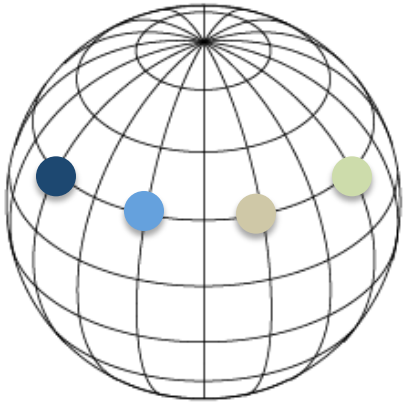
# Texturing

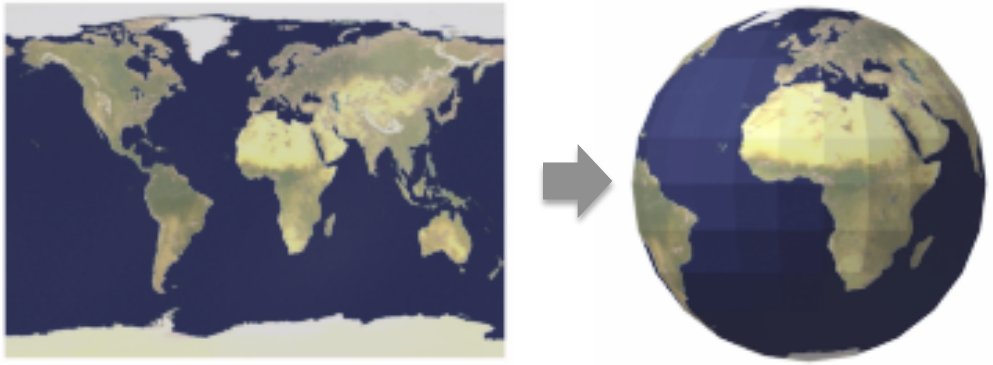
# Texture Mapping

- Have seen: colour can be assigned to vertices
- But: don't want to represent all this detail with geometry

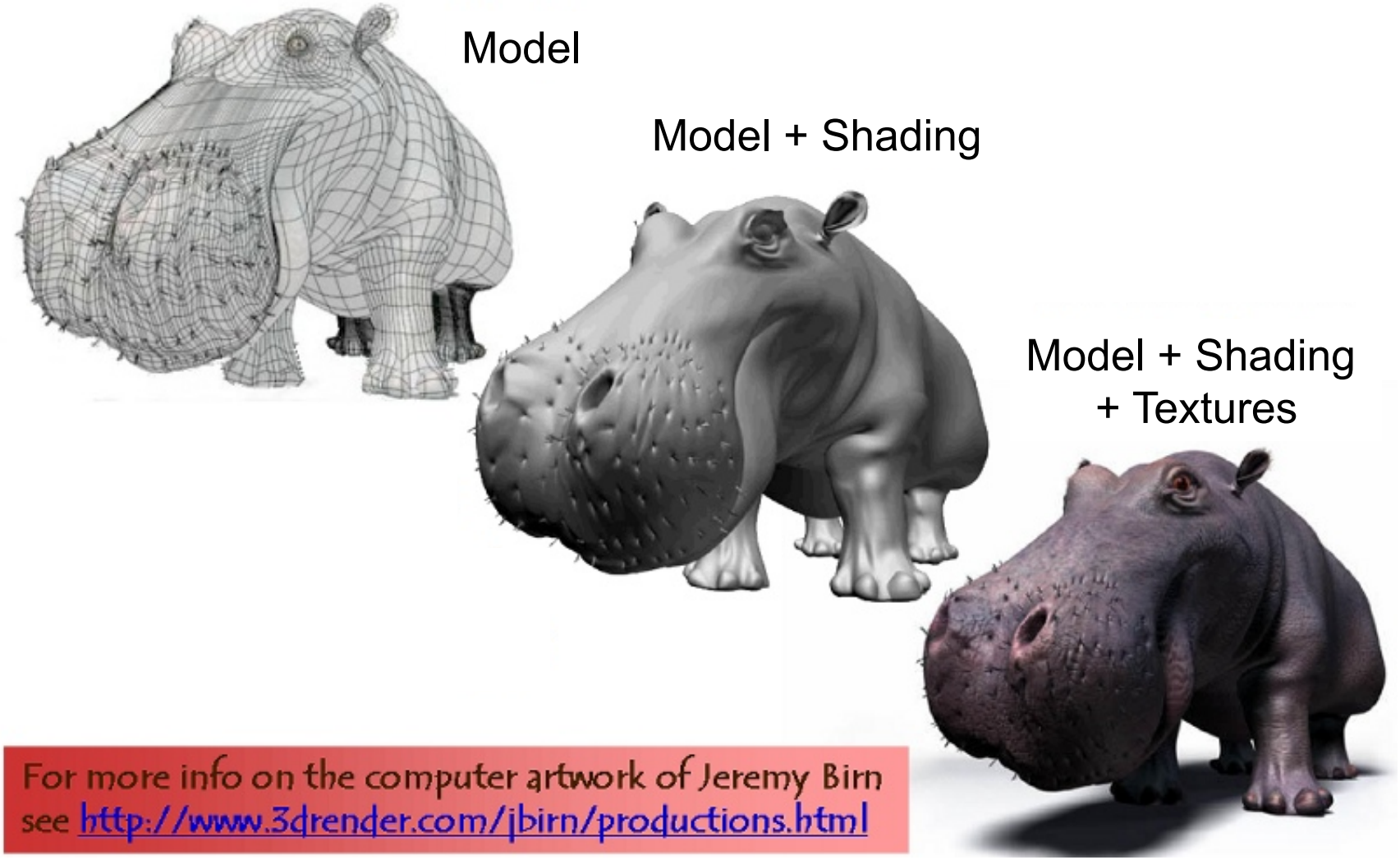
# Texture Mapping

- Considering small details
  - We may not want to add polygons to represent every detail
  - Instead, prefer to keep a large polygon and use an *image* to represent the details





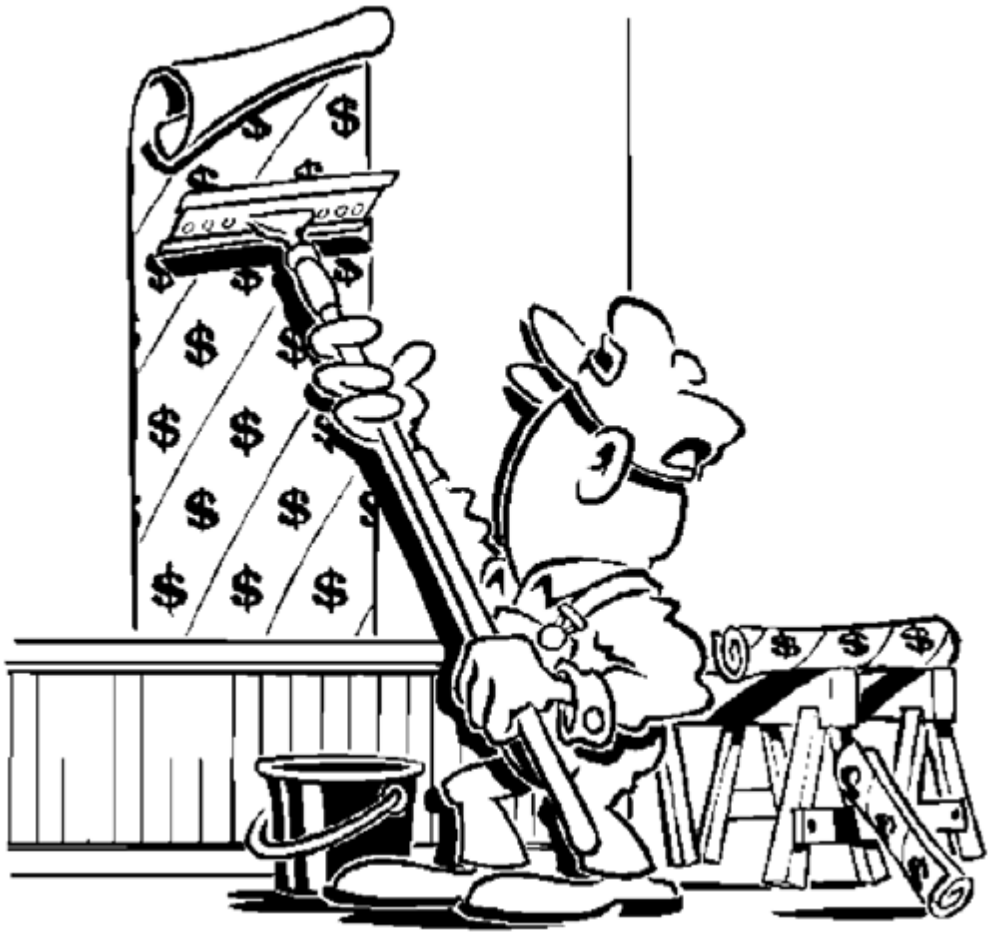
# The Quest for Visual Realism




For more info on the computer artwork of Jeremy Birn
see http://www.3drender.com/jbirn/productions.html

# Texture Mapping

- Increase the apparent complexity of simple geometry
- Efficient packing of flat detail
- Like wallpapering or gift-wrapping with stretchy paper



