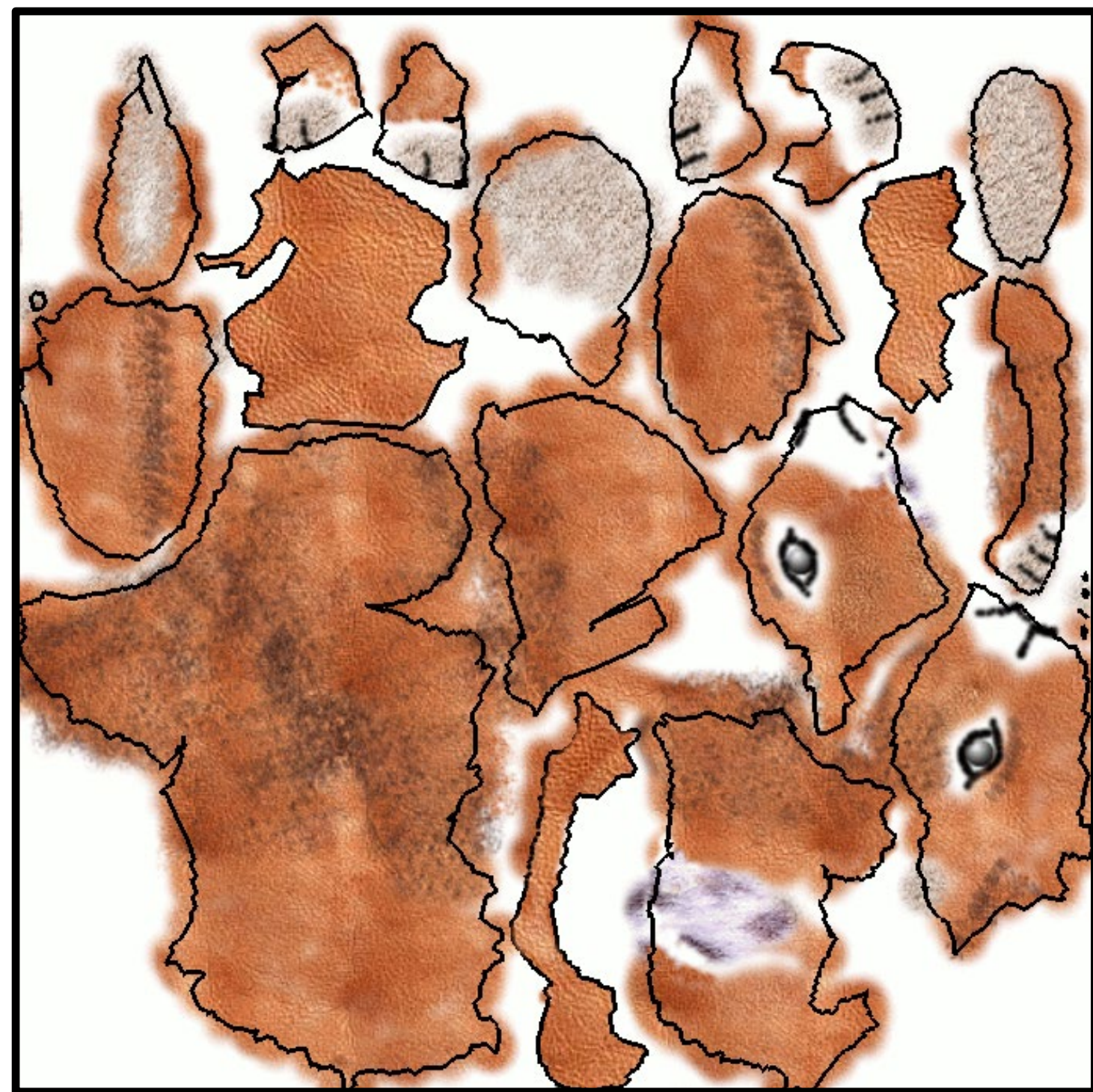


Texture mapping



15-468, 15-668, 15-868
Physics-based Rendering
Spring 2023, Lecture 4

Course announcements

- Programming assignment 0 posted, due whenever.
- Programming assignment 1 posted, due on Friday 2/10.
- Take-home quiz 1 posted, due **tonight**.
- Take-home quiz 2 posted, due Tuesday 2/7.
- First recitation tomorrow.

Overview of today's lecture

- Texture mapping.

Slide credits

Most of these slides were directly adapted from:

- Wojciech Jarosz (Dartmouth).

Why texture mapping?

Real objects have spatially varying details

Representing as geometry is correct, but tedious/expensive



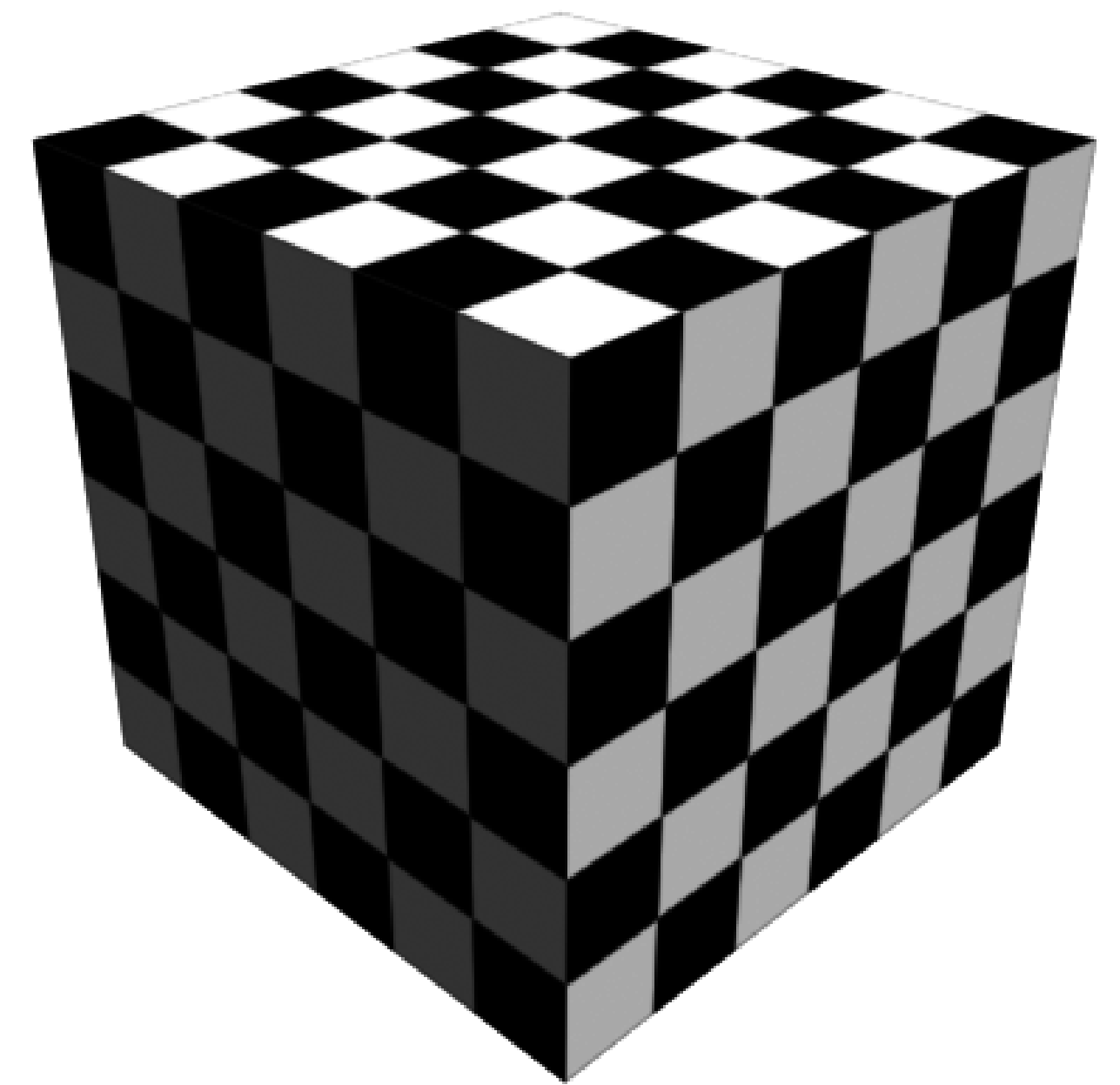
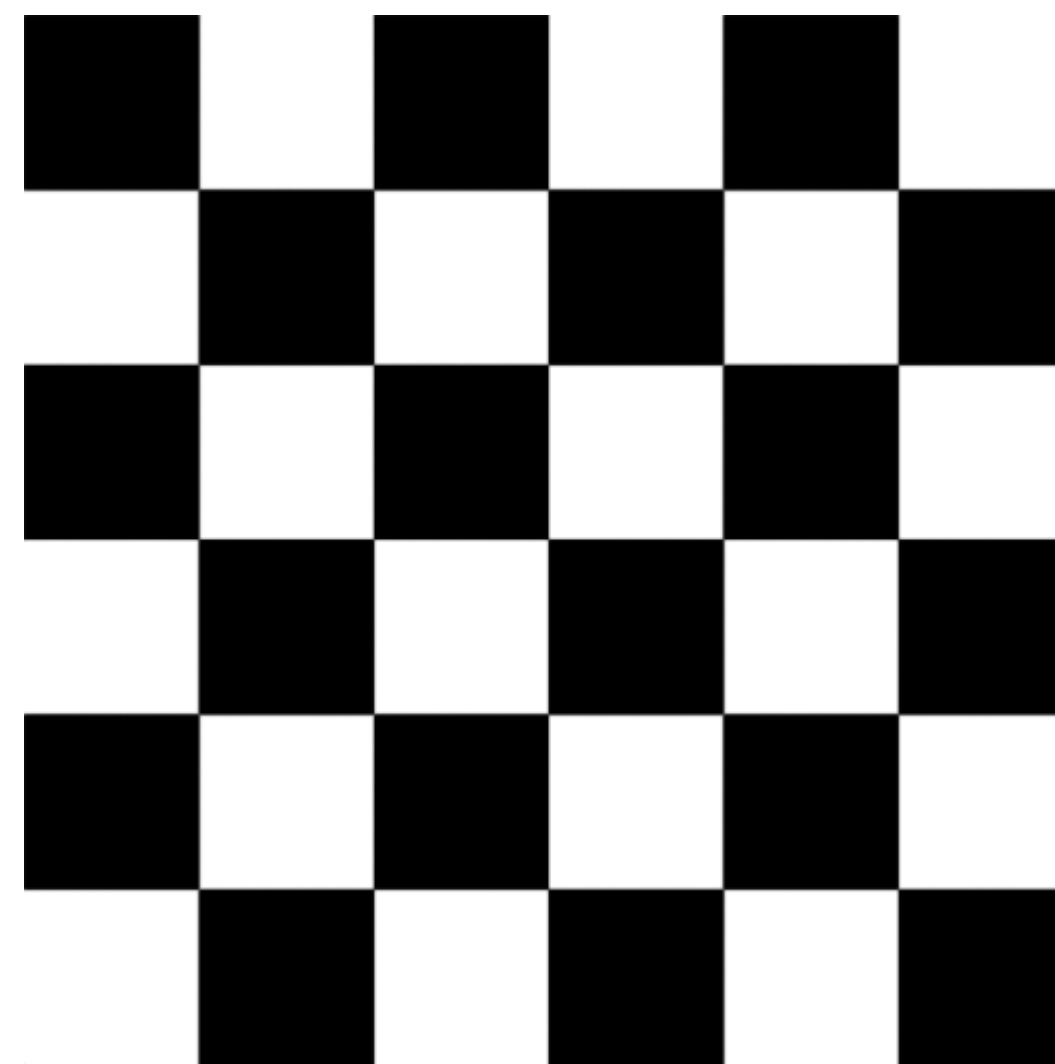
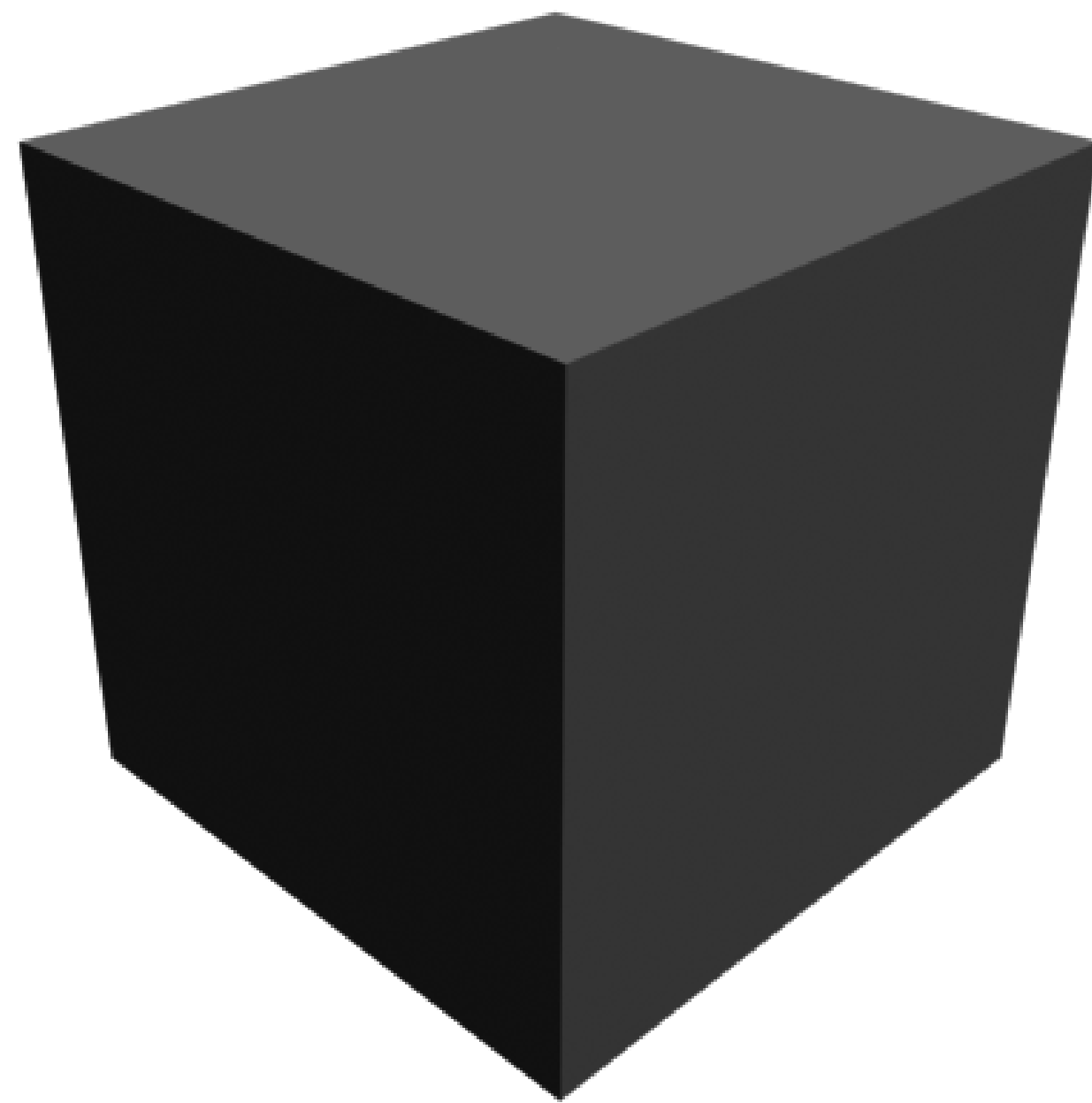
Why texture mapping?

Use simple geometry

Store varying properties in images

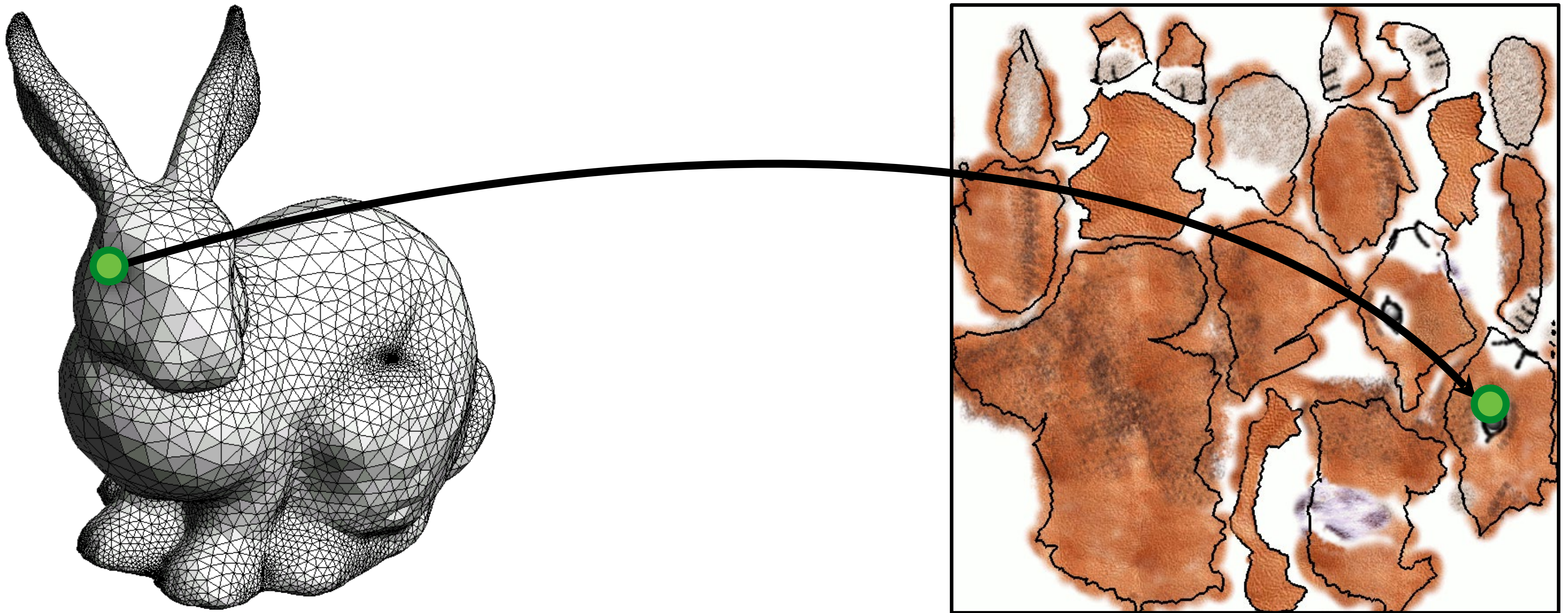
Map to objects

Proposed by
Ed Catmull in the 70s



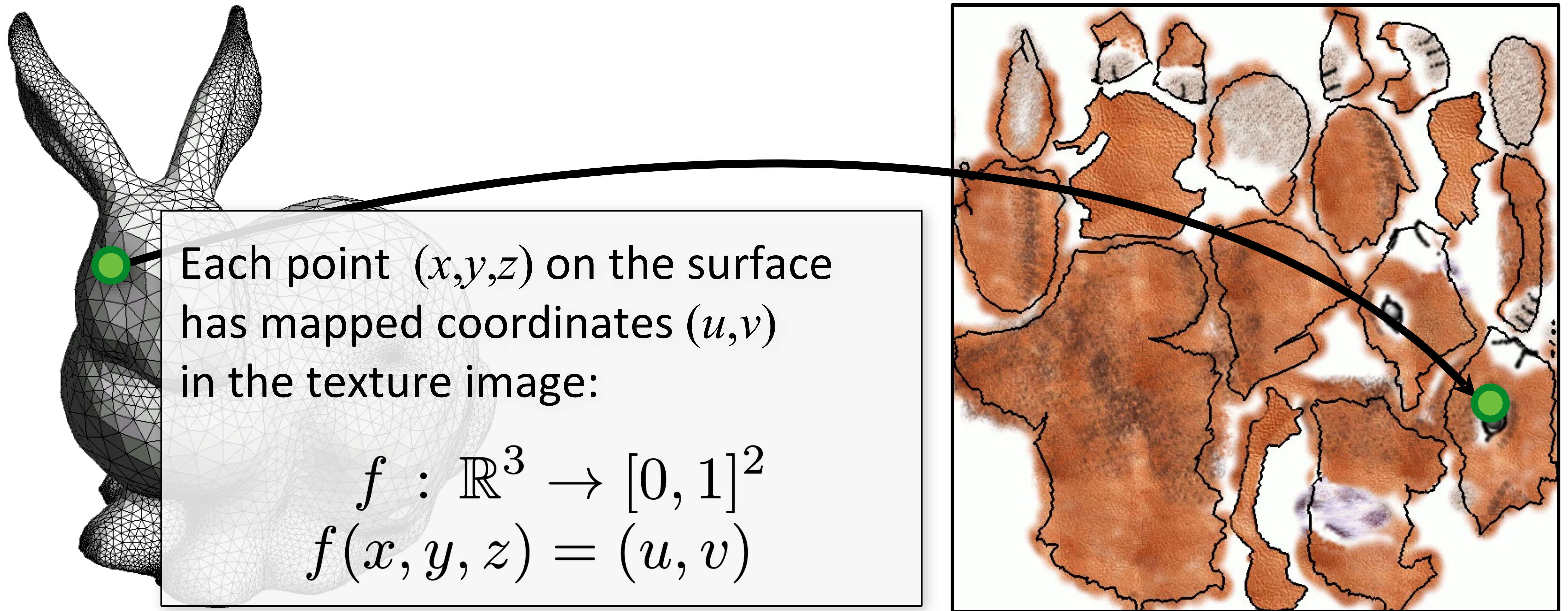
Texture mapping – definition

Mapping between the surface and the image



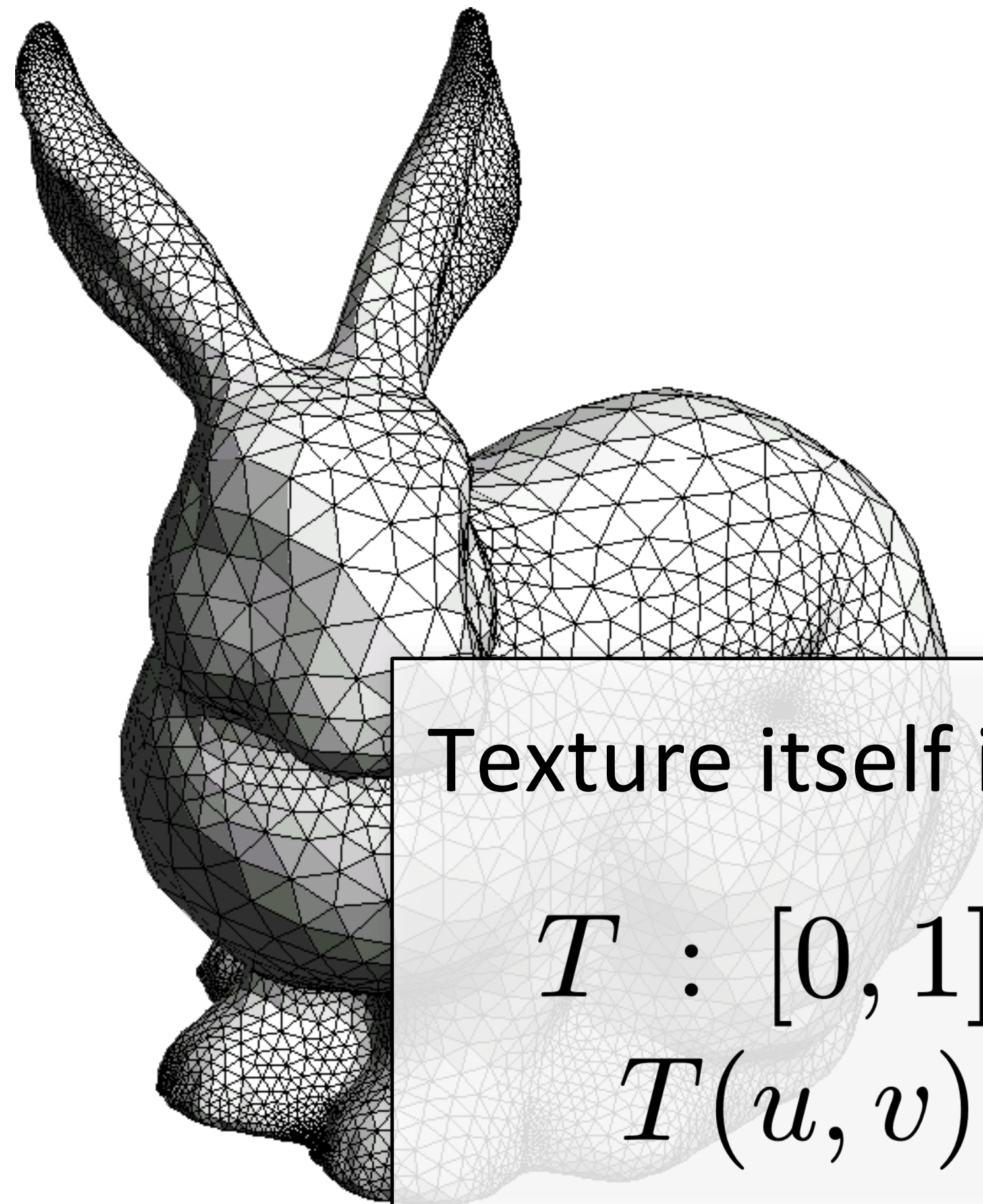
Texture mapping – definition

Mapping between the surface and the image



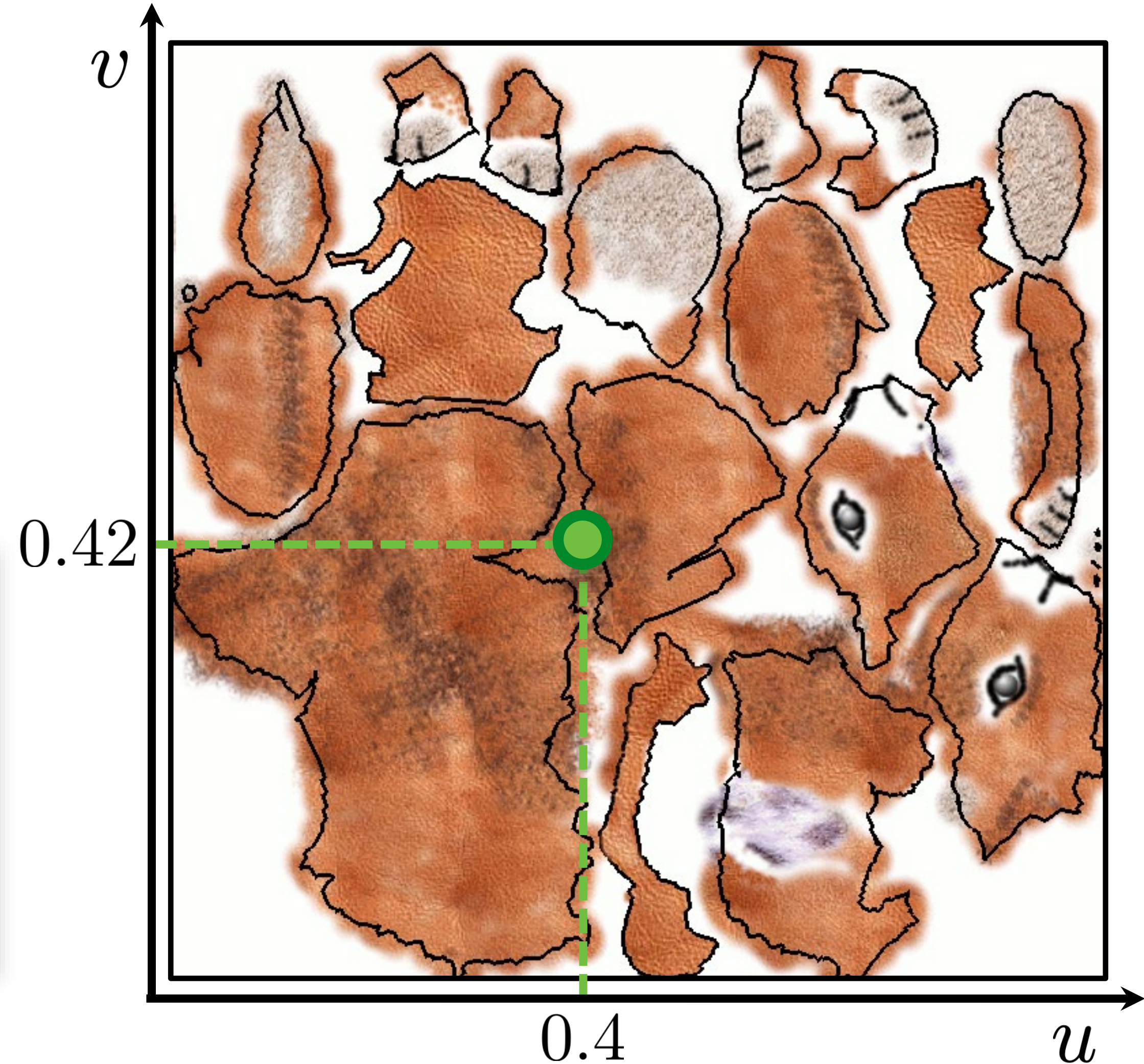
Texture mapping – definition

Mapping between the surface and the image



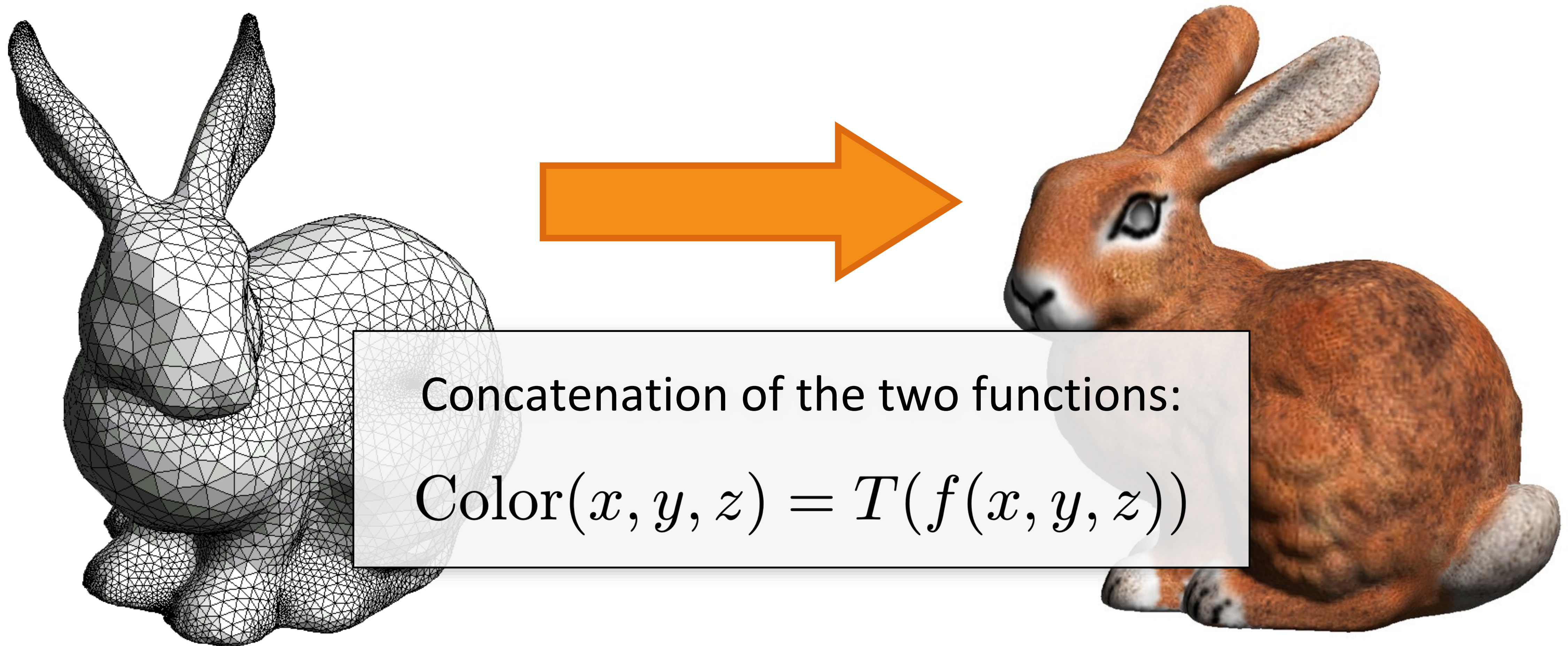
Texture itself is a function:

