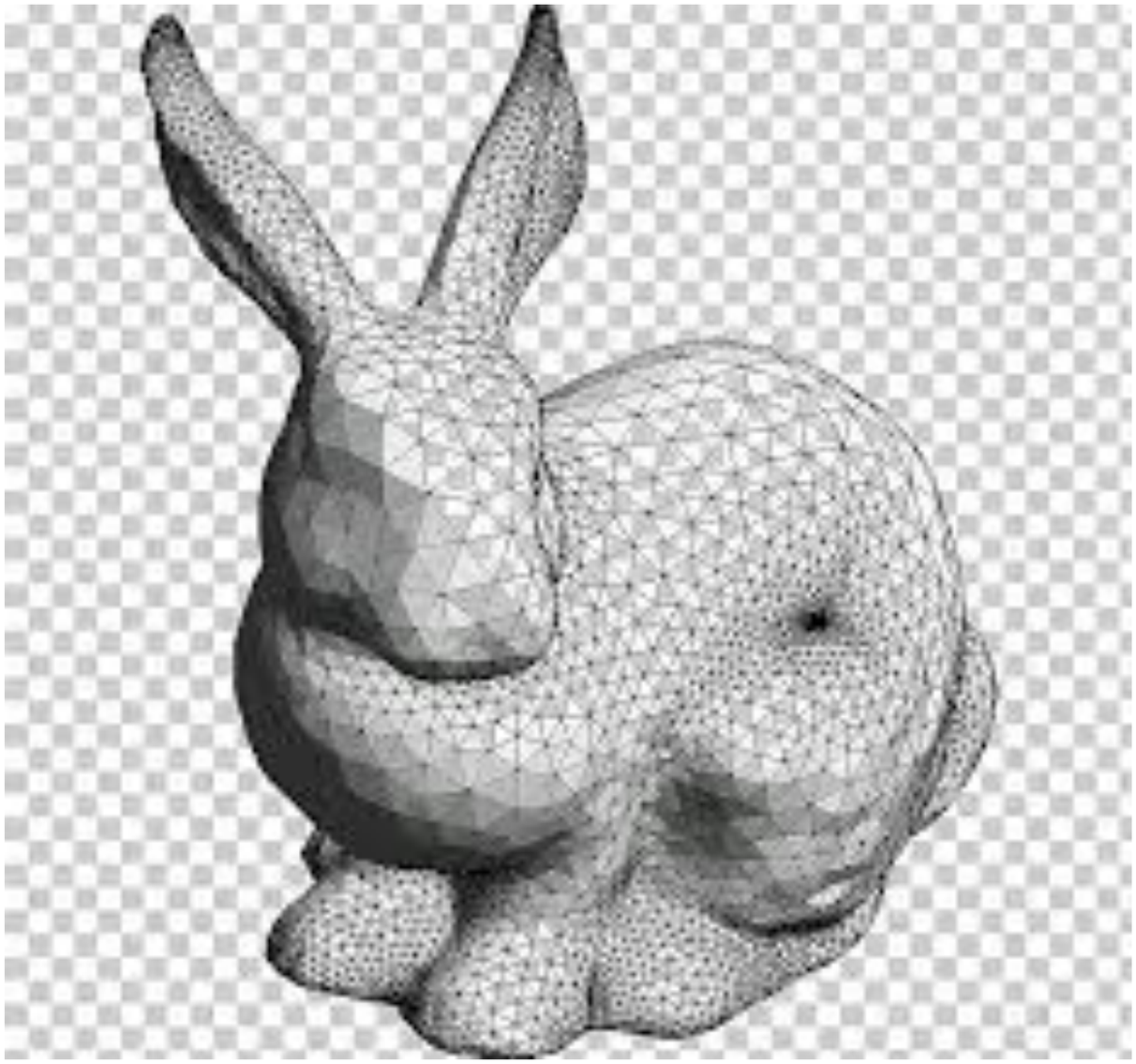
Faceted Objects



Estimating Surface Normals

Faceted Objects

Triangles represent geometry exactly.

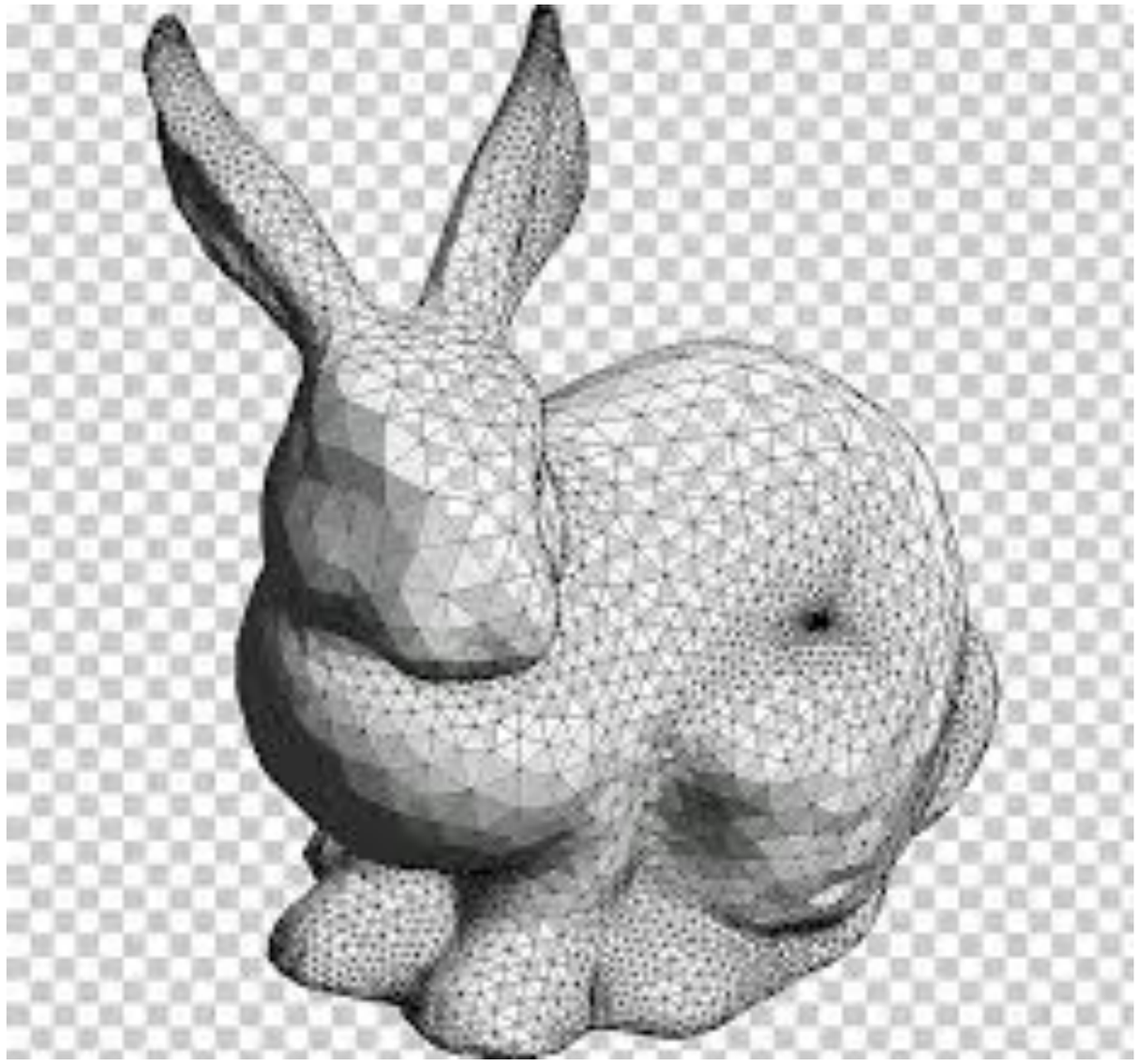


Estimating Surface Normals

Faceted Objects

Triangles represent geometry exactly.

Like the cube, all normals are normal to their triangles.



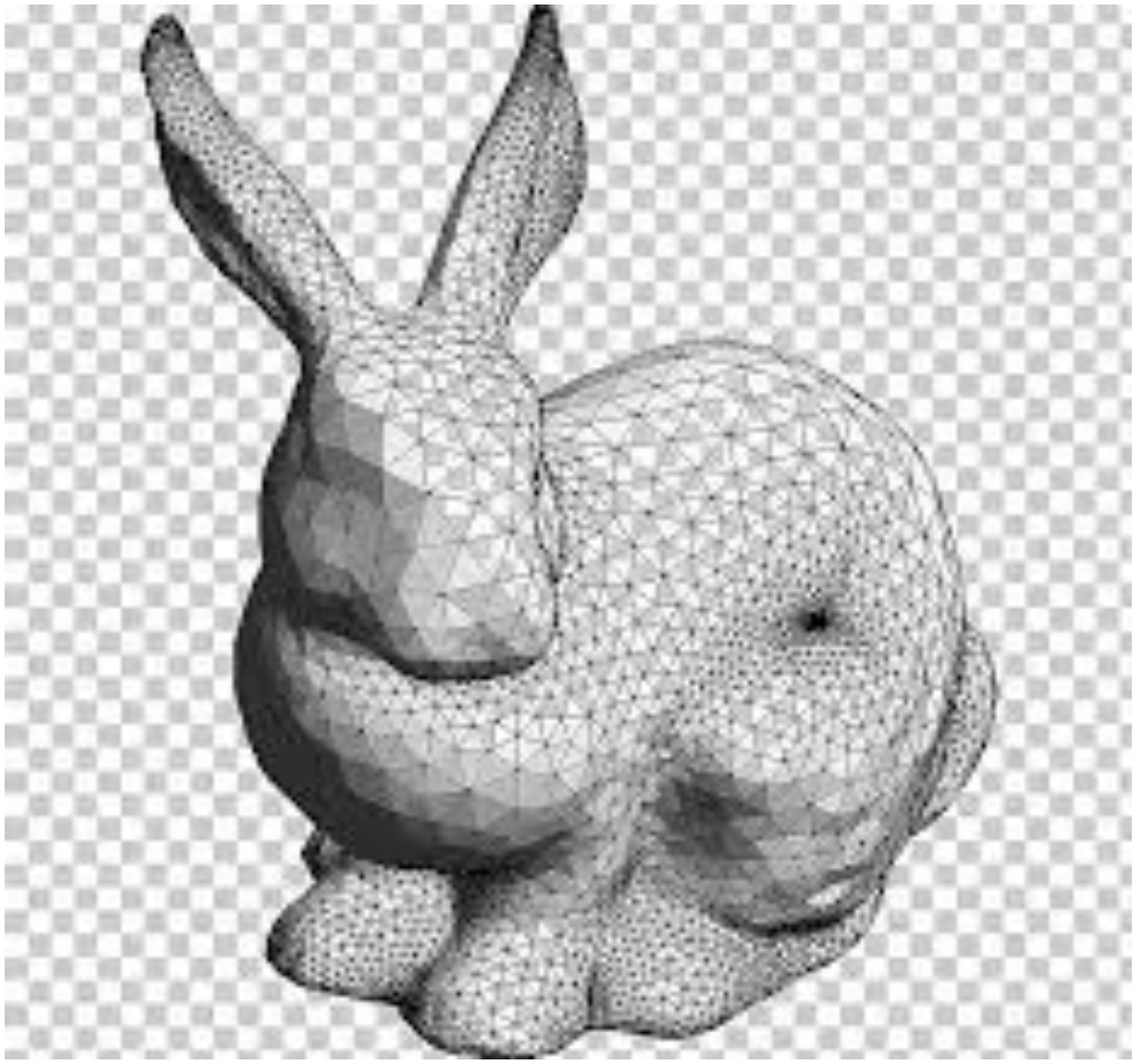
Estimating Surface Normals

Faceted Objects

Triangles represent geometry exactly.