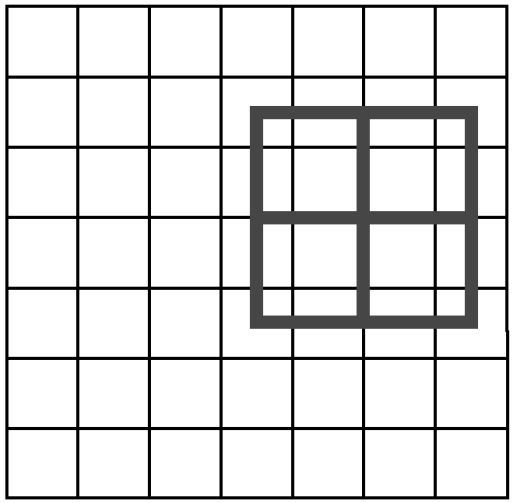
# Texture Mapping

- Standard texture mapping modifies diffuse component $k_{\mathrm{d}}$
  - Pasting a picture onto the polygon

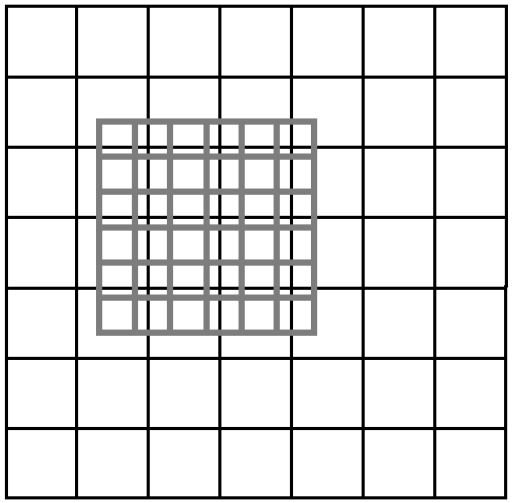
- A texture is a 2D array of texels storing RGB (or RGBA) components

# Difference between pixels and texels

There can be a different match between the pixels of the framebuffer and the texels of the texture



Magnification



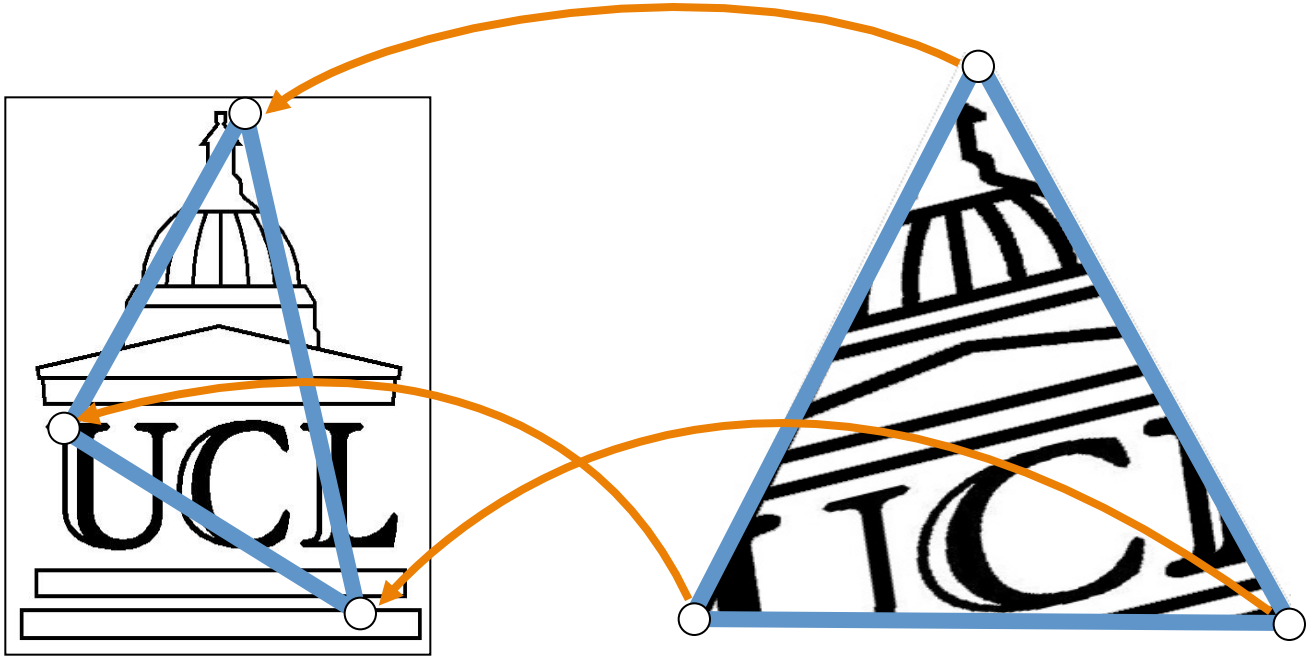
Minification

# Overview

- Texture mapping
  - Inverse and Forward Mapping
  - Bilinear interpolation
  - Perspective correction
- Mipmapping
- Other forms of mapping
  - Environment
  - Bump mapping

# Inverse Mapping

Each vertex is associated with a point on an image $(u, v)$



# Forward Mapping

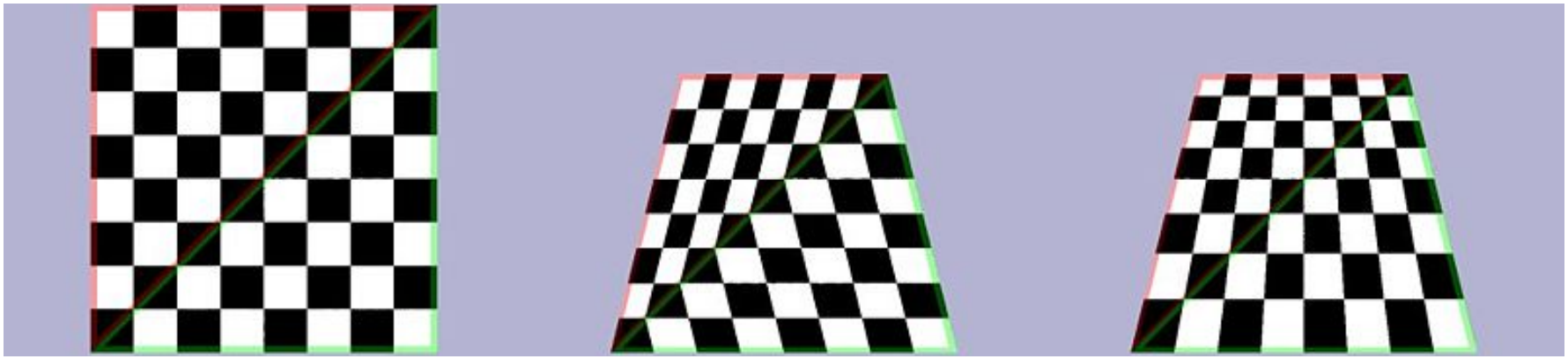
- For points in the texture, map onto the polygon
  - much harder to implement correctly, and harder to model
- Inverse mapping is much more commonly used
  - Most 3D modelers output $u, v$ co-ordinates for texture application

