Texture Mapping

Textures Provide Details

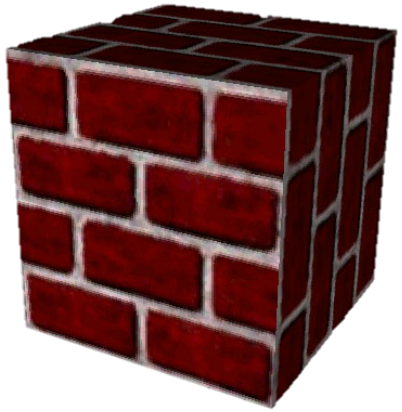
Makes Graphics Pretty

- Details creates immersion
- Immersion creates fun

Basic Idea

Paint pictures on all of your polygons

- adds color data
- adds (fake) geometric and texture detail

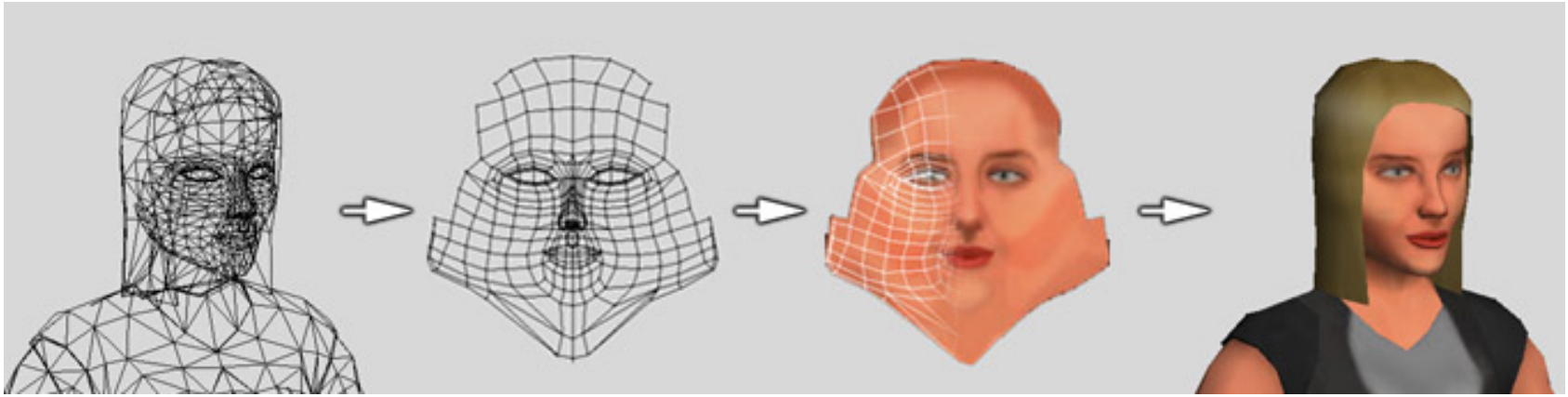


One of **the** basic graphics techniques

- tons of hardware support

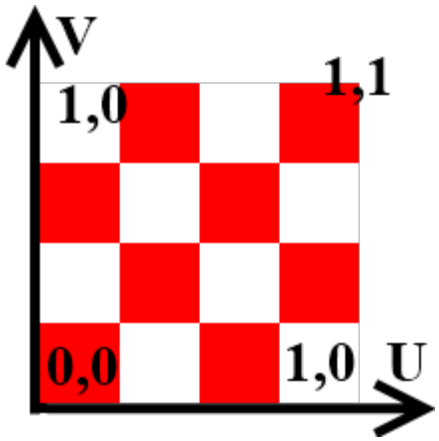
Texture Mapping

- Map between region of plane and arbitrary surface
- Ensure “right things” happen as textured polygon is rendered and transformed



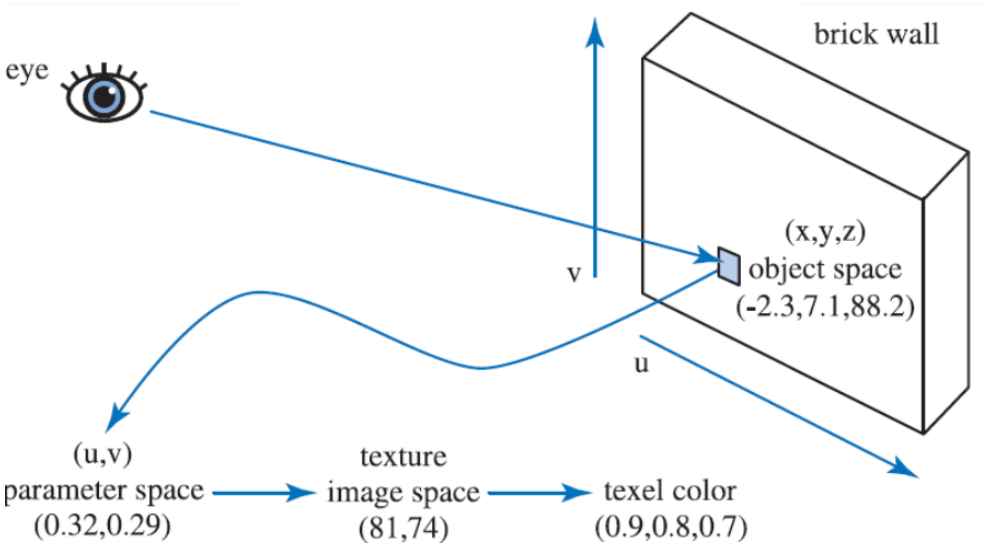
Parametric Texture Mapping

- Texture size and orientation tied to polygon
- Texture can modulate diffuse color, specular color, specular exponent, etc
- Separation of **texture space** from **screen space**
  - UV coordinates of range [0…1]



Retrieving Texel Color

- Compute pixel (u,v) using barycentric interpolation
- Look up texture pixel (texel)
- Copy color to pixel
- Apply shading