$$T : [0, 1]^2 \rightarrow \text{RGB}$$
$$T(u, v) = (r, g, b)$$



Texture mapping – definition

Mapping between the surface and the image

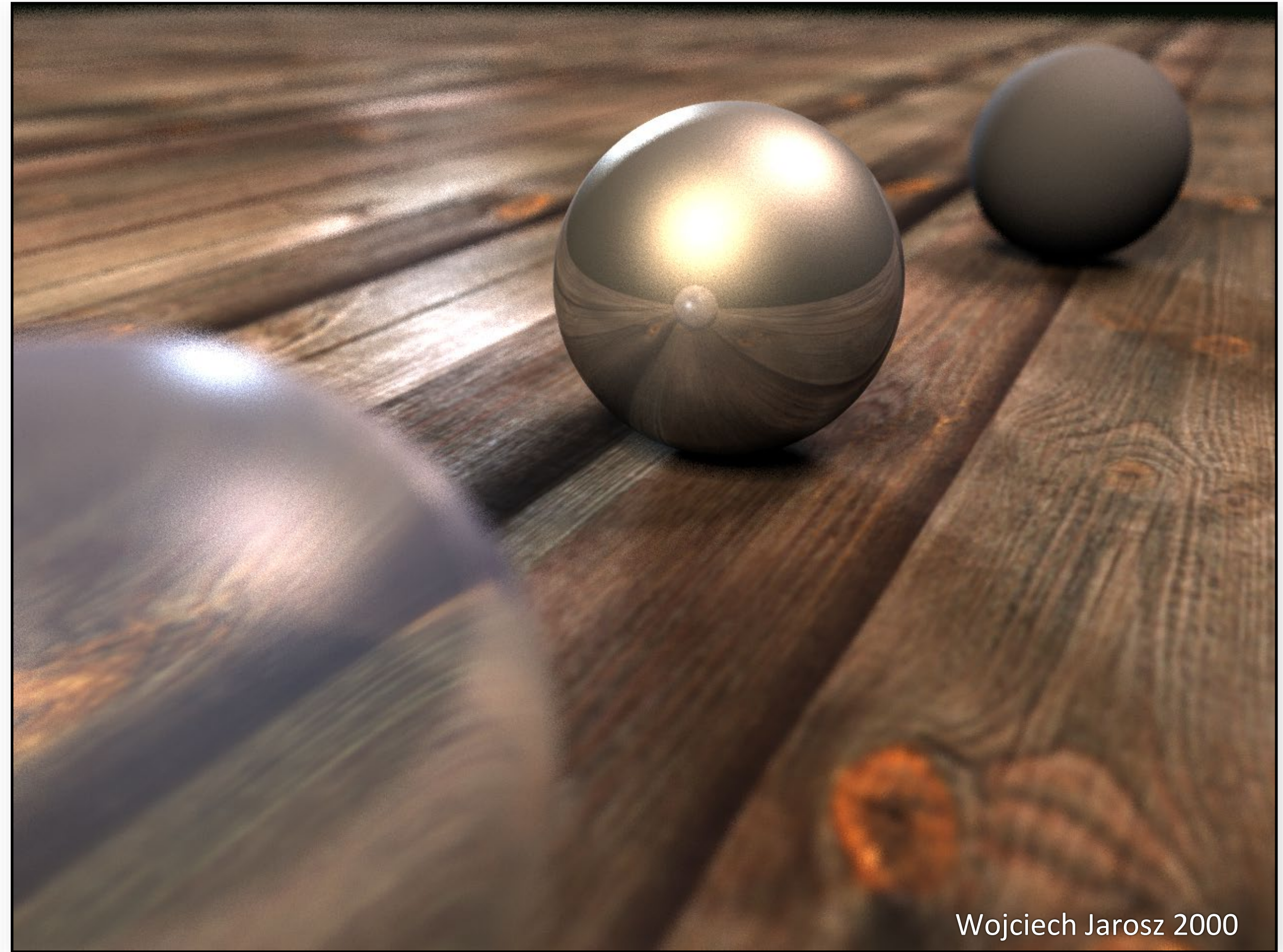
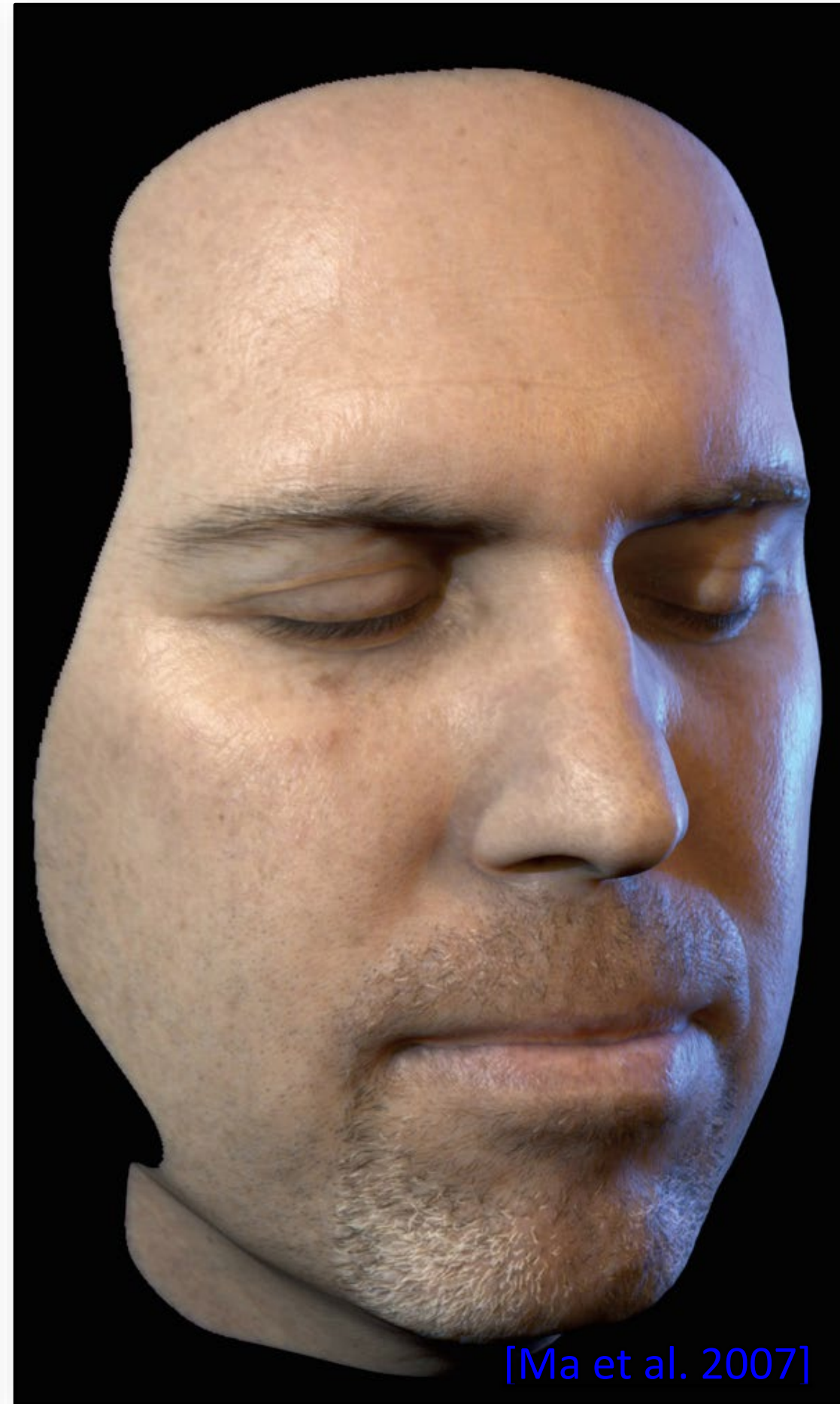
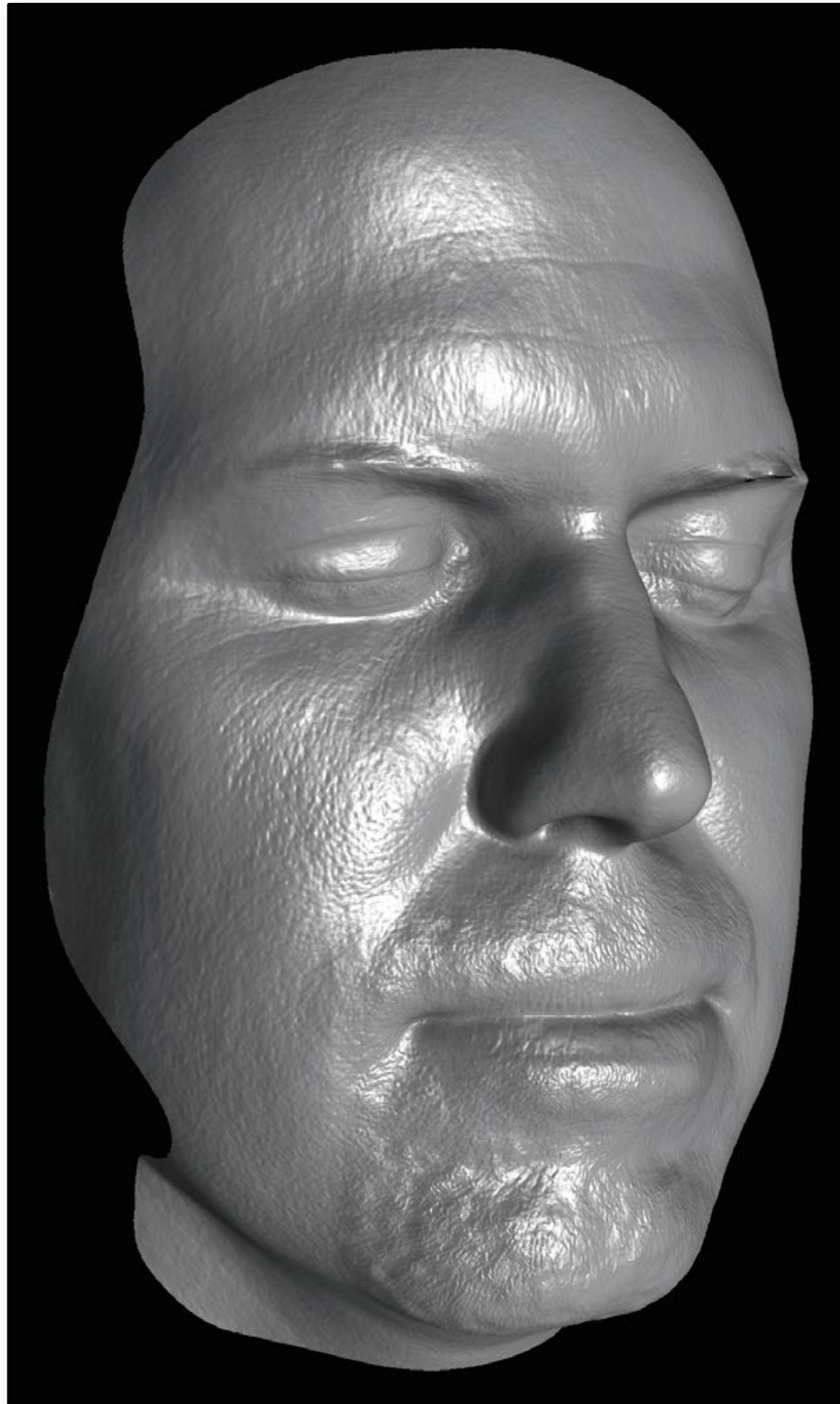


Concatenation of the two functions:

$$\text{Color}(x, y, z) = T(f(x, y, z))$$

Why texture mapping?

Produces compelling results



Agenda

How do we map between surface and texture?

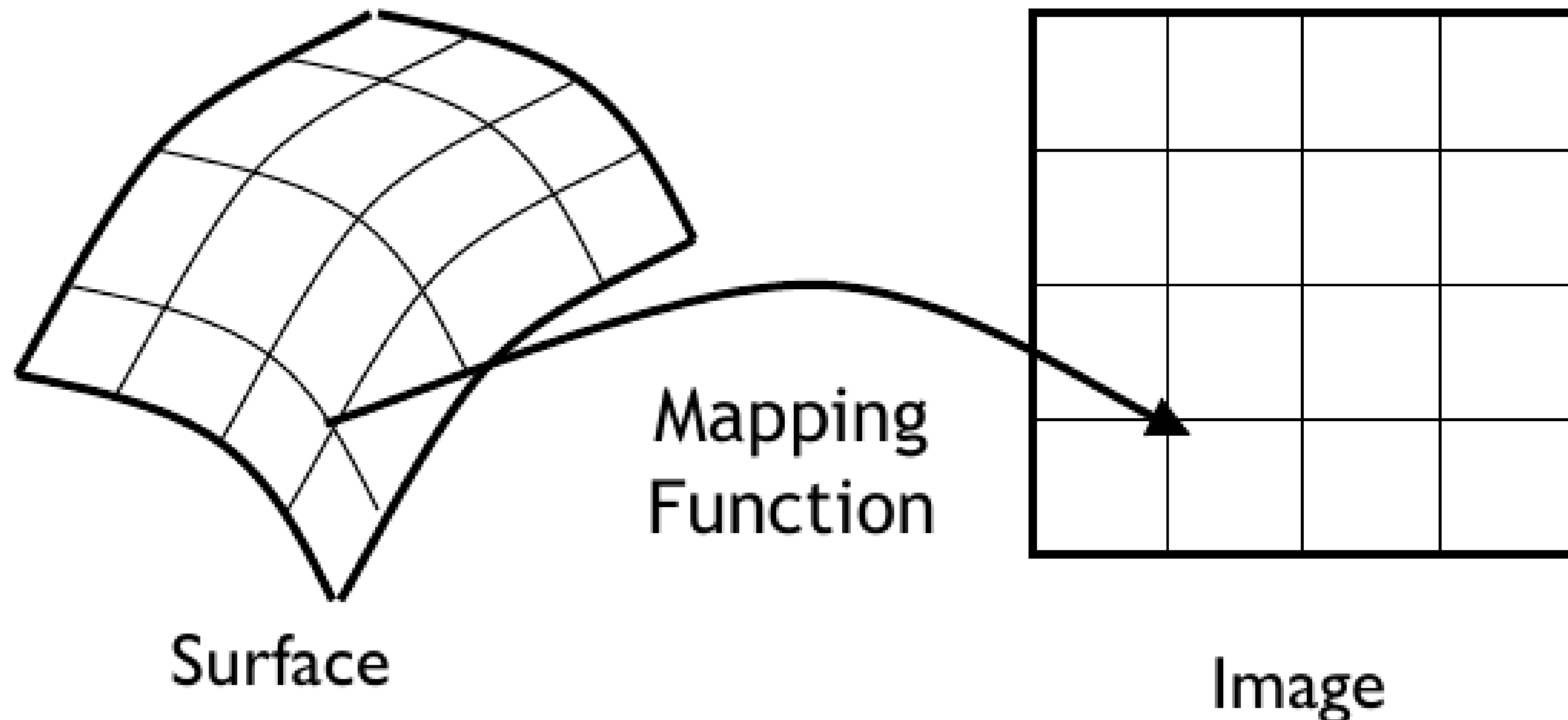
What do we map onto the surface?

Surface parameterization

A surface in 3D is a two-dimensional thing

How do we map a surface point to a point in a texture?

- Defining 2D coordinates is parameterizing the surface



Surface parameterization

A surface in 3D is a two-dimensional thing

How do we map a surface point to a point in a texture?

- Defining 2D coordinates is parameterizing the surface

Examples:

- cartesian coordinates on a rectangle (or other planar shape)
- cylindrical coordinates (θ, y) on a cylinder
- latitude and longitude on the Earth's surface
- spherical coordinates (θ, ϕ) on a sphere

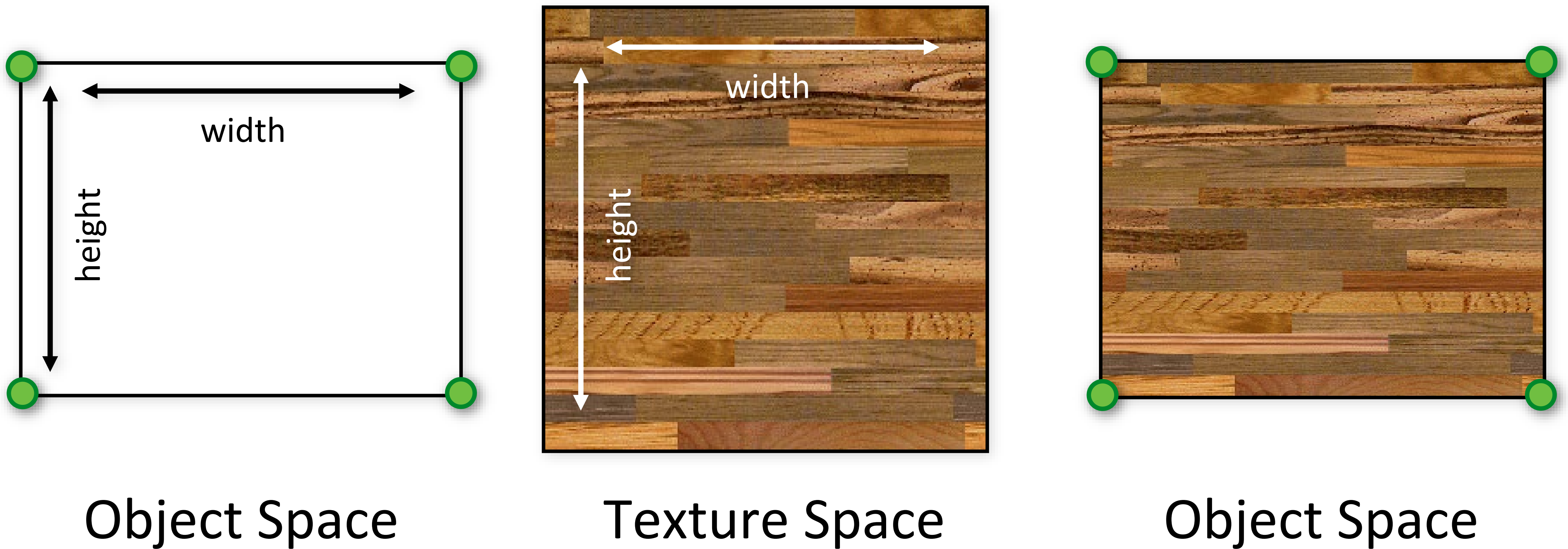
A rectangle

Image can be mapped directly, unchanged



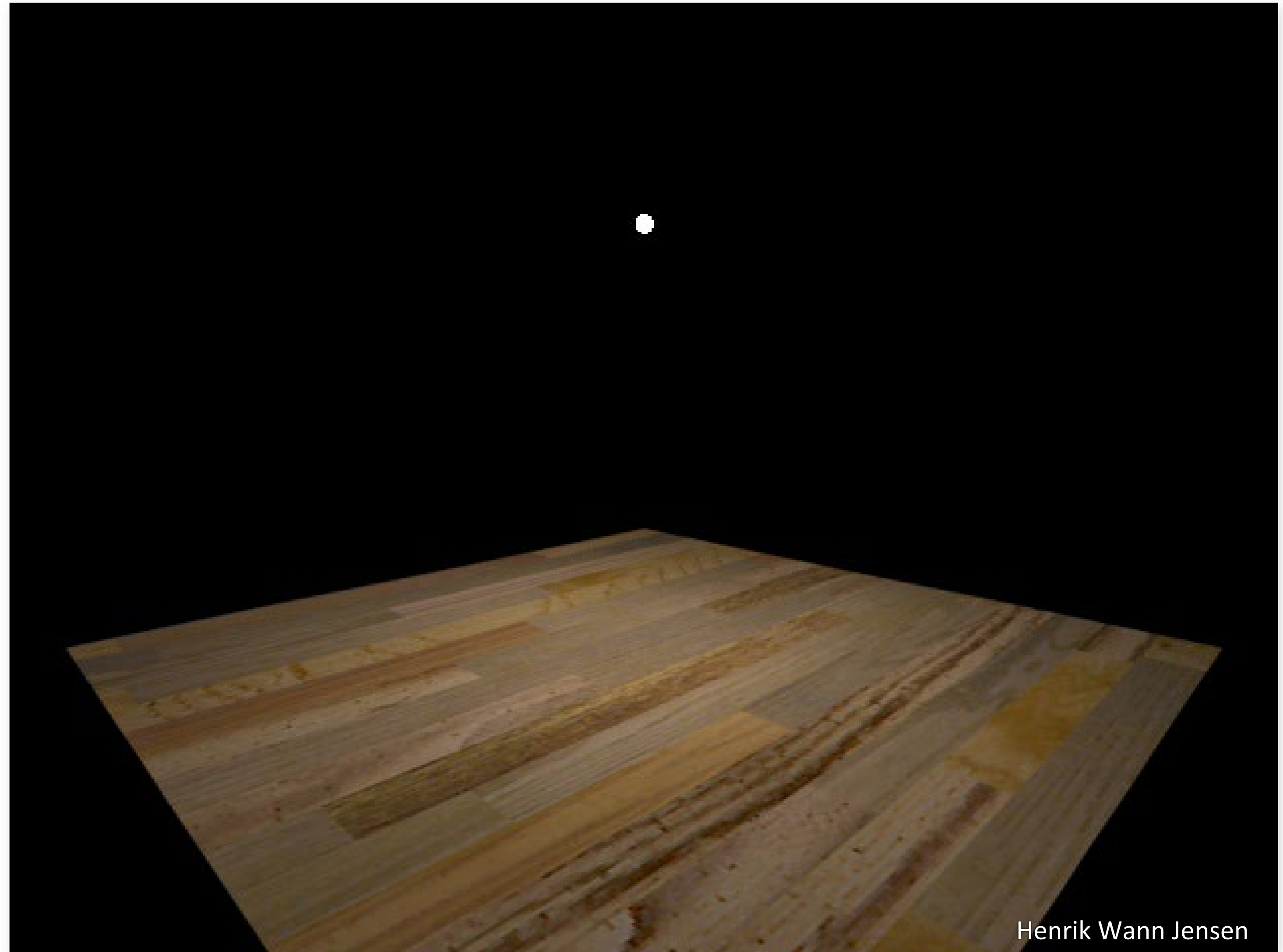
Texturing a rectangle

Image can be mapped directly, unchanged



A textured rectangle

$$\text{albedo} = T(u, v)$$



Transformation of shape

$$\text{albedo} = T([u, v](x, y, z))$$

$$\text{albedo} = T([u, v](\text{transform}^{-1}(x, y, z)))$$



Henrik Wann Jensen



Henrik Wann Jensen

Transformation of texture

$$\text{albedo} = T([u, v](x, y, z))$$

$$\text{albedo} = T(\textit{transform}([u, v](x, y, z)))$$



Henrik Wann Jensen



Henrik Wann Jensen

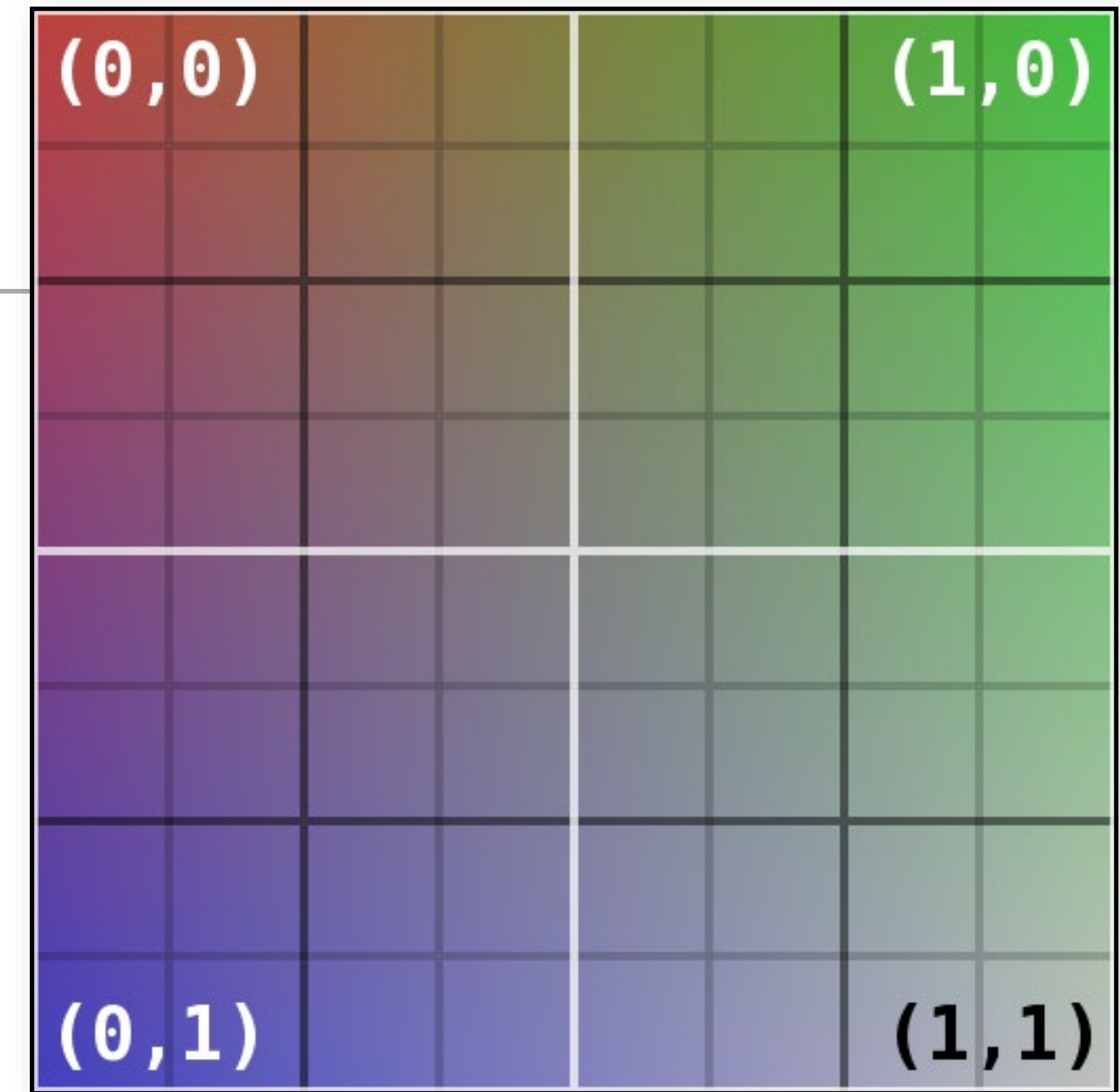
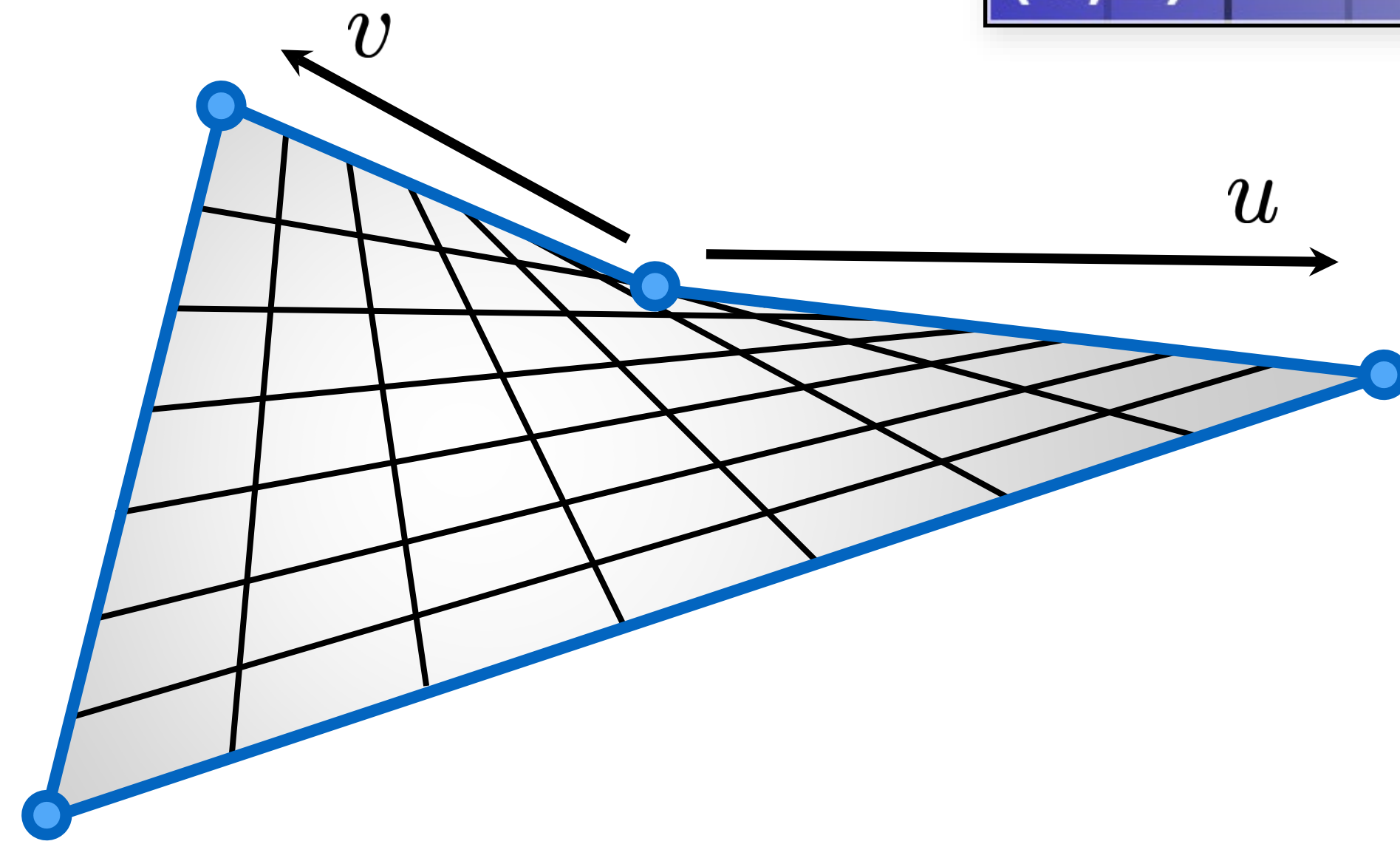
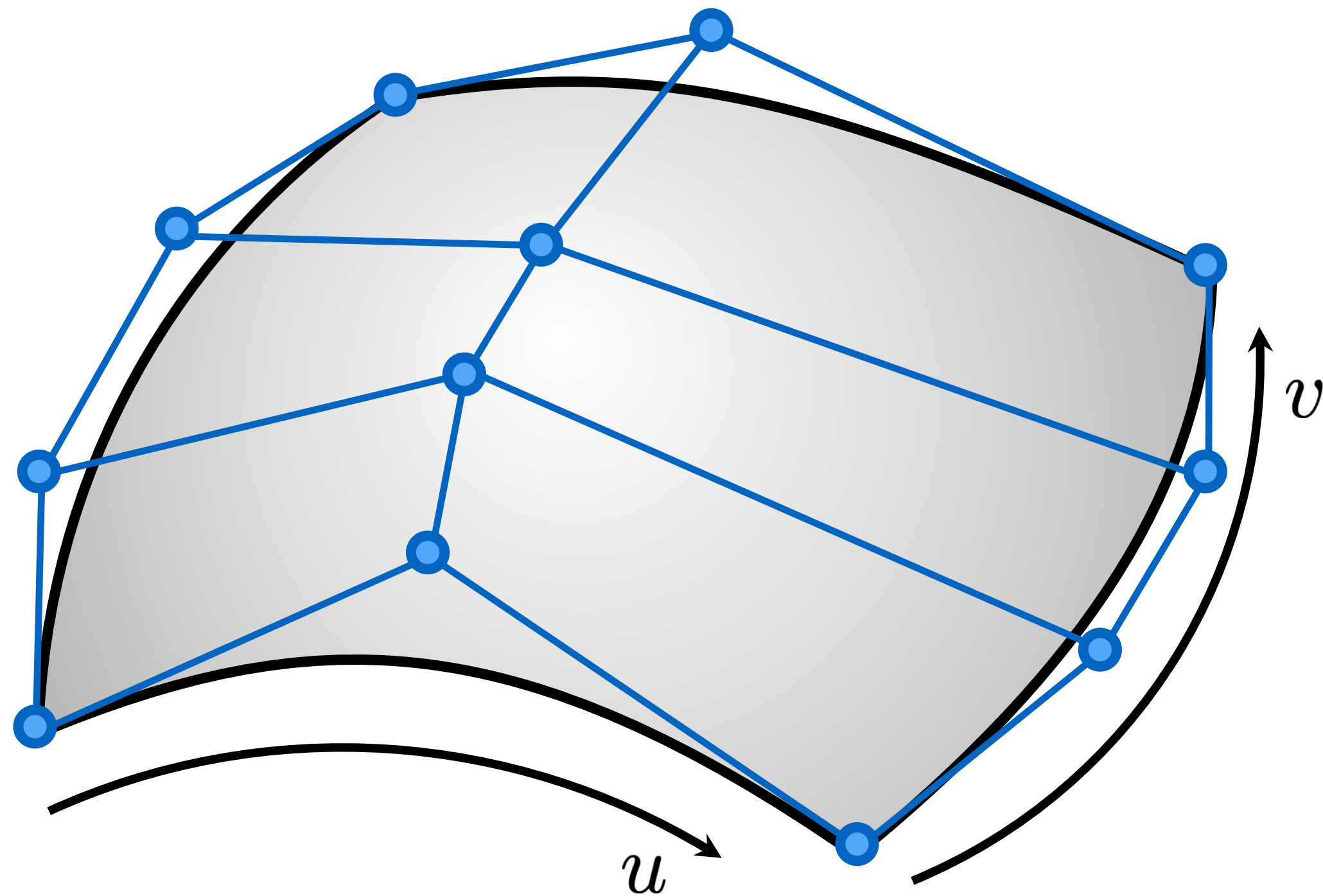
What if the object is not rectangular?