# The Problem

Texture

Side view

Side view

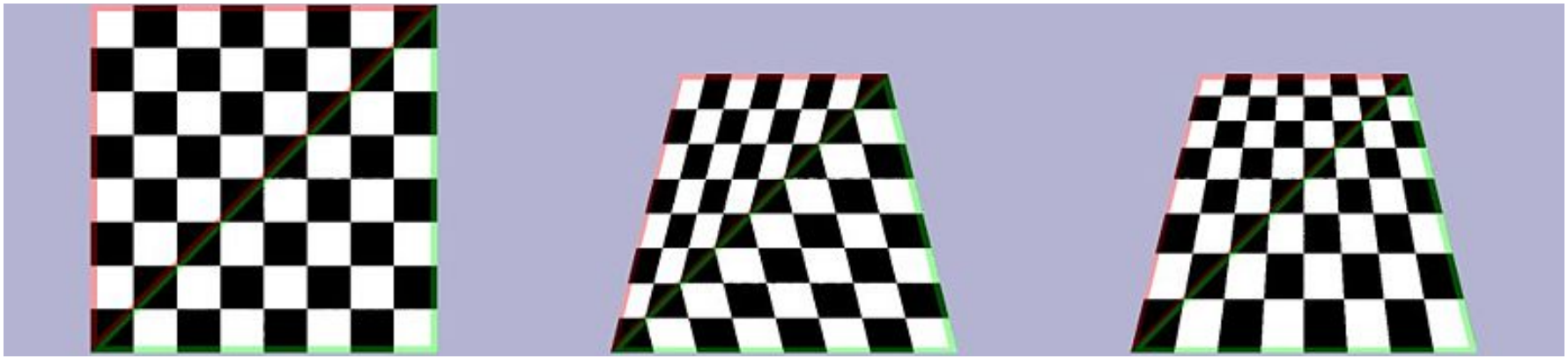


# Why?

Texture

Side view

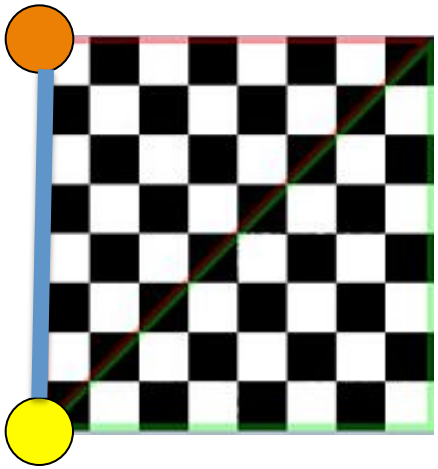
Side view

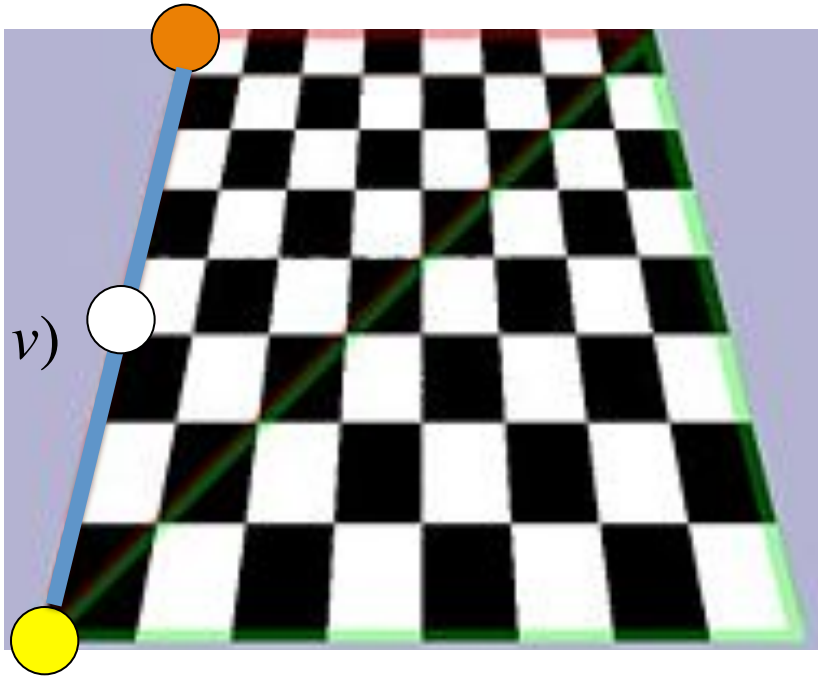


Linear 2D interplation

Linear 3D interplation

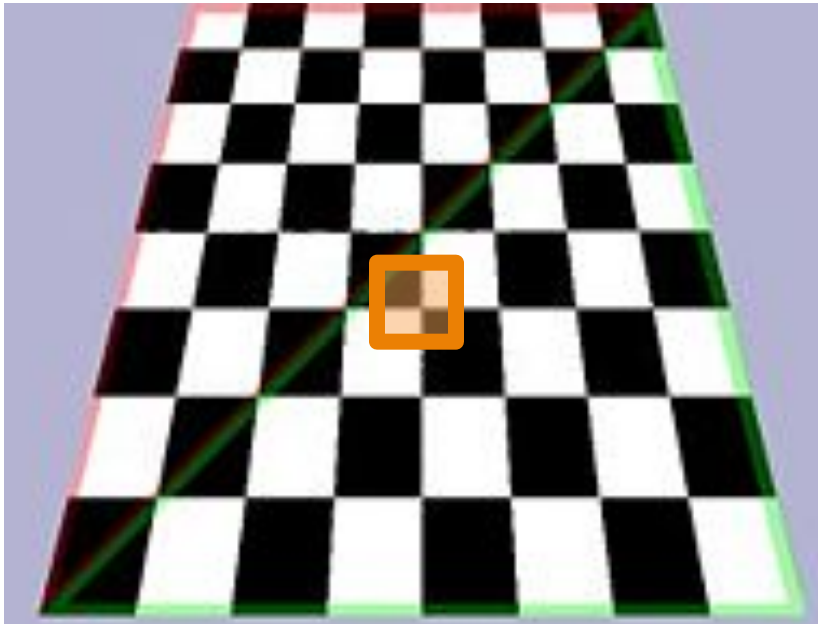
# Perspective interpolation





# Pixels and texels

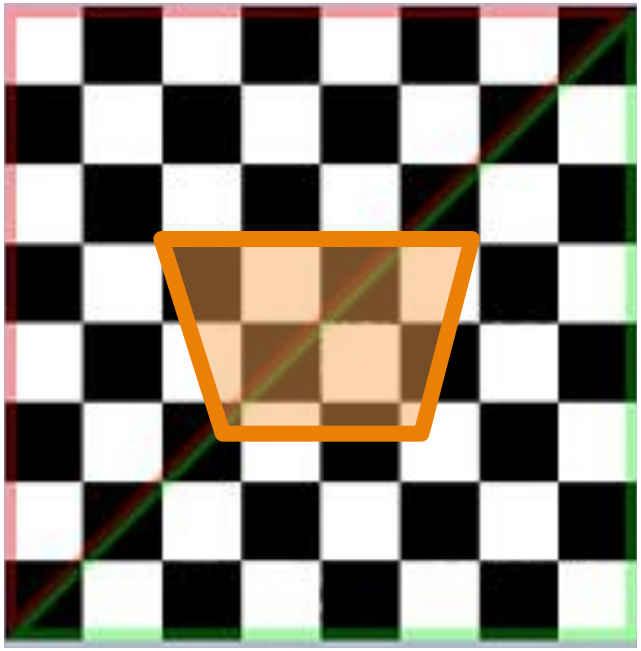
Texture



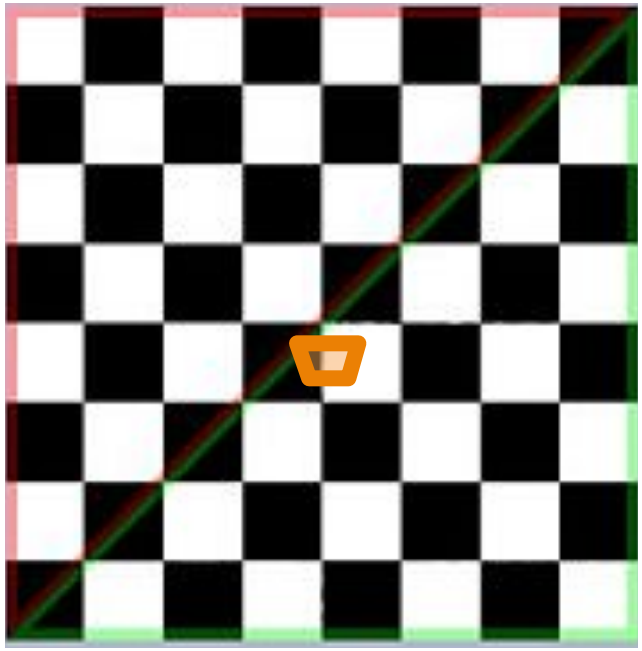
3D view

# Sampling

- A pixel maps to a non-rectangular region
- Usually only perform map on centre of pixel
- Problem: Under and over samping