Understanding Texture Maps

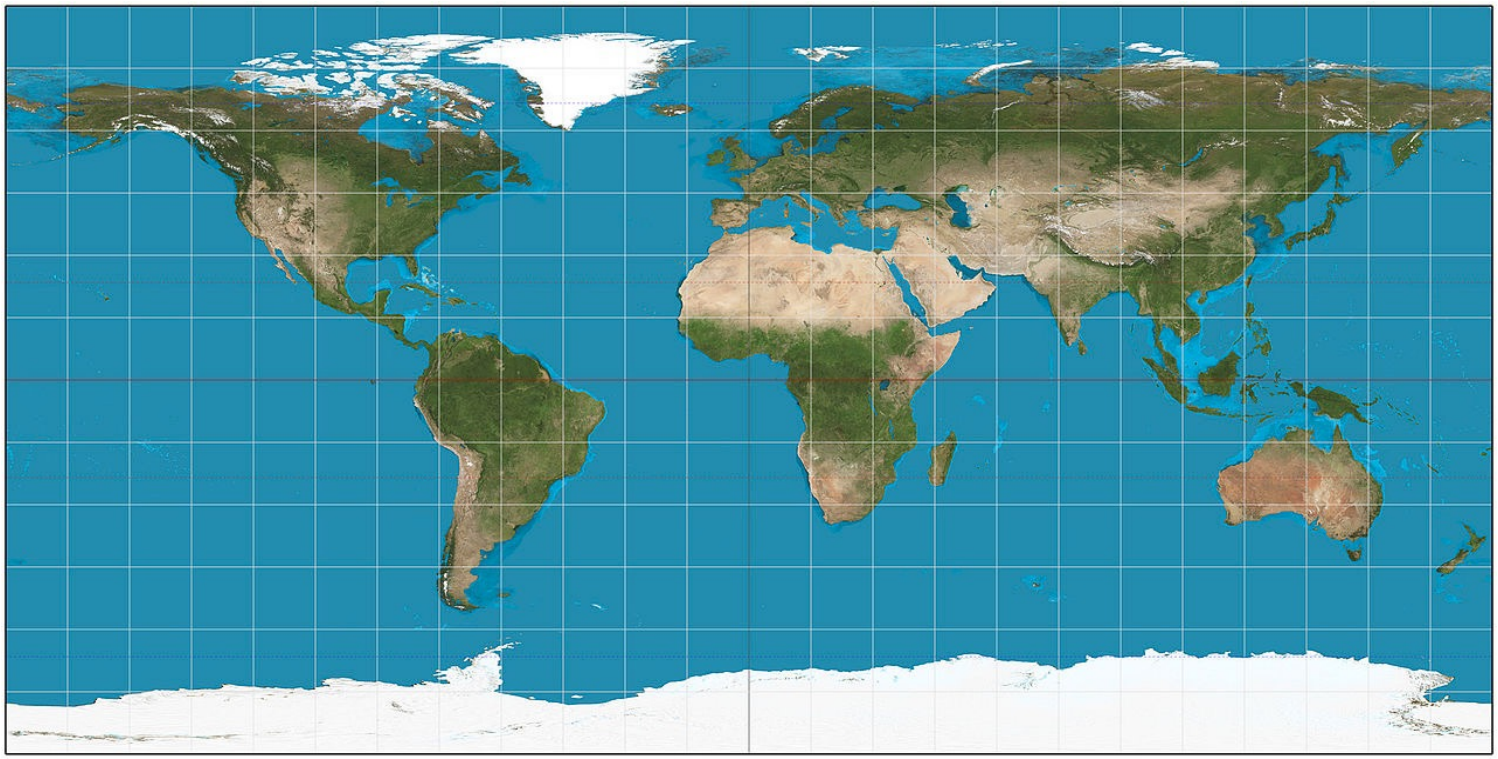
- Parameterization related to:
  - Texture mapping
  - UV coordinates
  - UV unwrapping
- Usually means assigning U and V coordinates to every pixel
- Or U and V for every vertex, then interpolate

How to Parameterize?

Classic problem: How to parameterize the earth (sphere)?

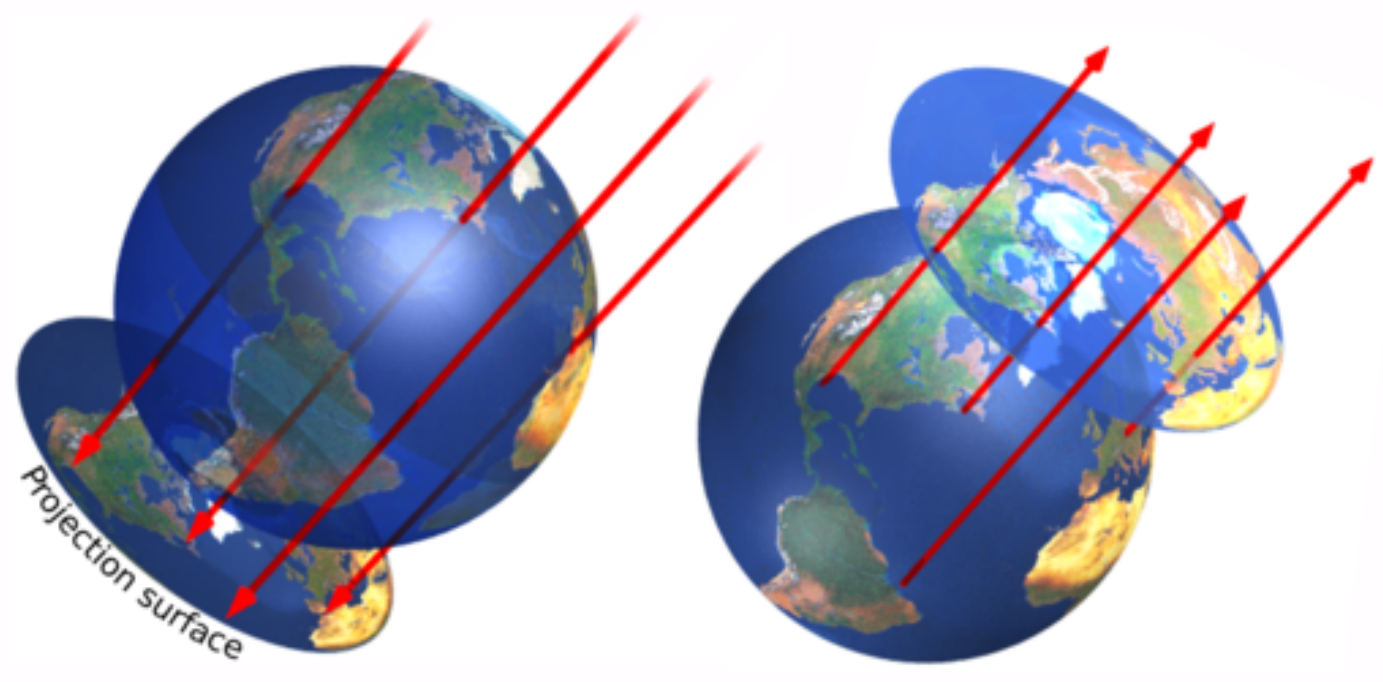
Very practical, important problem in Middle Ages…