Need to determine mapping from 3D coordinates to 2D texture coordinates

- Parametric surfaces
- Projection mapping
- UV mapping

Parametric surfaces

Parametric surfaces have a natural 2D coordinate system.



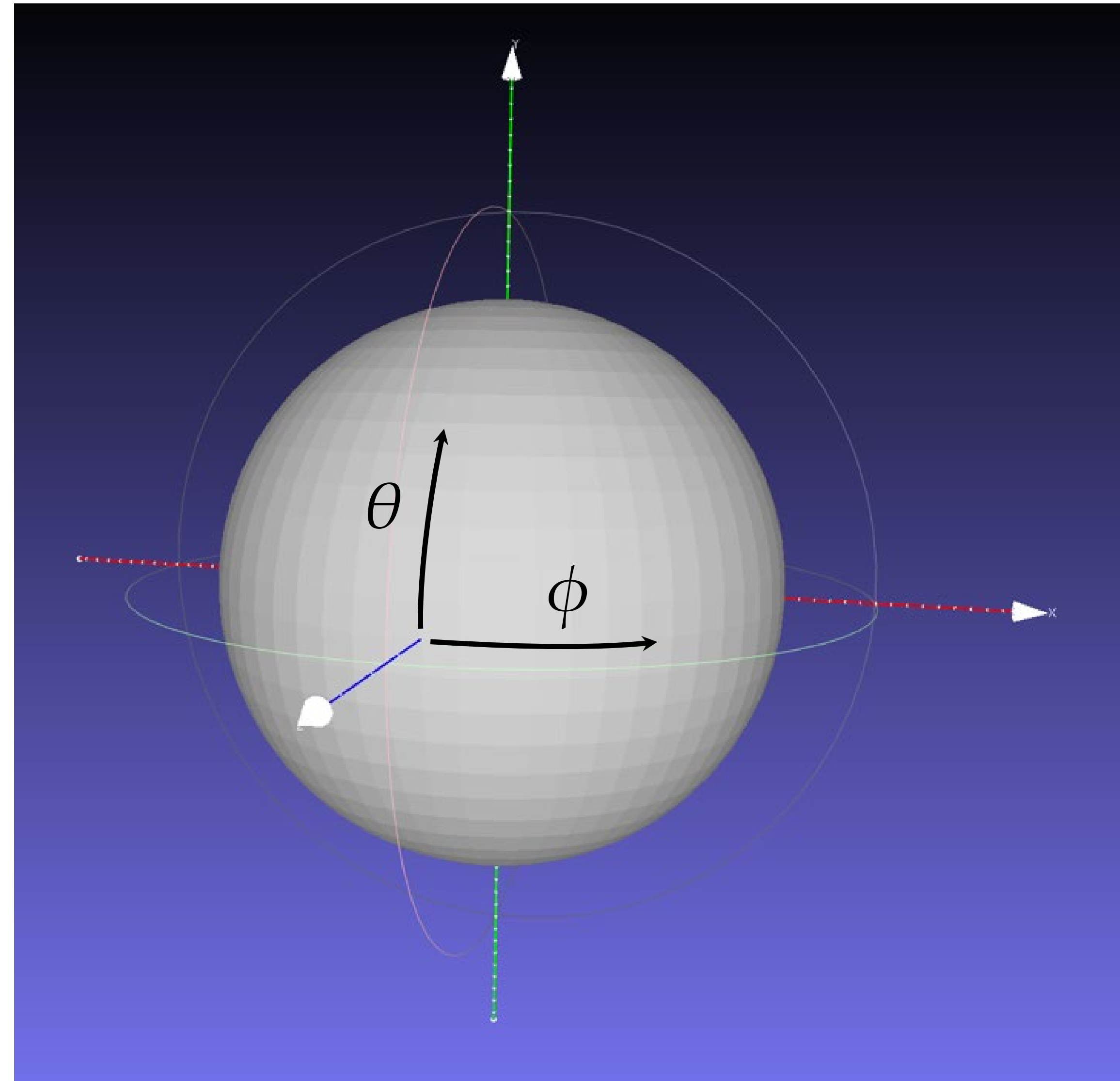
Can also define a sphere parametrically

Position:

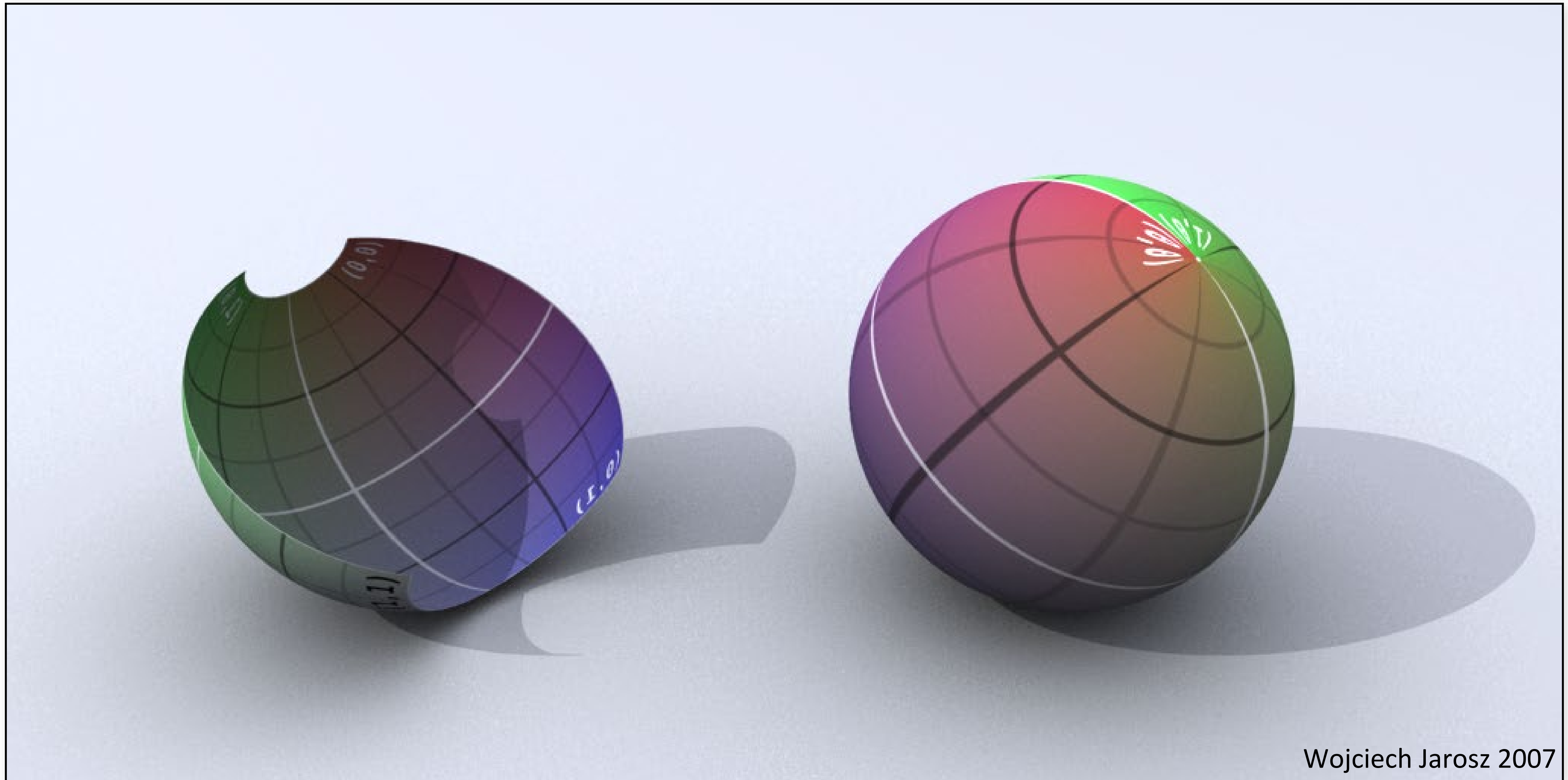
$$x = \cos \theta \sin \phi$$

$$y = \sin \theta$$

$$z = \cos \theta \cos \phi$$

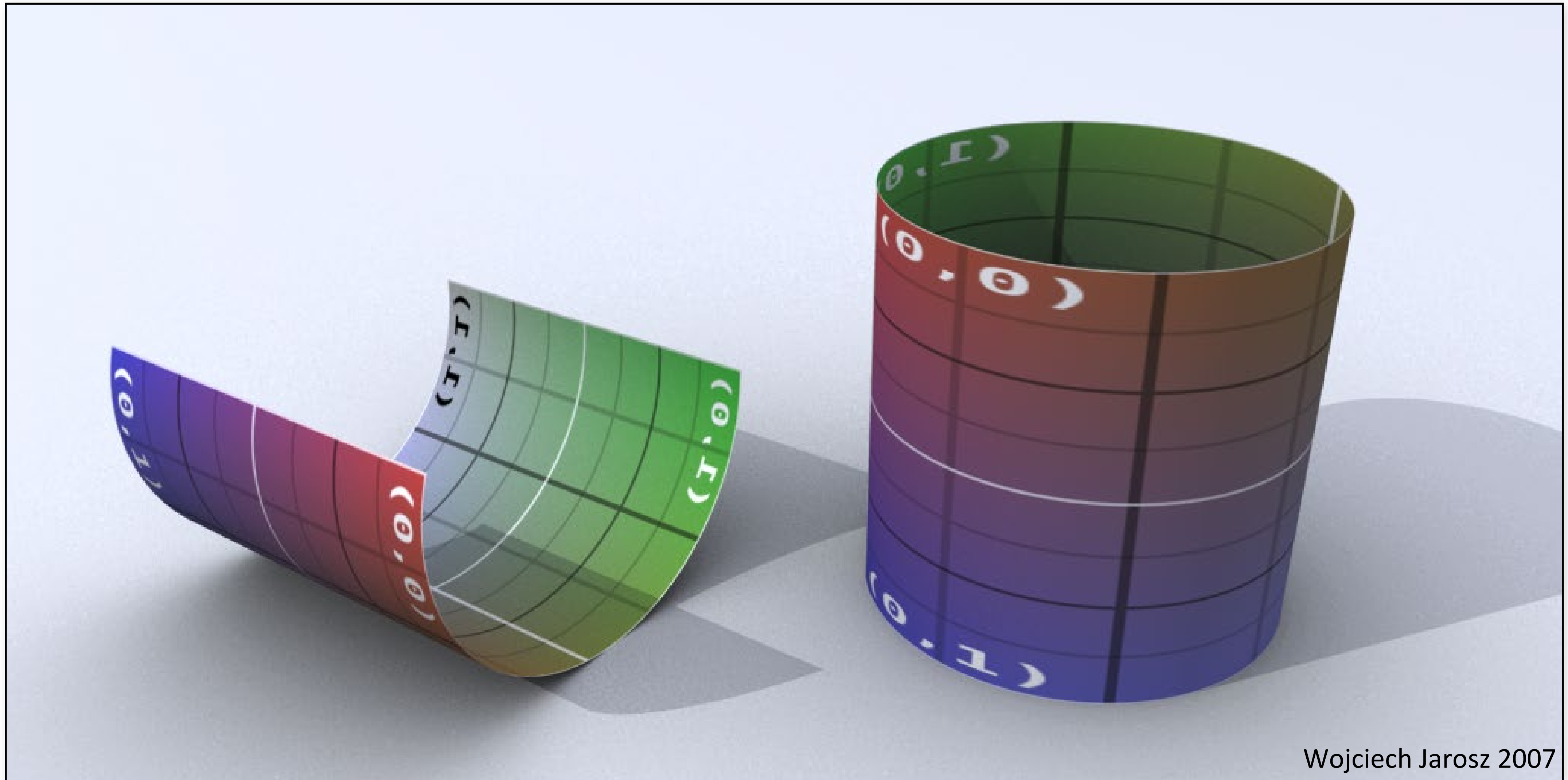


Parametric spheres



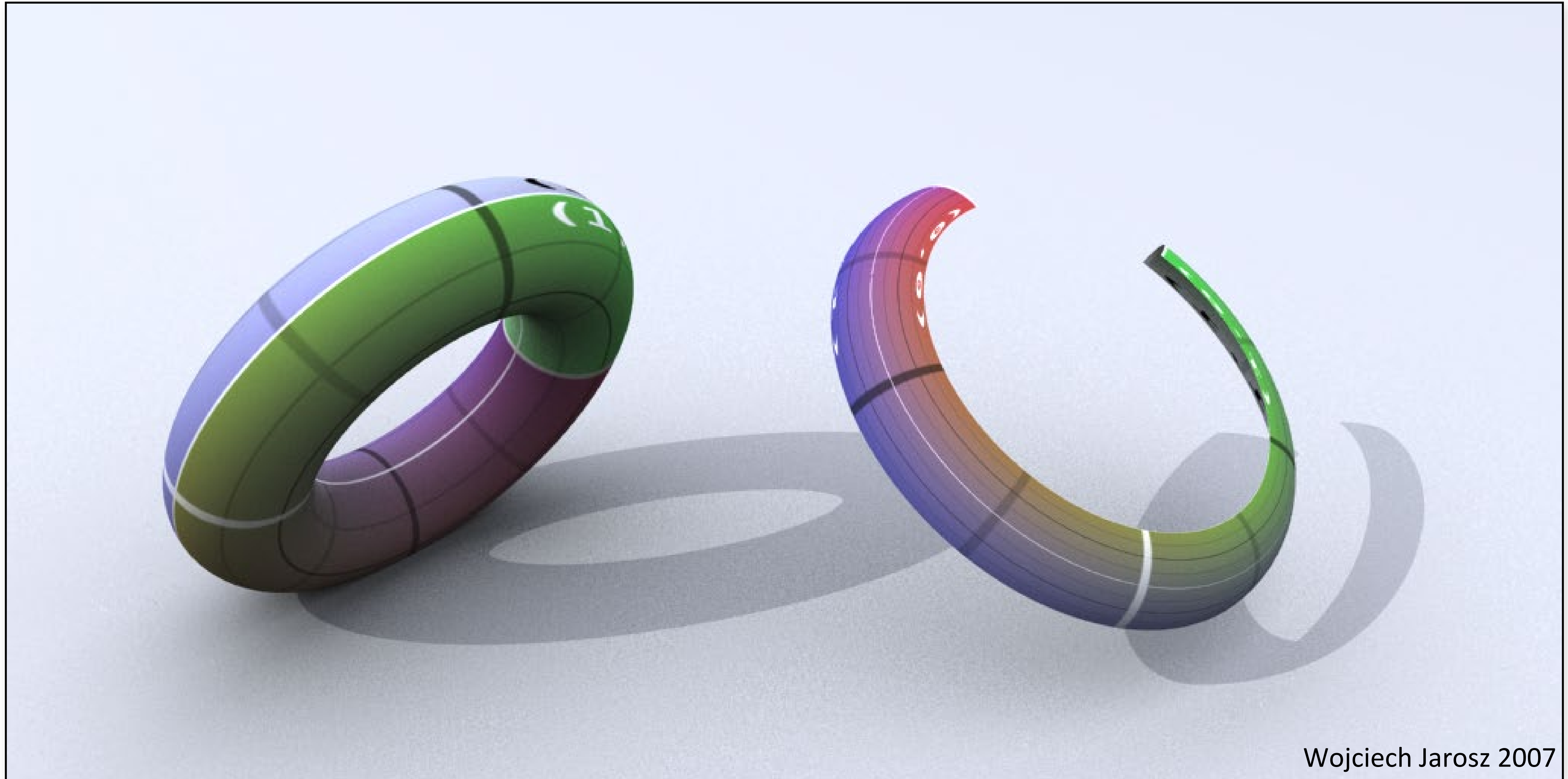
Wojciech Jarosz 2007

Parametric cylinders

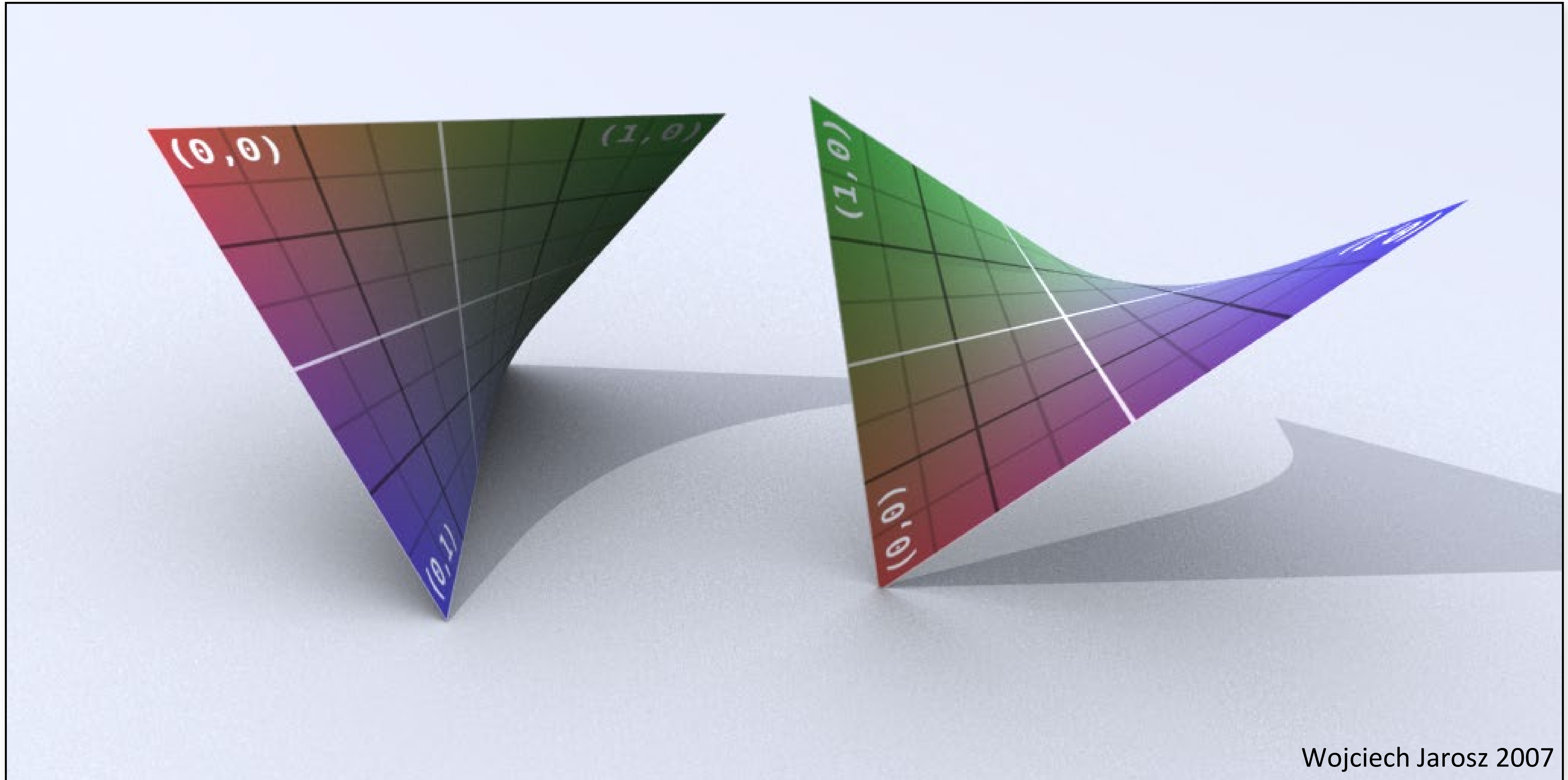


Wojciech Jarosz 2007

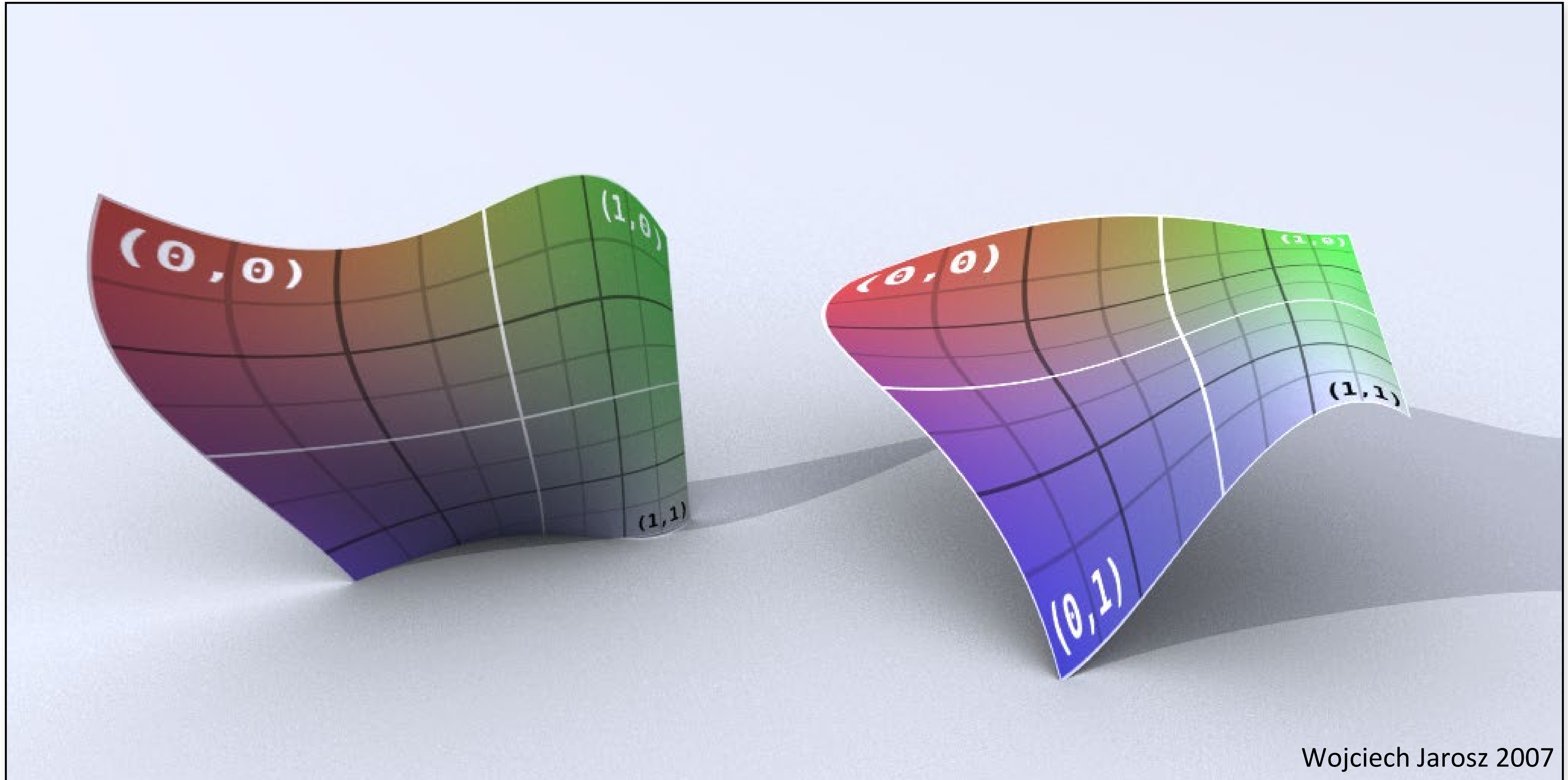
Parametric toruses



Bilinear patches



Bicubic patches



Creating parameterizations

For non-parametric surfaces it is trickier

Need to create a parametrization!

- Projection mapping
- UV mapping

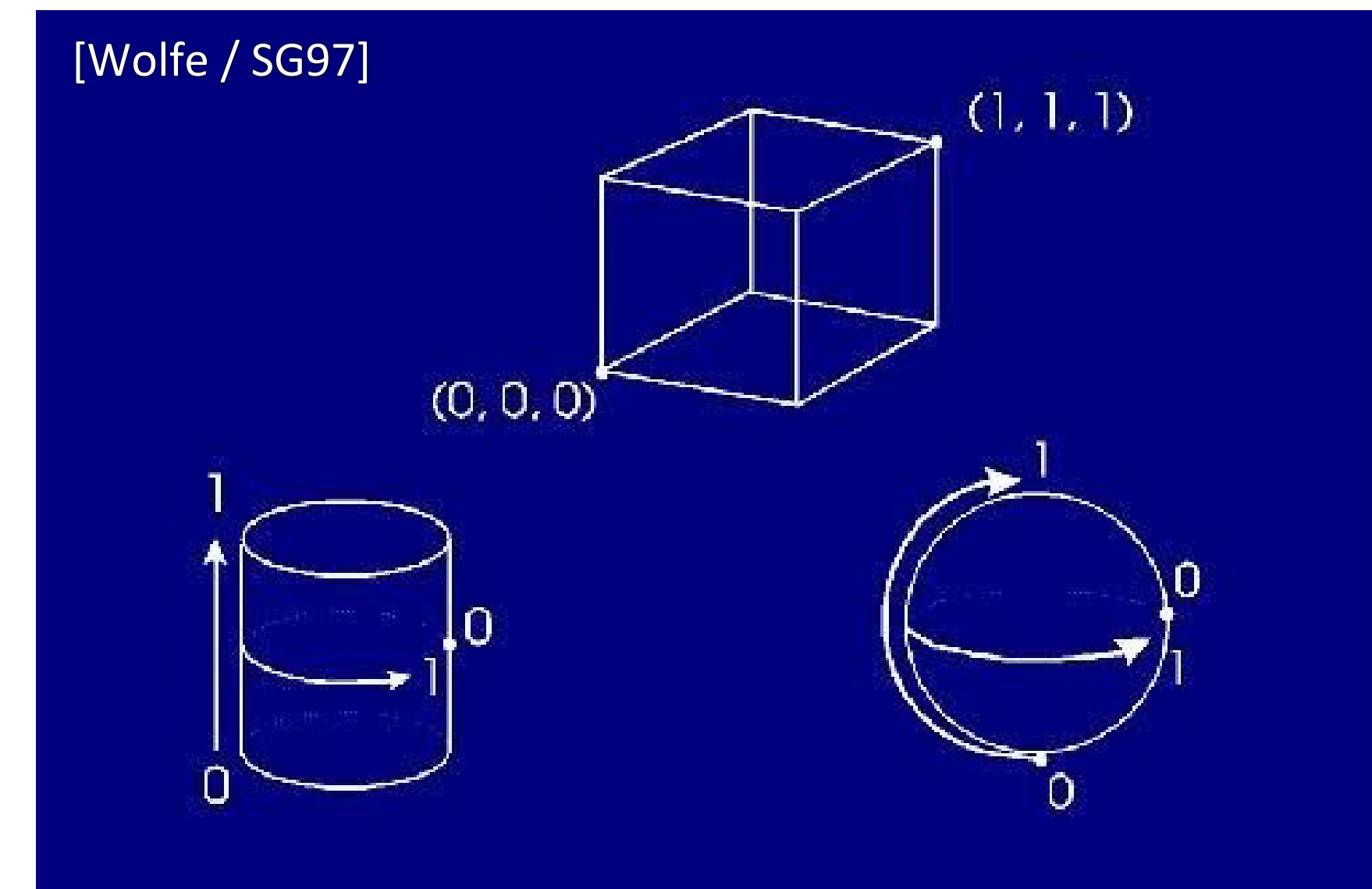
Projection mapping

Maps 3D surface points to 2D image coordinates

$$f : \mathbb{R}^3 \rightarrow [0, 1]^2$$

Different types of projections

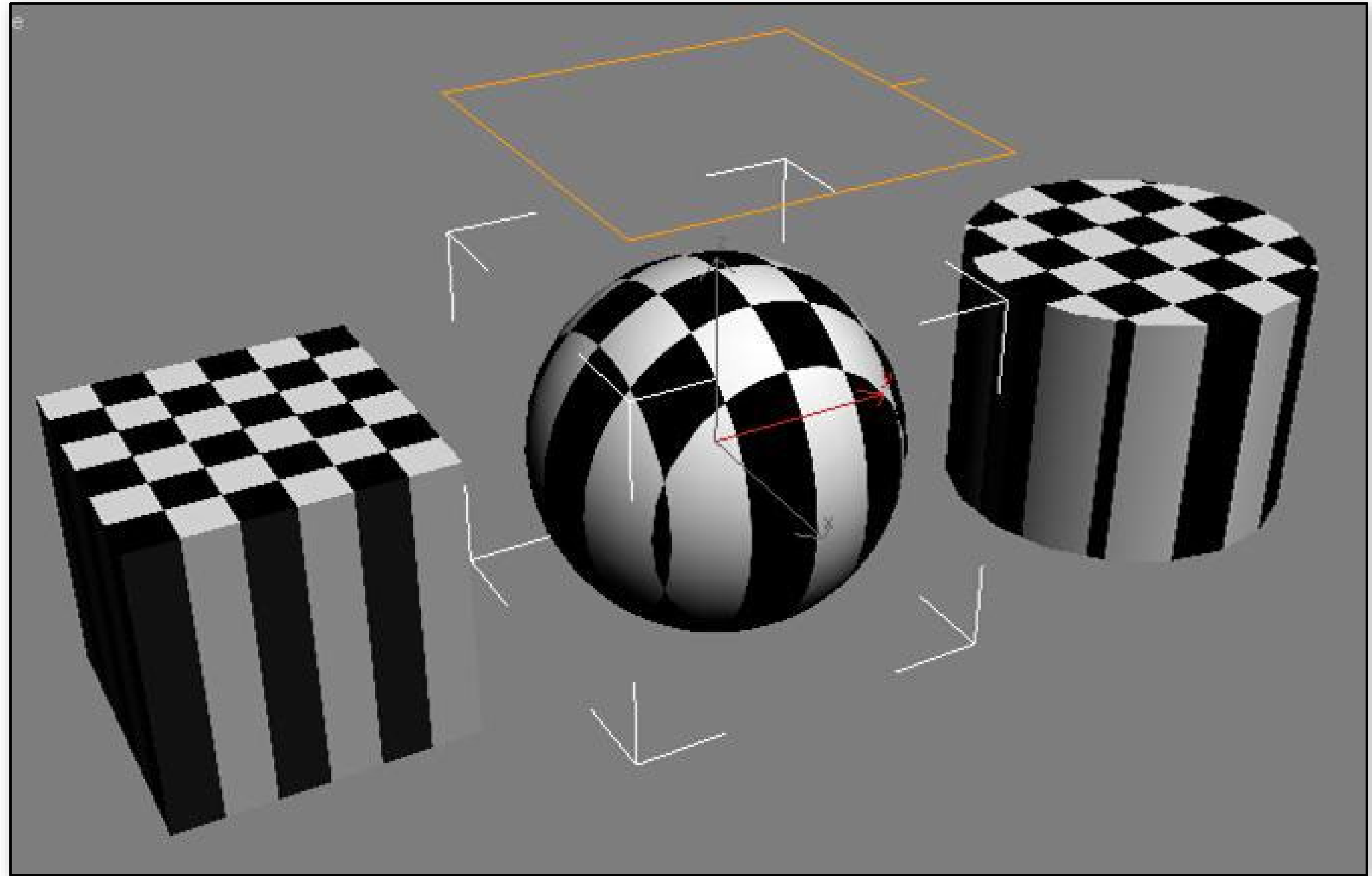
- often corresponding to simple shapes
- useful for simple objects