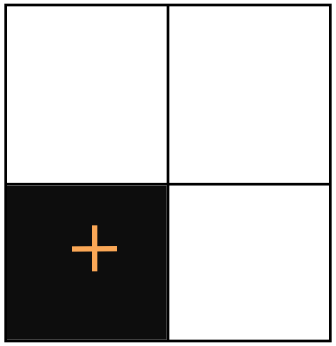
Oversampling



Undersampling

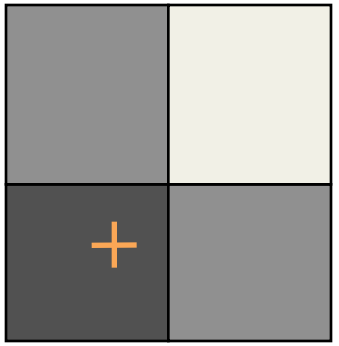
# Undersampling solution: Filtering

Nearest neighbour



Pick texel with closest centre

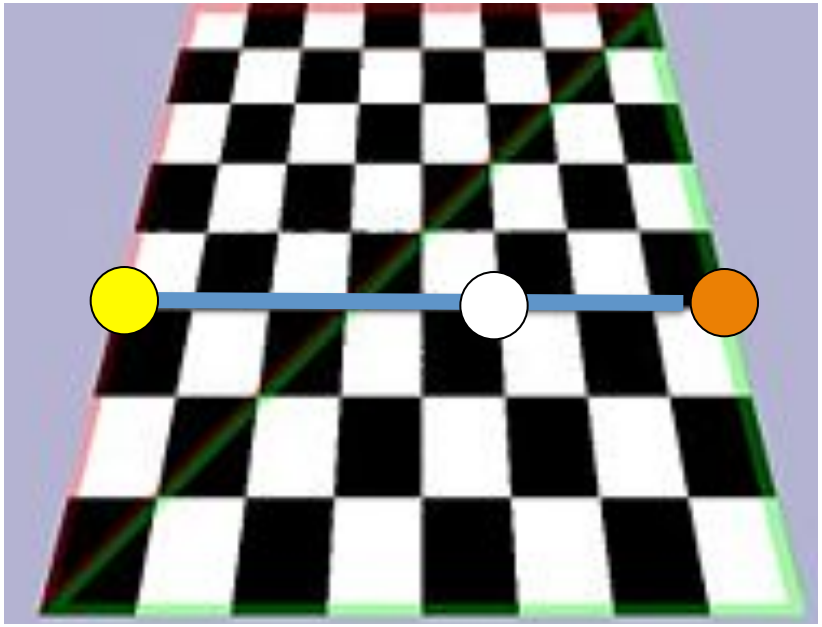
Bilinear



Weighted average based on distance to texel centre

# Filtering

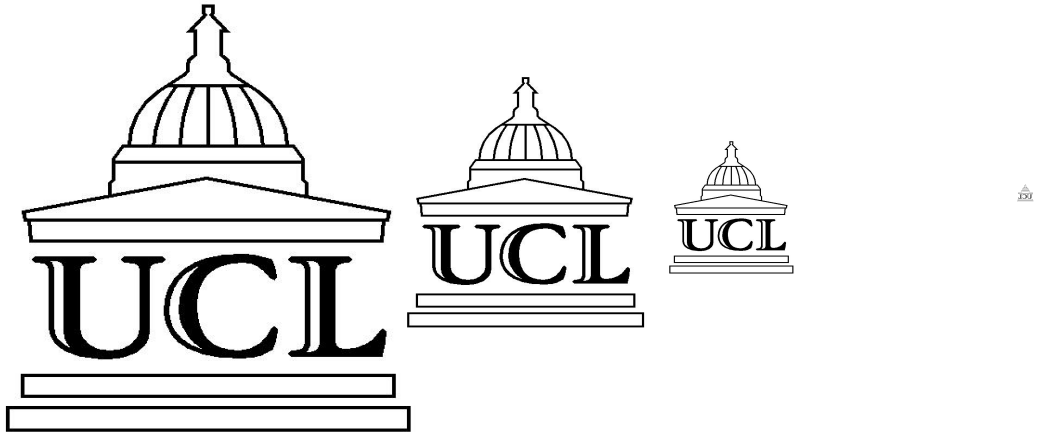
- Bilinear filtering (partially) solves the undersampling problem since it provides smooth shading between texels



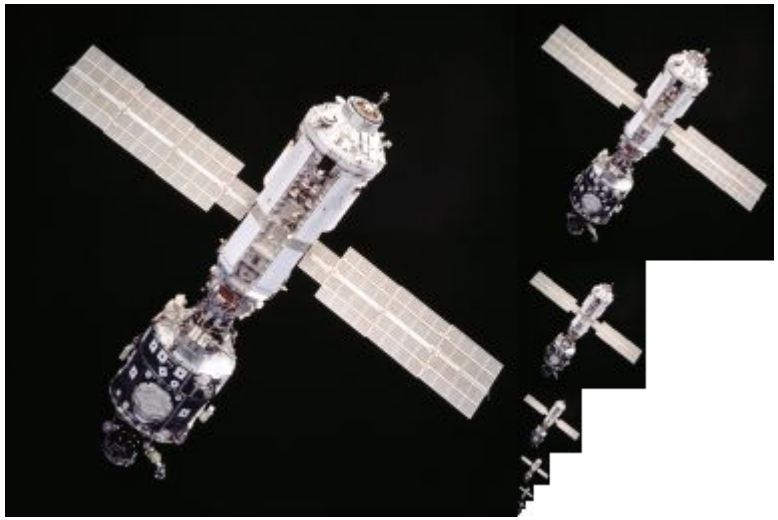
# MIP-Mapping

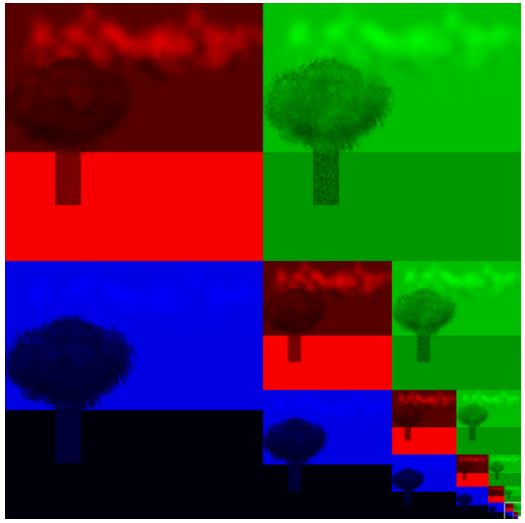
- When oversampling we use **MIP-mapping**
- Resample image at lower resolution
- Create a “pyramid” of textures.
- Interpolate texture between two adjacent layers

# Texture Pyramid



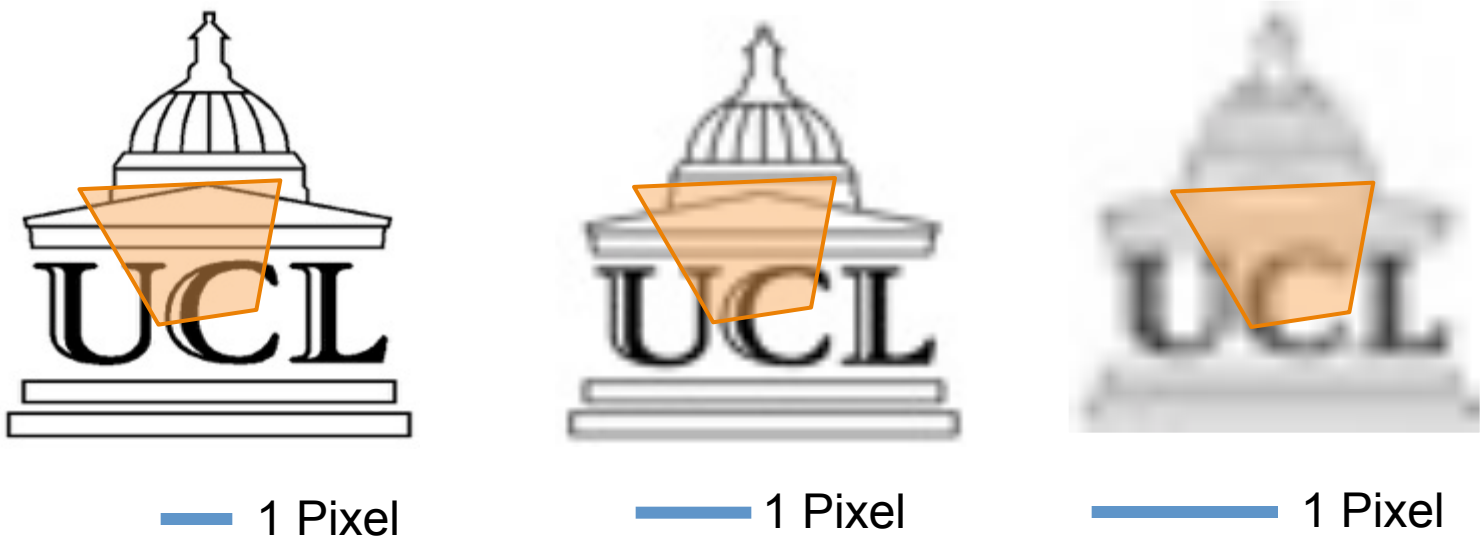

128x128 64x64 32x32 ... 1x1





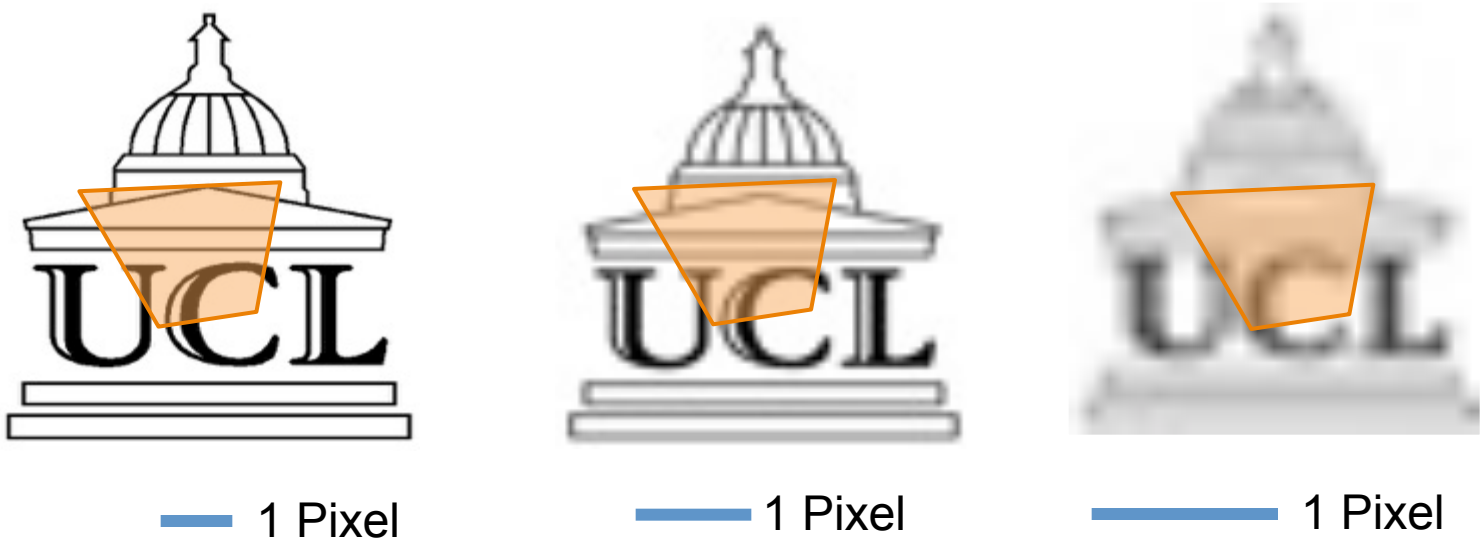
# Linear MIP Sampling

Choose the level of the MIP-map based on the $\mathrm{d}u$ and $\mathrm{d}v$ for $\mathrm{d}x$ and $\mathrm{d}y$ are closest to 1 pixel