Latitude & Longitude



Distorts areas and angles

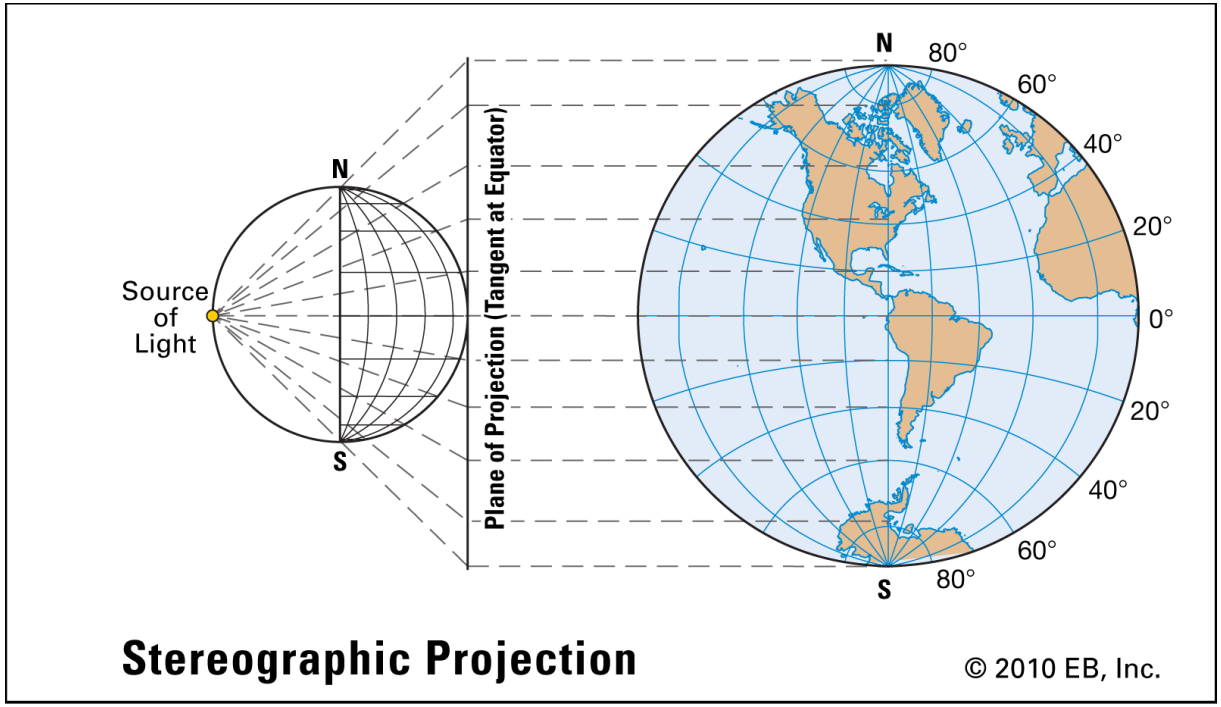
Planar Projection



Covers only half of the earth

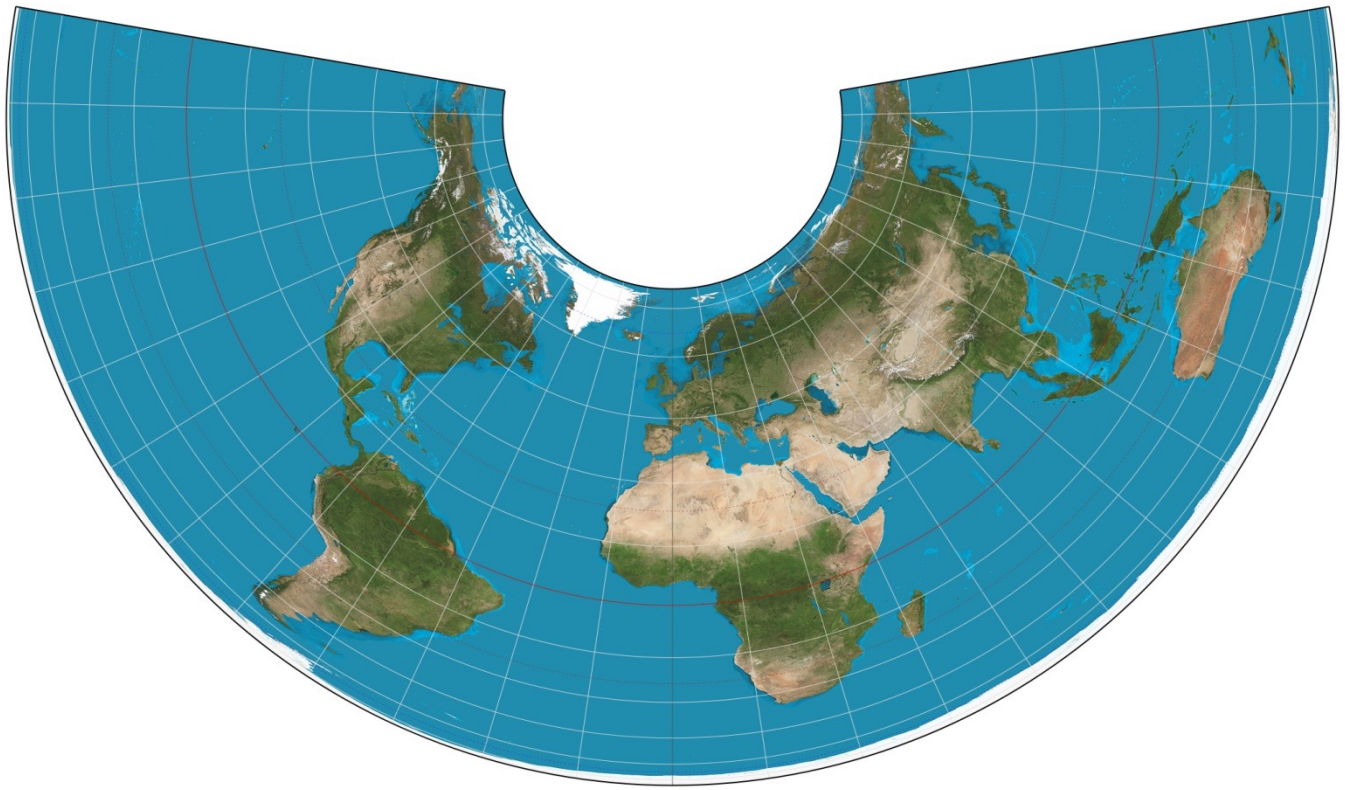
Distorts areas and angles

Stereographic Projection



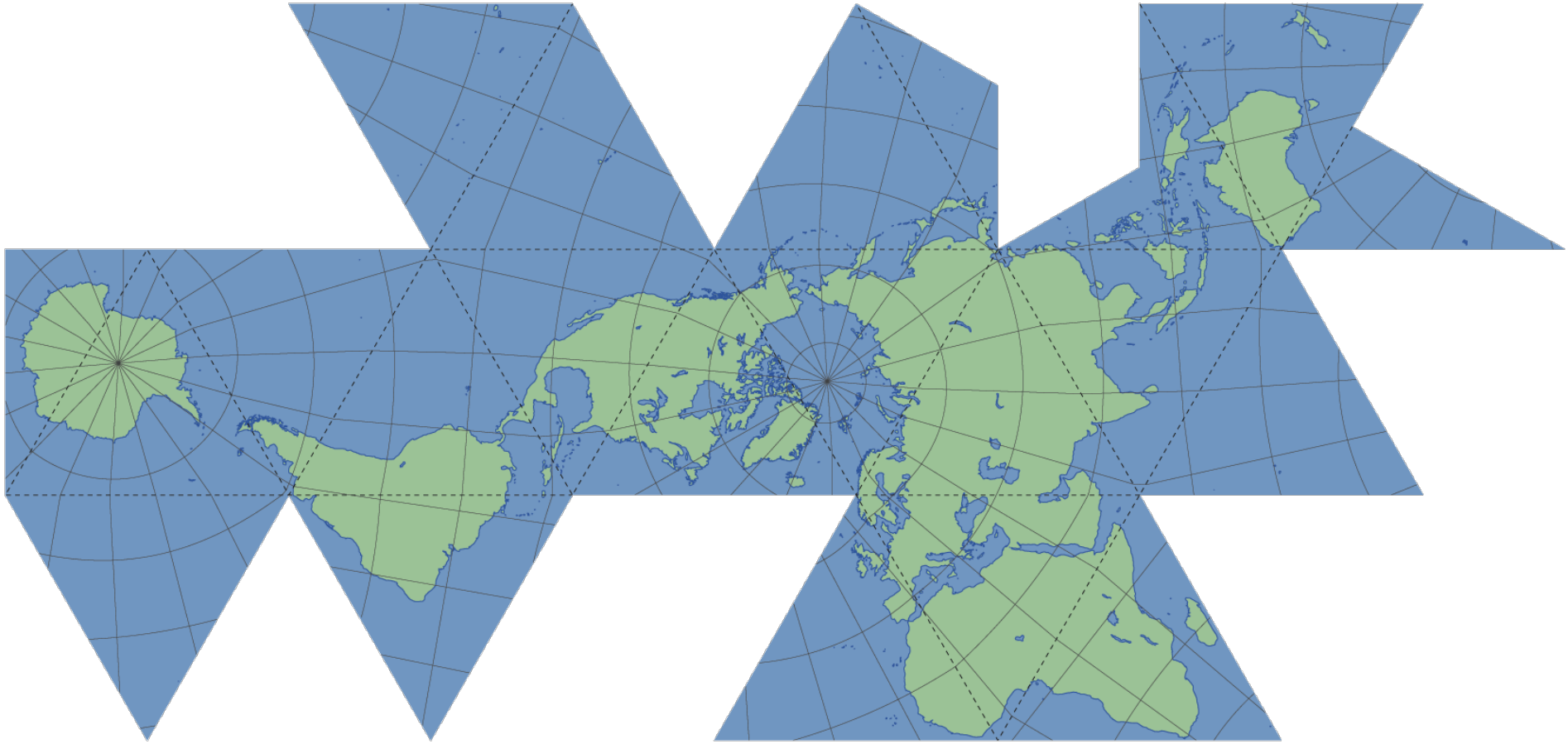
Distorts areas

Albers Projection



Preserves areas, distorts aspect ratio

Fuller Parameterization



No Free Lunch

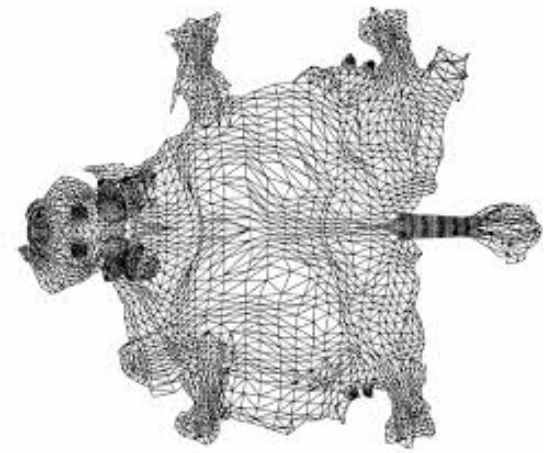