Like the cube, all normals are normal to their triangles.

Smooth Objects



Estimating Surface Normals

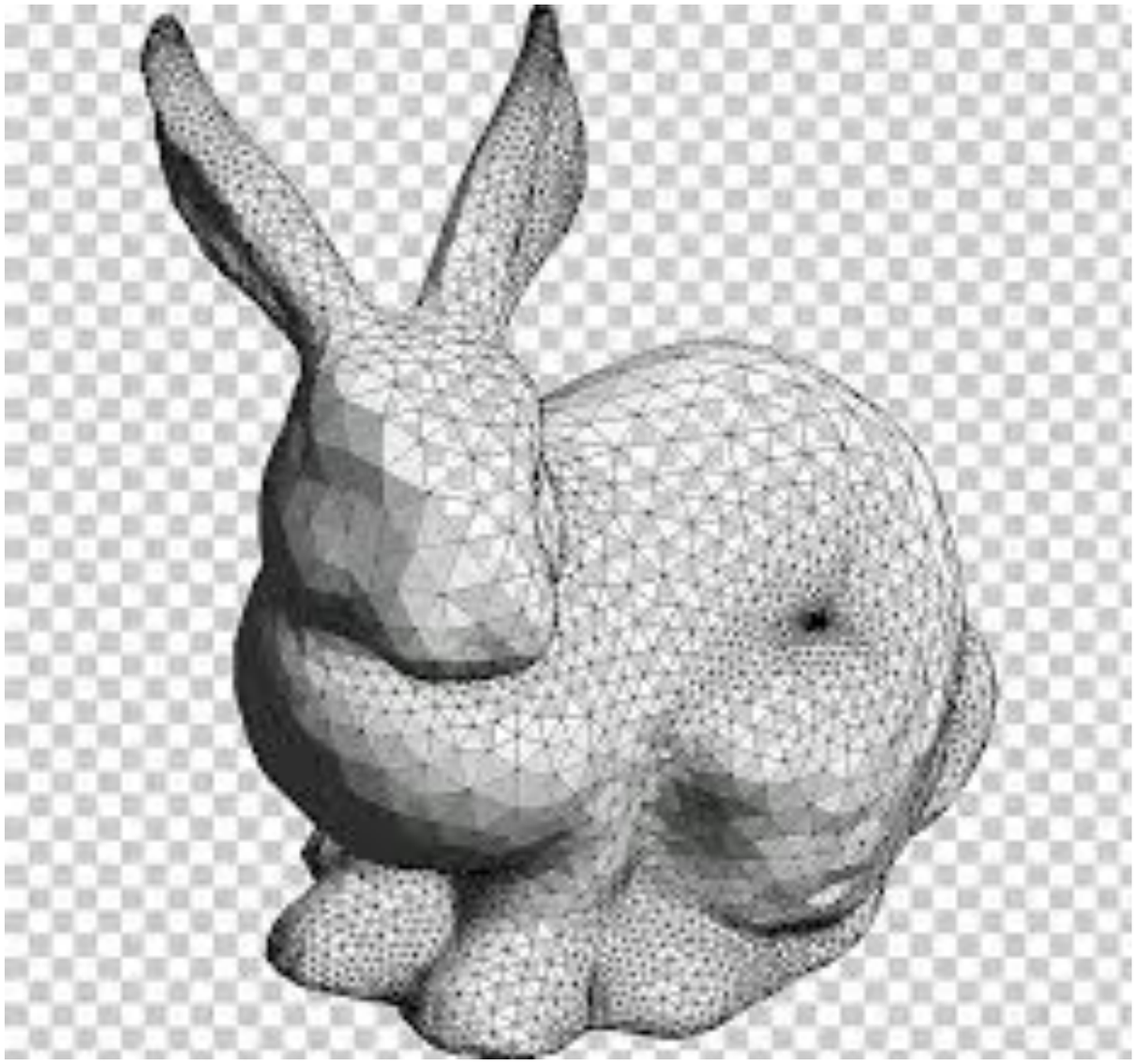
Faceted Objects

Triangles represent geometry exactly.

Like the cube, all normals are normal to their triangles.

Smooth Objects

Triangles approximate smooth geometry.



Estimating Surface Normals

Faceted Objects

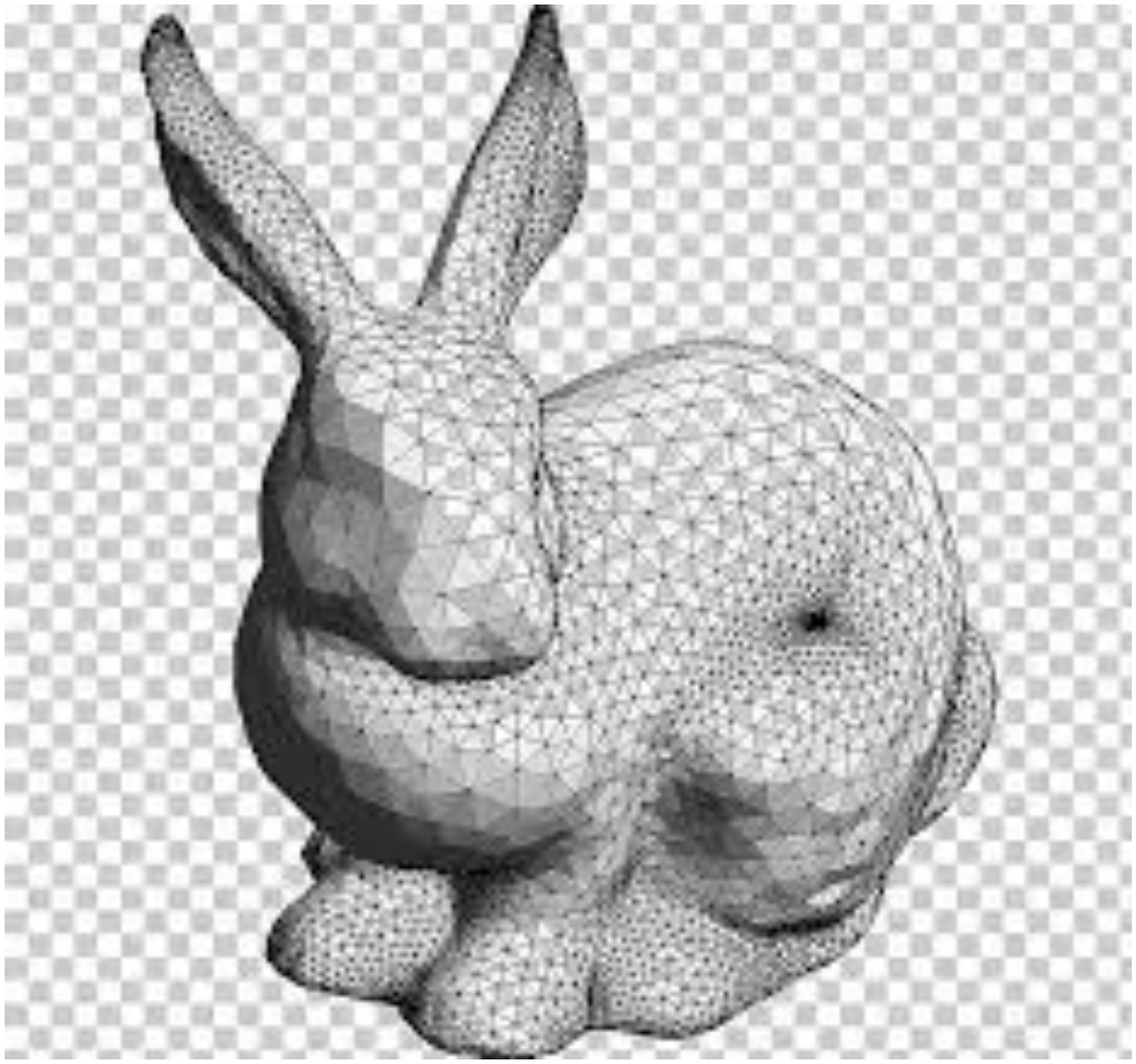
Triangles represent geometry exactly.

Like the cube, all normals are normal to their triangles.

Smooth Objects

Triangles approximate smooth geometry.

Vertex normal is the average of all surrounding triangle normals.