# Tri-linear MIP Sampling

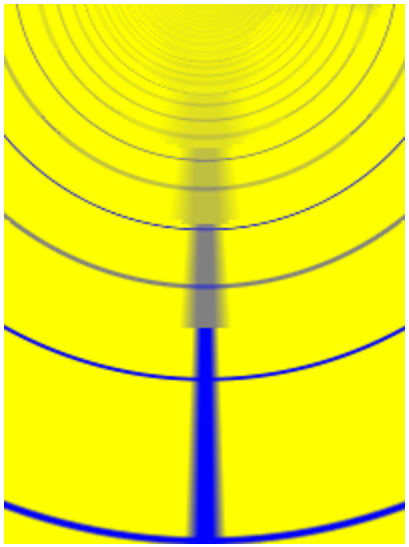
Choose **two** level and after interpolating within the levels, interpolate between the outcome



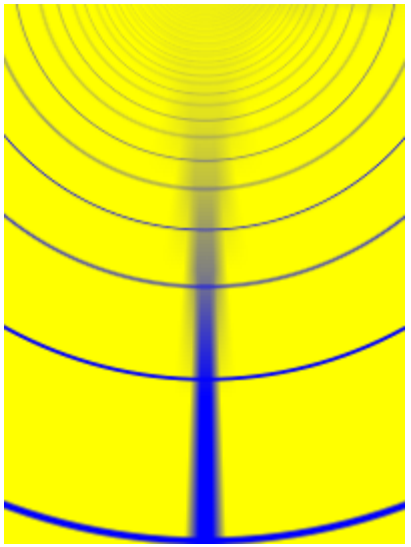
# Filtering Examples

Nearest Neighbor

Bilinear Filtering

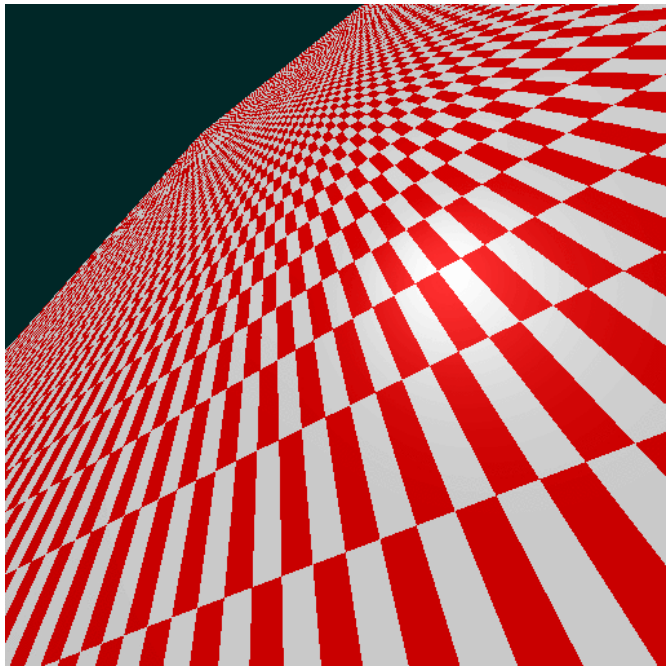


Bilinear Filtering (distinct MIP map levels)

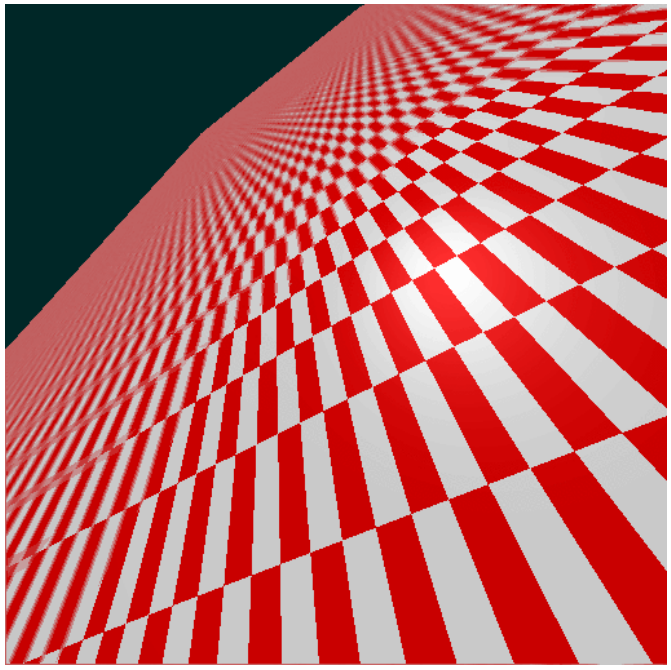


Trilinear Filtering (Mip mapping)