Every parameterization of the earth either:

- distorts areas
- distorts distances
- distorts angles

Good Parameterizations

- Low area distortion
- Low angle distortion
- No obvious seams
- One piece

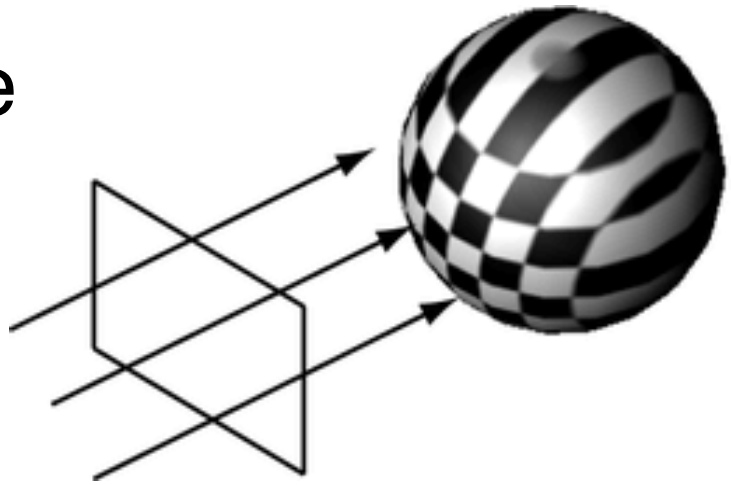
- How do we achieve this?



Planar Parameterization

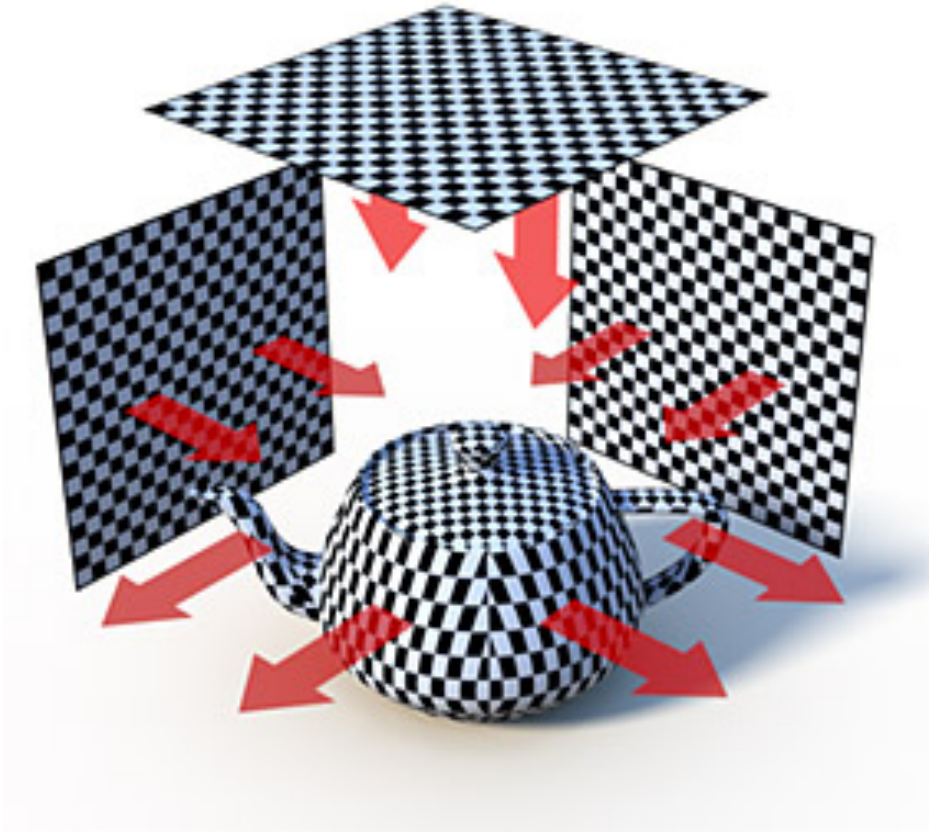
Project surface onto plane

- quite useful in practice
- only partial coverage
- bad distortion when normals perpendicular