Estimating Surface Normals

Faceted Objects

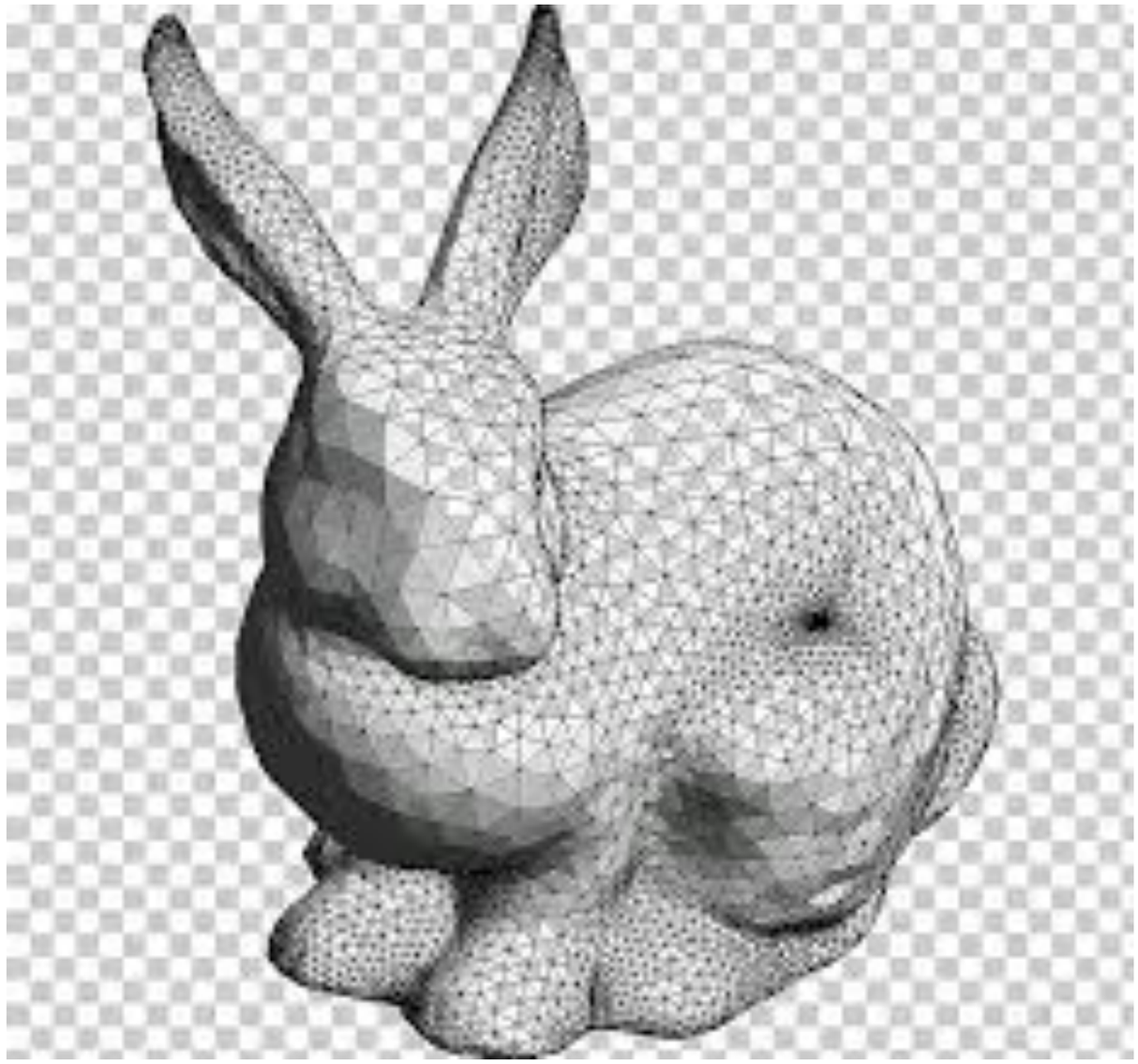
Triangles represent geometry exactly.

Like the cube, all normals are normal to their triangles.

Smooth Objects

Triangles approximate smooth geometry.

Vertex normal is the average of all surrounding triangle normals.



← This is the last part of A1.

Finding a vector normal to a triangle

Finding a vector normal to a triangle

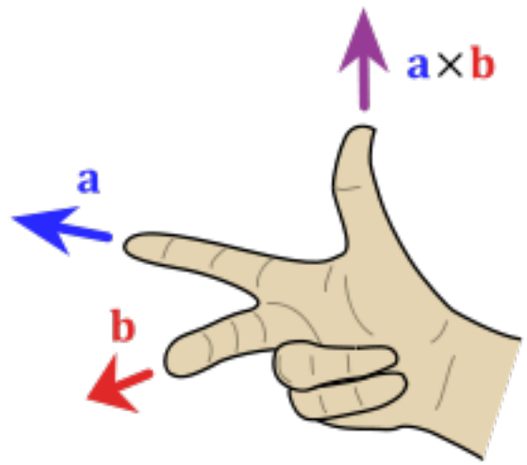
- What is a vector orthogonal to triangle (a, b, c)?

Finding a vector normal to a triangle

- What is a vector orthogonal to triangle (a, b, c)?
  - Hint 1: cross product

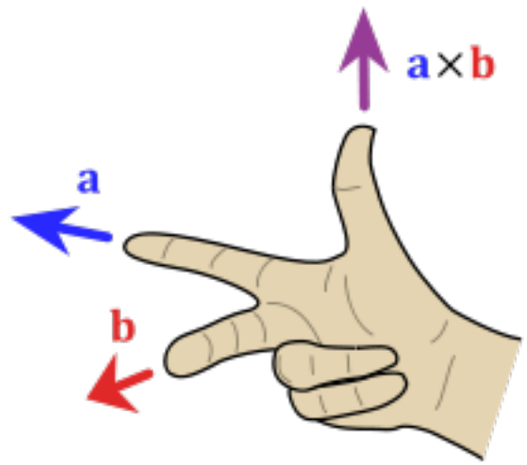
Finding a vector normal to a triangle

- What is a vector orthogonal to triangle (a, b, c)?
  - Hint 1: cross product
  - Hint 2: right-handed coordinate system



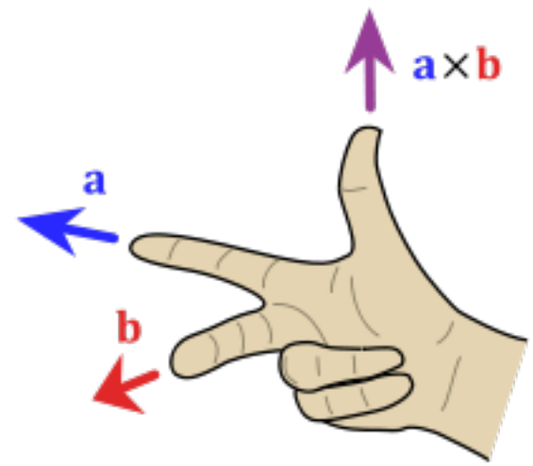
Finding a vector normal to a triangle

- What is a vector orthogonal to triangle (a, b, c)?
  - Hint 1: cross product
  - Hint 2: right-handed coordinate system
  - Hint 3: normal vectors are usually kept unit length



Finding a vector normal to a triangle

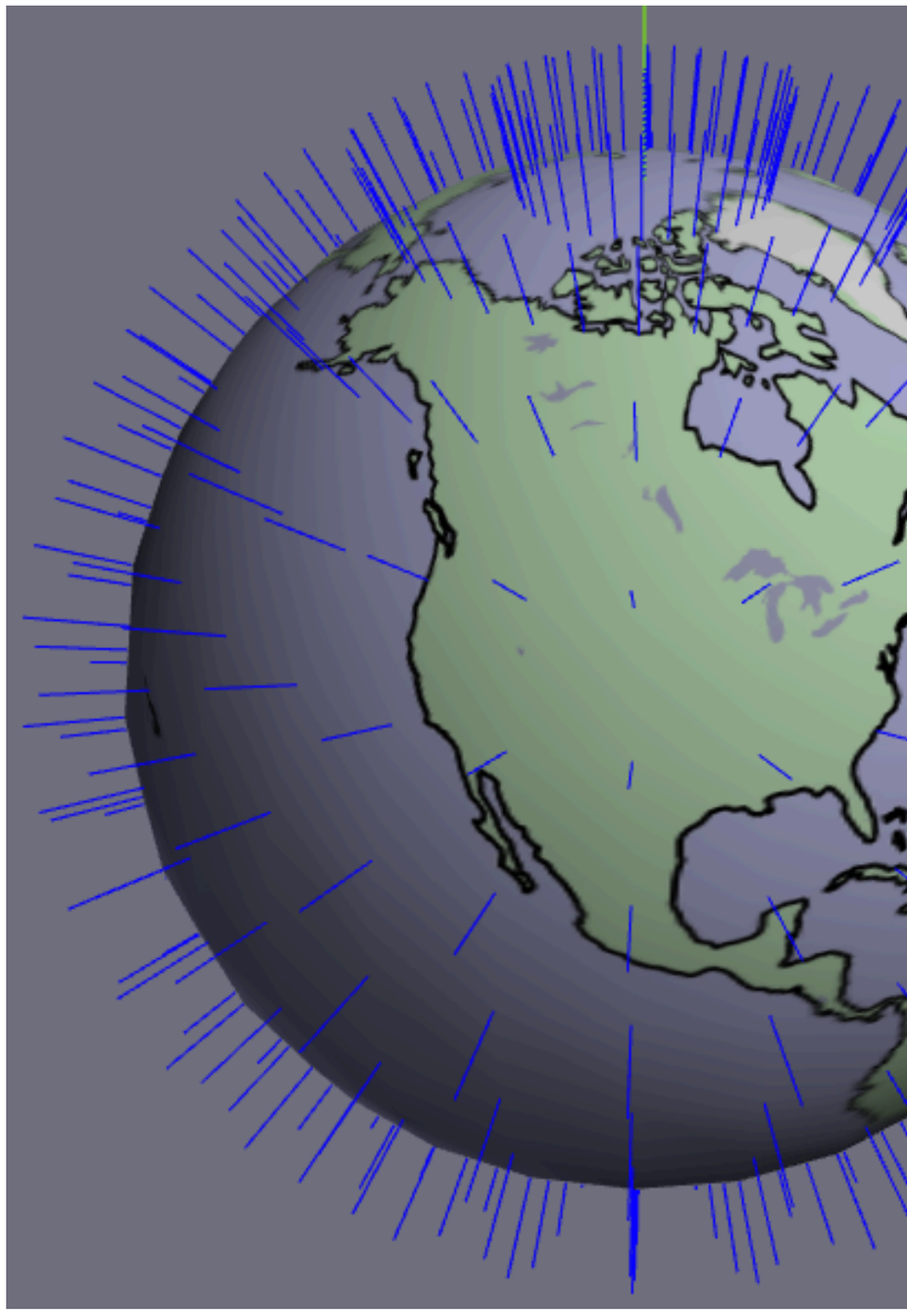
- What is a vector orthogonal to triangle (a, b, c)?
  - Hint 1: cross product
  - Hint 2: right-handed coordinate system
  - Hint 3: normal vectors are usually kept unit length
  - Hint 4 (for posterity): Section 2.7.2 in the book.



Questions?