# More Examples



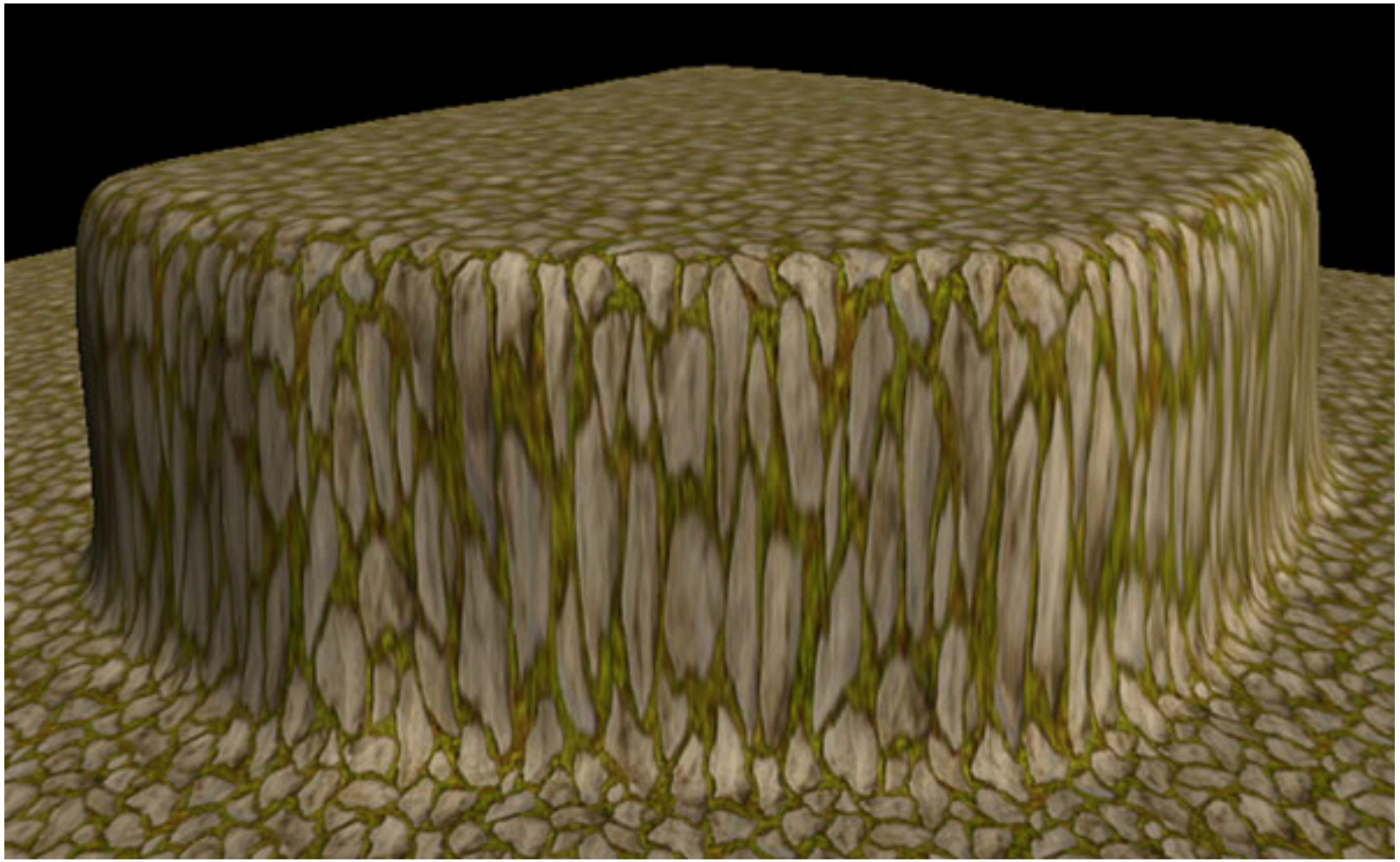
Nearest Neighbor



Mip Mapping

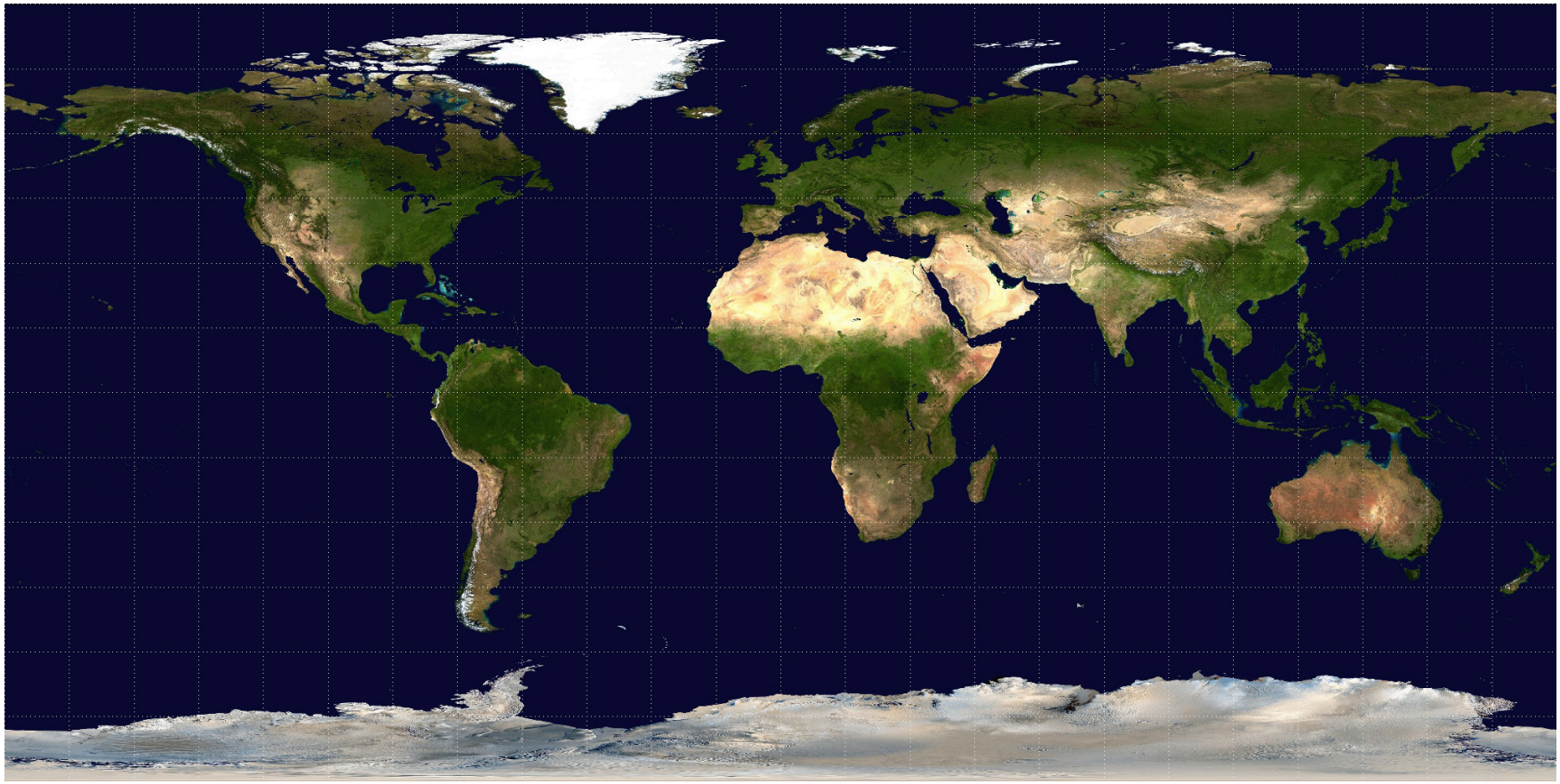
Rendering speed increases since the number of texture pixels ("texels") being processed can be much lower with the simple textures. Artifacts are reduced since the mipmap images are effectively already anti-aliased, taking some of the burden off the real-time renderer.