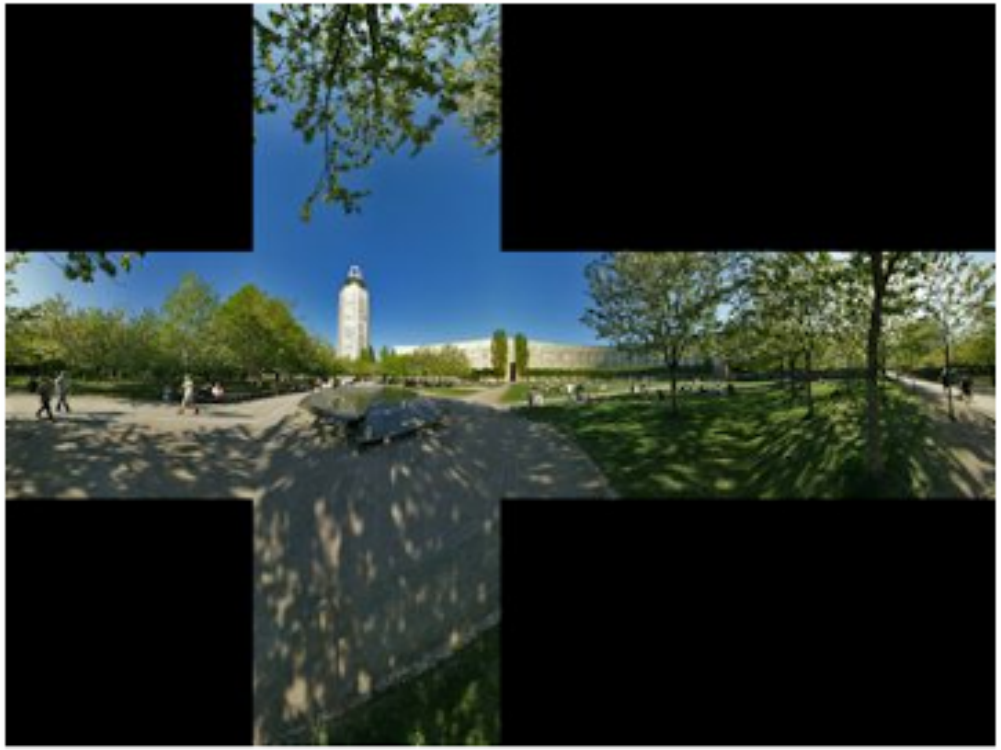


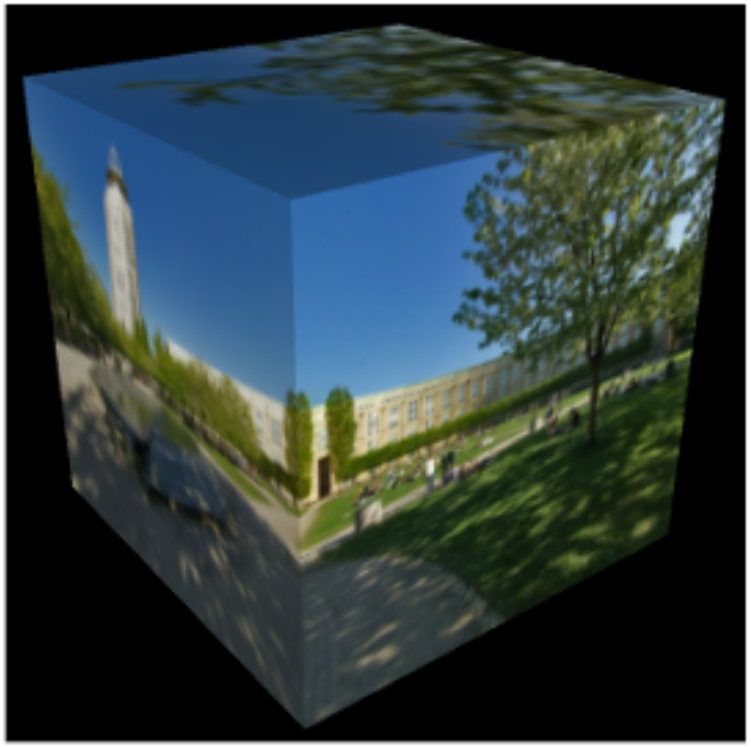
Planar Parameterization

In practice: combine multiple views



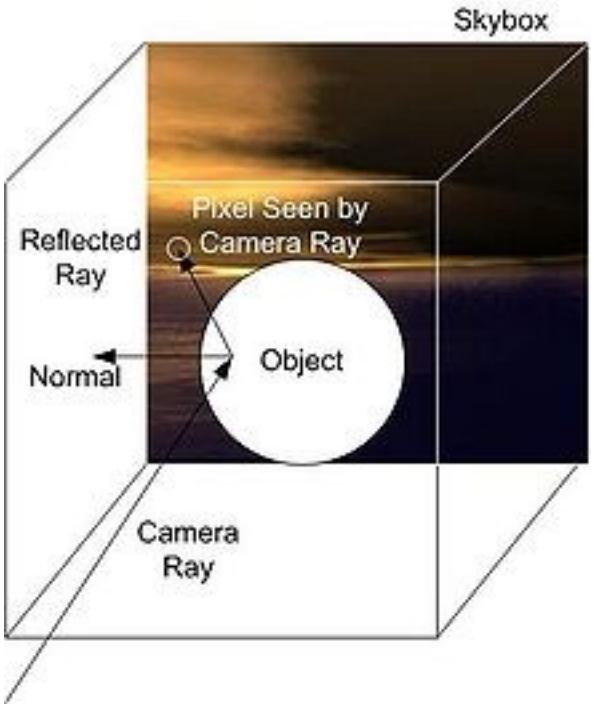
Cube Map/Skybox

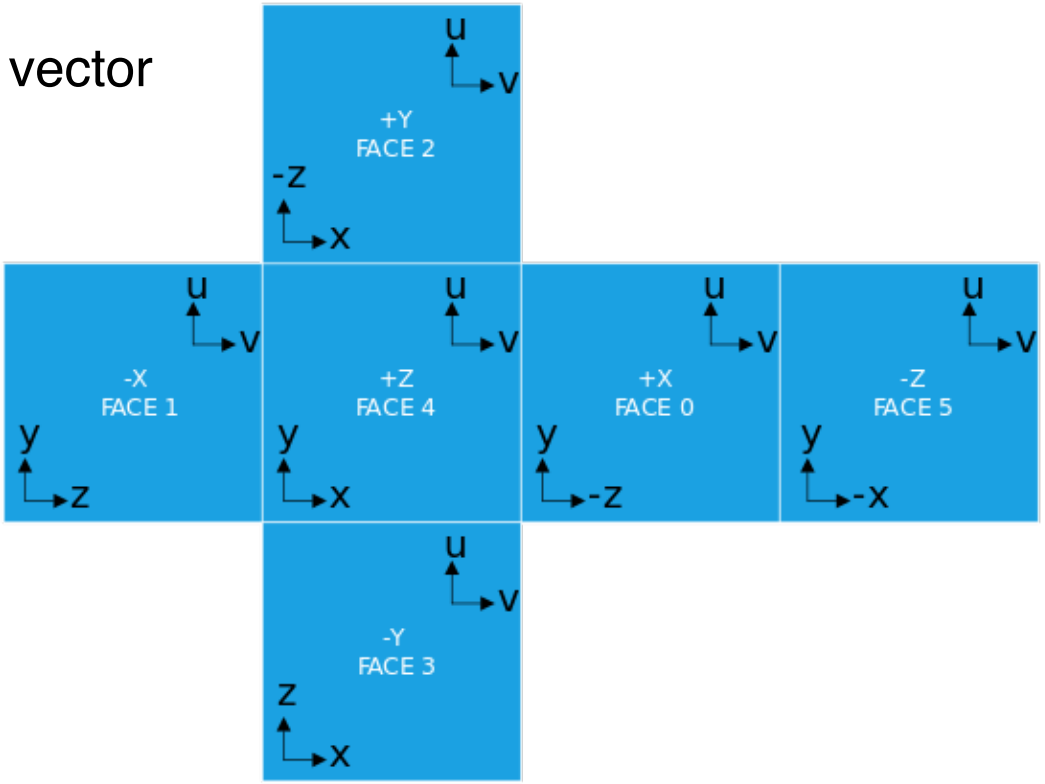




Cube Map Textures

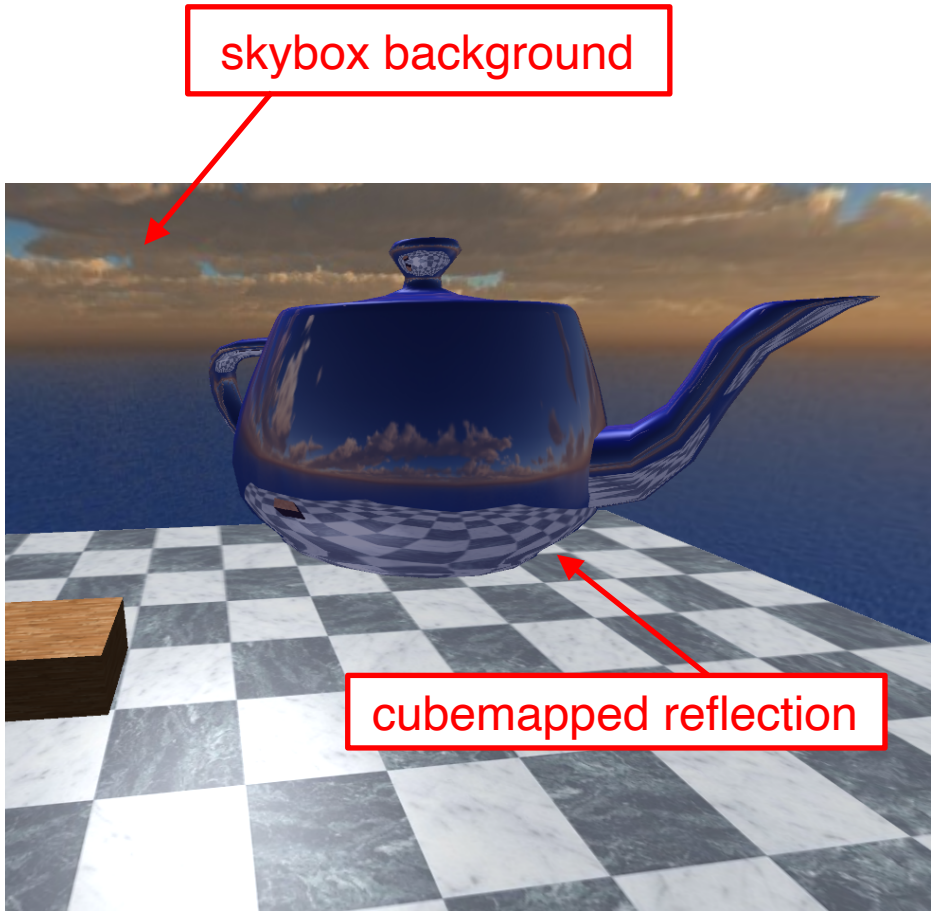
- 6 2D images arranged like faces of a cube
  - +X, -X, +Y, -Y, +Z, -Z
  - Index by unnormalized vector