Projections — planar

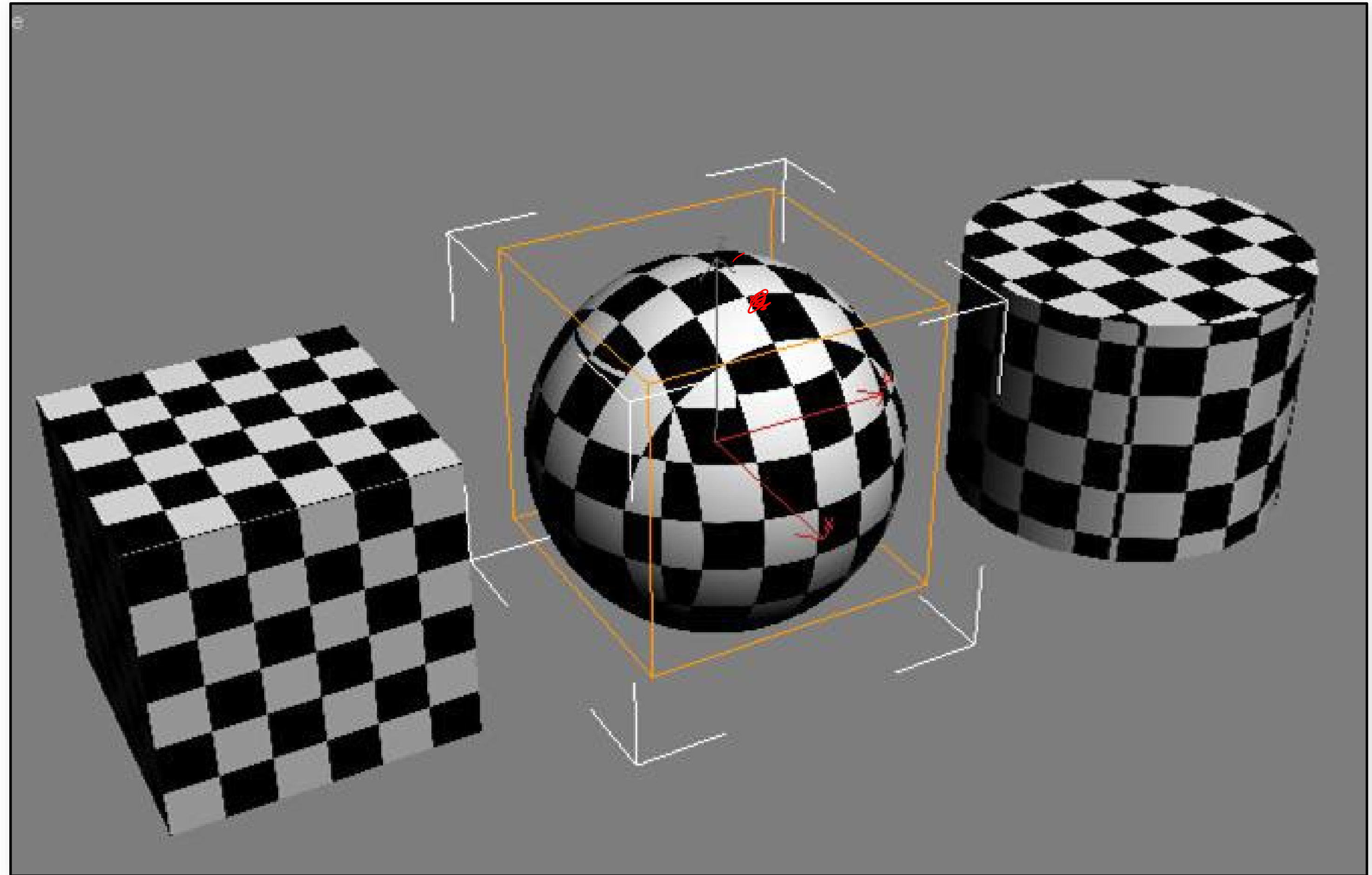
Planar projection along xy plane of size (w, h)

$$f(\mathbf{p}) = \begin{bmatrix} \frac{p_x}{w} \\ \frac{p_y}{h} \end{bmatrix}$$



Projections — cubical

Planar projection onto one of 6 faces of a cube based on surface normal

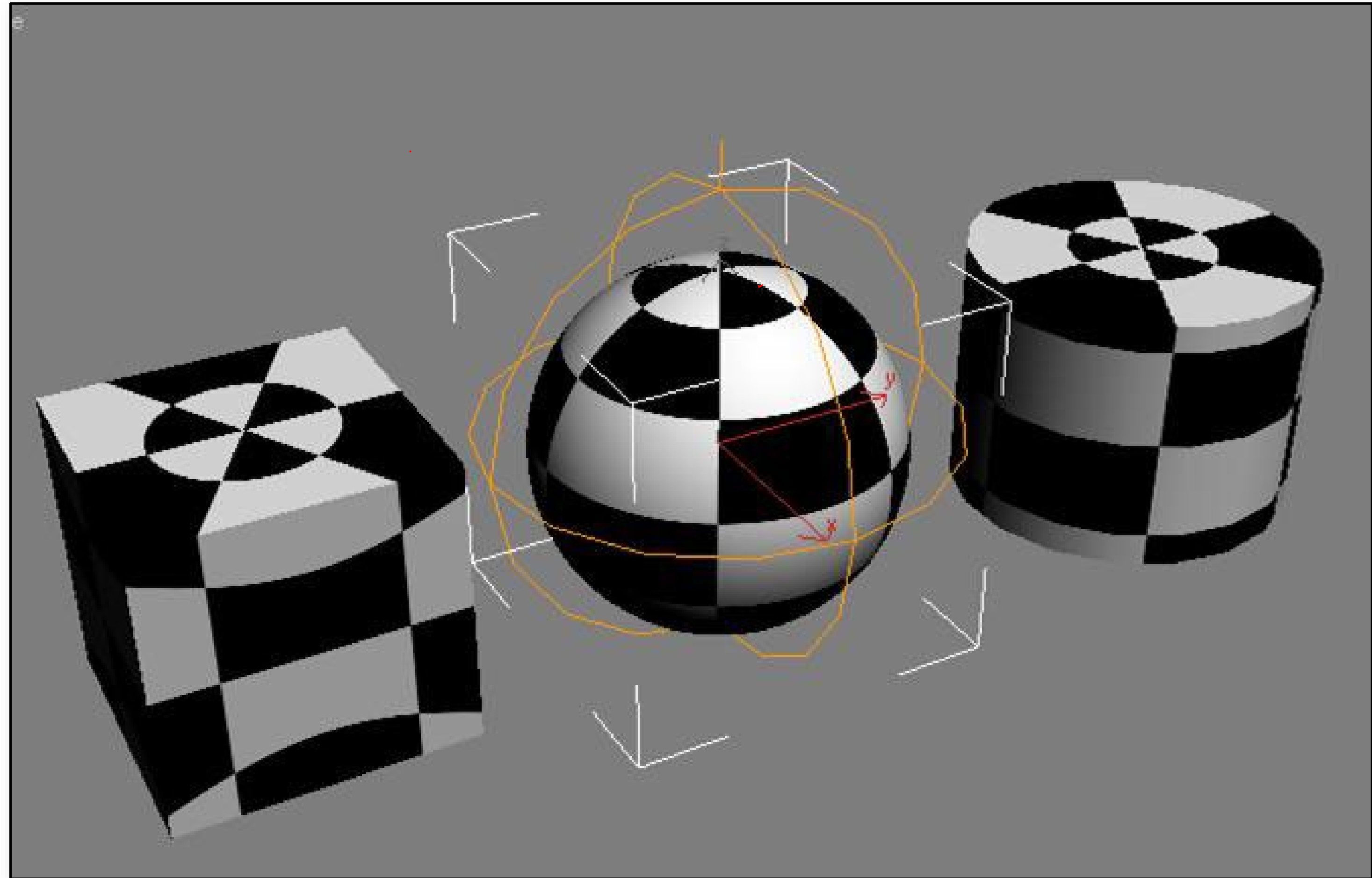


Projections — spherical

Project point onto unit sphere

- compute spherical coordinates

$$f(\mathbf{p}) = \begin{bmatrix} \frac{\phi}{2\pi} \\ \frac{\theta}{\pi} \end{bmatrix}$$



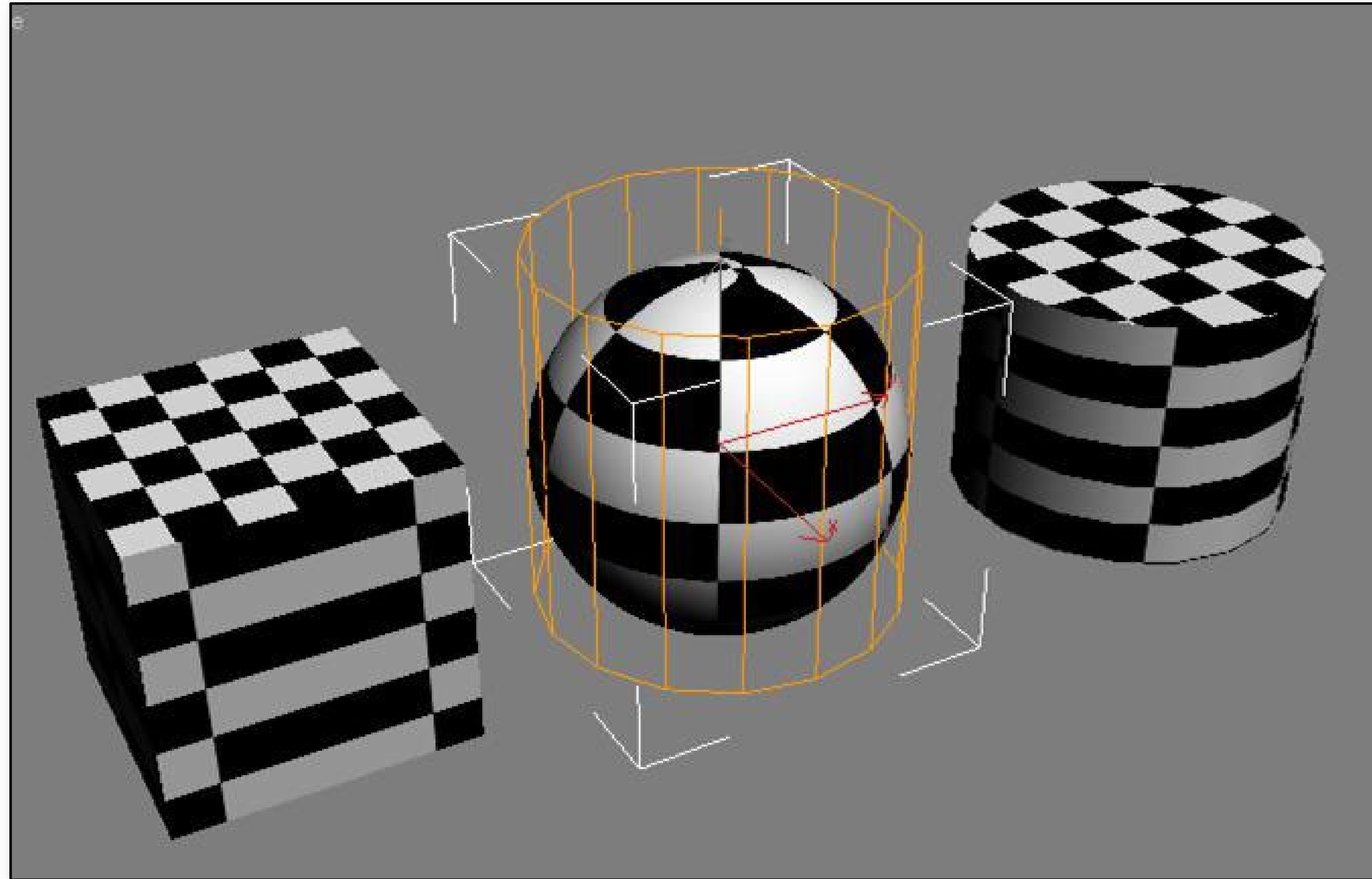
Projections — cylindrical

Project point onto cylinder of height h

- compute cylindrical coordinates

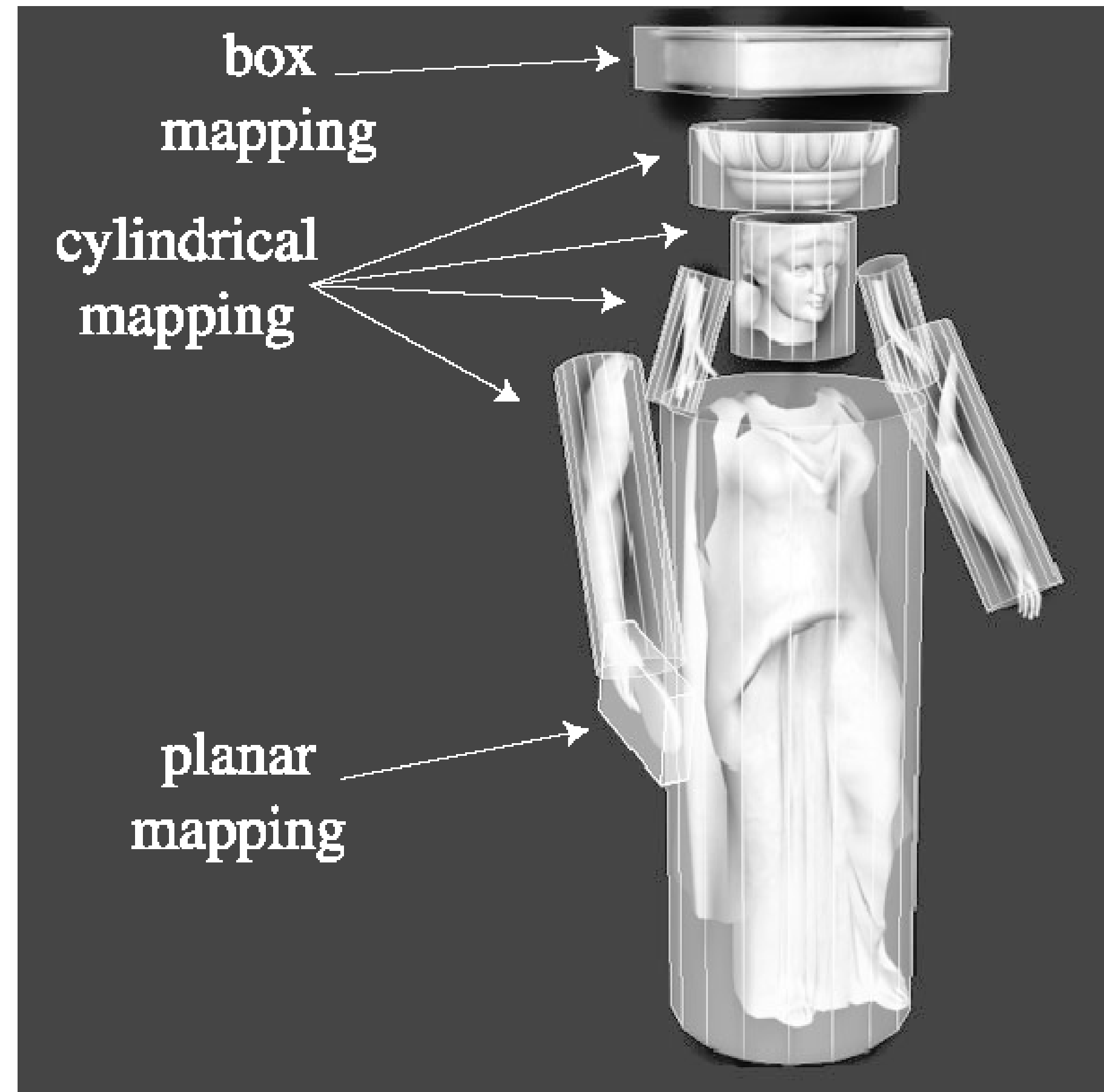
$$f(\mathbf{p}) = \begin{bmatrix} \frac{\phi}{2\pi} \\ \frac{p_y}{h} \end{bmatrix}$$

- treat caps separately



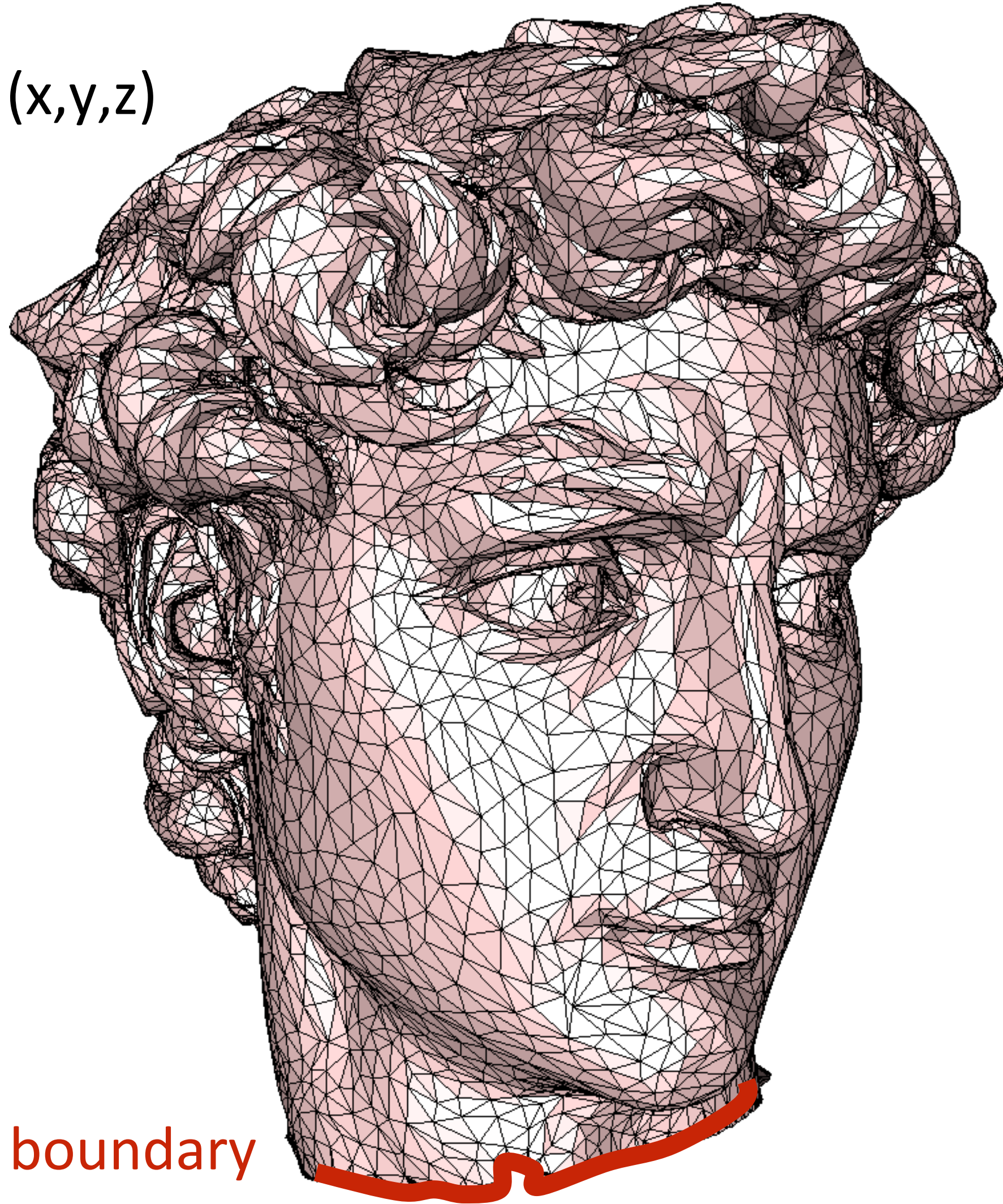
Combining projections

Non-parametric surfaces: project pieces to parametric surface



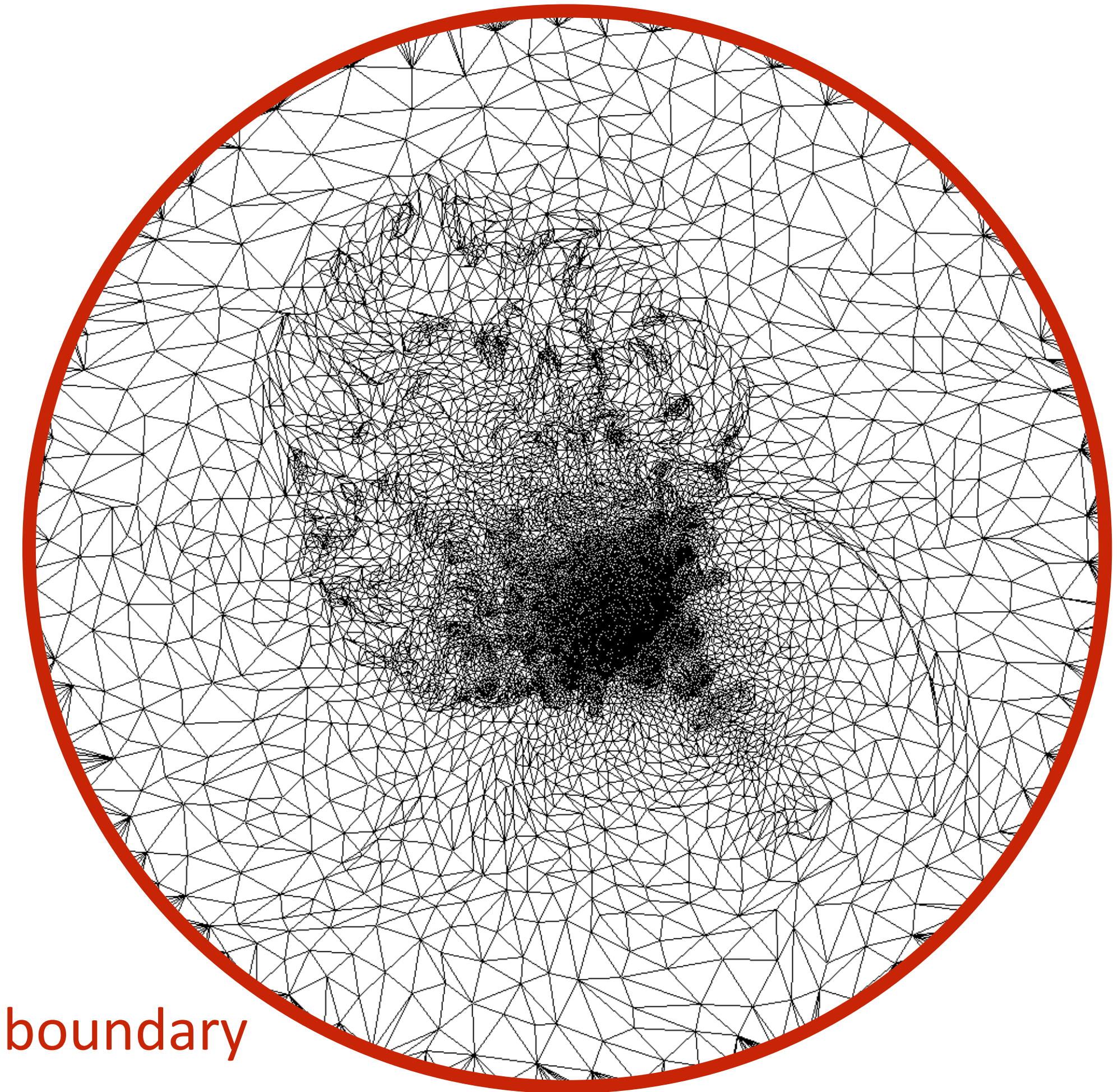
Creating UV parameterizations

3D space (x,y,z)



boundary

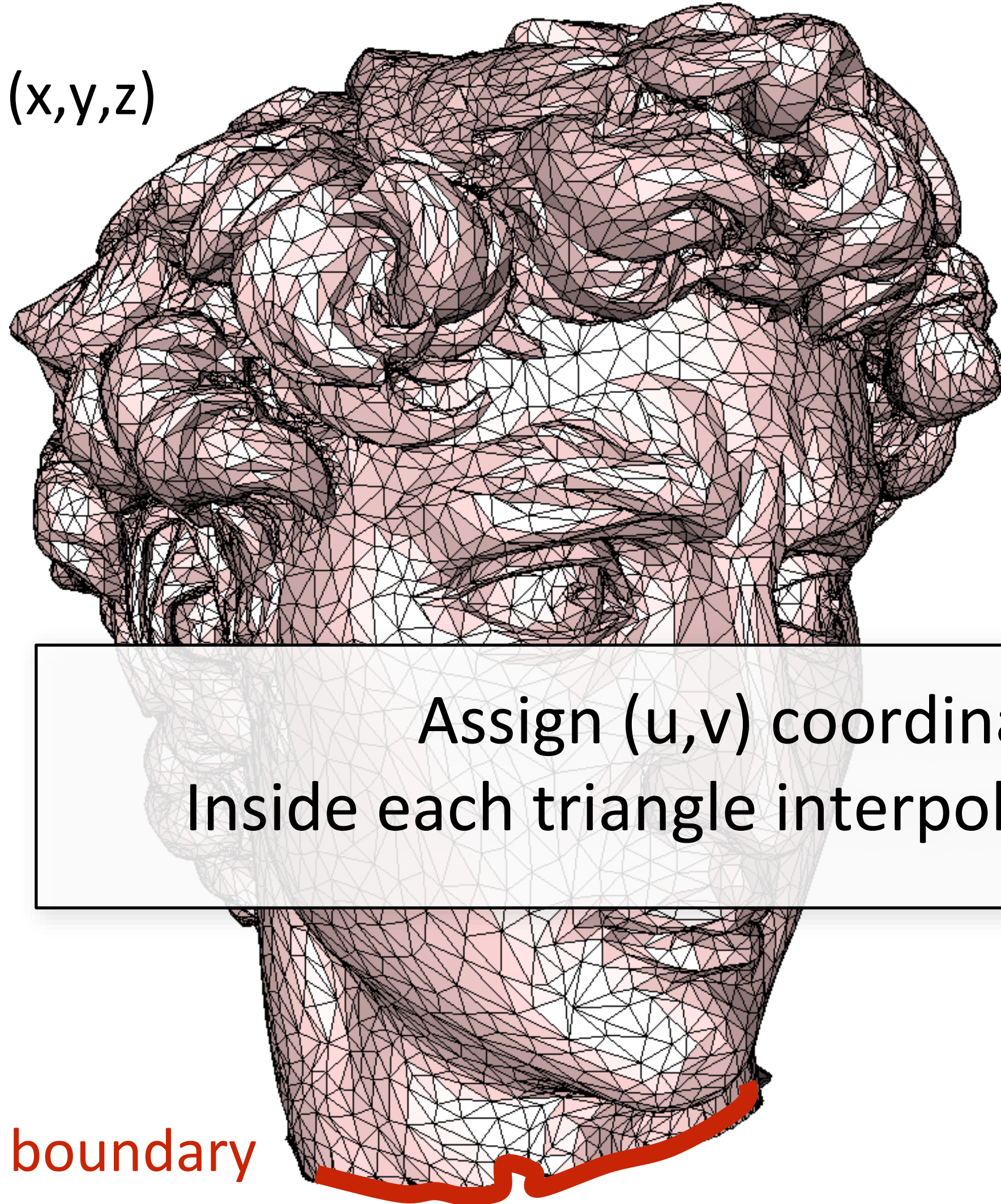
2D parameter domain (u,v)



boundary

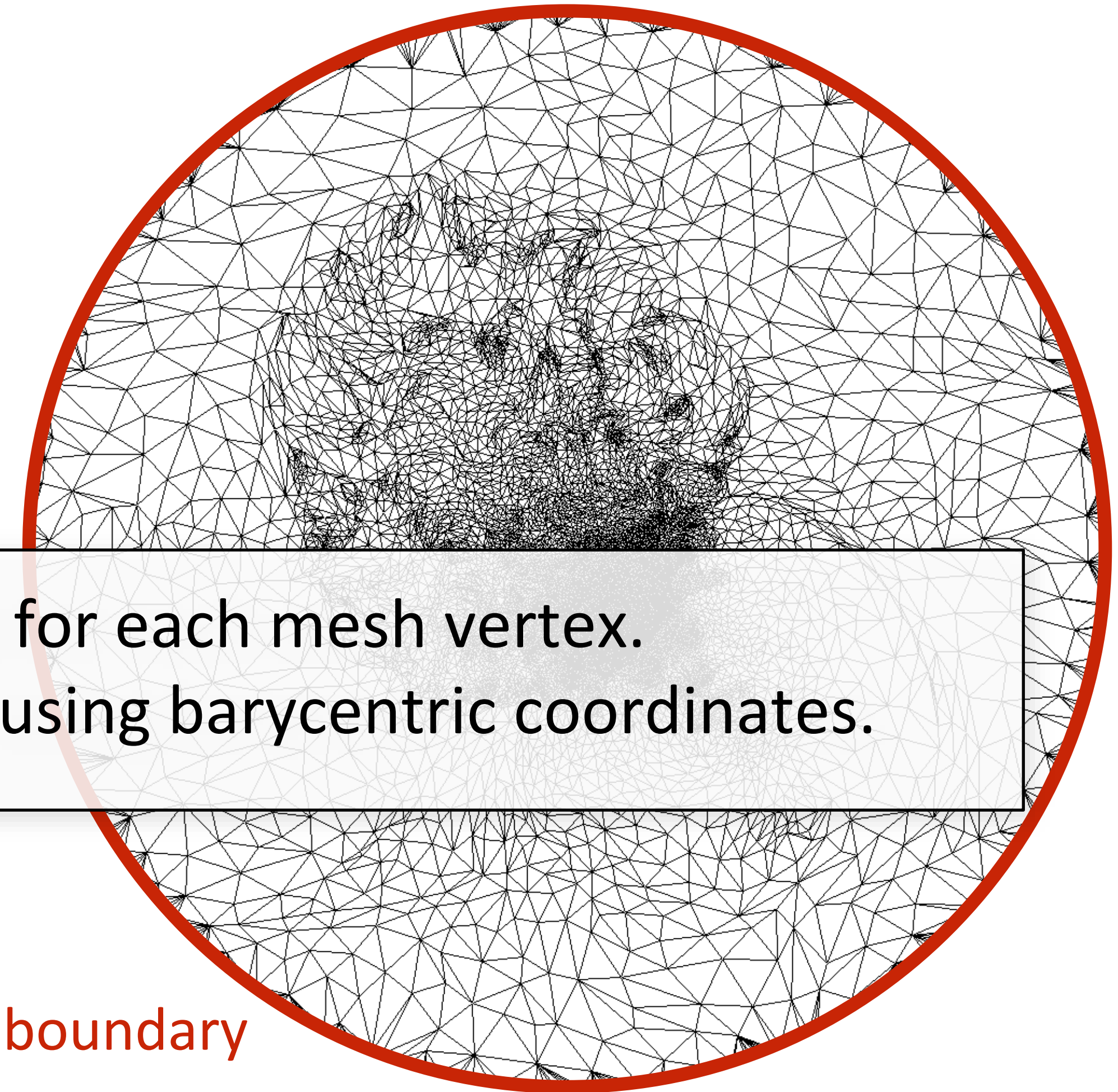
Creating UV parameterizations

3D space (x,y,z)