# Where does $u, v$ come from?



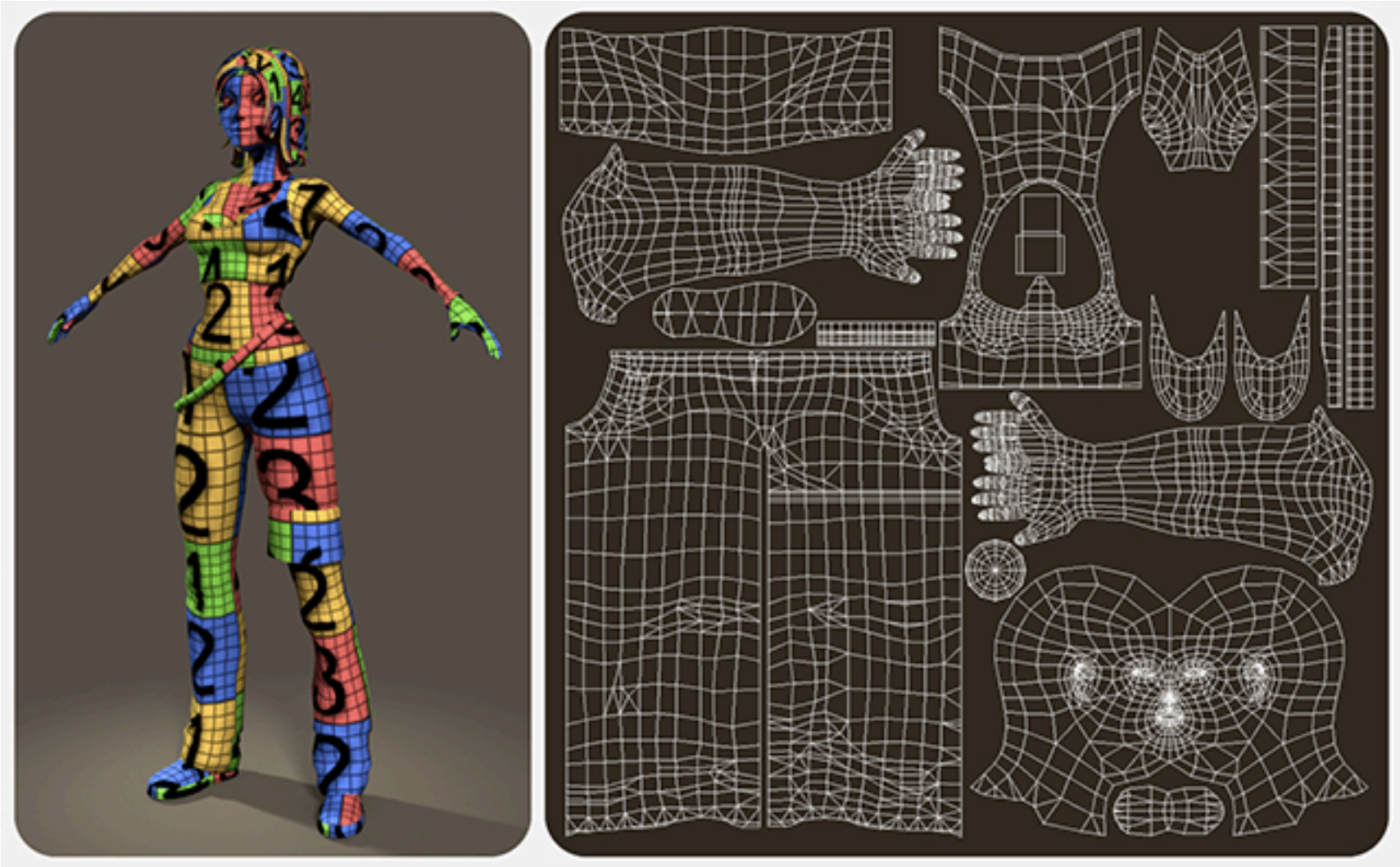
Planar projection

# Where does $u, v$ come from?



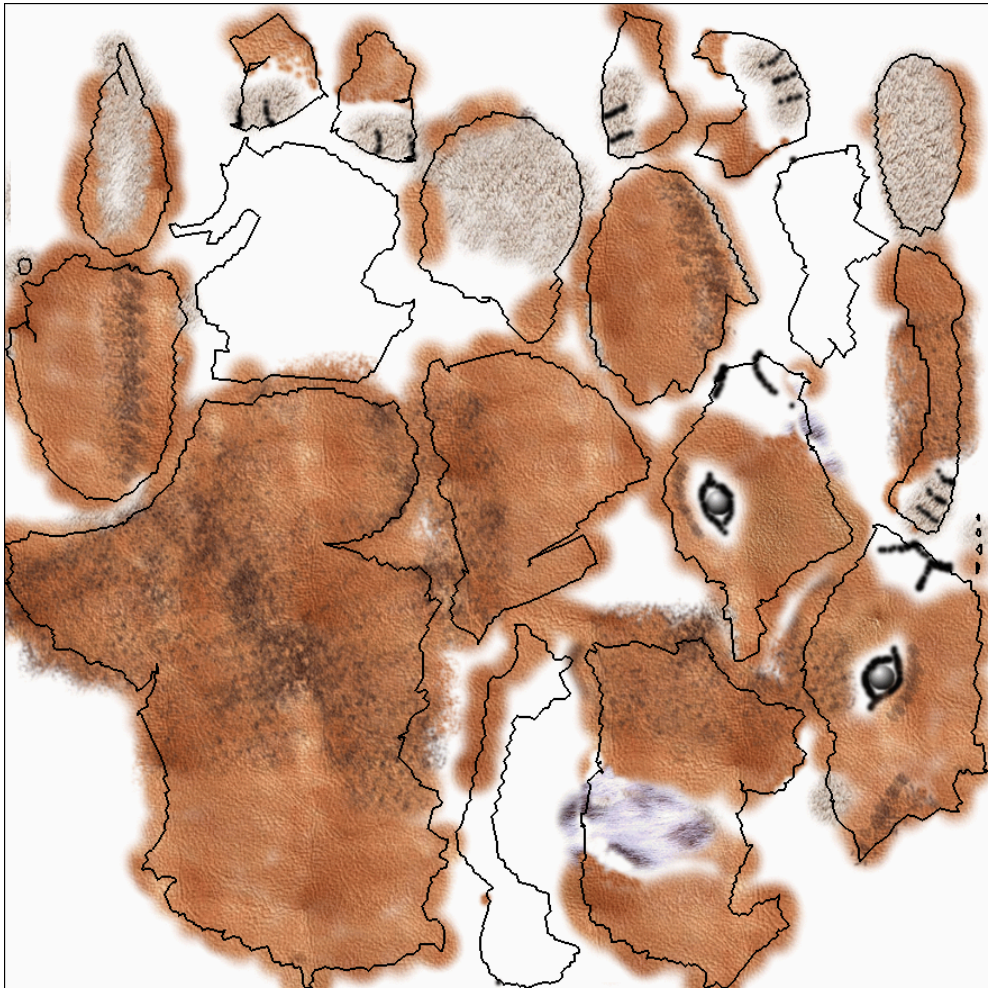
Spherical projection

# Where does $u, v$ come from?



Charts, done manually

# Where does $u, v$ come from?



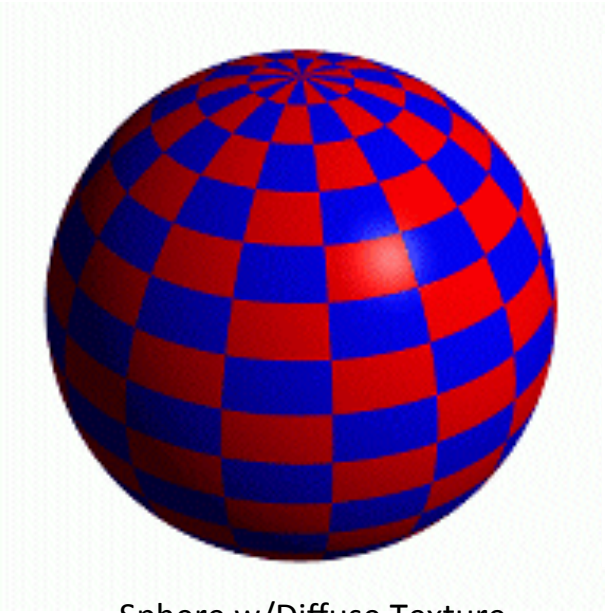
Charts, done automatically

# Other Forms of Texture Mapping

1. Bump Mapping
2. Displacement Mapping
3. Environment Mapping

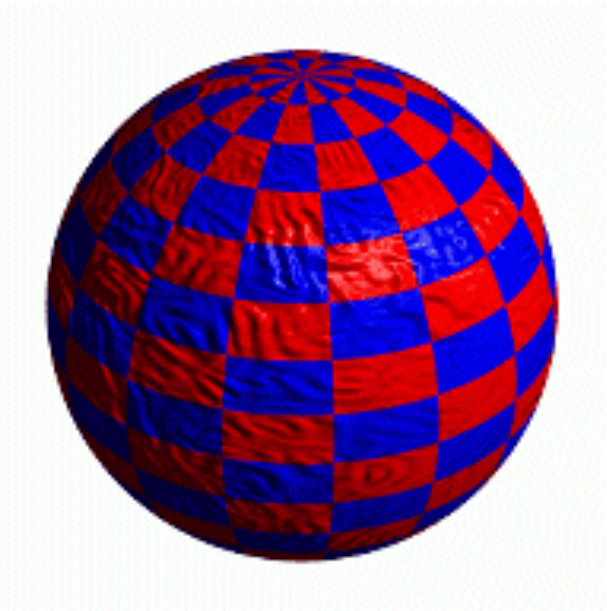
# Bump Mapping

- Use textures to alter the surface normal
  - Does not change the actual shape of the surface
  - Just shaded as if it were a different shape



Sphere w/Diffuse Texture

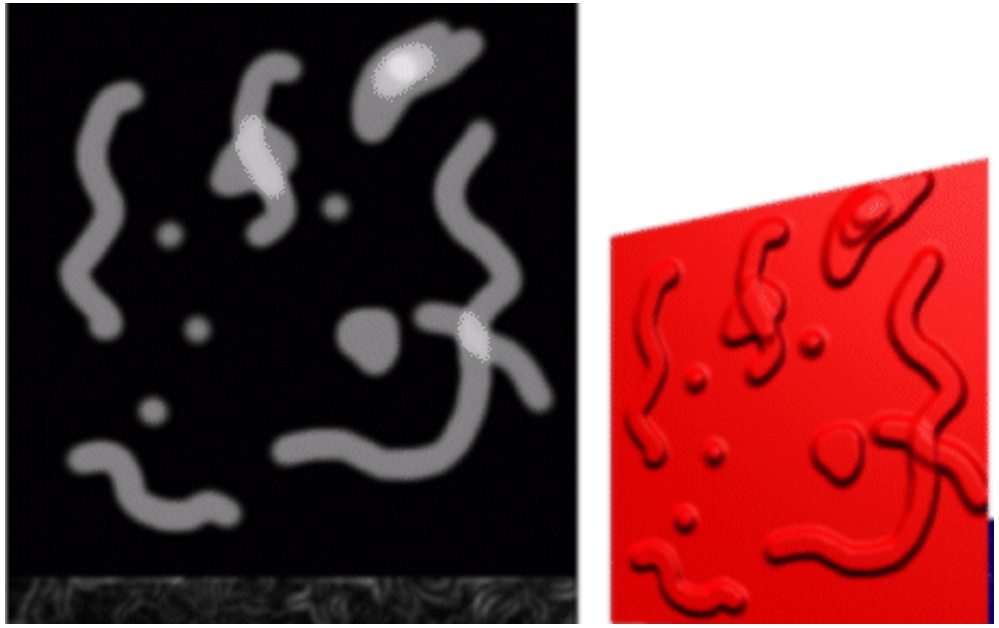
Swirly Bump Map



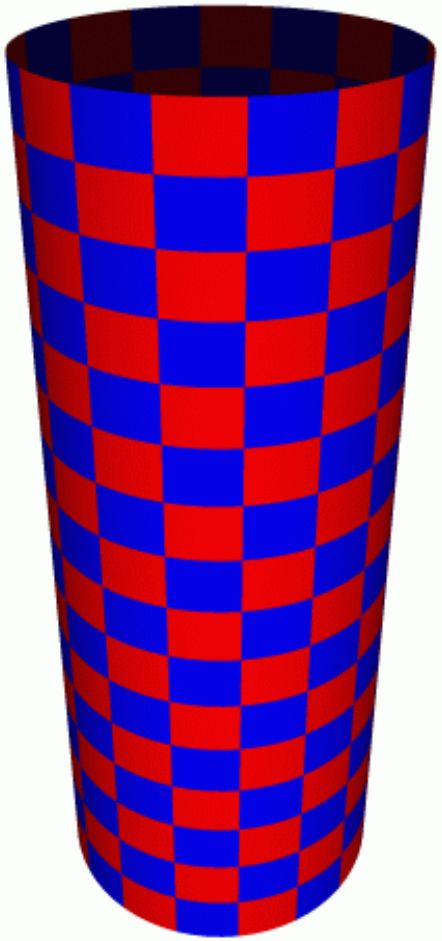
Sphere w/Diffuse Texture & Bump Map

# Bump Mapping

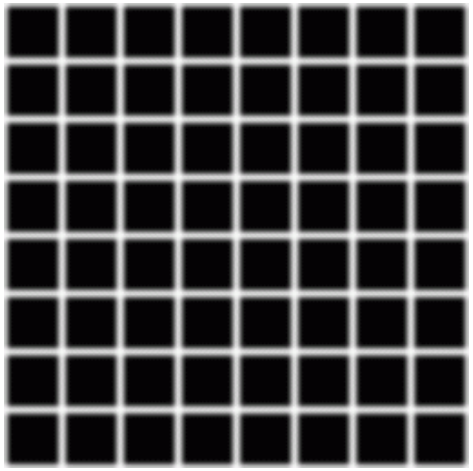
- Treat the texture as a single-valued height function
- Compute the normal from the partial derivatives in the texture