Cube Map vs Skybox

- Cube maps map reflections to emulate reflective surface (e.g. environment mapping in local illumination)
- Skyboxes provide scene information where there is no geometry
- Same mathematical idea — just different use cases!

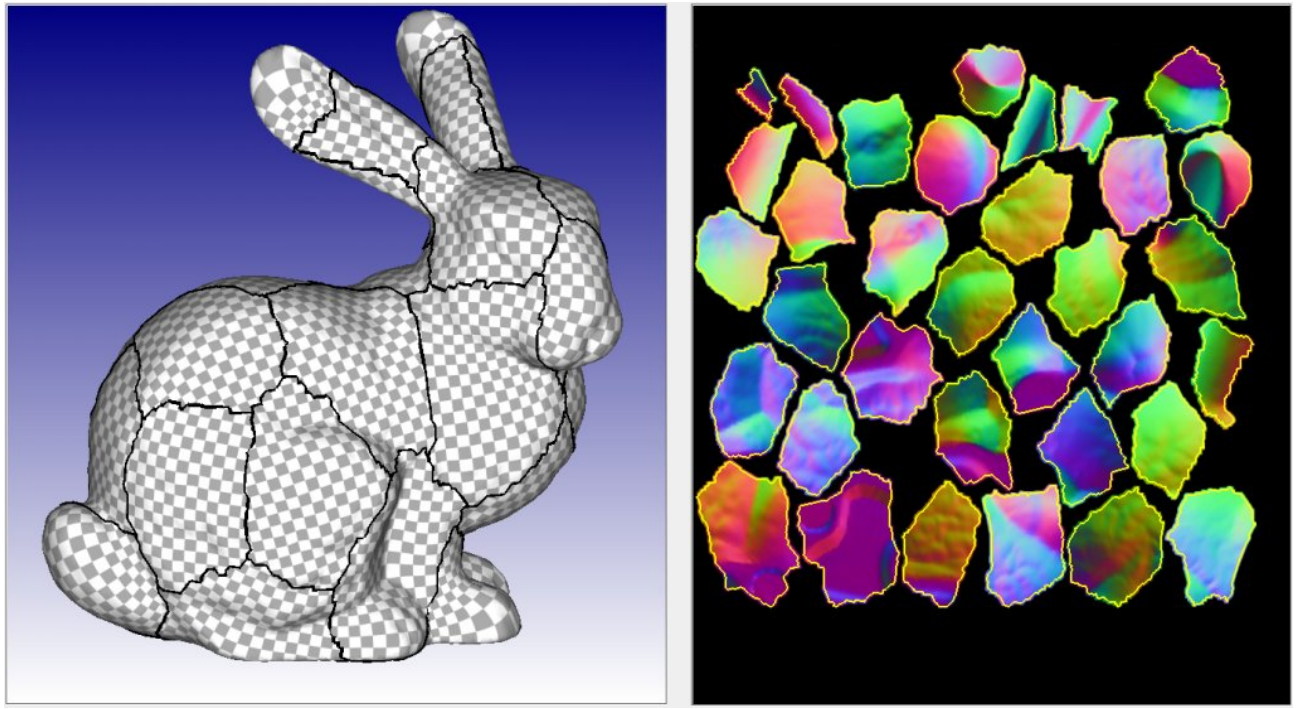


Storing Textures

- Texture sizes traditionally powers of 2
- Textures usually compressed on GPU
- Textures can be 3D
  - Huge memory hog!