boundary

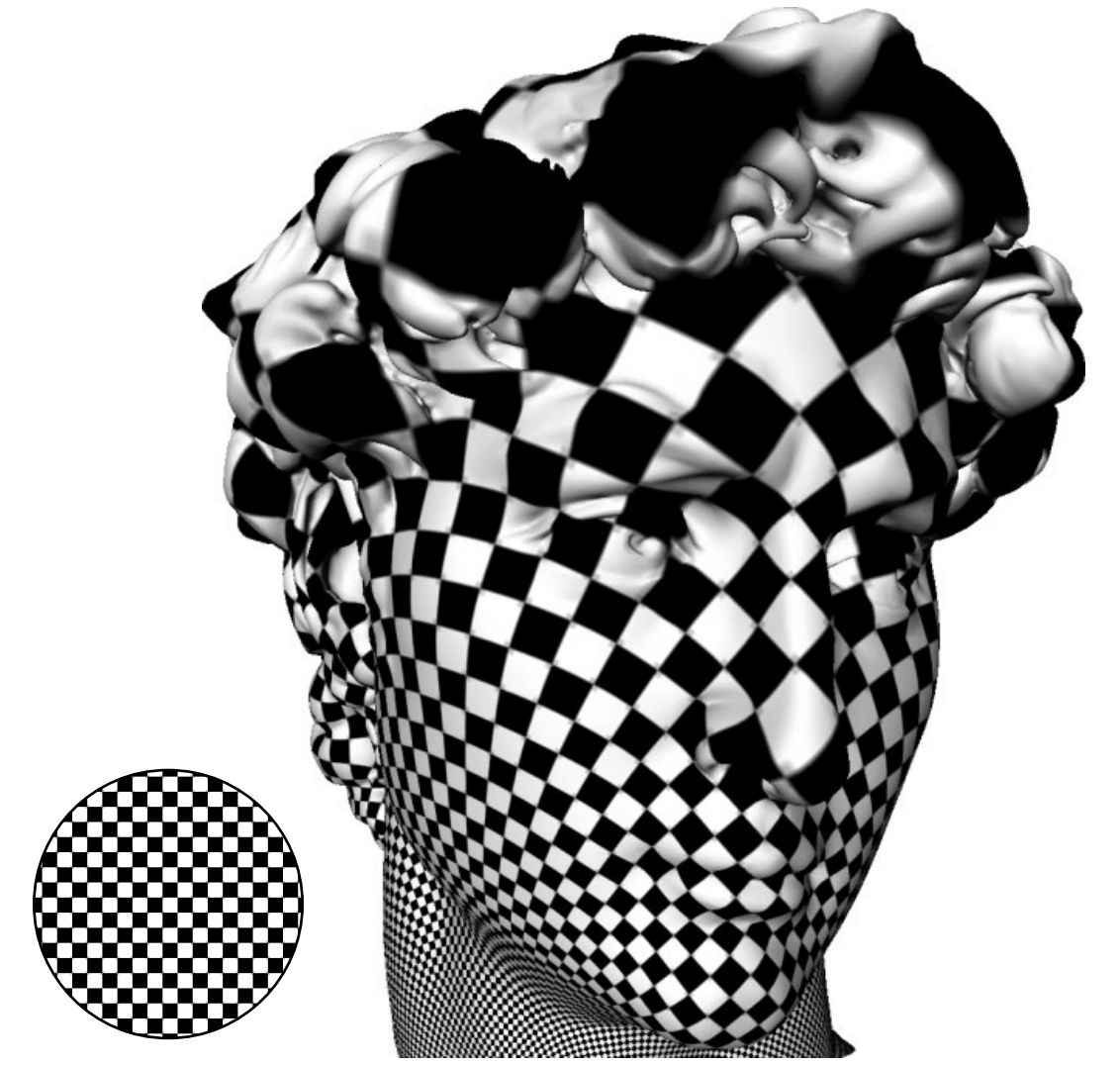
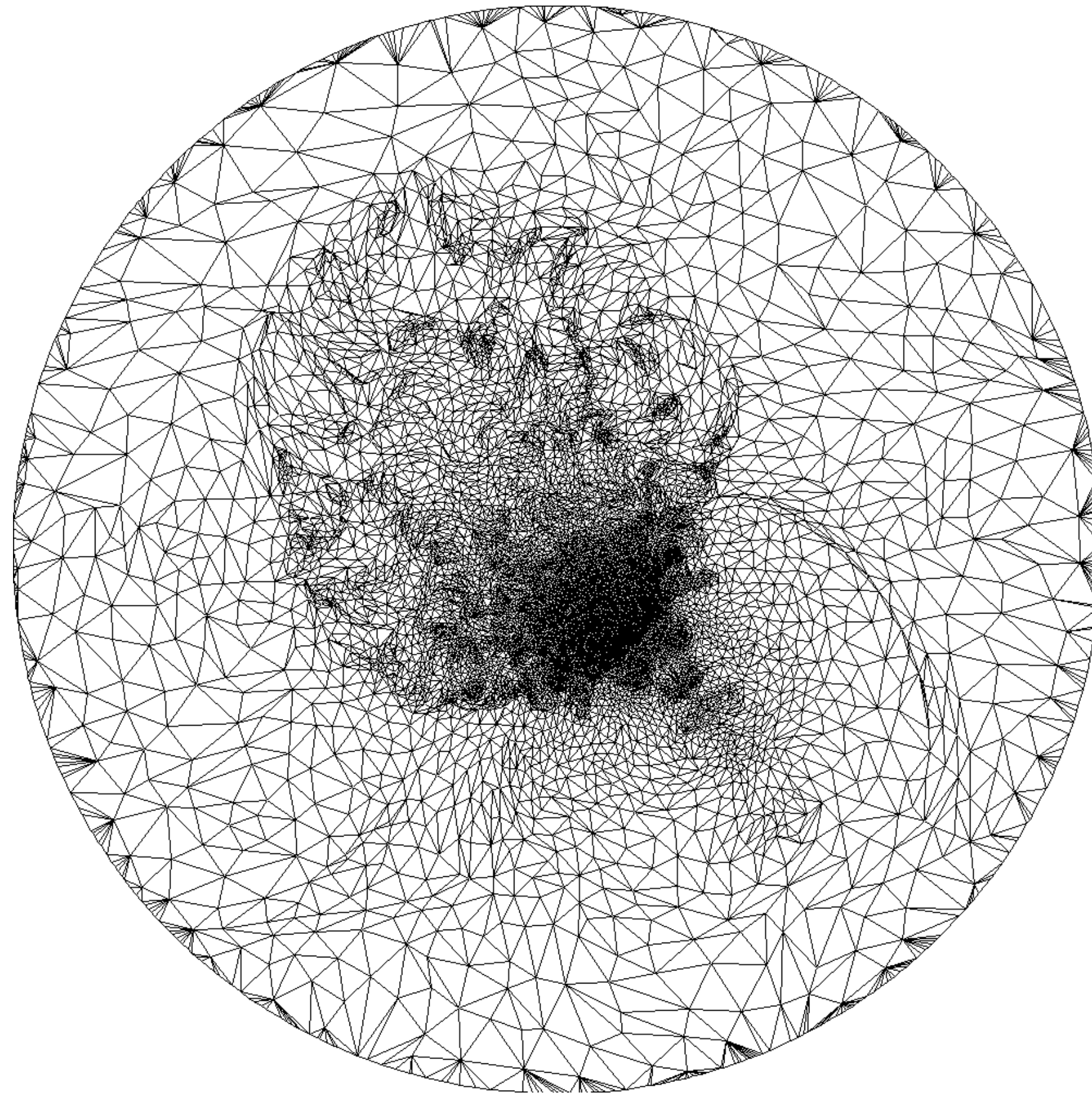
2D parameter domain (u,v)



boundary

Assign (u,v) coordinates for each mesh vertex.
Inside each triangle interpolate using barycentric coordinates.

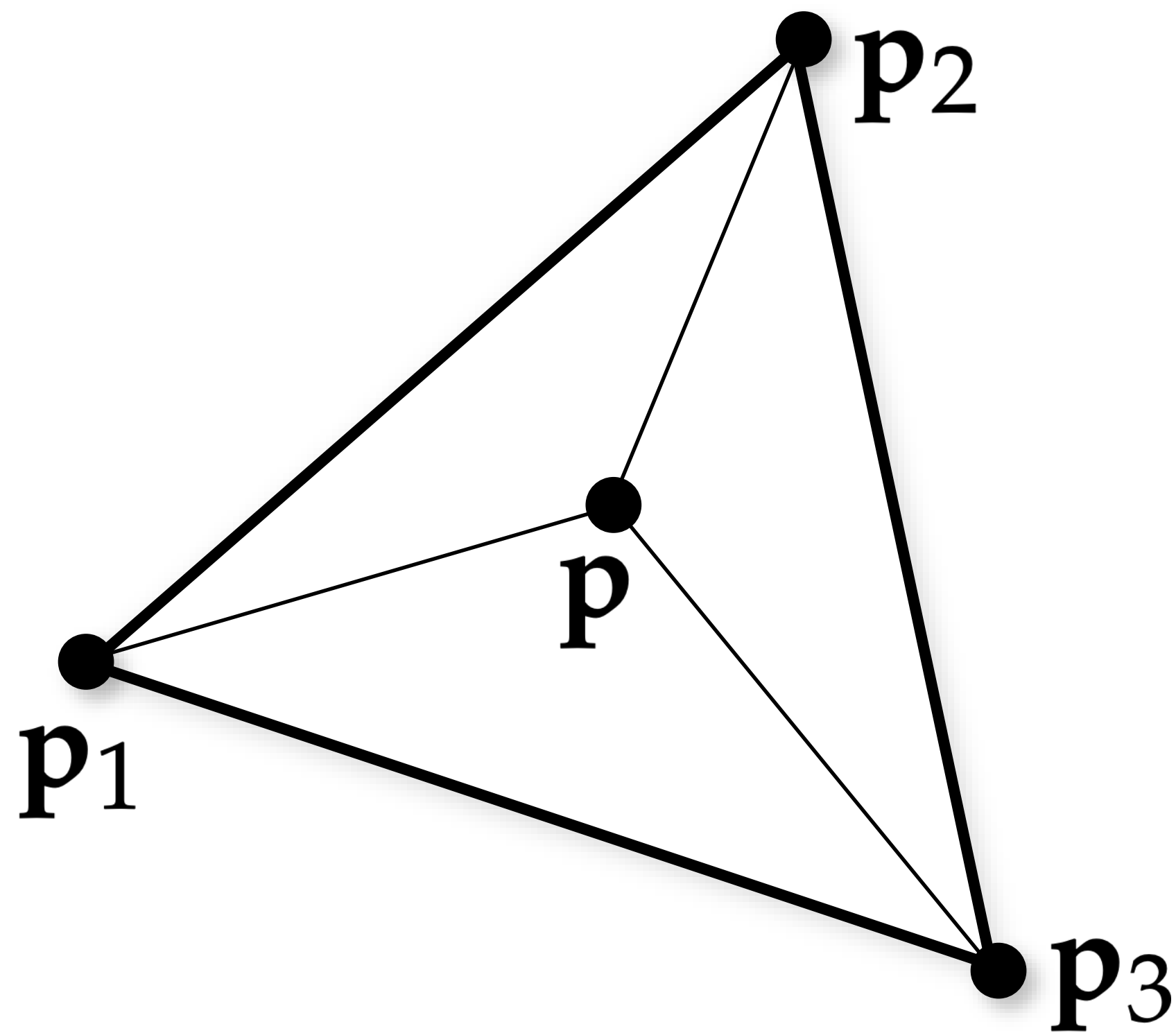
Creating UV parameterizations



After a slide by Daniele Panozzo

Barycentric coordinates

Barycentric interpolation: $\mathbf{p}(\alpha, \beta, \gamma) = \alpha \mathbf{p}_1 + \beta \mathbf{p}_2 + \gamma \mathbf{p}_3$



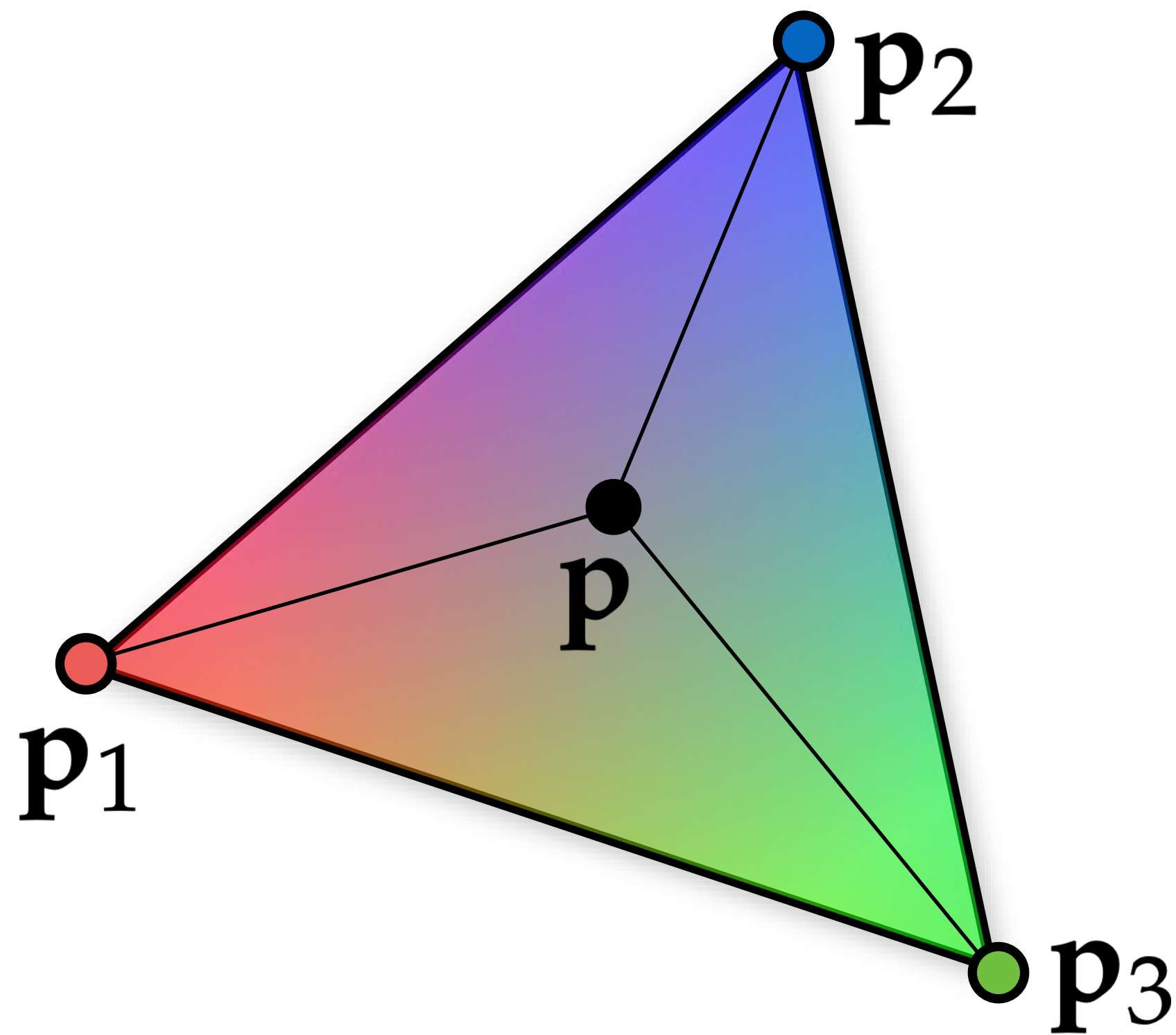
Can use this eqn. to
interpolate any vertex
quantity across triangle!

Barycentric coordinates

Barycentric interpolation:

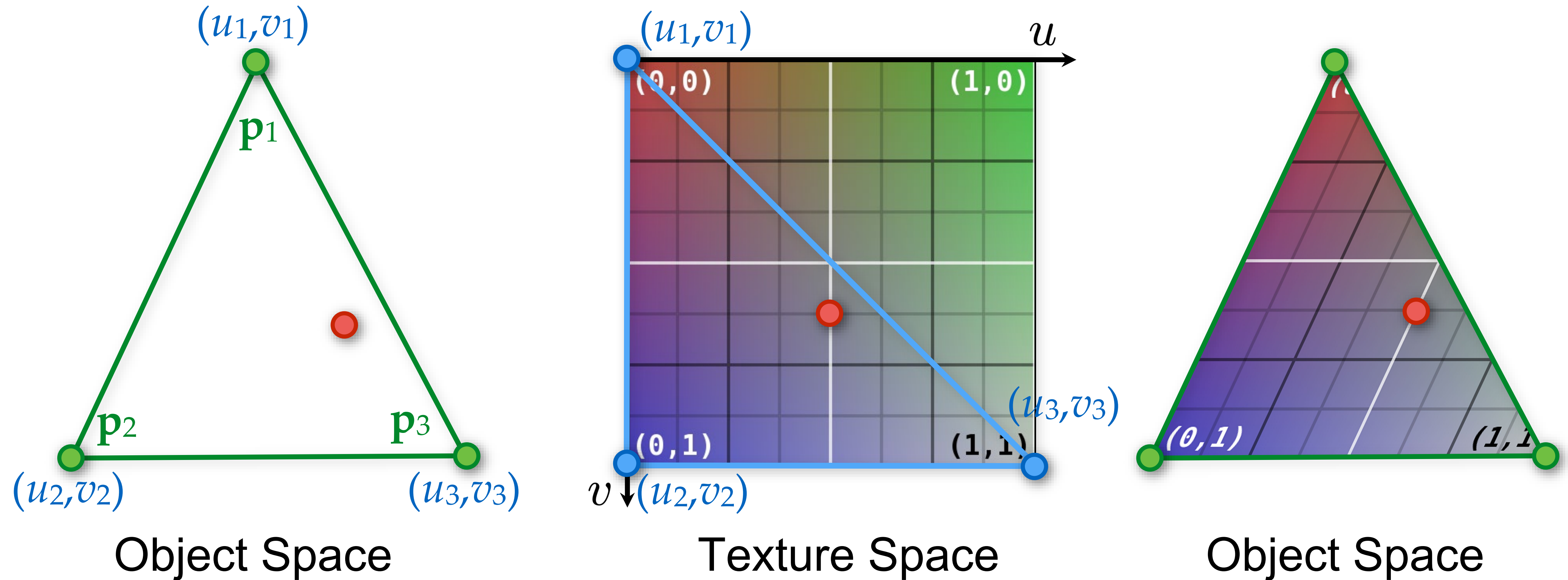
$$\mathbf{p}(\alpha, \beta, \gamma) = \alpha \mathbf{p}_1 + \beta \mathbf{p}_2 + \gamma \mathbf{p}_3$$

$$\mathbf{c}(\alpha, \beta, \gamma) = \alpha \mathbf{c}_1 + \beta \mathbf{c}_2 + \gamma \mathbf{c}_3$$



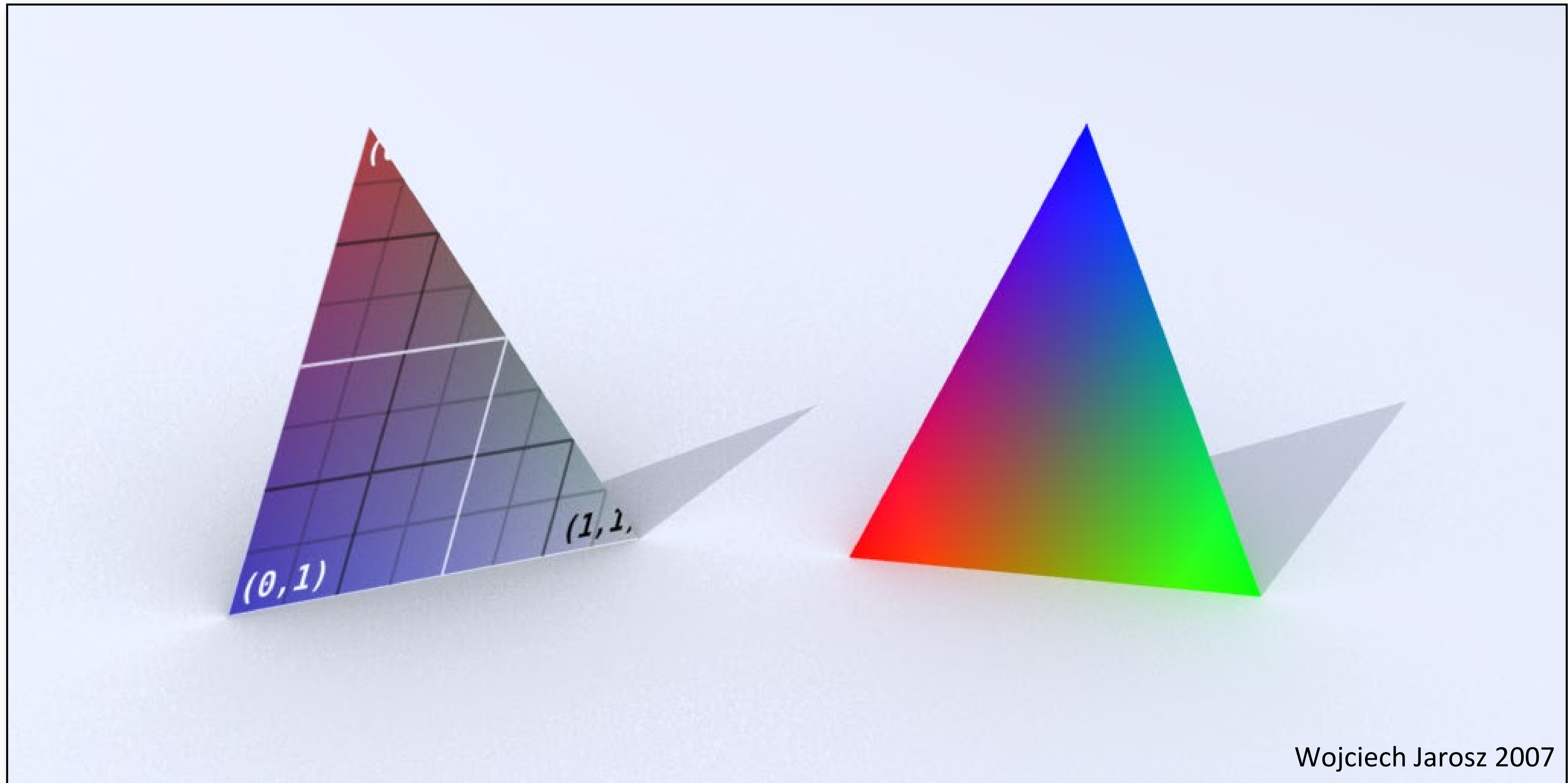
Can use this eqn. to
interpolate any vertex
quantity across triangle!

UV texturing

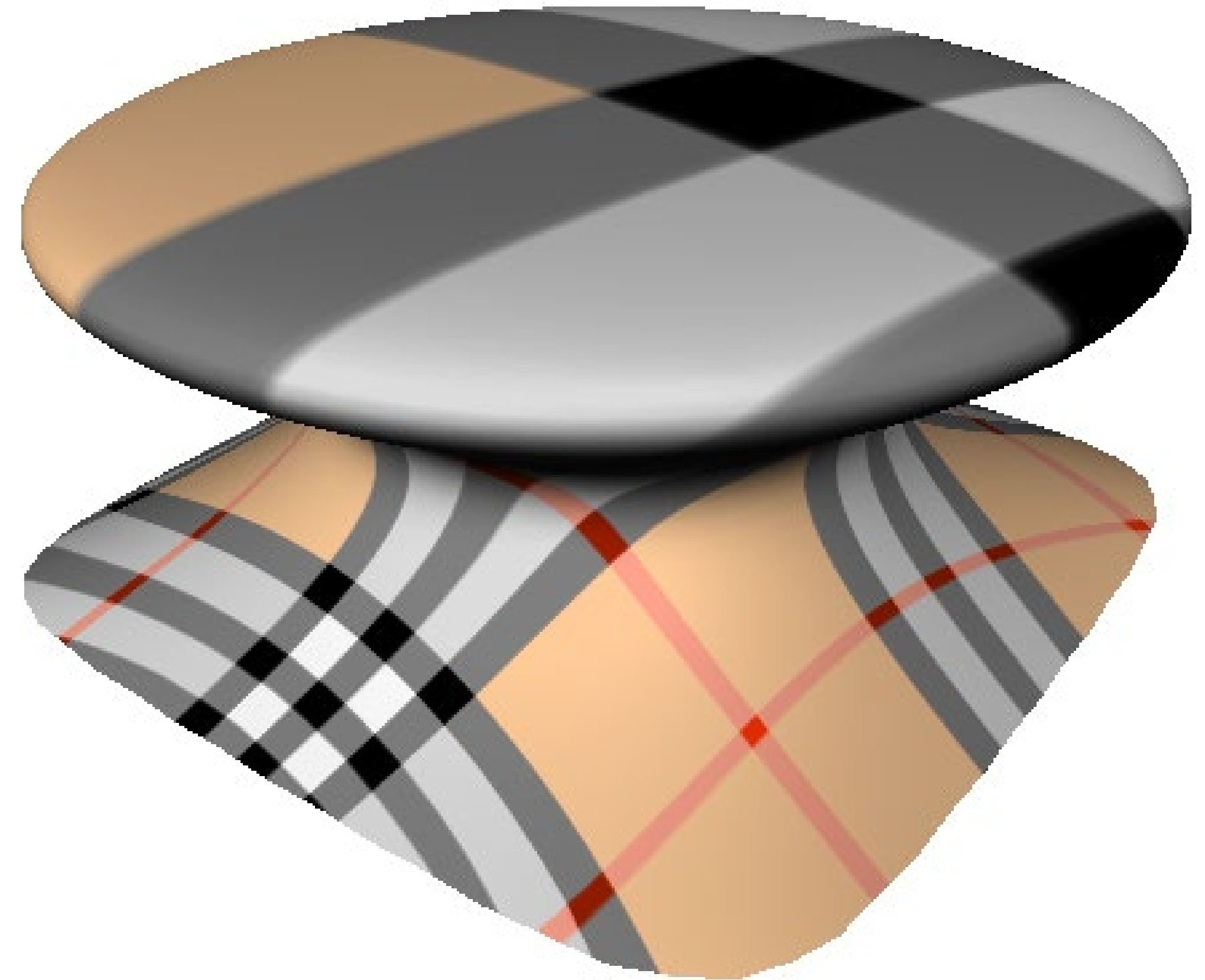
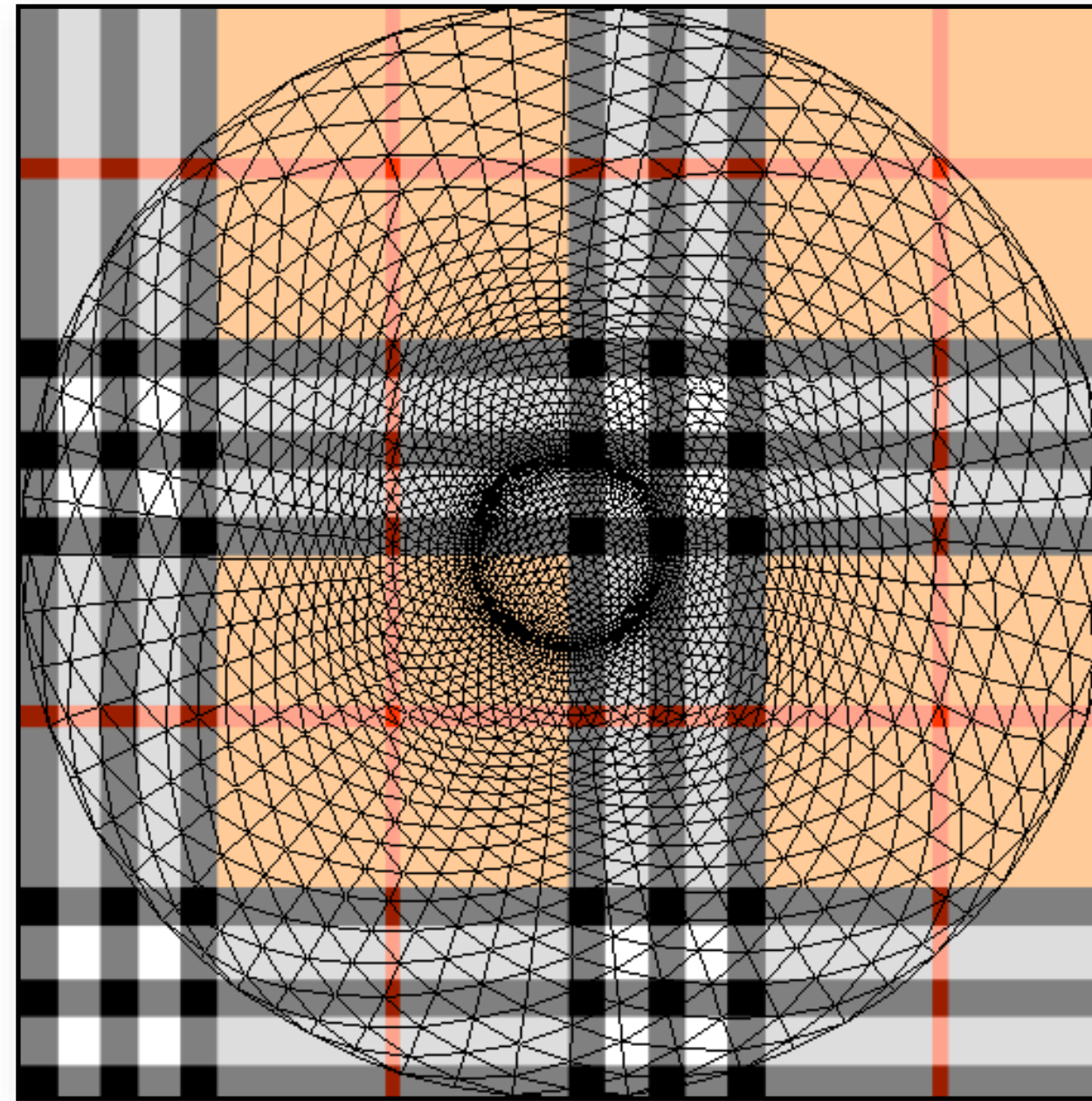
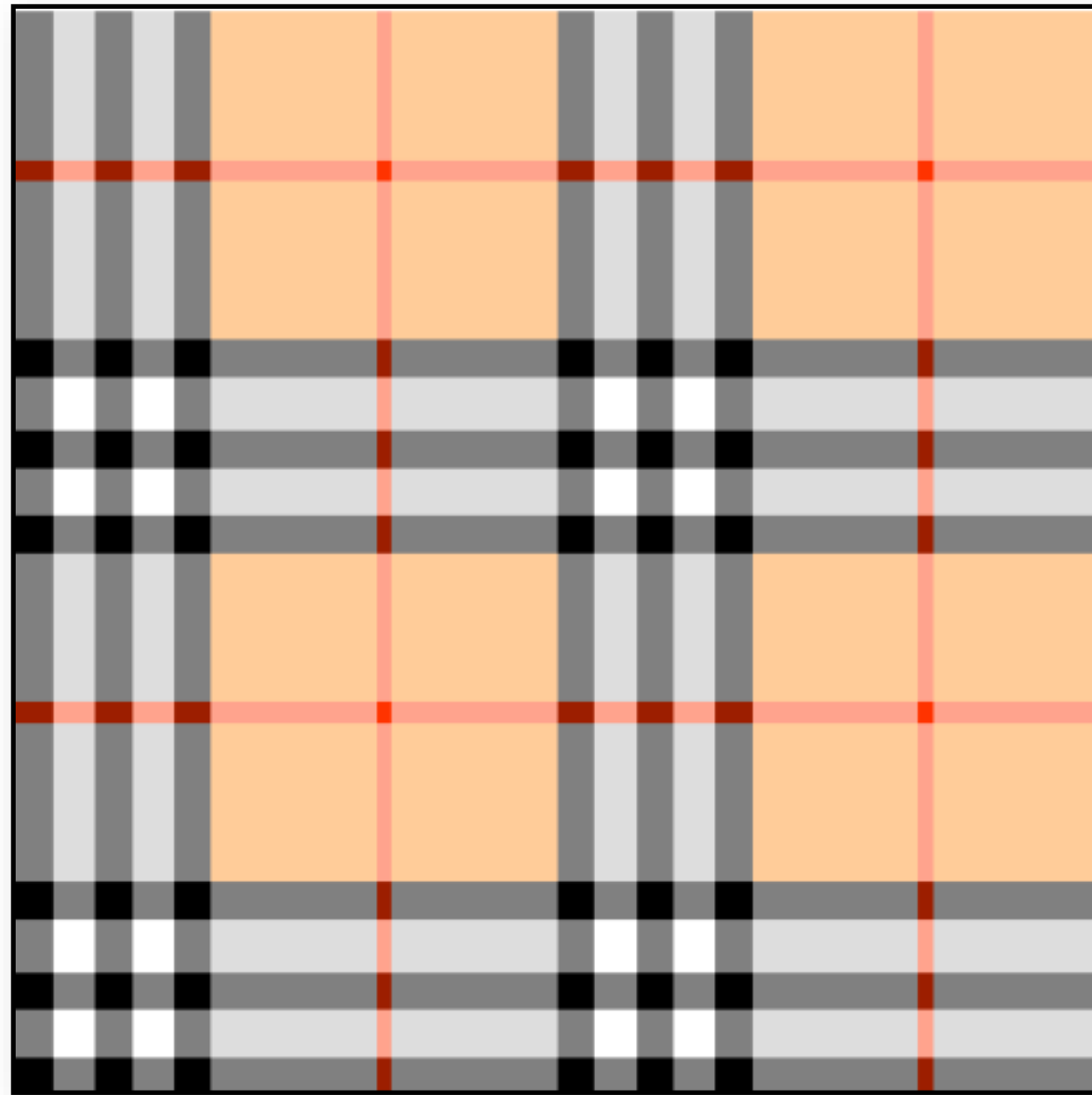