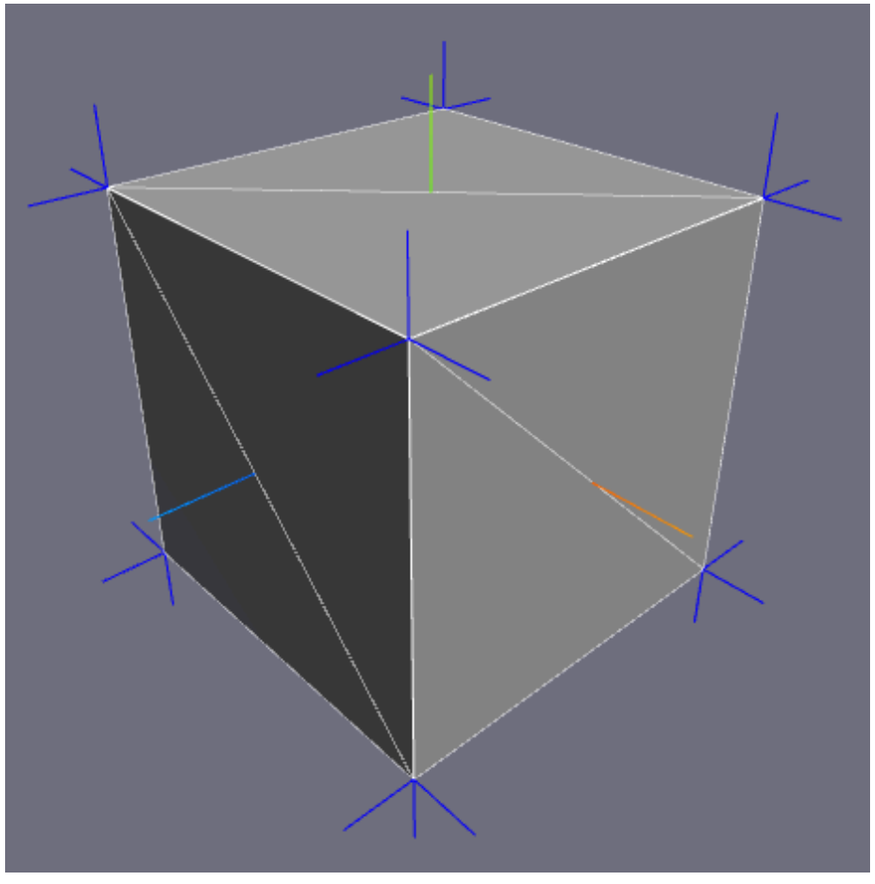
Data on Meshes

- What do we need to store at vertices?
  - **Surface Normals**
    to more accurately portray geometry
  - **Texture Coordinates**
    to paste image data onto surfaces
  - **Positions!?** (last lecture)
    just another piece of per-vertex data!

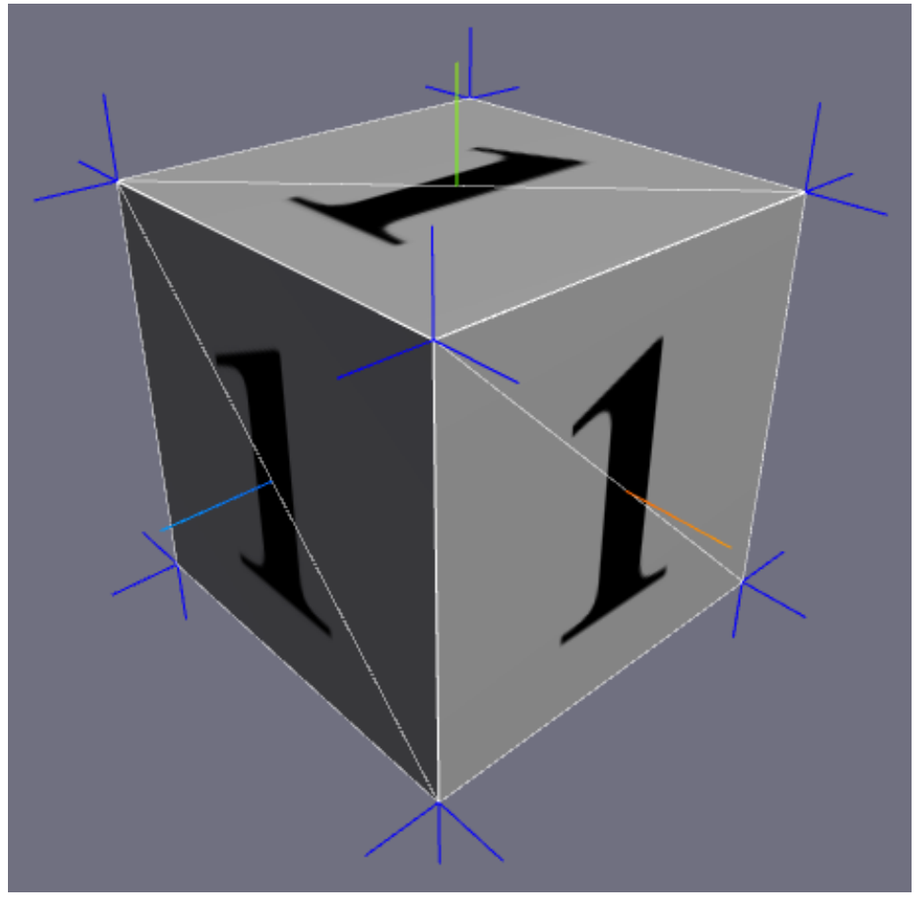


Textures

You are here:

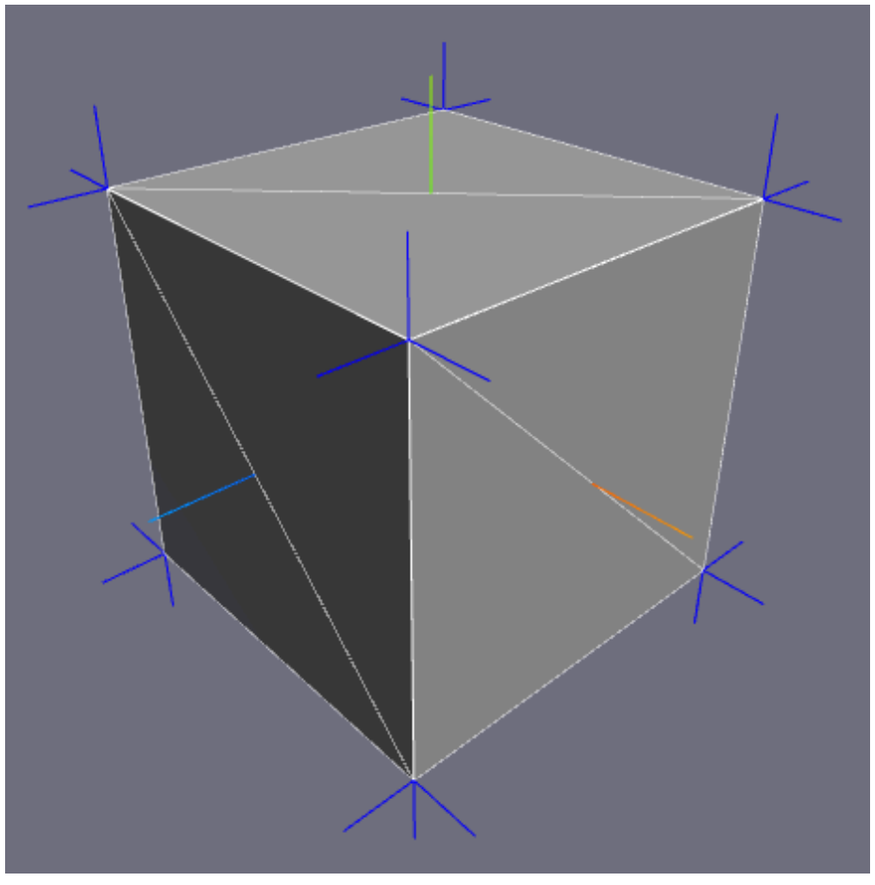


You wish to be here:

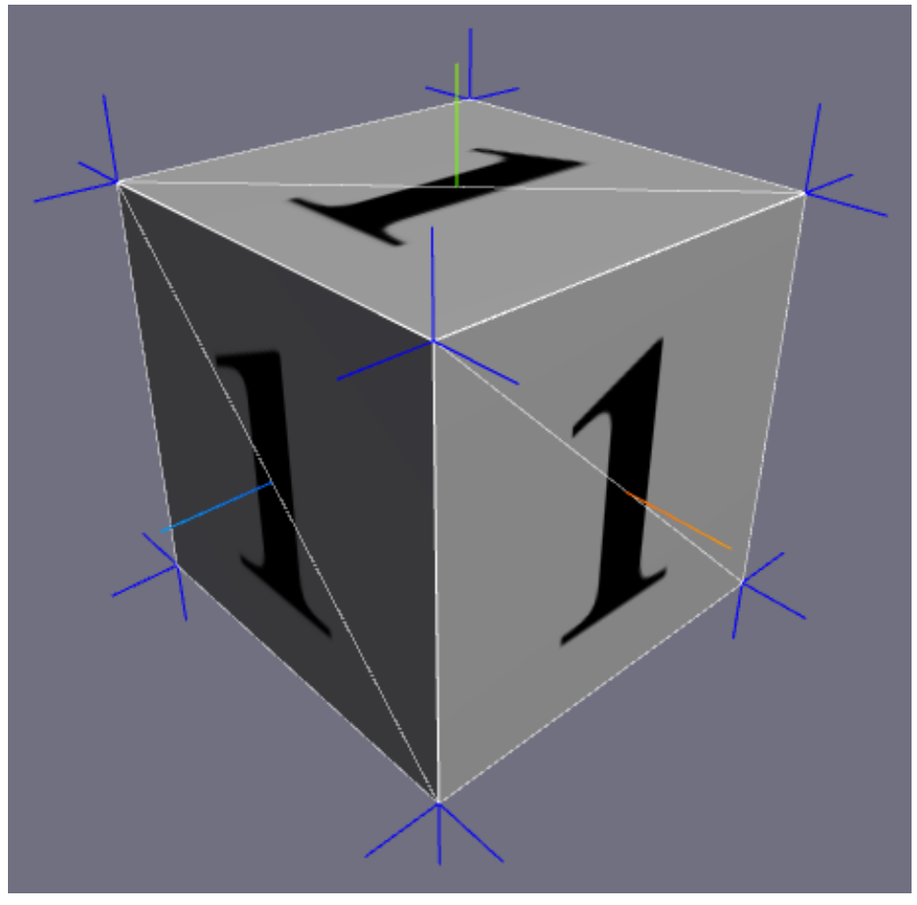


Textures

You are here:



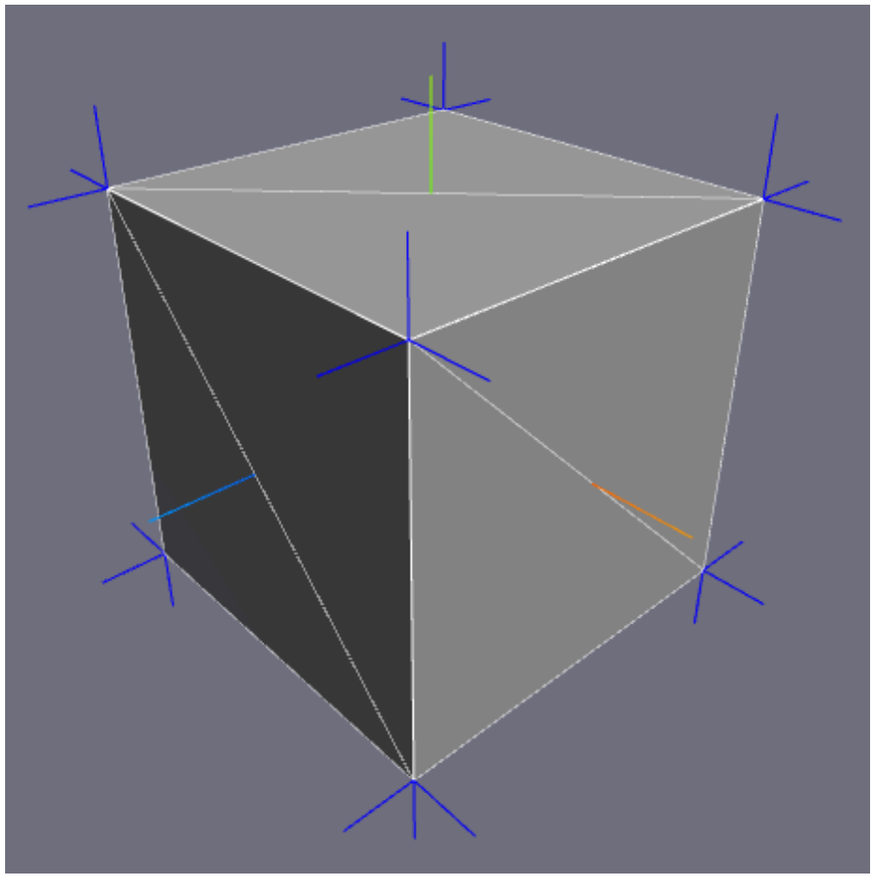
You wish to be here:



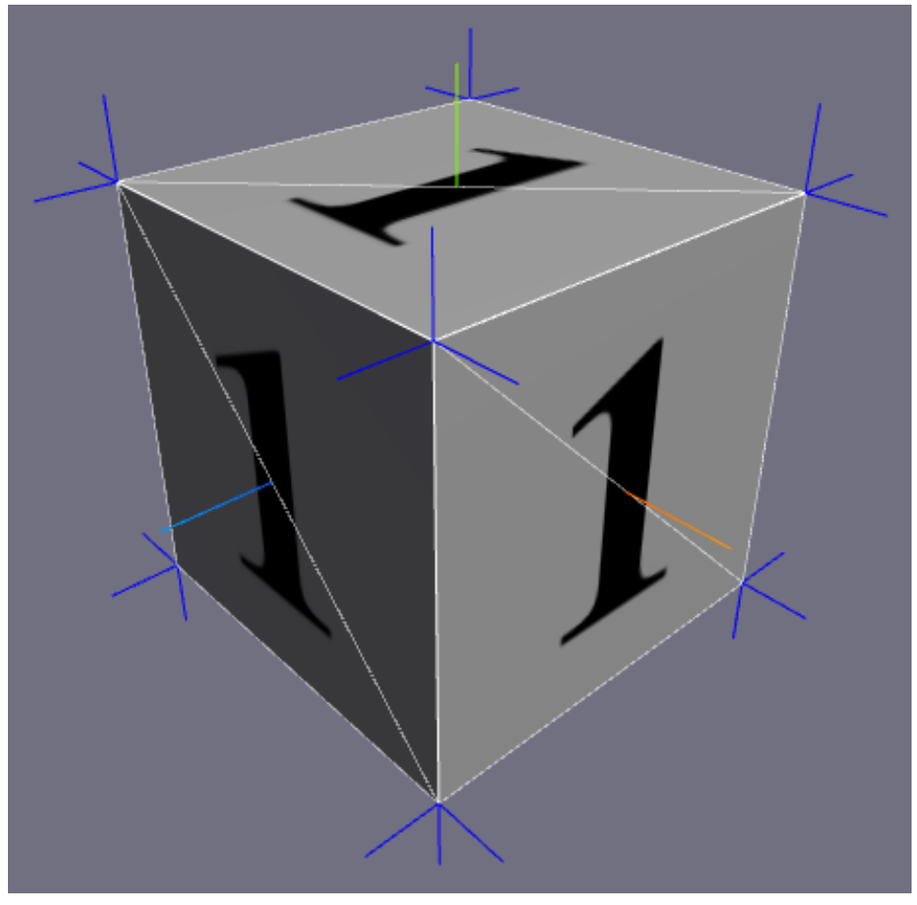
Using current machinery: store a color at each vertex and interpolate between them.

Textures

You are here:



You wish to be here:

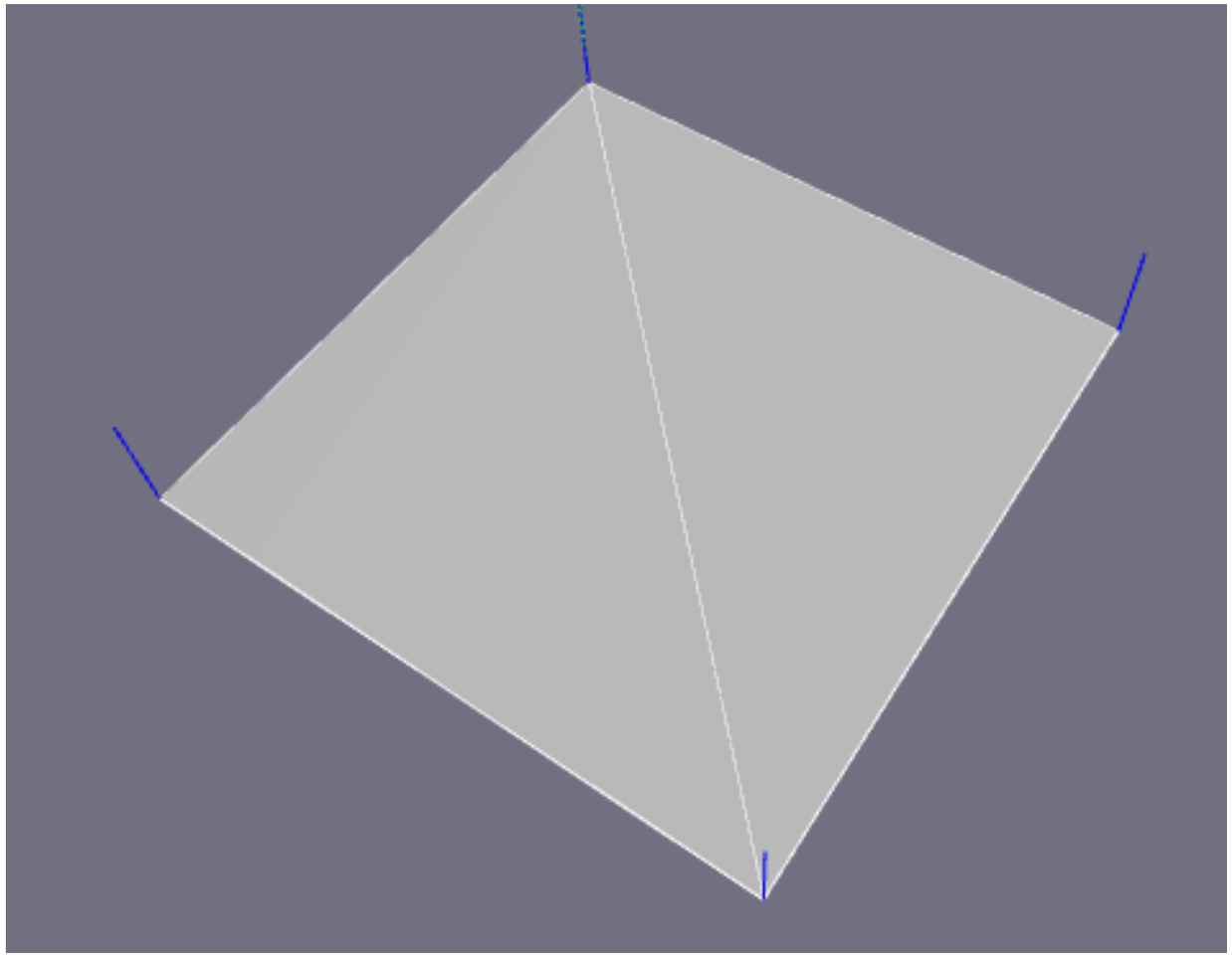


Using current machinery: store a color at each vertex and interpolate between them.