Texture Atlas

Break up surface into easy pieces, parameterize separately



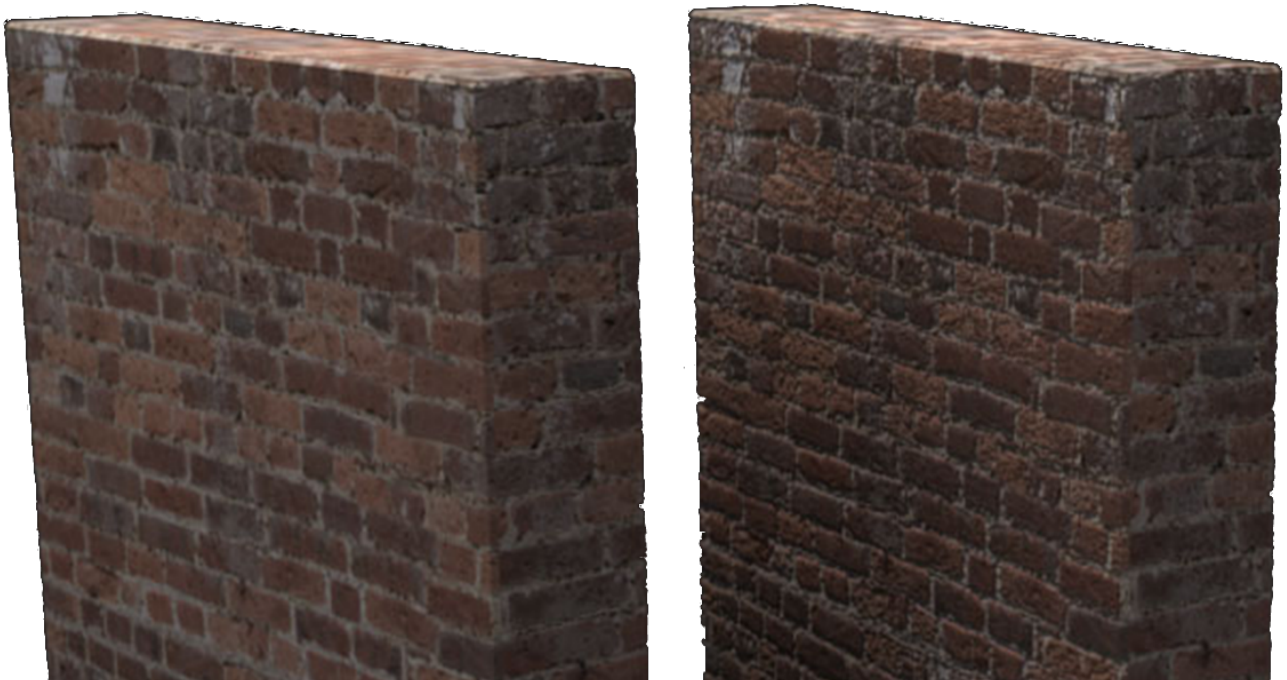
Texture Atlas

More and more automatic methods exist…

but artists traditionally hand-painted UV coords and UV "unwrapping" is still a tedious process

Texture Mapping Flaws

Texture mapping adds fake geometric details but still looks flat

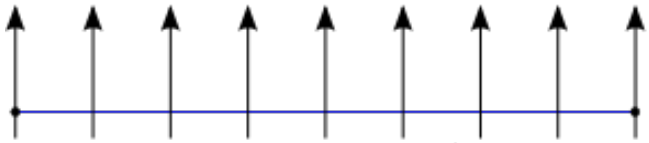


How do we fix this?

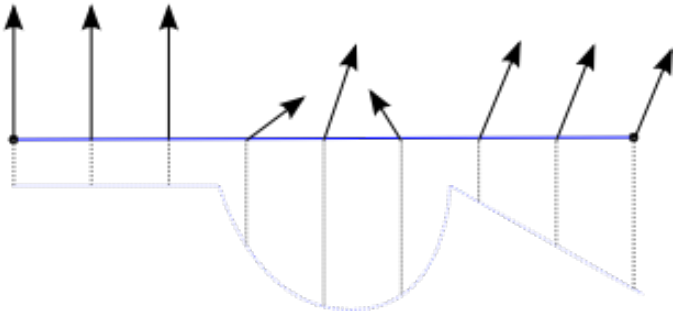
Normal Map

Key idea: modify **normals** of flat face

Unmapped face



Normal-mapped face

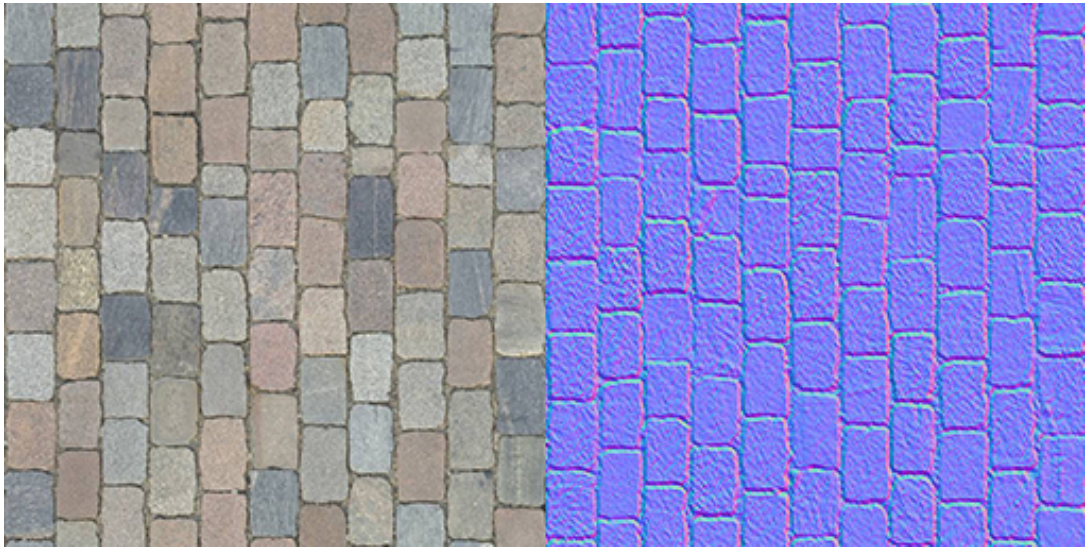


Rendered surface

- **is** flat
- shaded as if it were bumpy

Normal Map

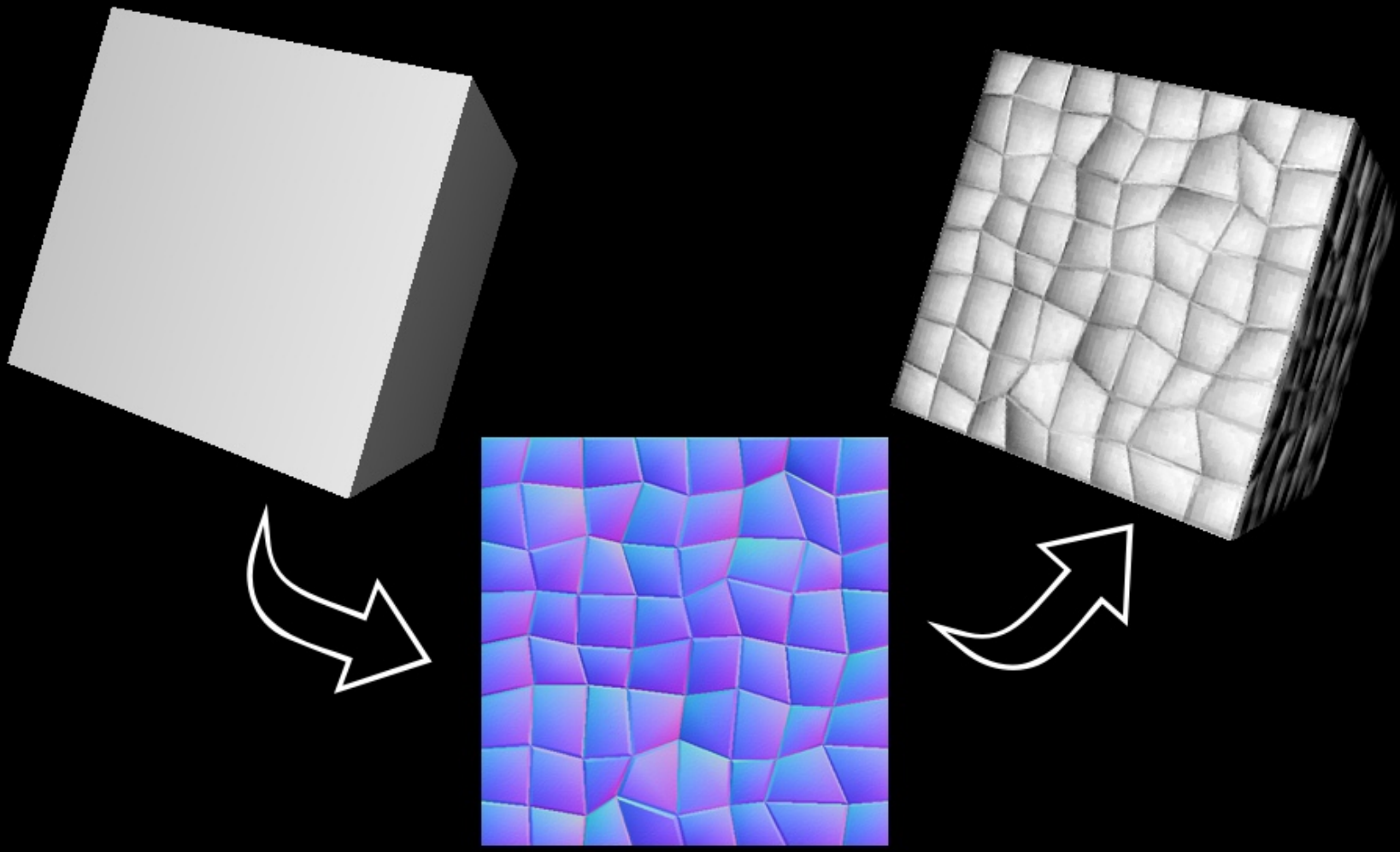
How to represent normals?