$$(u, v) = \alpha(u_1, v_1) + \beta(u_2, v_2) + \gamma(u_3, v_3)$$

UV texturing

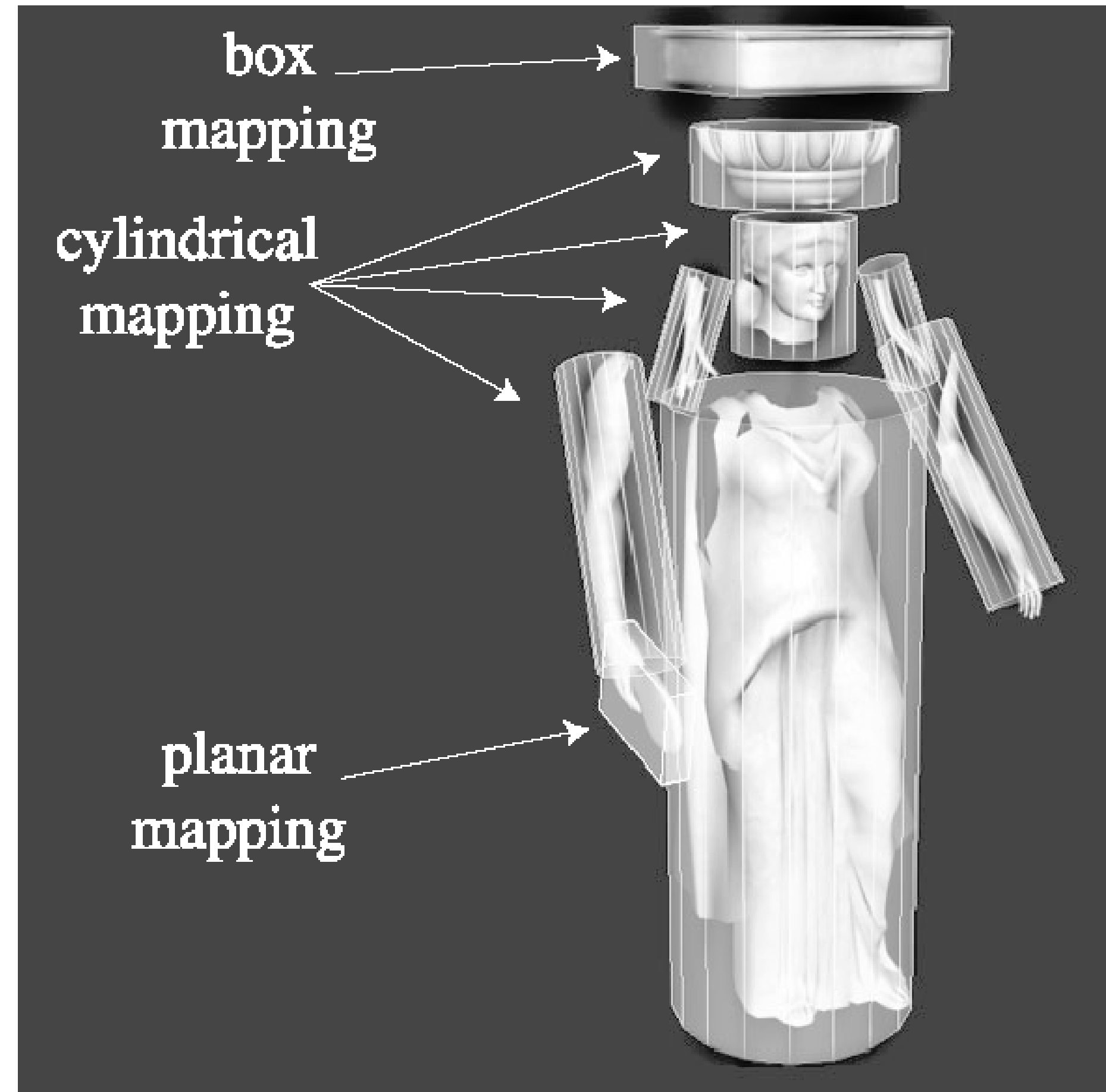


Area distortion vs. angle distortion



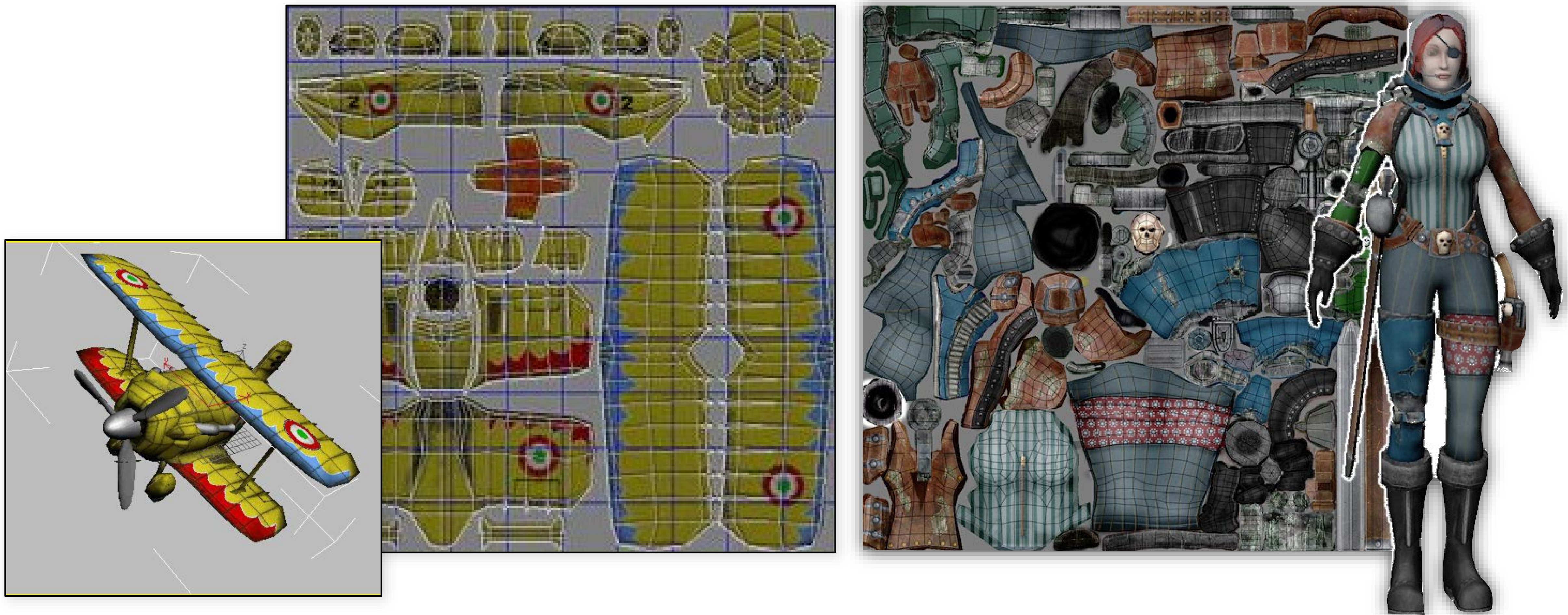
Creating UV parameterizations

Can compute vertex UVs using projections



Creating UV parameterizations

“Atlas” - break up model into single texture



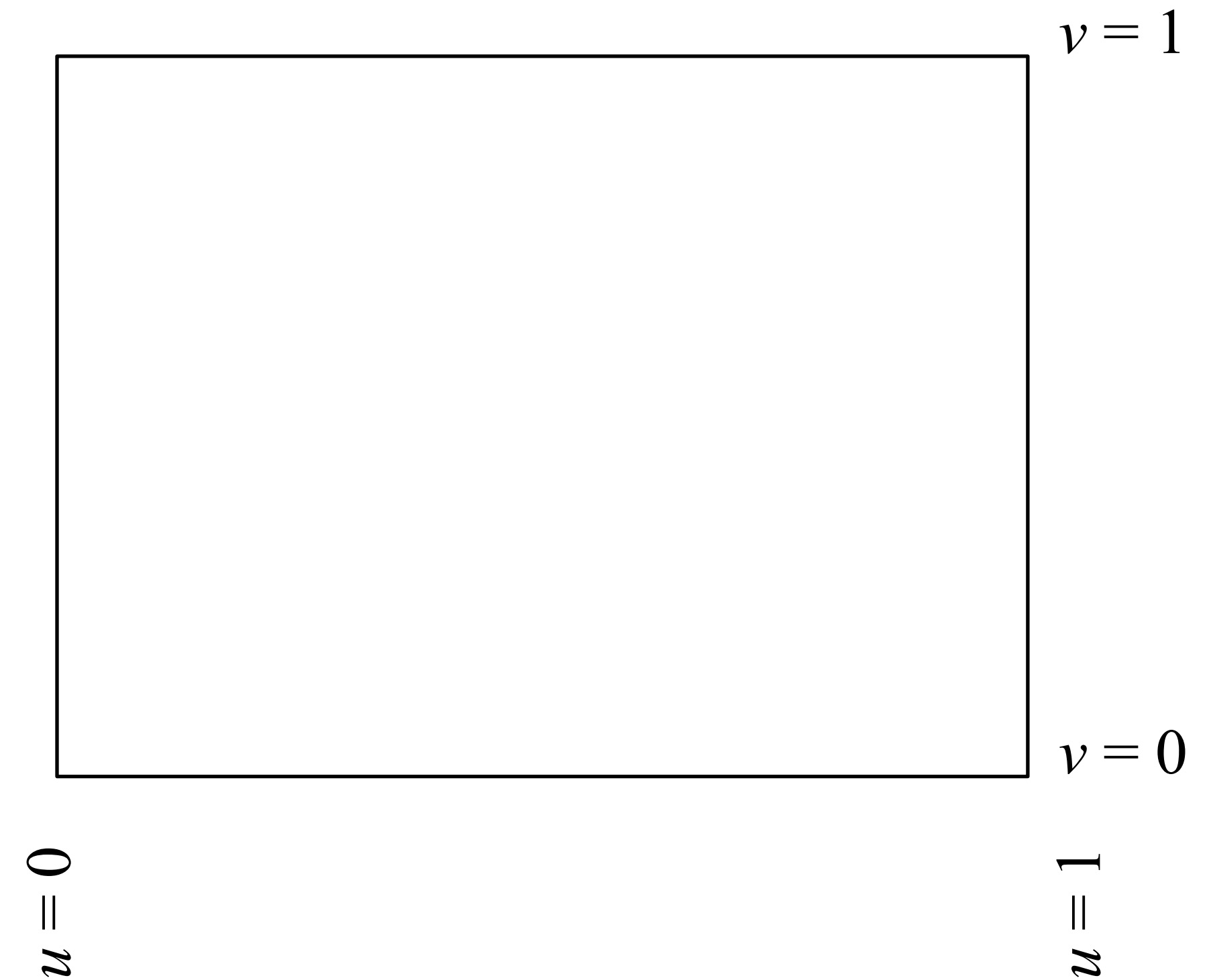
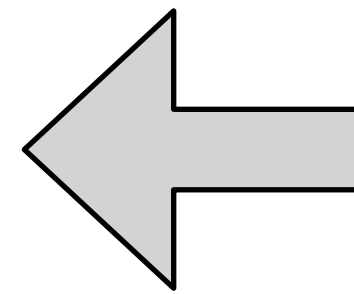
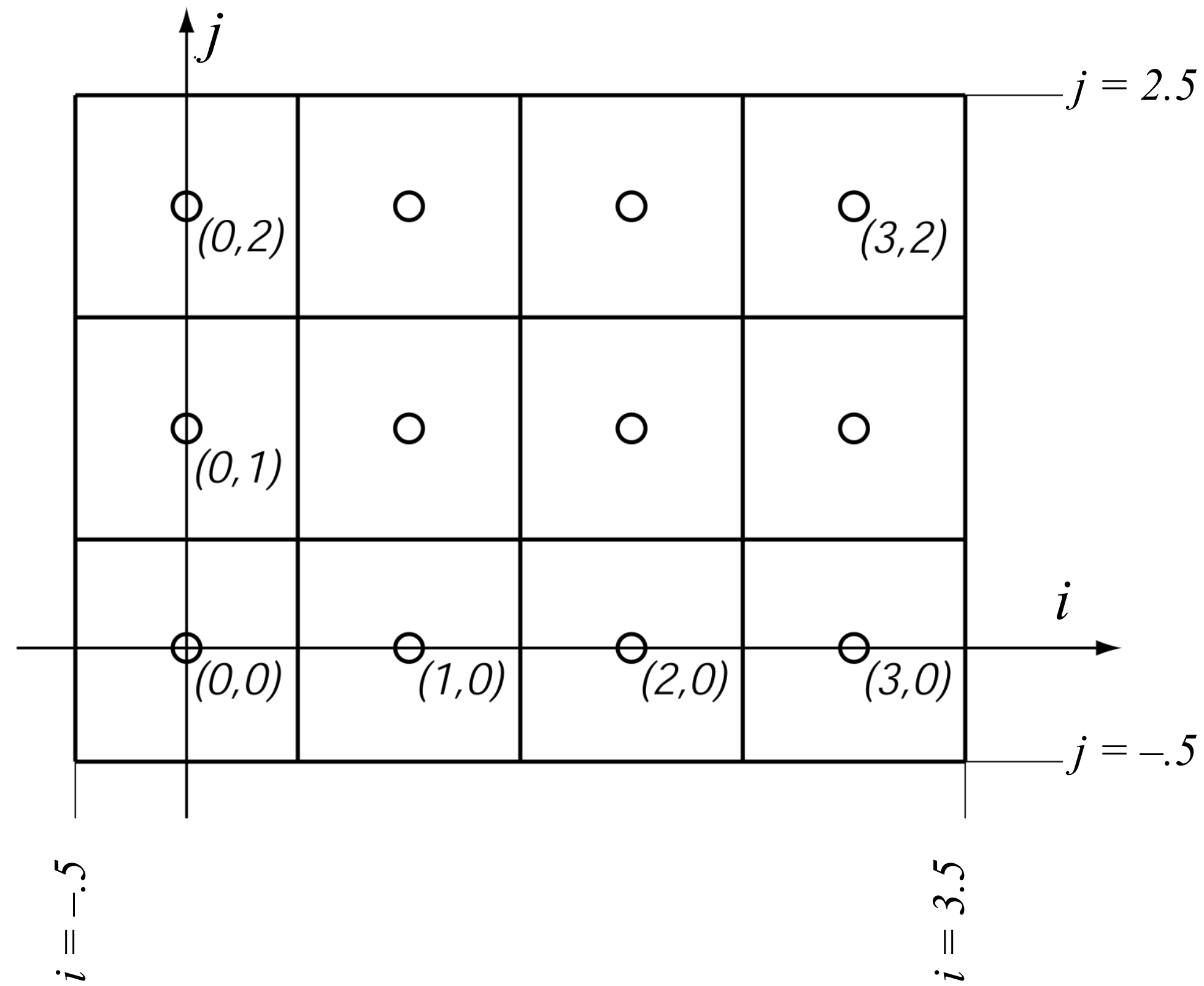
Texture lookup

Texture lookups and wrapping

In shading calculation, when you need a texture value you perform a *texture lookup*

Convert (u, v) texture coordinates to (i, j) texel coordinates, and read a value from the image

Obtaining (i,j) from (u,v)



$$i = u n_x - 0.5$$
$$j = v n_y - 0.5$$

for an image of
 n_x by n_y pixels

Looking up texture values

Lookup locations will fall at fractional texel coordinates

- simplest: round to nearest (nearest neighbor lookup)
- various ways to be smarter and get smoother results

Texture lookups and wrapping

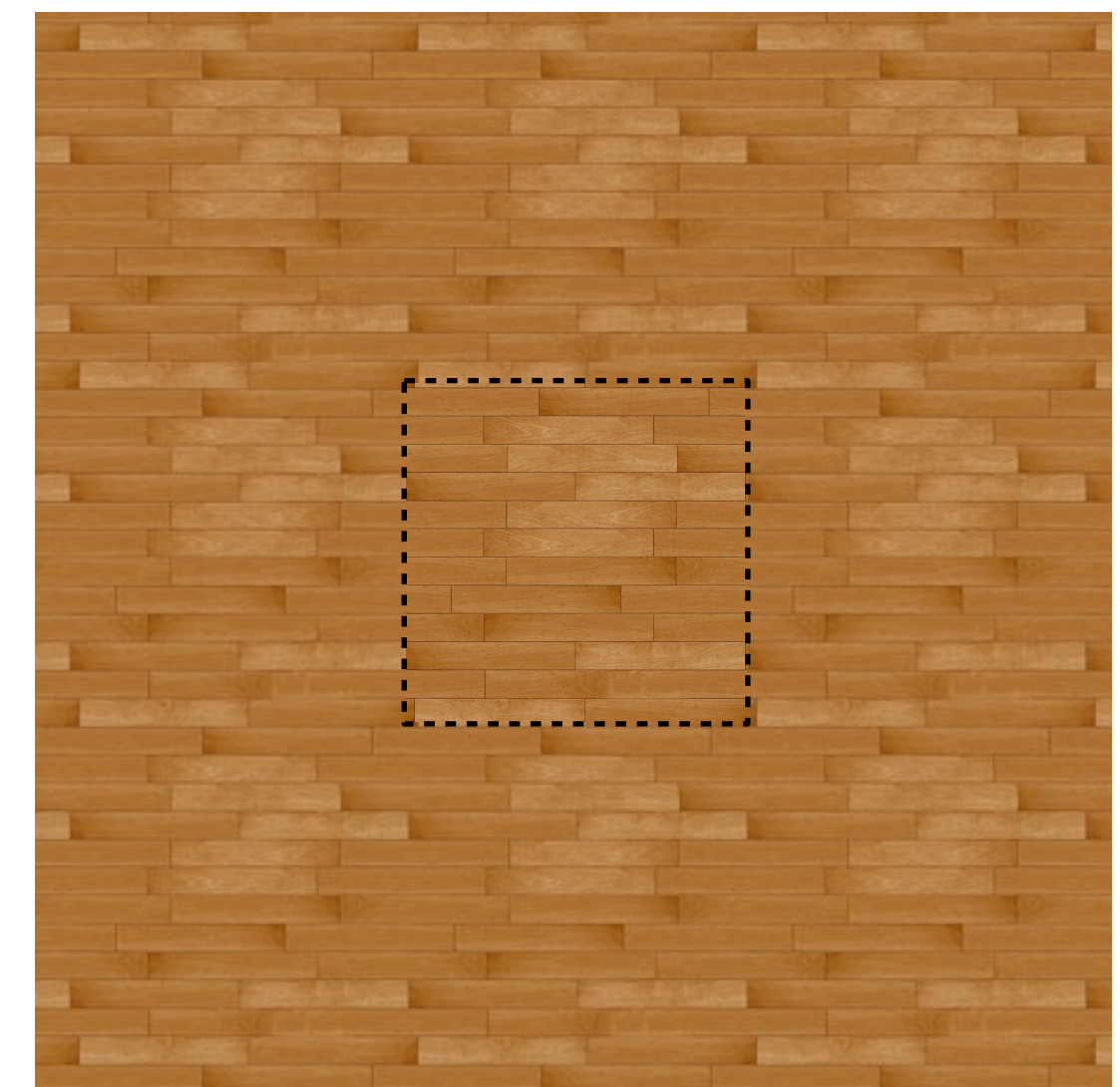
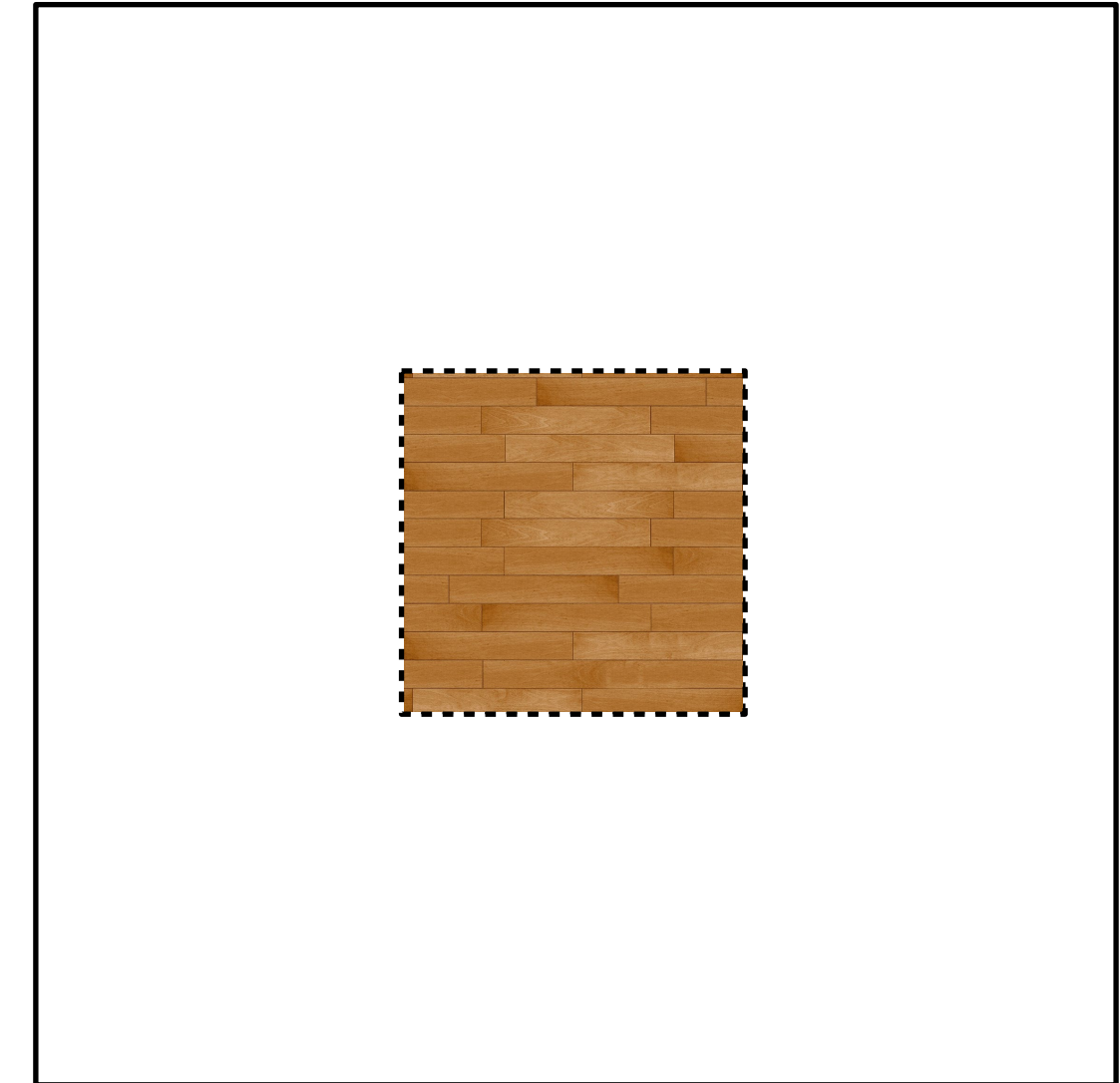
What if i and j are out of range?

Option 1, clamp: take the nearest pixel that is in the image

$$i_{\text{pixel}} = \max(0, \min(n_x - 1, i_{\text{lookup}}))$$

Option 2, wrap: treat the texture as periodic, so that falling off the right side causes the look up to come in the left

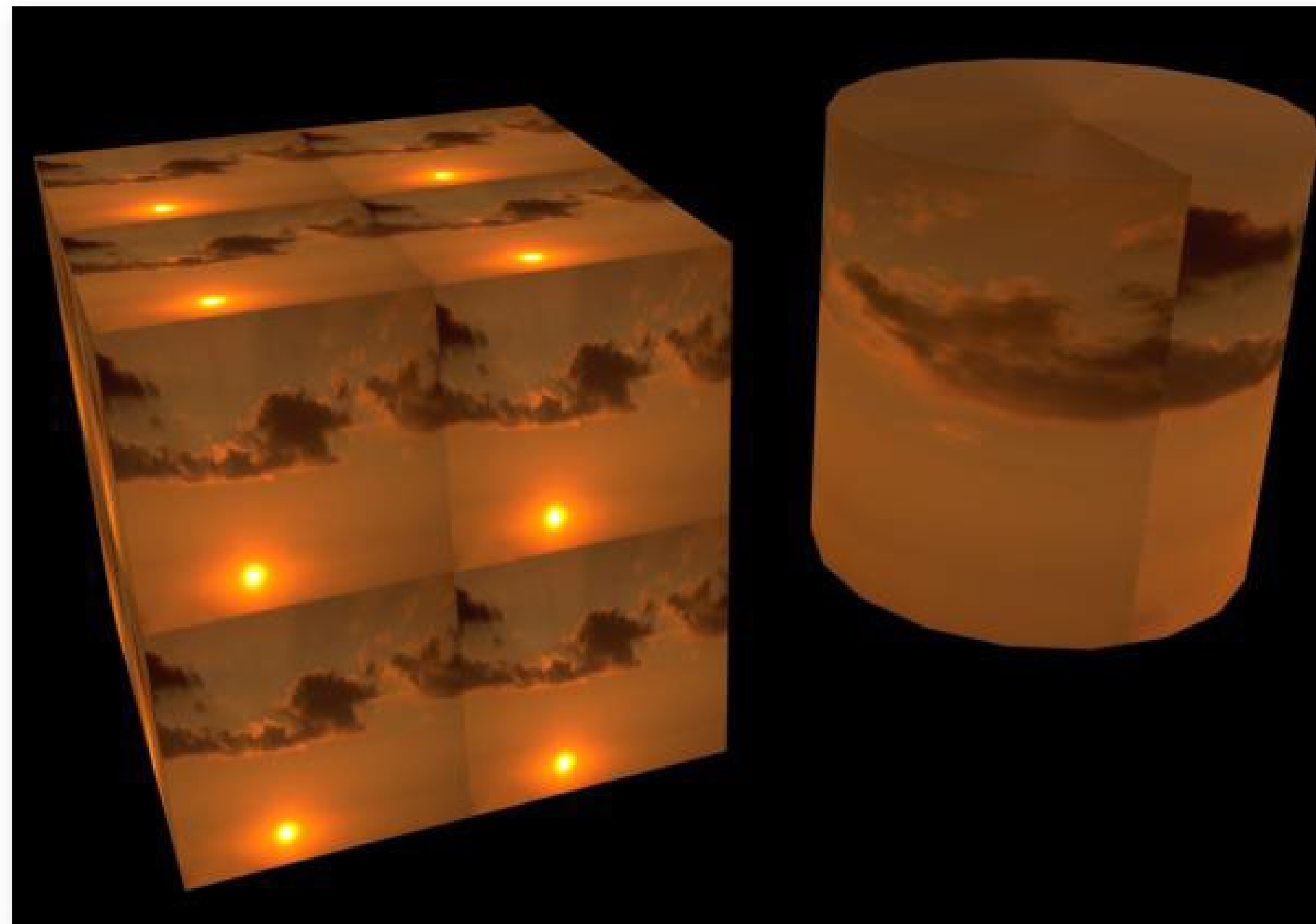
$$i_{\text{pixel}} = \text{remainder}(i_{\text{lookup}}, n_x)$$