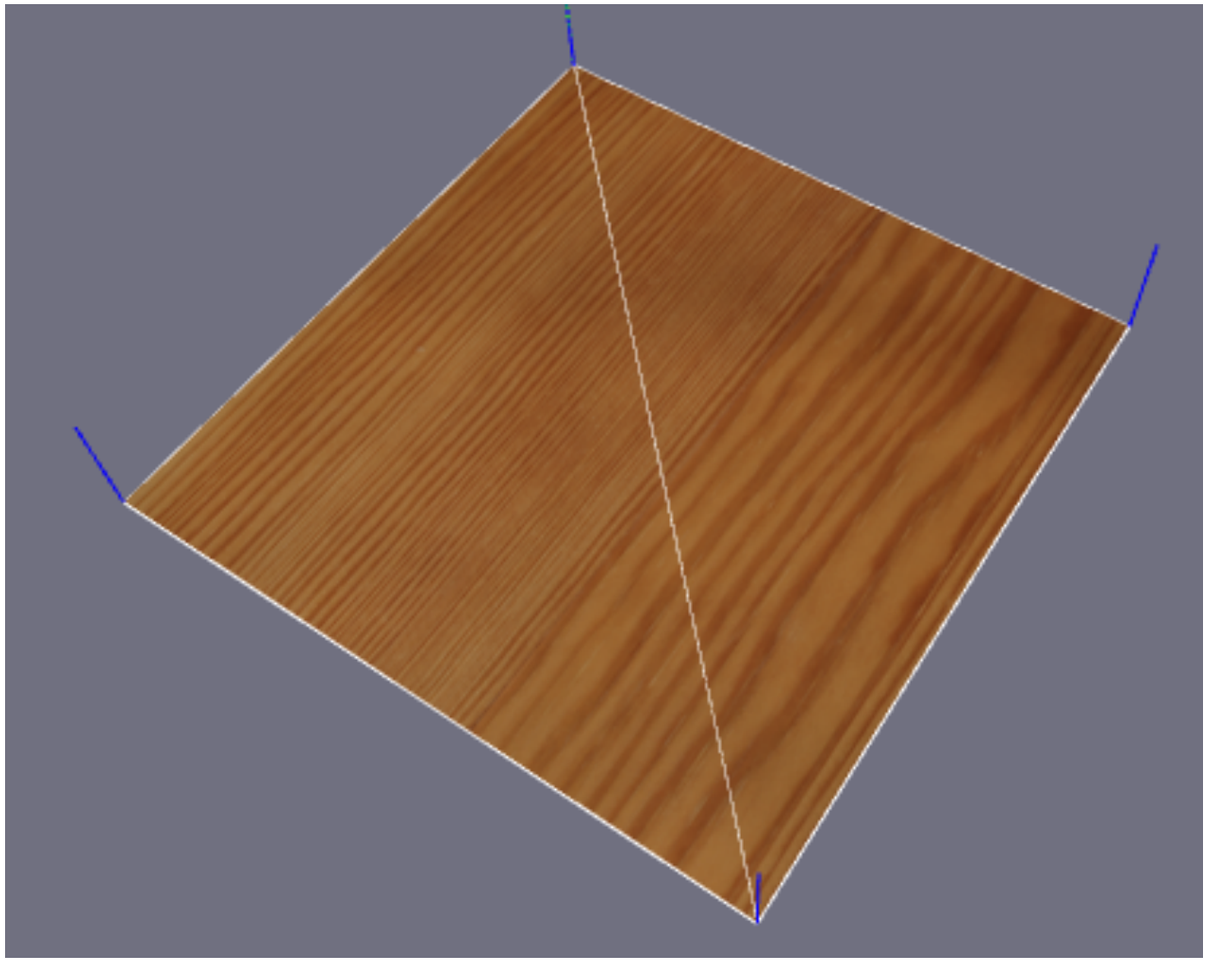
We'd need a bunch more triangles.

Textures

You are here:

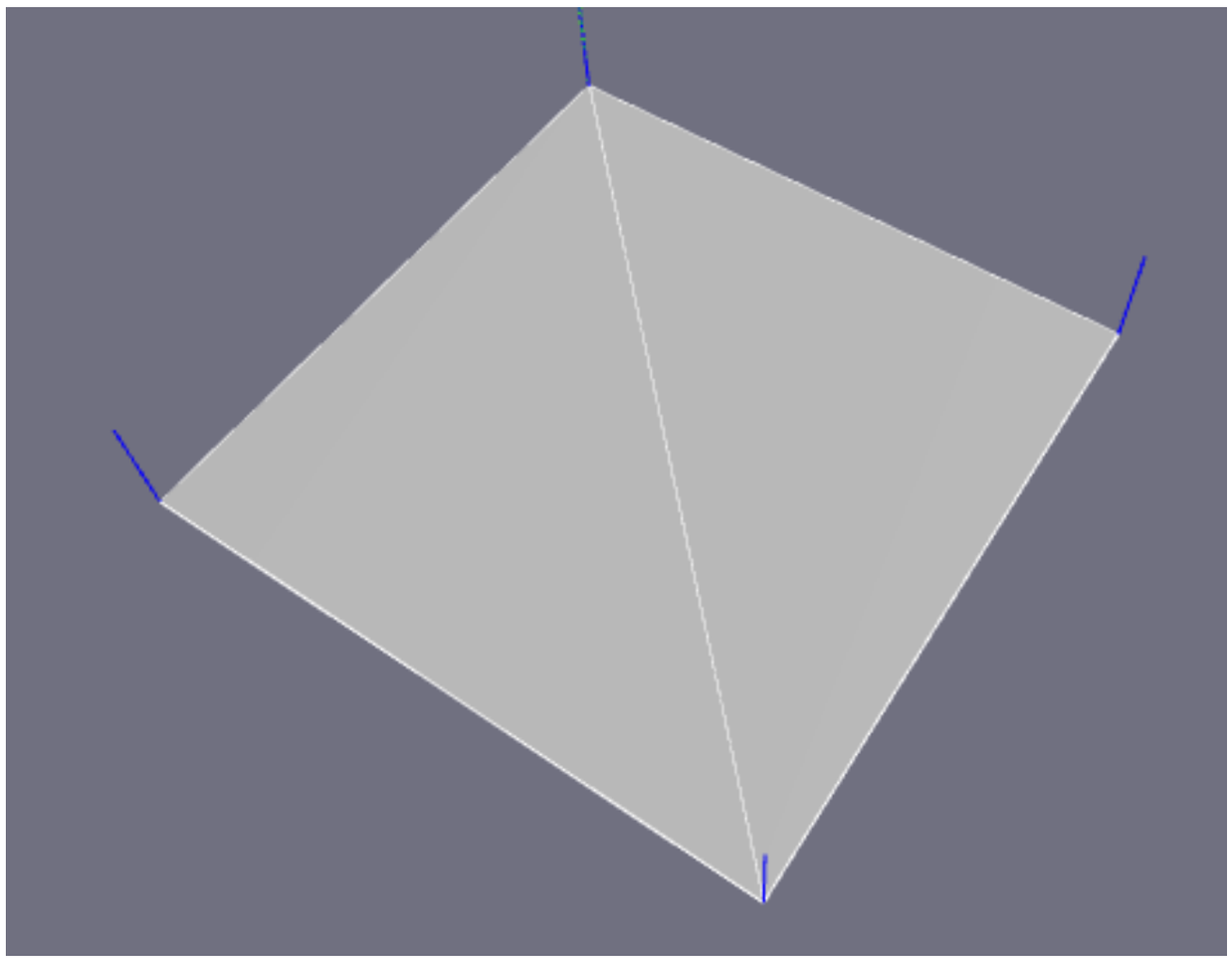


You wish to be here:

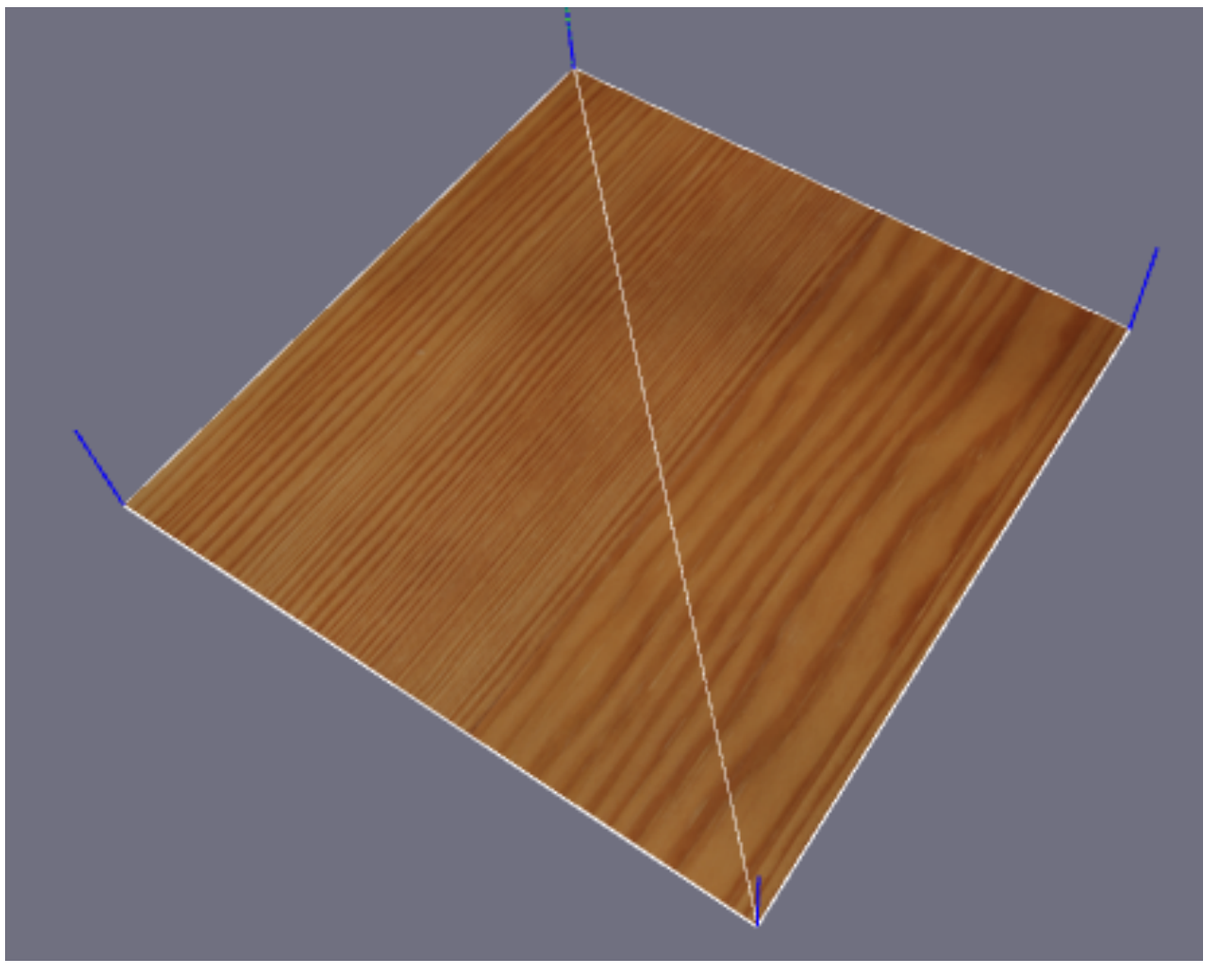


Textures

You are here:



You wish to be here:



We'd need a **bunch** more triangles.

Textures

- Store **spatially varying surface properties:**
  - color is an intuitive example, but many other things too; anything that changes over the surface but doesn't affect geometry (much)
  - roughness, faked lighting effects, normals(!?), bumps

What is a texture?

- A texture is basically a 2D image that stores some **spatially-varying surface property**.