Encode as second texture (same size)

- (r,g,b) encodes coordinates of normal (x, y, z)

Applying Normal Map



Bump Mapping

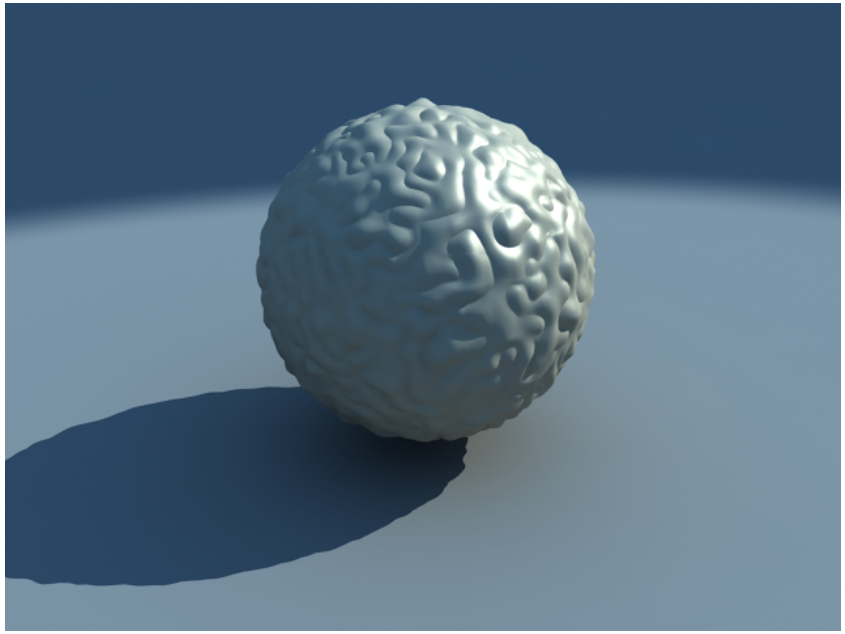
Older technique: give **offset height** only

- Less flexible than normal mapping
- Can be converted to normal maps
- Artists may use terms interchangeably

Displacement Map

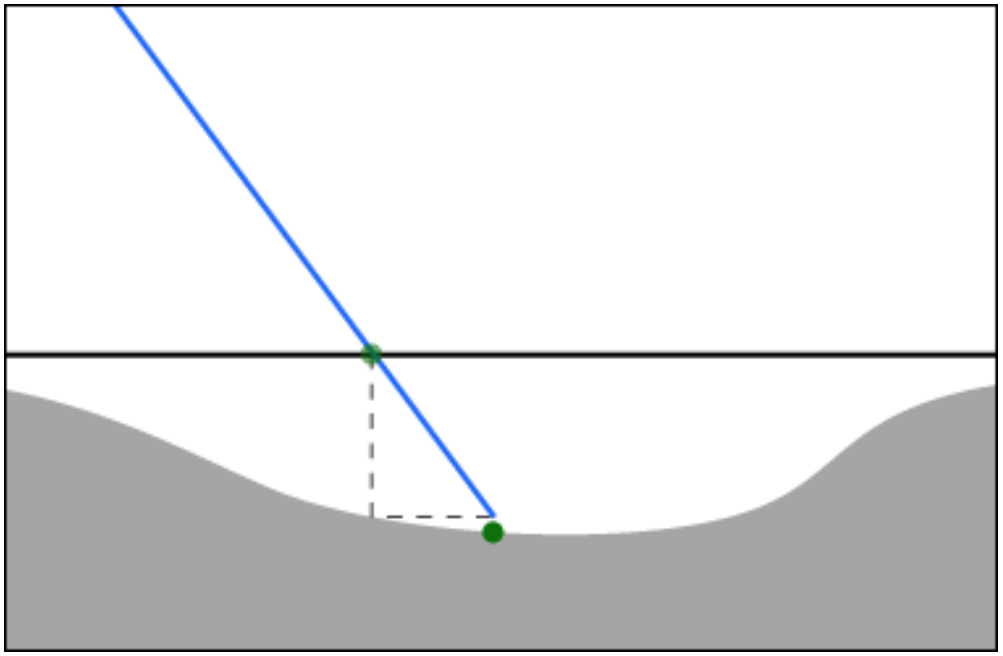
Like normal map, but change normals **and** geometry

- Fully correct
- Slow
- Increasingly common in games due to hardware improvements

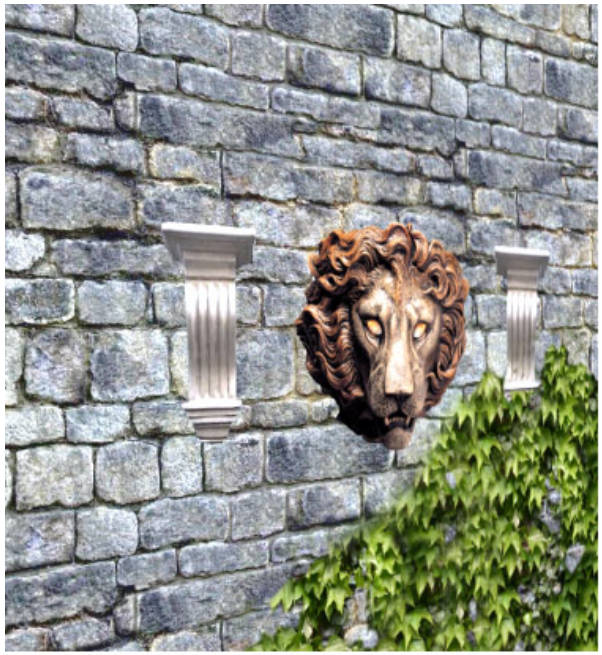


Parallax Map

Take into account **shift in texture coordinates**



Parallax Map Example



Texture Mapped

Normal Mapped

Parallax Mapped