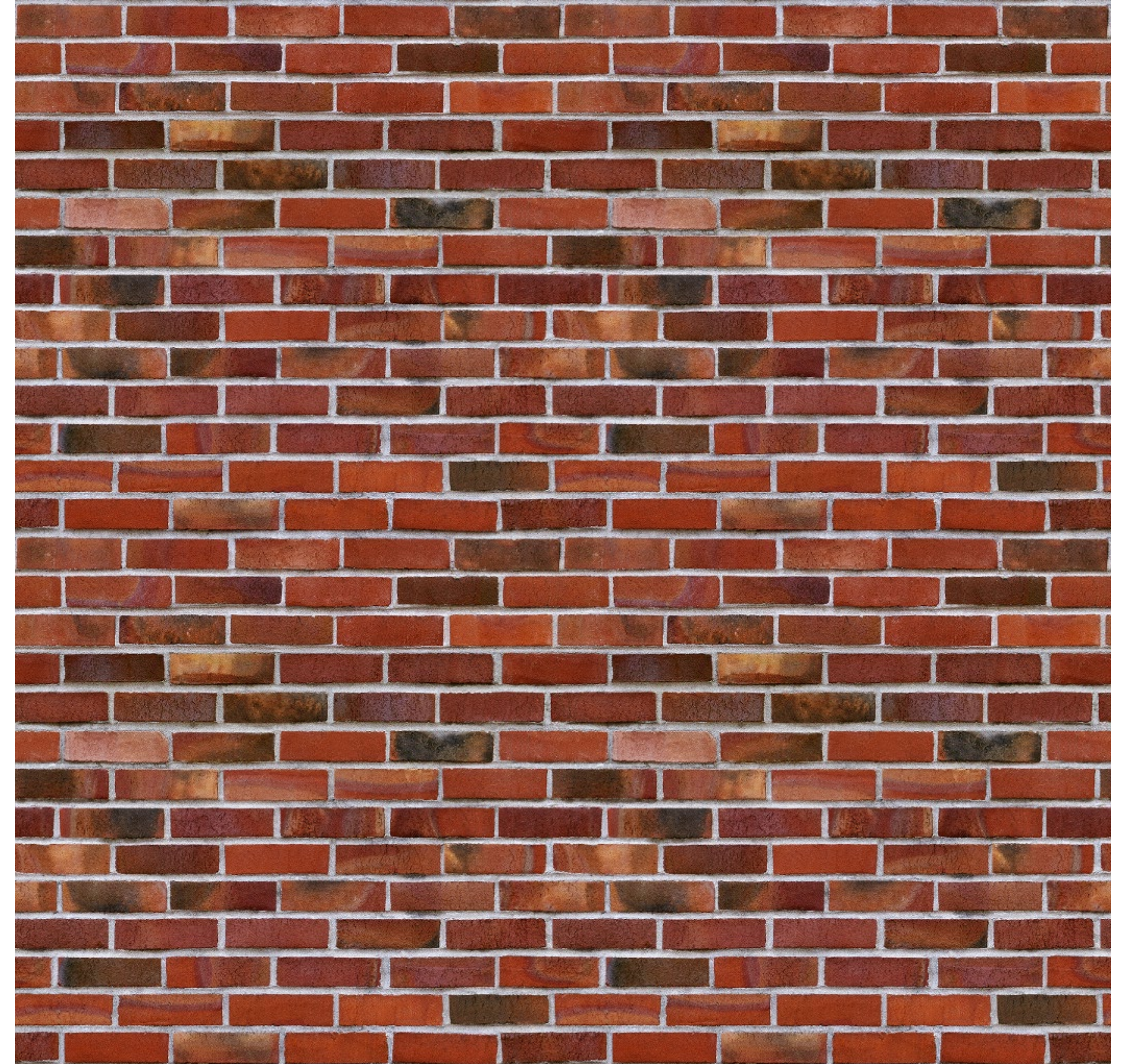
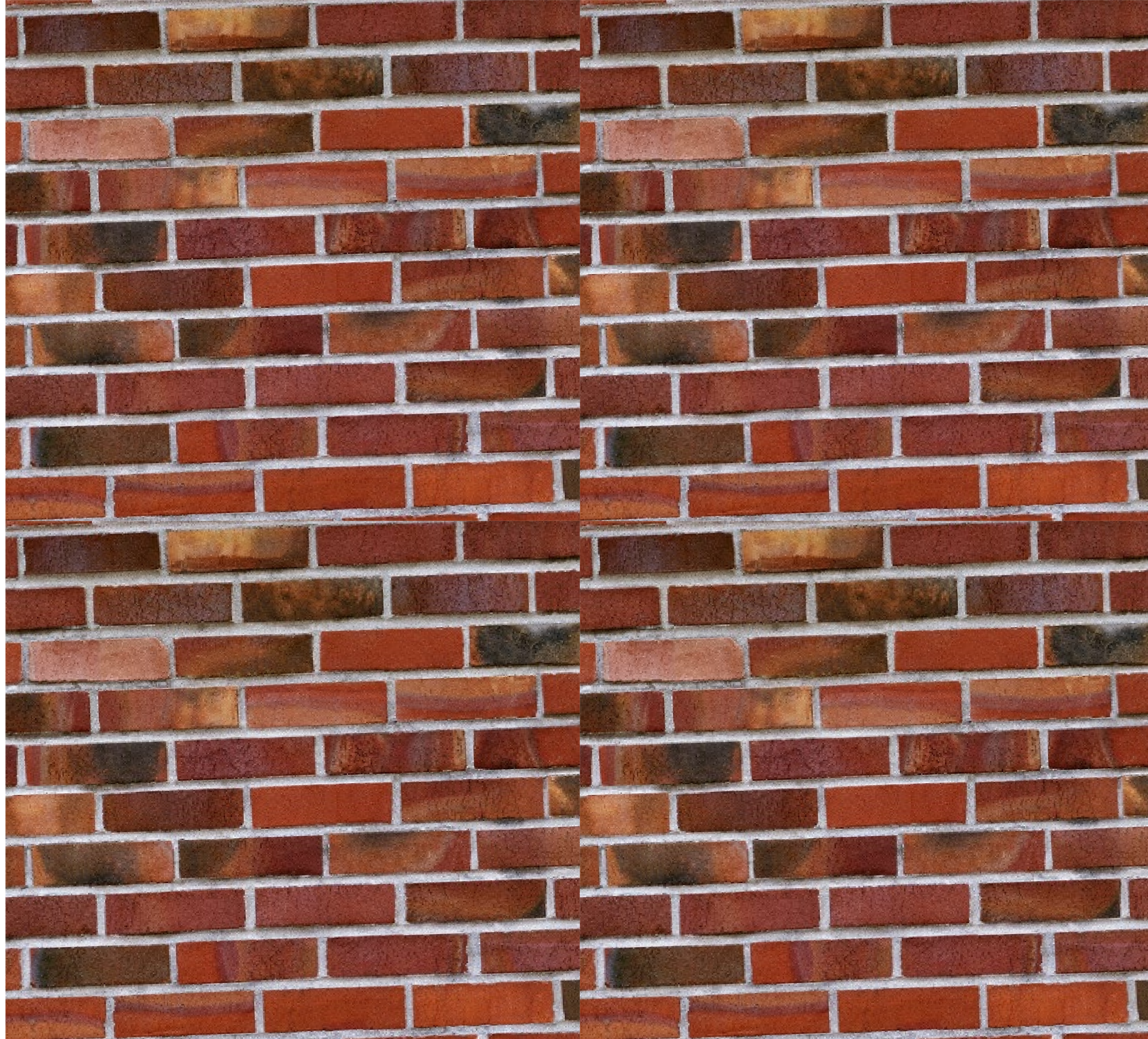
Texture mapping artifacts

Tiling textures might introduce seams

- discontinuities in the mapping function
- change textures to be "tileable" when possible



Seamlessly “tileable” textures



Pixels versus texels

In general, we will not have a 1-to-1 mapping between image pixels and texture pixels, or “texels”

Two issues arise:

- **Magnification:** Texel size larger than pixel size
- **Minification:** Texel size smaller than pixel size (potential aliasing)

Texture Filtering - Magnification



Nearest Neighbor

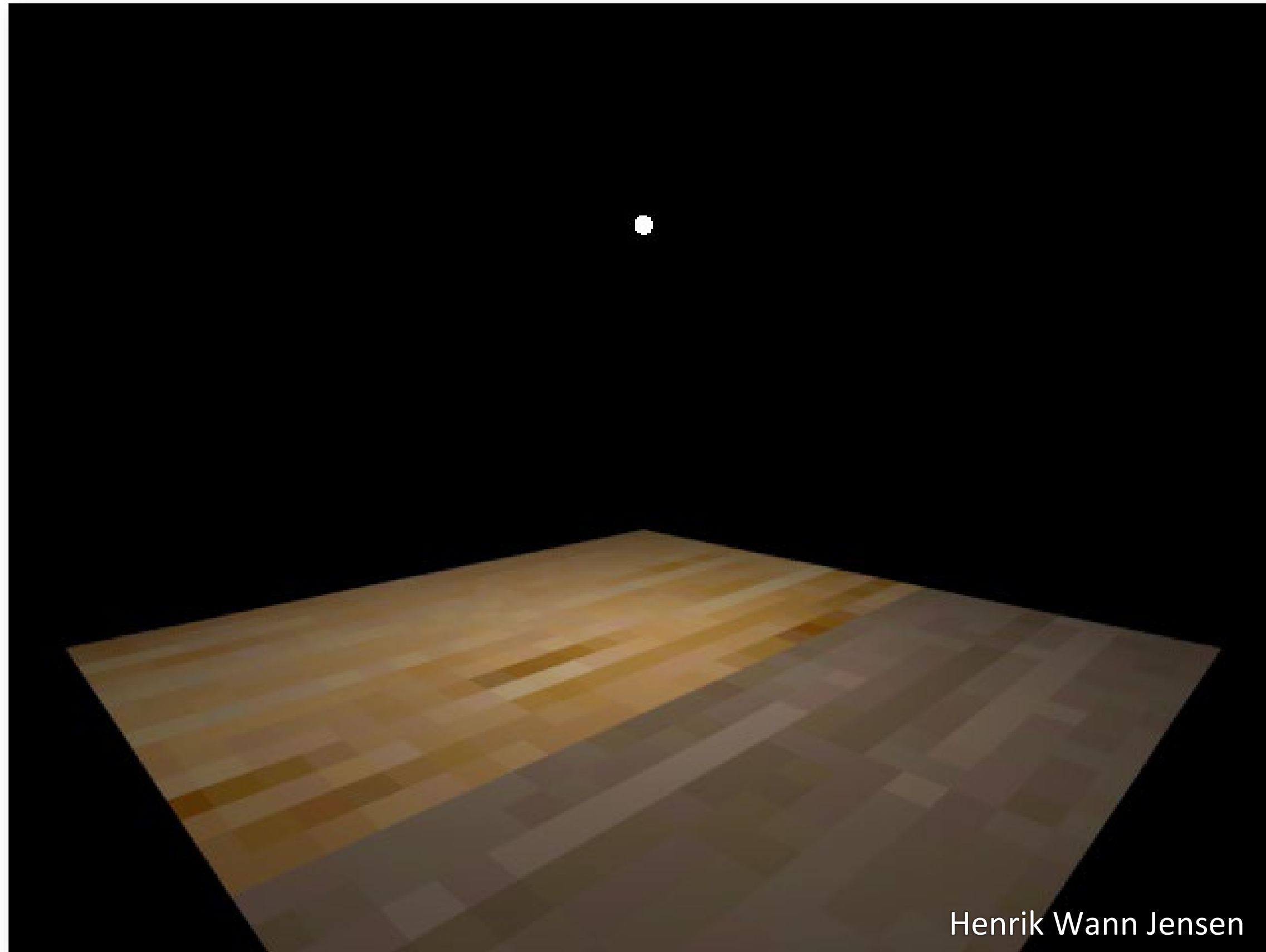


Bilinear

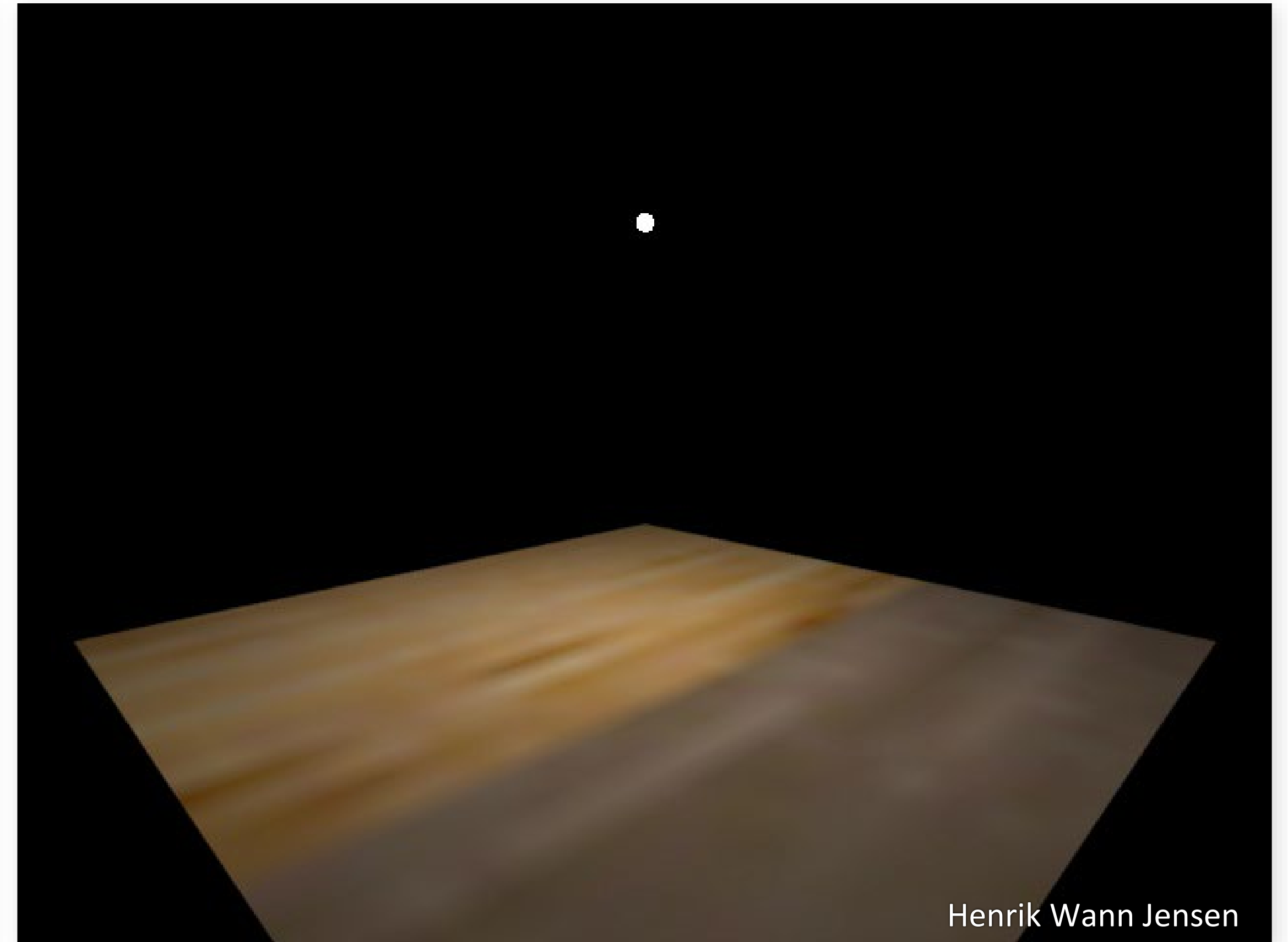


Bicubic

Texture Filtering - Magnification



Nearest Neighbor

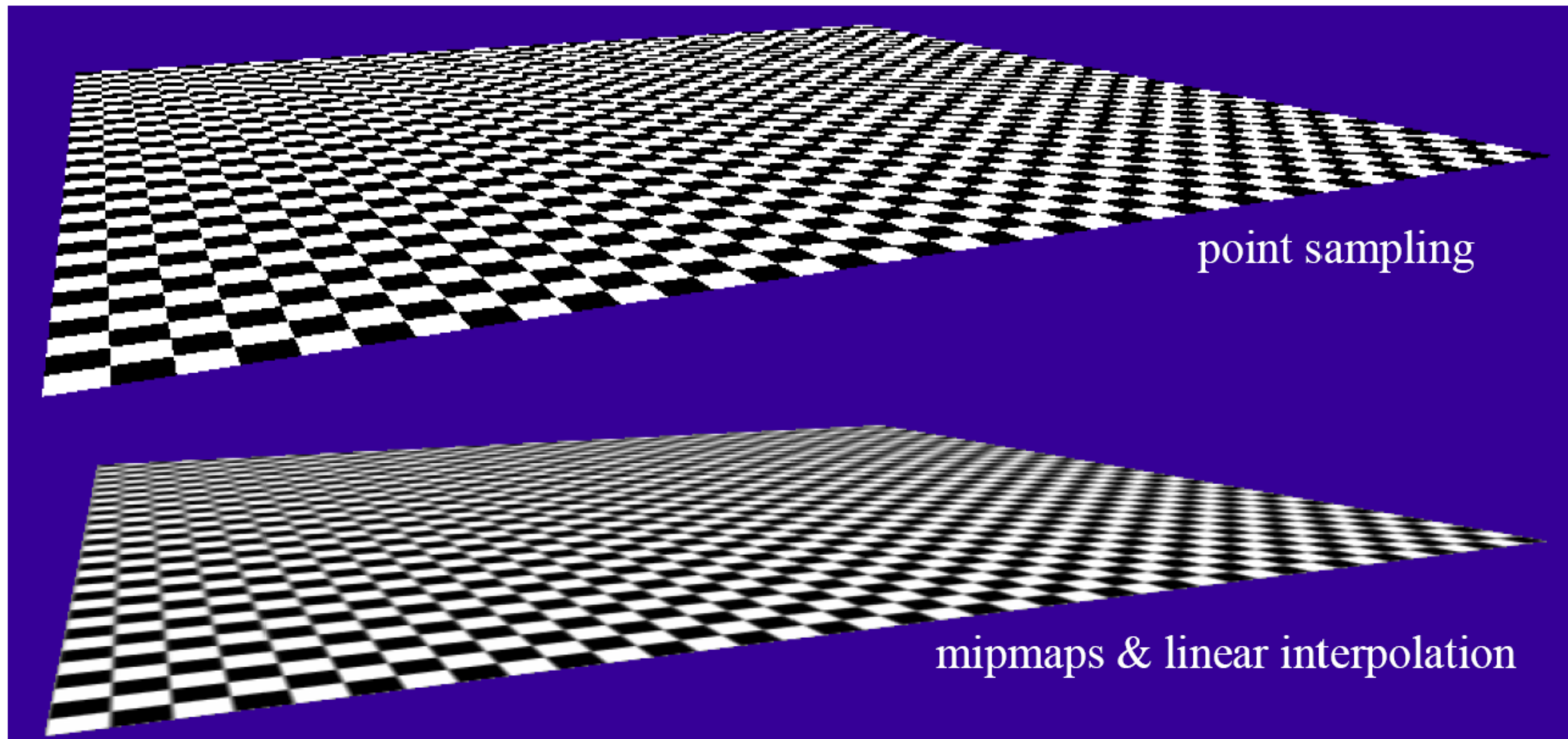


Bilinear

Texture minification (aliasing)

Point-sampling introduces artifacts (aliasing)

- need average of texture within area of a pixel



Texture minification

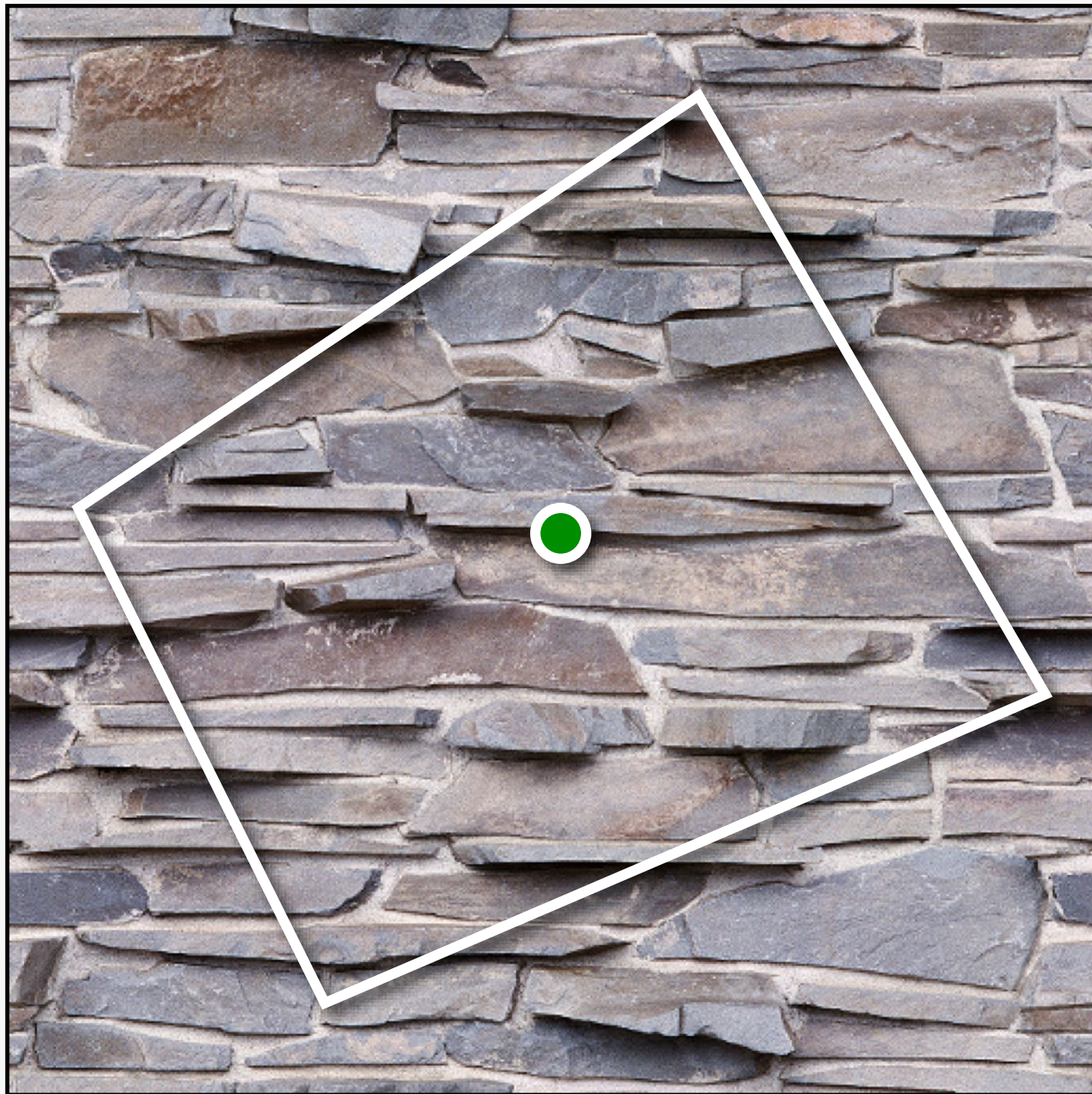
In ray tracing, in theory, you're already sending many rays randomly through area of pixel

Minification artifacts will go away, eventually

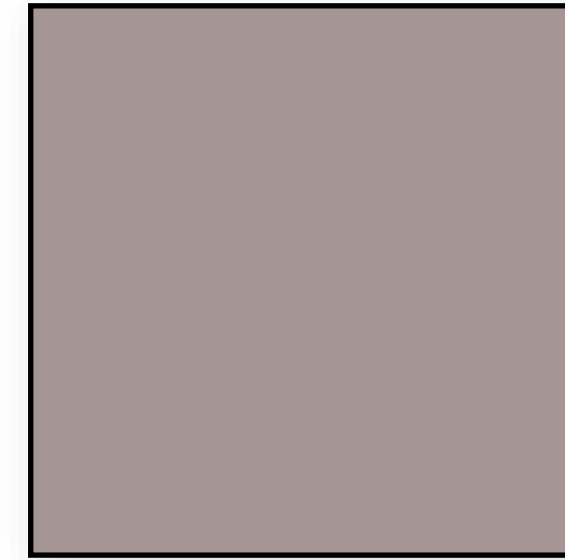
In production, texture filtering techniques still useful for improved speed/quality

- mipmapping
- ripmapping
- summed area tables

Texture filtering - minification

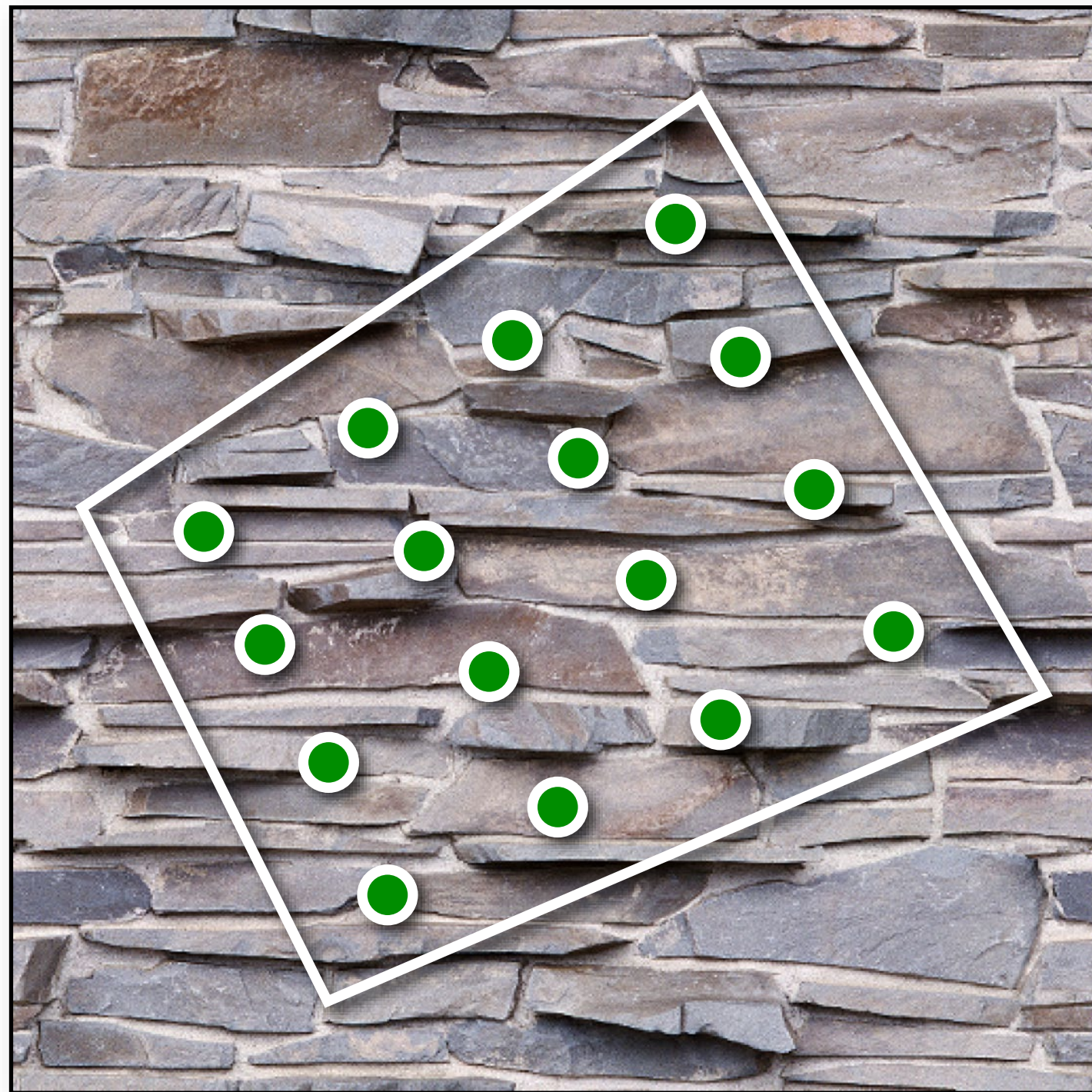


point sample



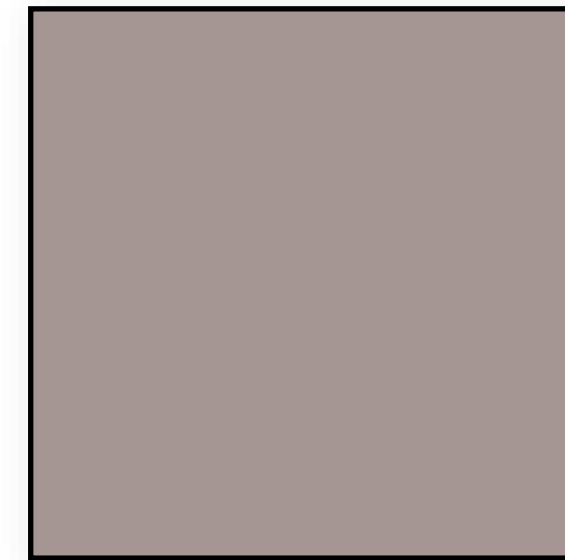
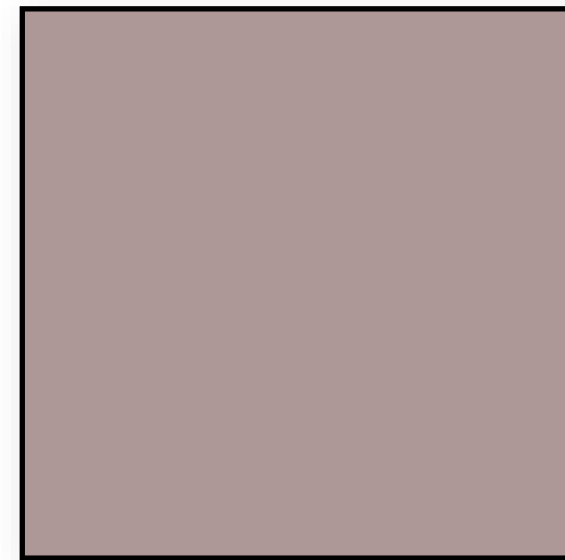
area average

Texture filtering - minification



4x4 supersampling

expensive!