  (use color for intuition, but keep in mind it's more general)

2D grid of values ("texels")
u, v coordinates in [0, 1]

Texture Mapping

- To use this, we need a **mapping** (function)
  - from the surface we're modeling/rendering
  - to (u,v) **texture coordinates**

- **Simplest possible example:**
  a 2x2 tabletop in the xz plane

- When rendering, non-vertex points get colors via **interpolated** (u,v) coordinates.

Texture Mapping: nontrivial surfaces

Rendering (later!)

Texture mapping

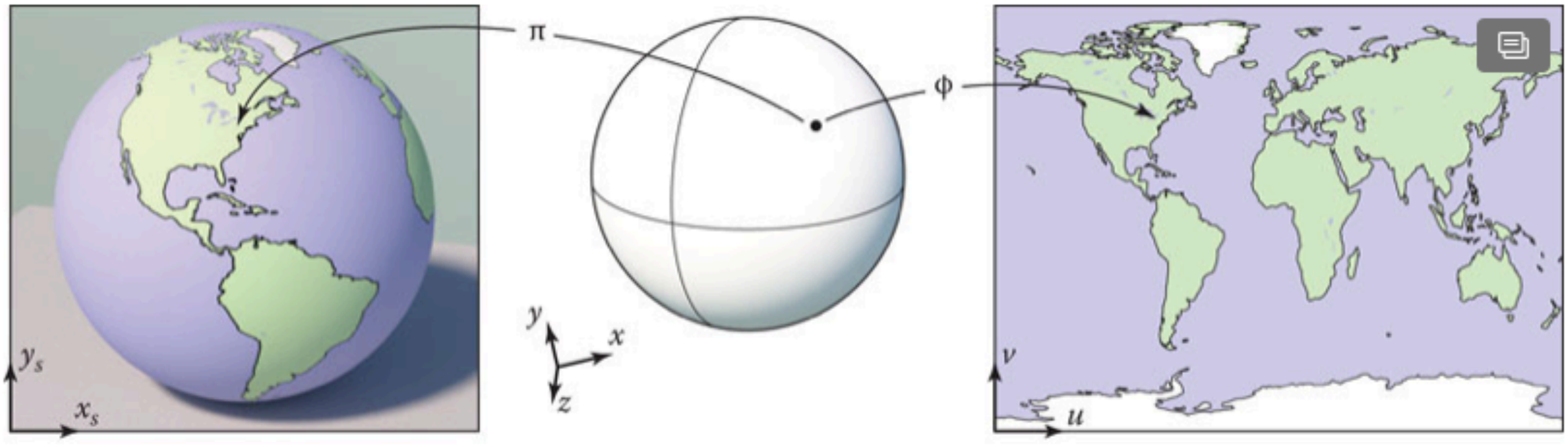


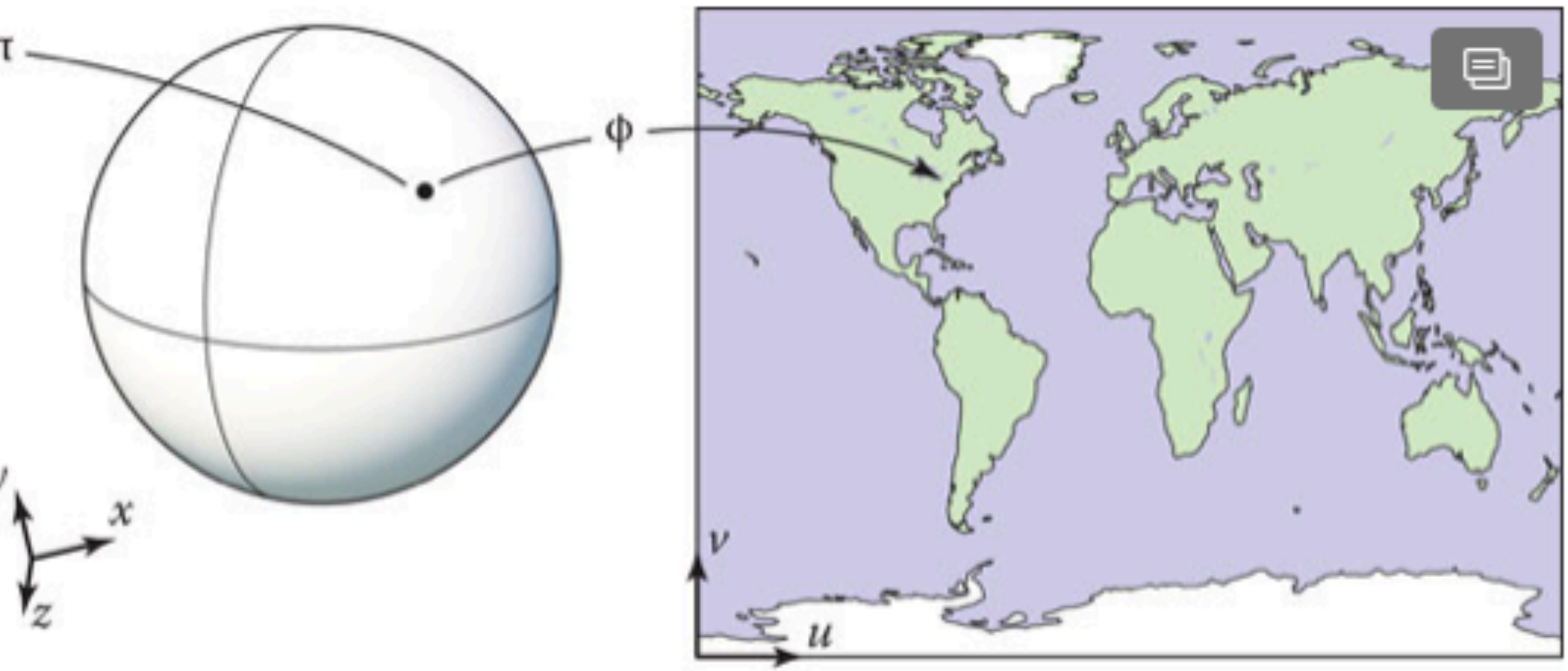
Image space

Surface *S* in world space

Texture space, *T*

Texture Mapping: nontrivial surfaces

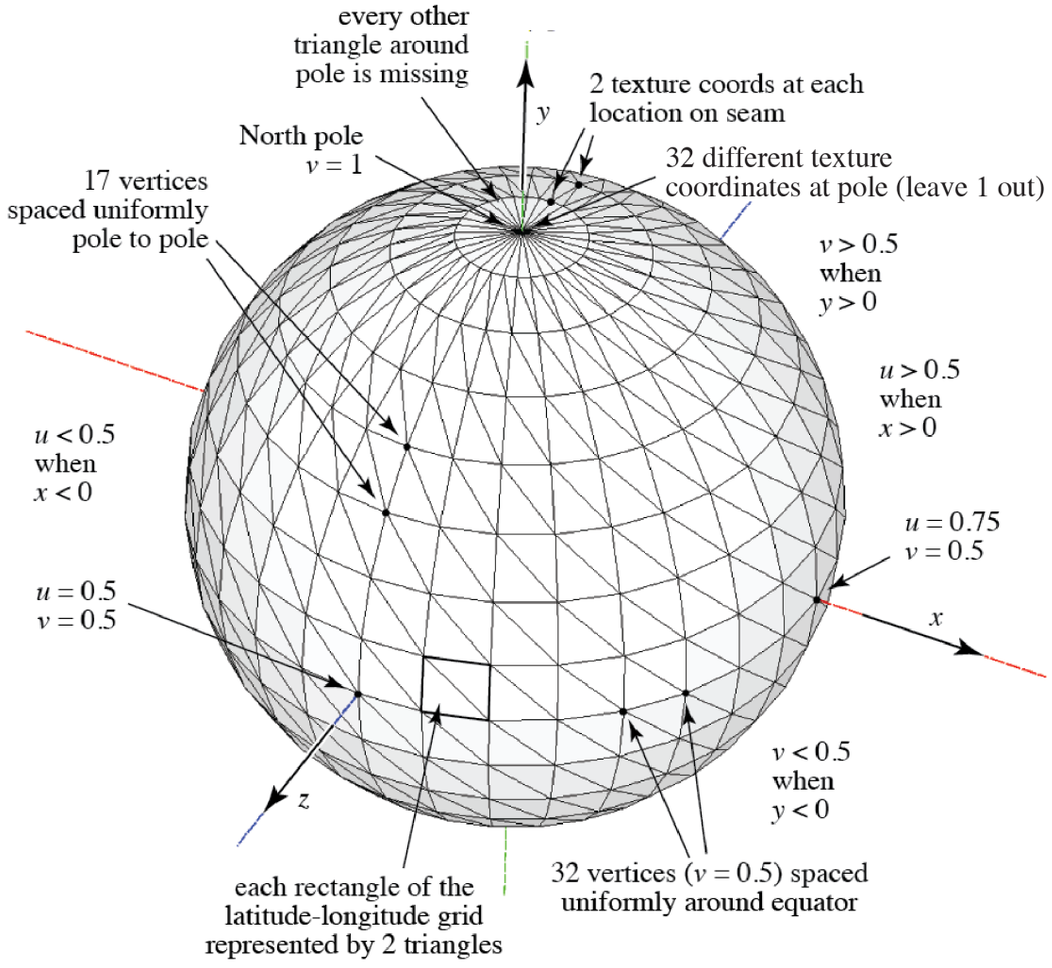
- Map from point on sphere to point in (u,v)