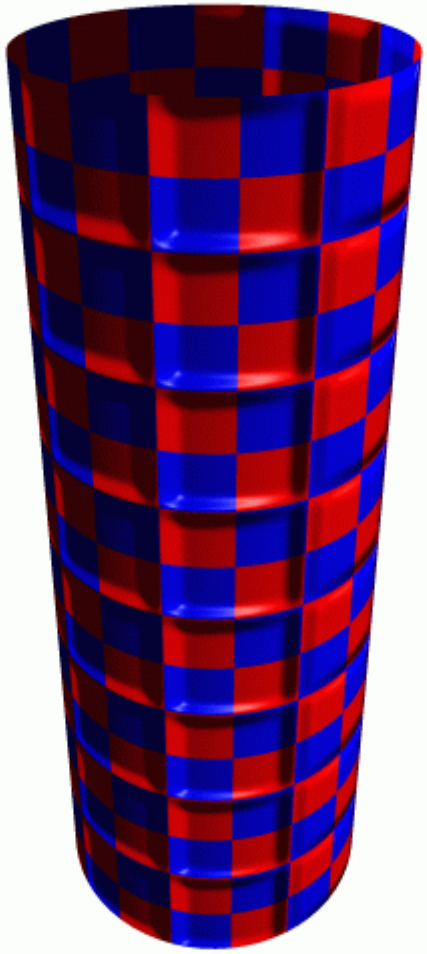
# Another Bump Map Example



Cylinder w/Diffuse Texture Map



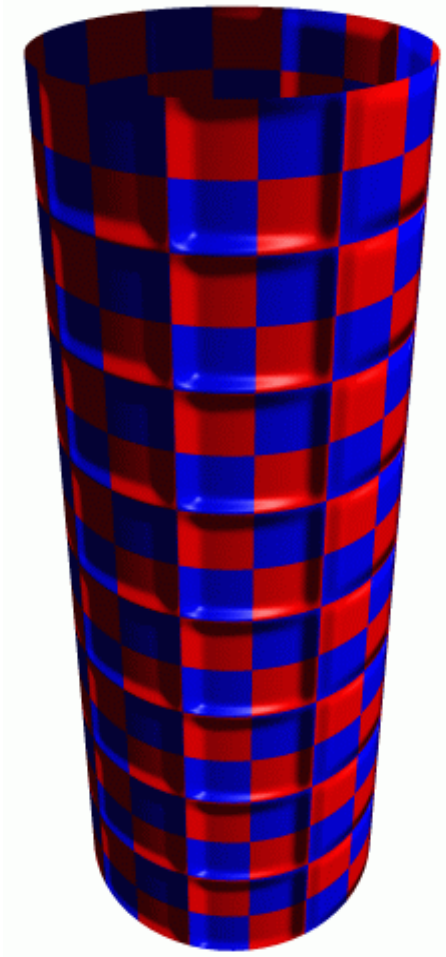
Bump Map



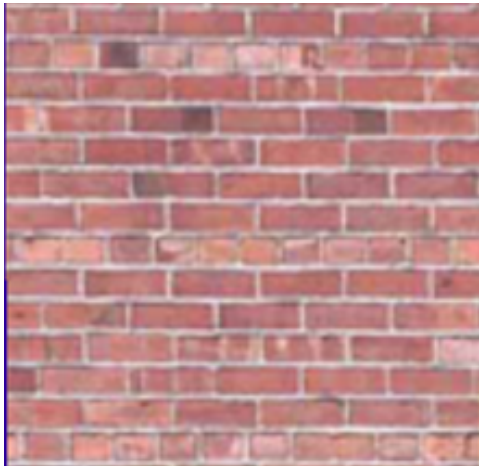
Cylinder w/Texture Map & Bump Map

# What's Missing?

- There are no bumps on the silhouette of a bump-mapped object

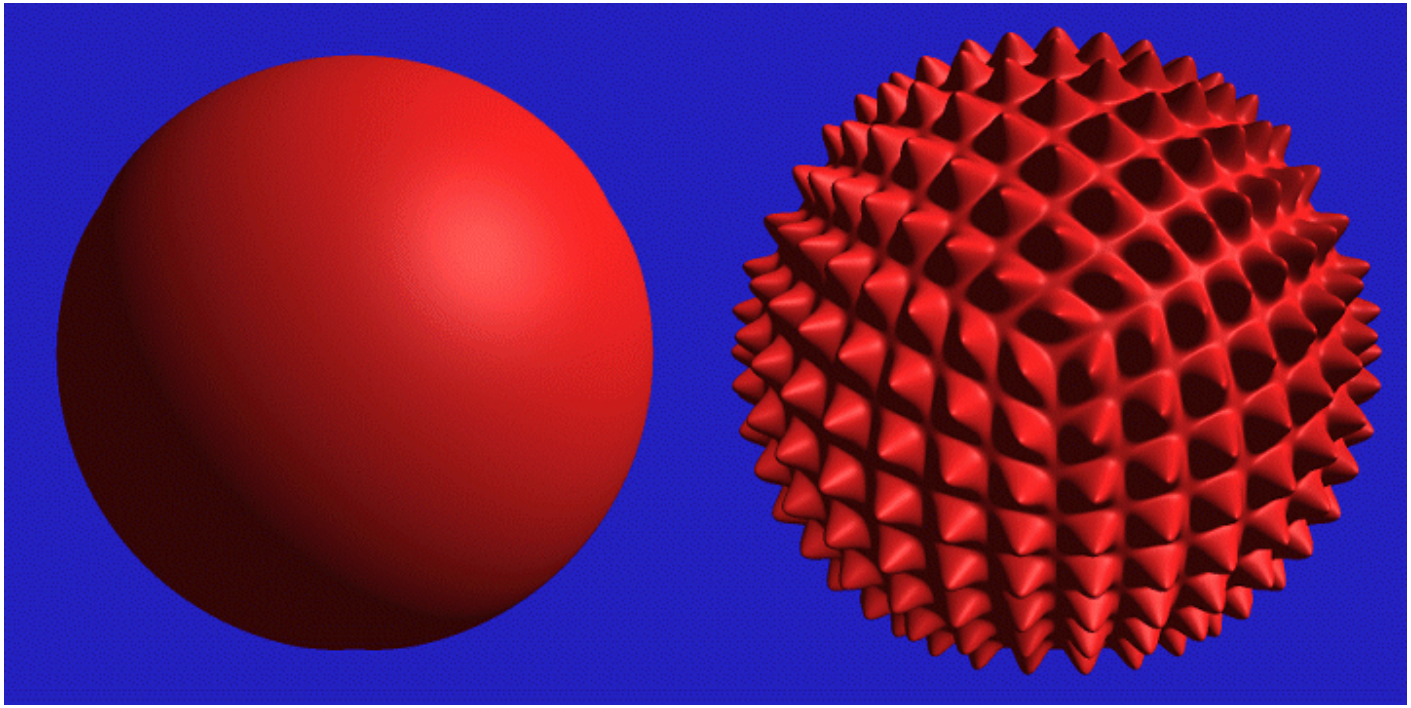


- Bump maps don’t allow self-occlusion or self-shadowing



# Displacement Mapping

- Use the texture map to actually move the surface point
- The geometry must be displaced before visibility is determined



# Displacement Mapping

Image from:

*Geometry Caching for Ray-Tracing Displacement Maps*

by Matt Pharr and Pat Hanrahan.

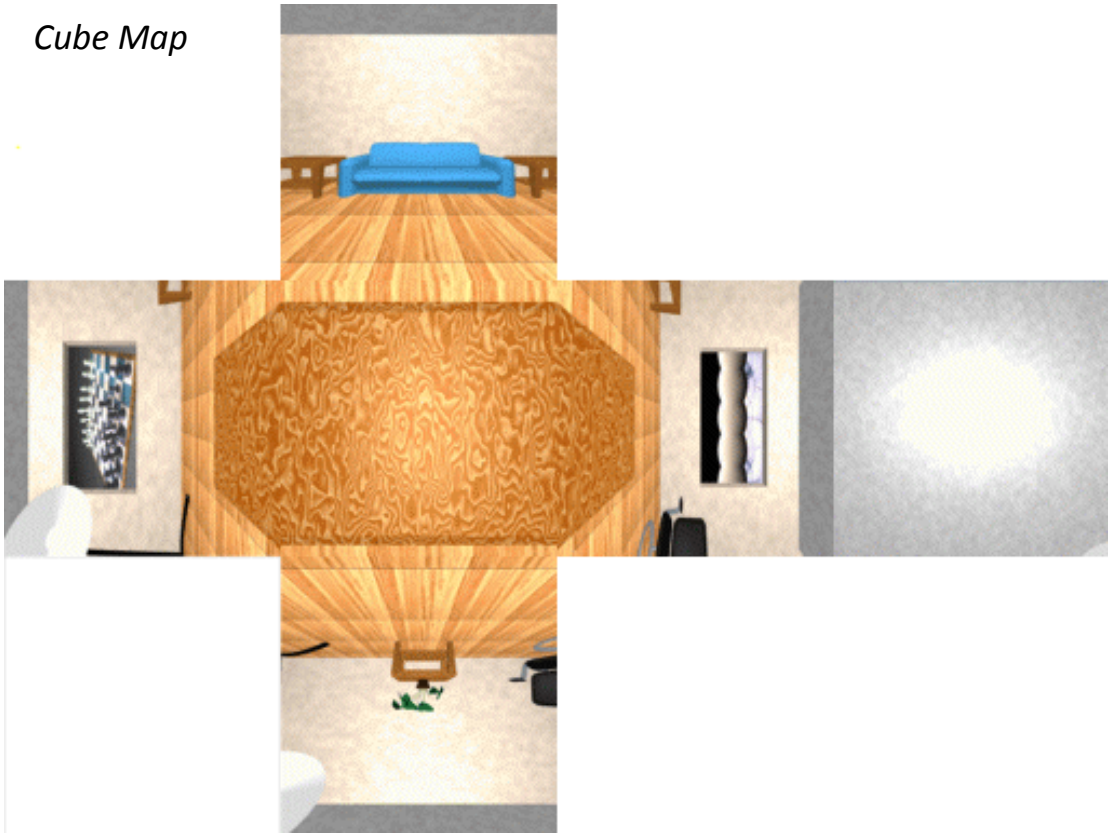
*Note the detailed shadows cast by the stones*

# Environment Maps

- We can simulate reflections by using the direction of the reflected ray to index a spherical texture map at "infinity".
- Assumes that all reflected rays begin from the same point.

# What's the Best Layout?

*Cube Map*



*Lattitude Map*

*GL Map*

# Environment Mapping Example

Terminator II

# Recap

- Texture Mapping adds detail to otherwise simple geometry
- “Texture” can mean different modifications to the calculation of lighting, or can even displace geometry locally
- Sampling issues are very important
- To some extent current graphics cards are built around attempting to do texturing efficiently.