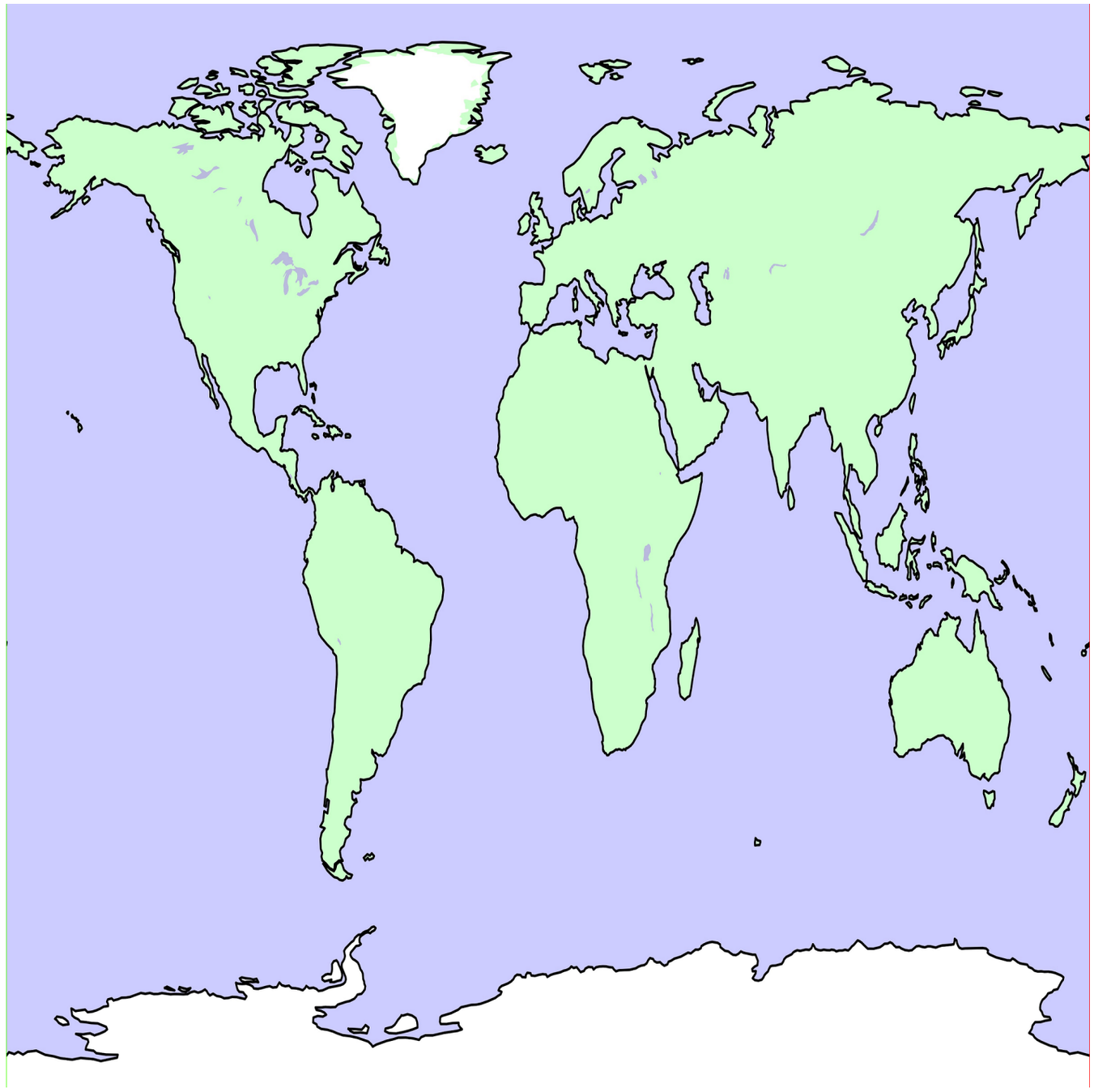


Surface *S* in world space

Texture space, *T*

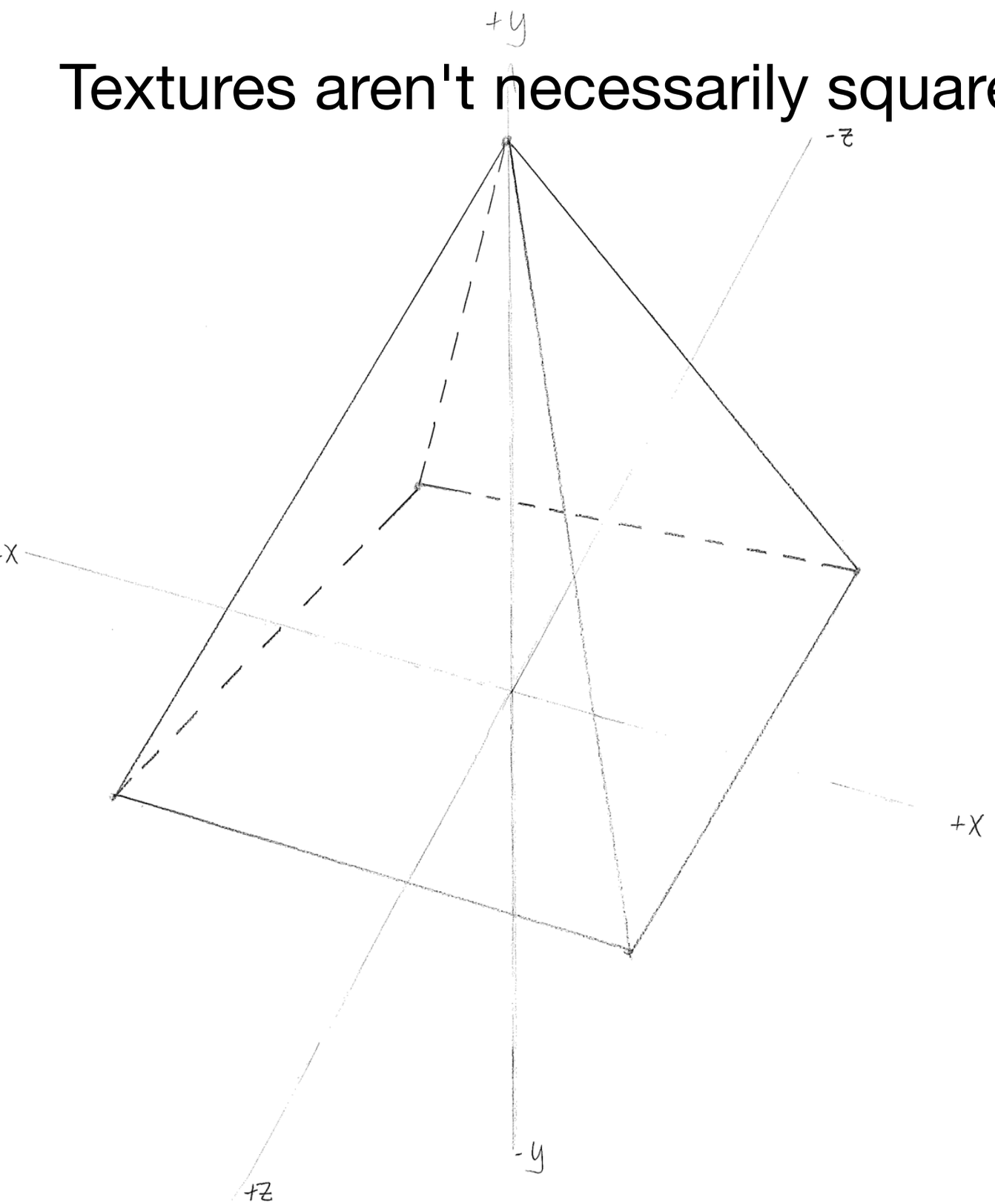
A1 sphere - demo

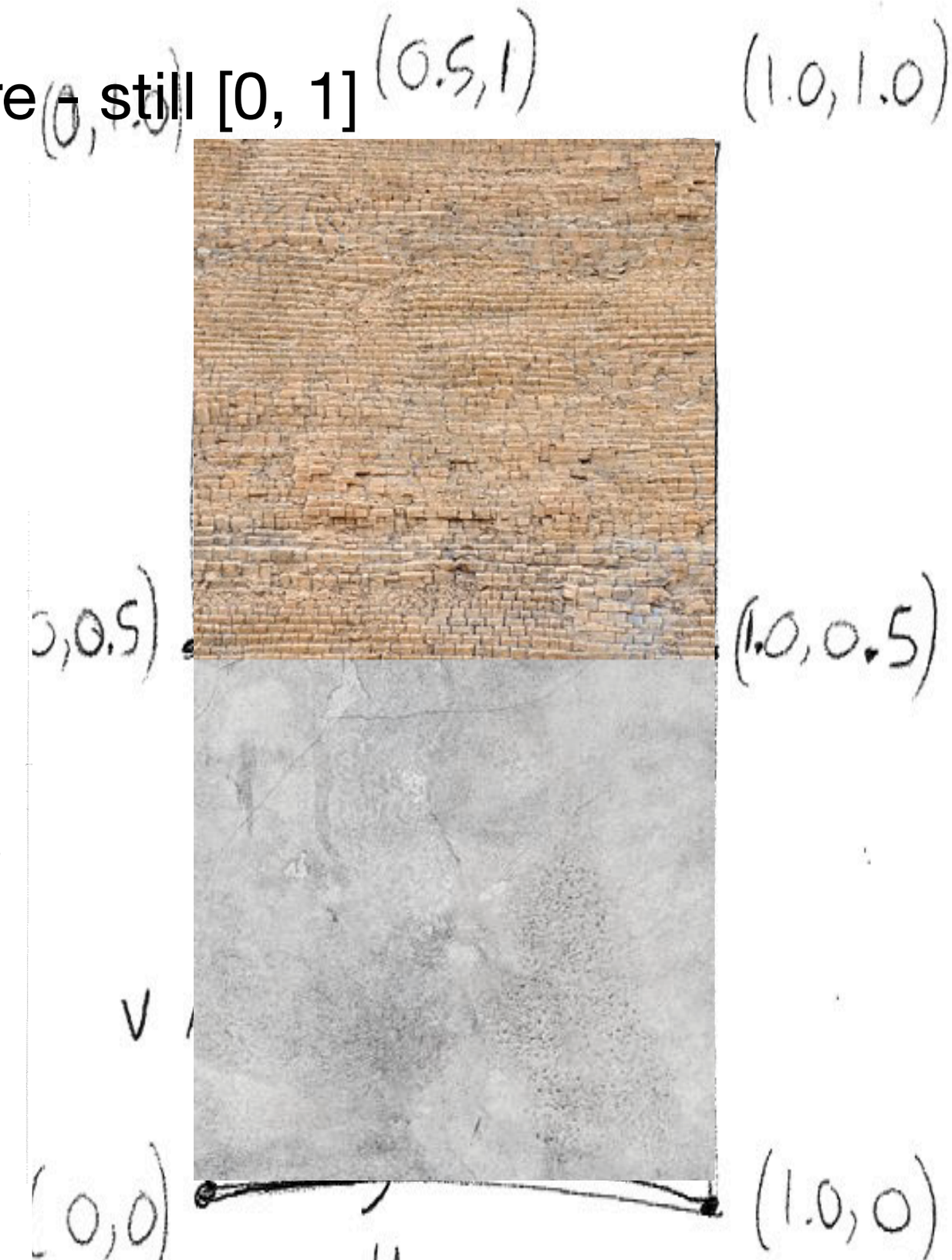




Texture Mapping the Pyramid

Textures aren't necessarily square - still [0, 1]





Texture Mapping the Pyramid

Textures aren't necessarily square - still [0, 1]