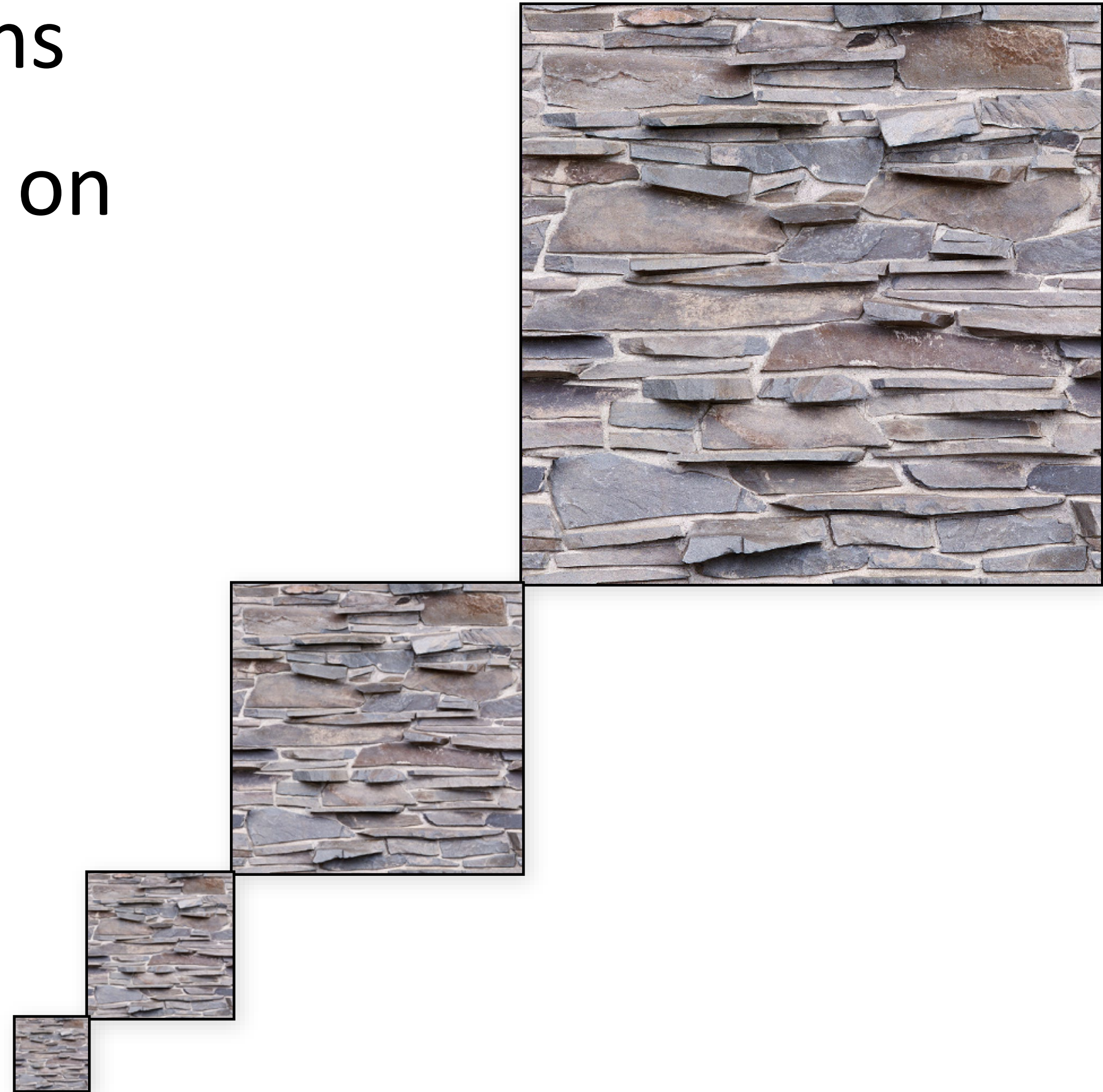


area average

Mipmapping

Store textures at different resolutions

Look up in appropriate image based on projected pixel size



Mipmapping

Bilinear: Look up in closest resolution and bilinearly interpolate

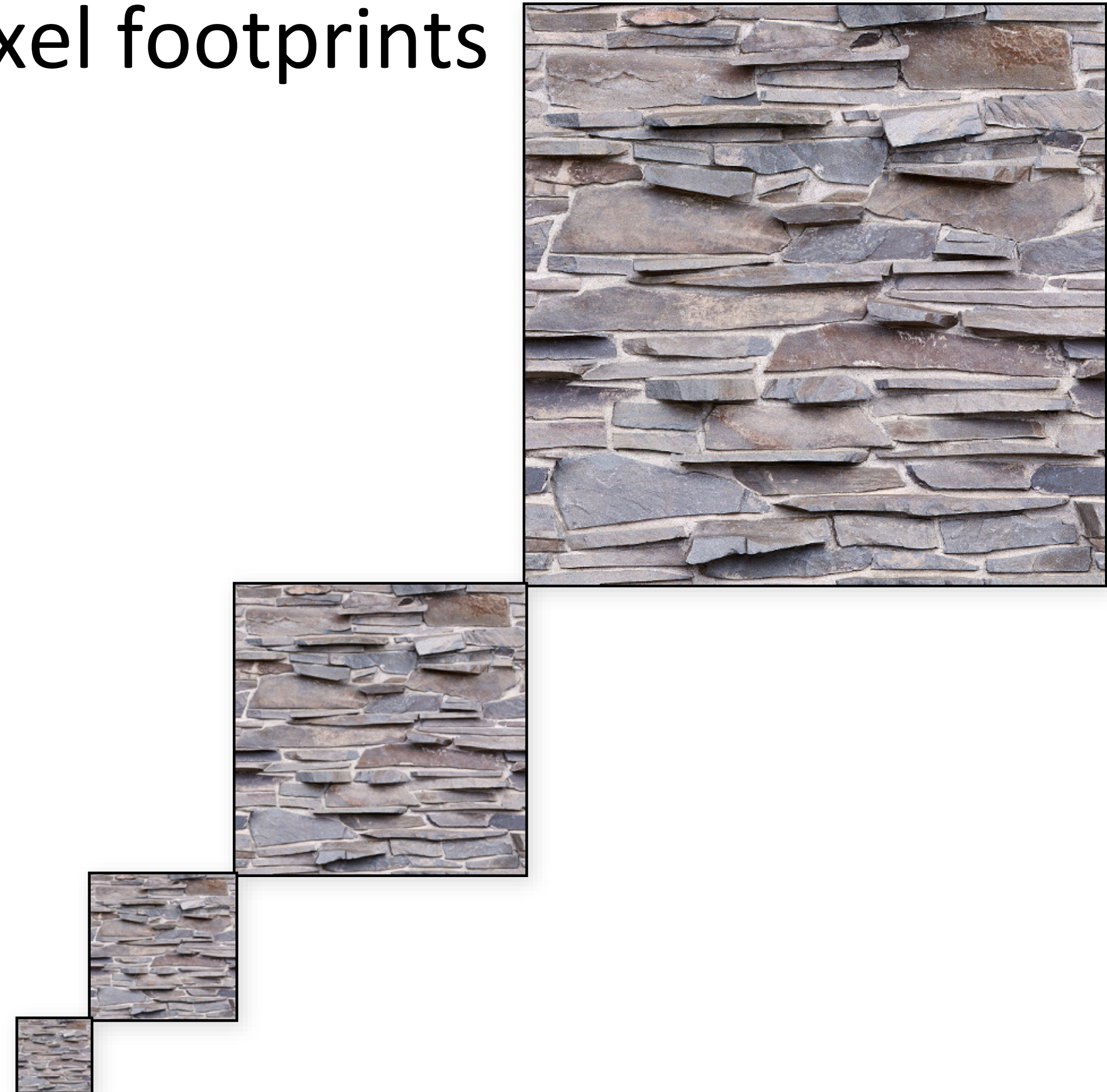
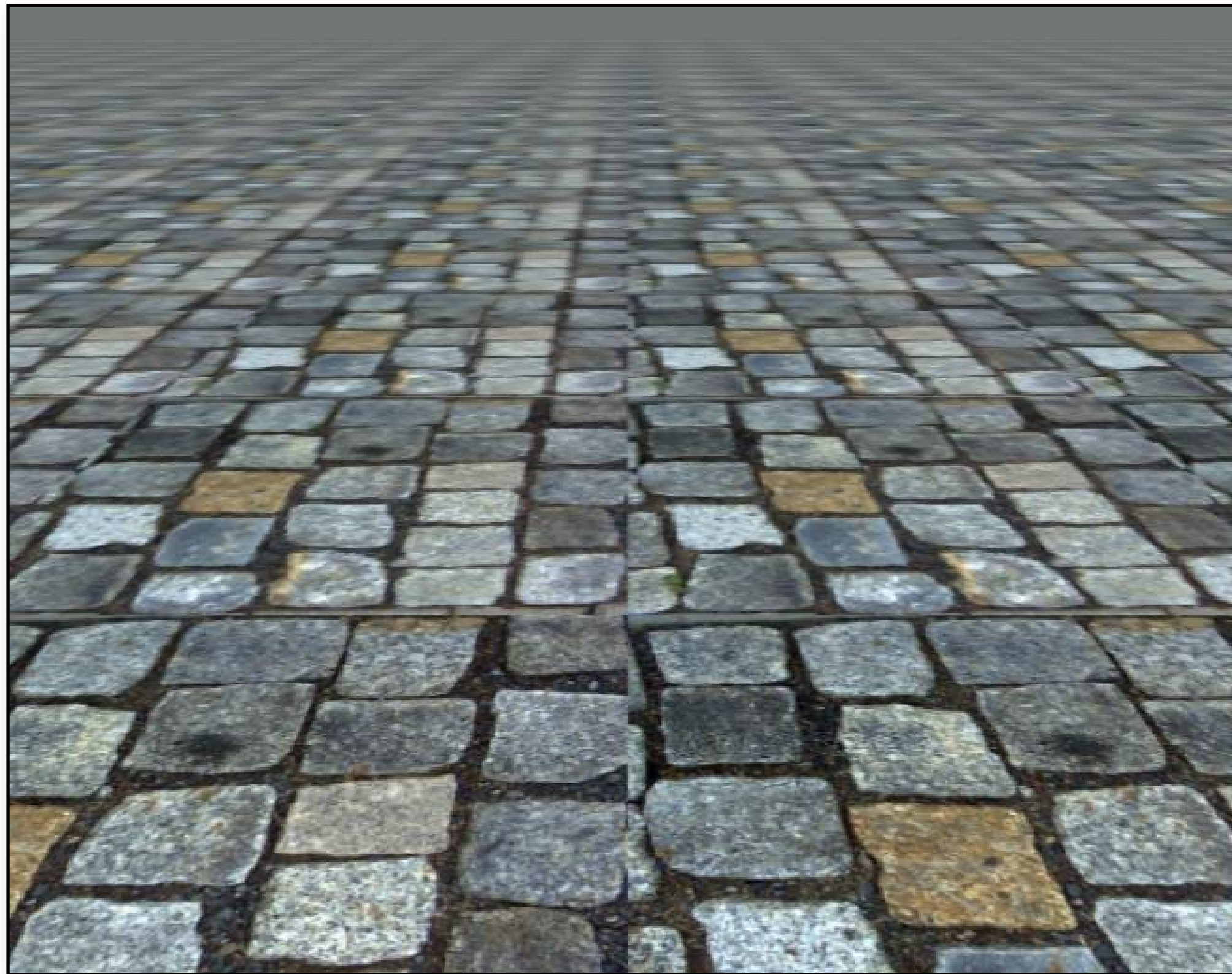
- transition artifacts

Trilinear: linearly interpolate between two closest levels



Mipmapping

Mipmaps average in squares, but pixel footprints are not square! → overblurring



Ripmap

One possible improvement:

- downscale independently in x, y



So far

How do we map between surface and texture?

- parametric surfaces
- projection mapping
- uv texturing
- texture lookup

Agenda

What do we map onto the surface?

- reflectance (color, diffuse + specular coeffs., etc)
- surface normal (bump mapping)
- geometry (displacement mapping)
- illumination (environment, reflection, shadows)

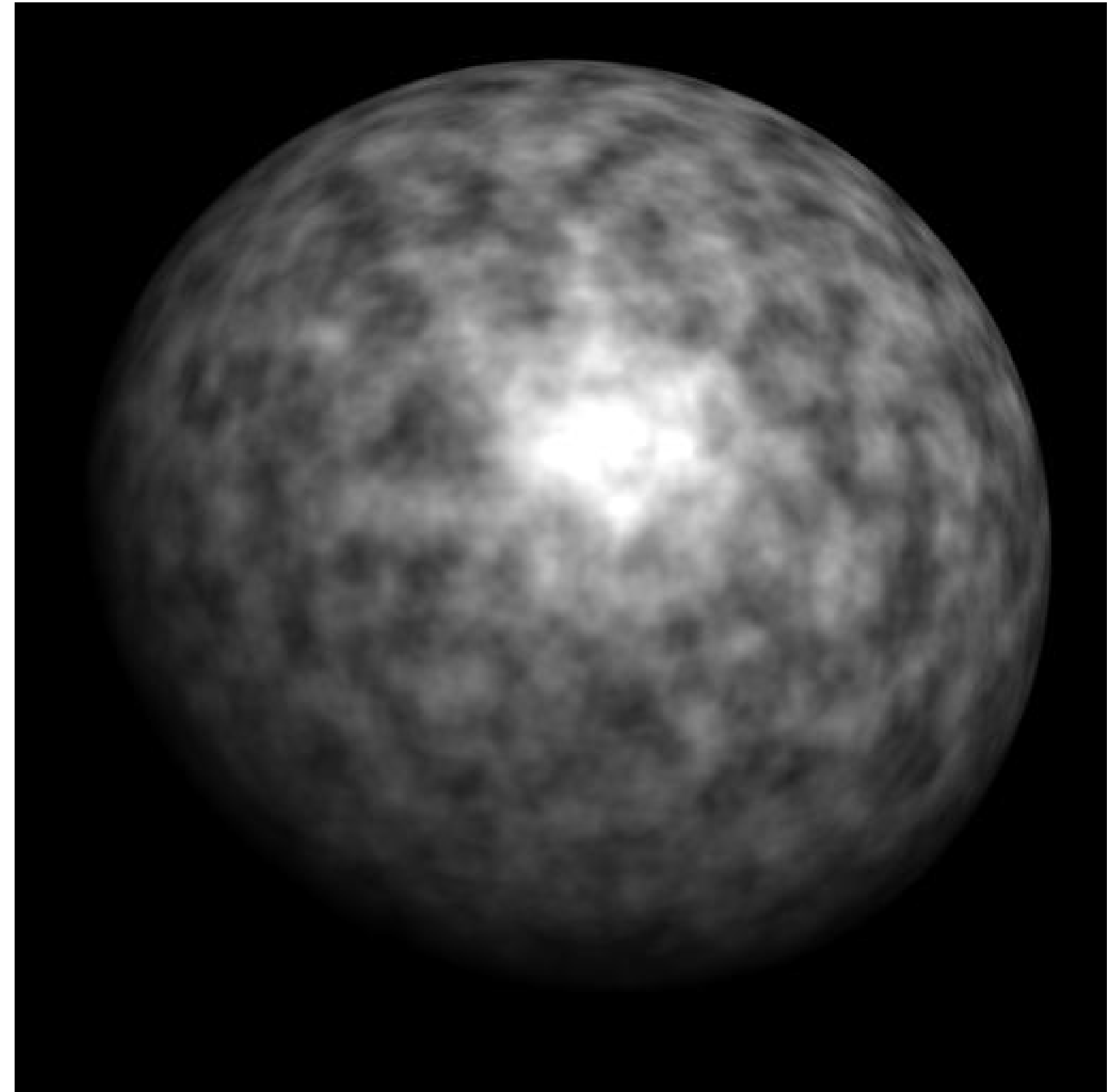
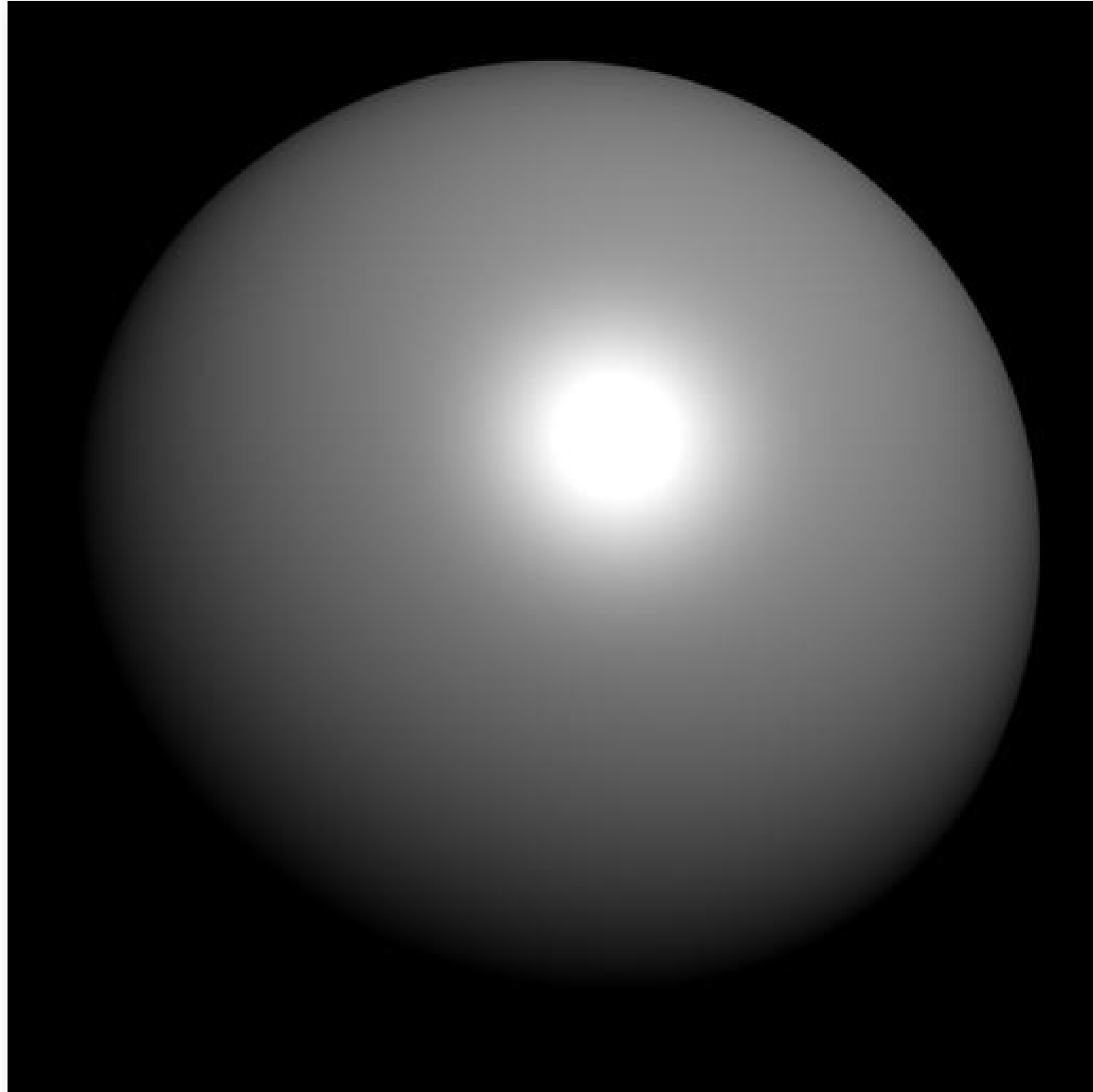
Texturing reflectance
properties

Texture mapping

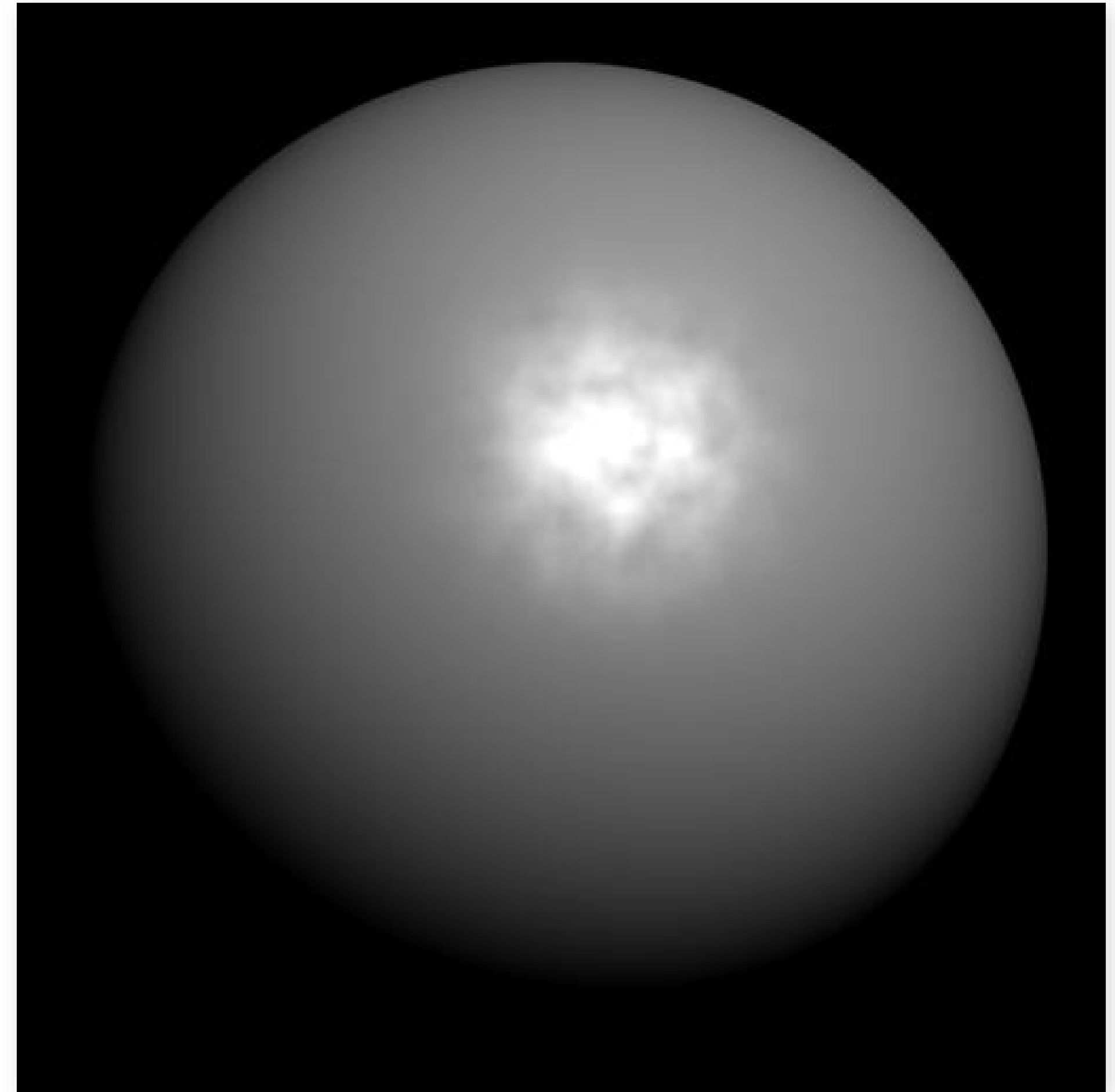
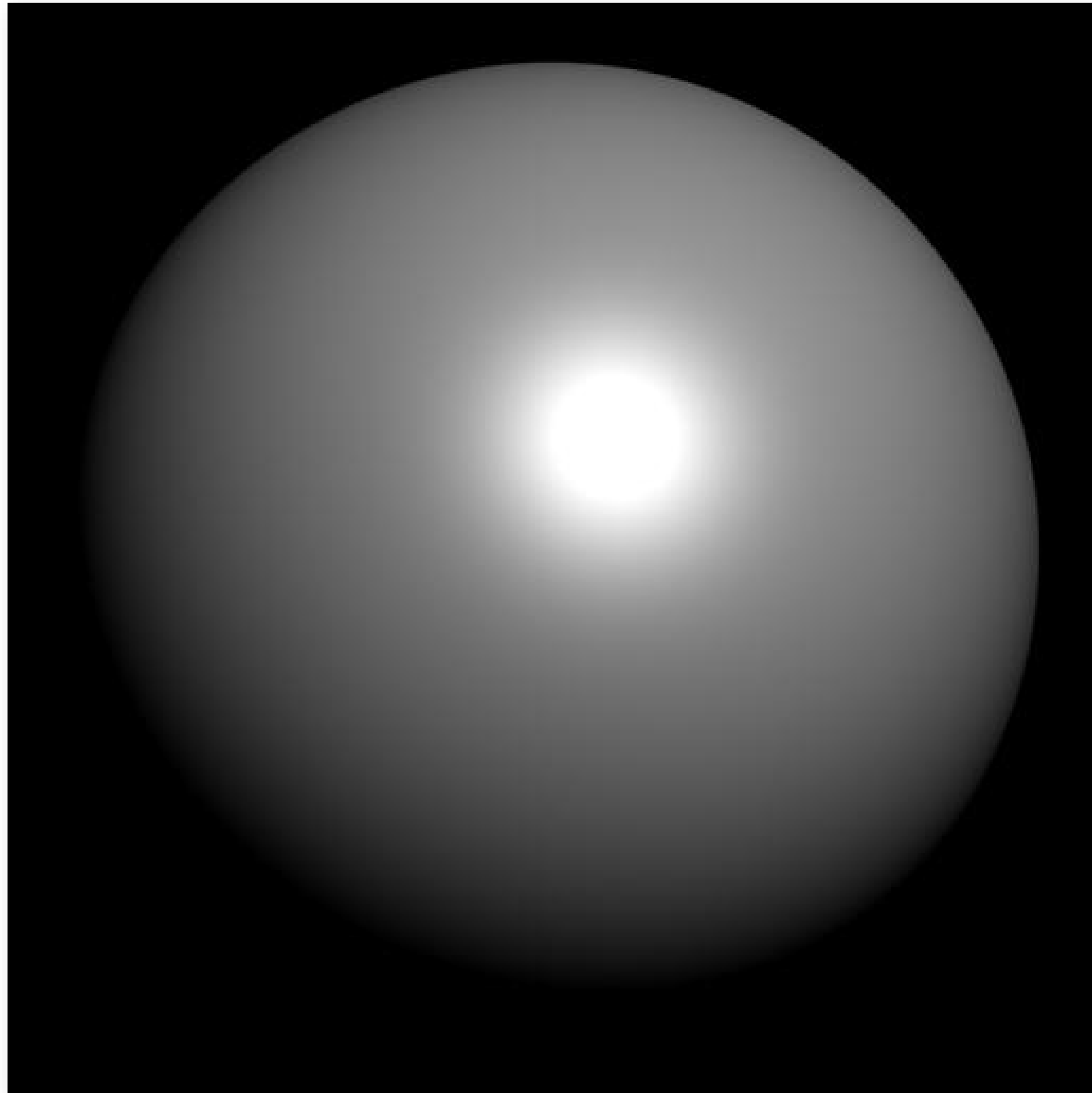
Surface properties are not the same everywhere

- diffuse color varies due to changing pigmentation
- roughness varies due to changing roughness and surface contamination

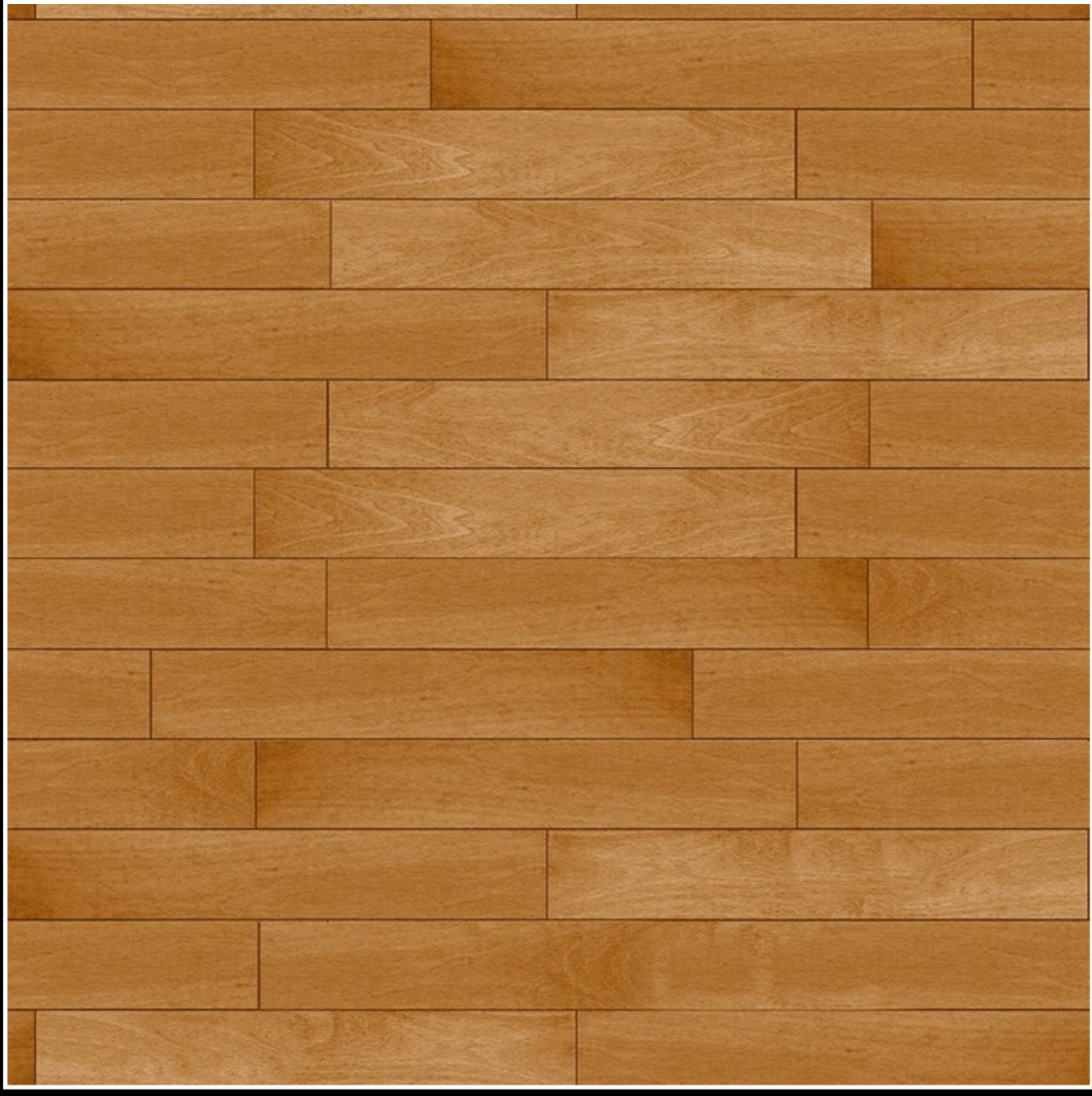
Diffuse coefficient



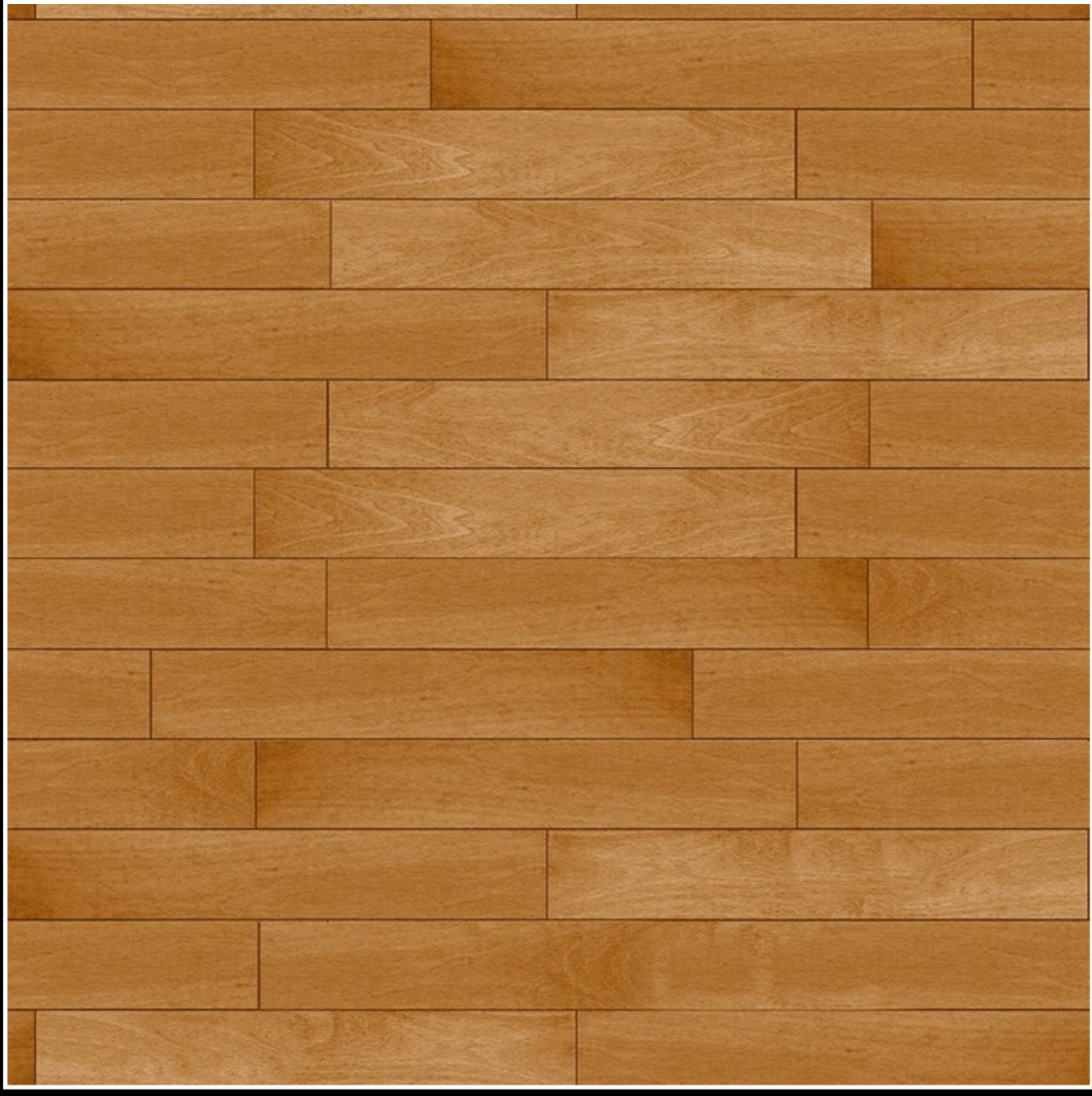
Specular “shininess” coefficient



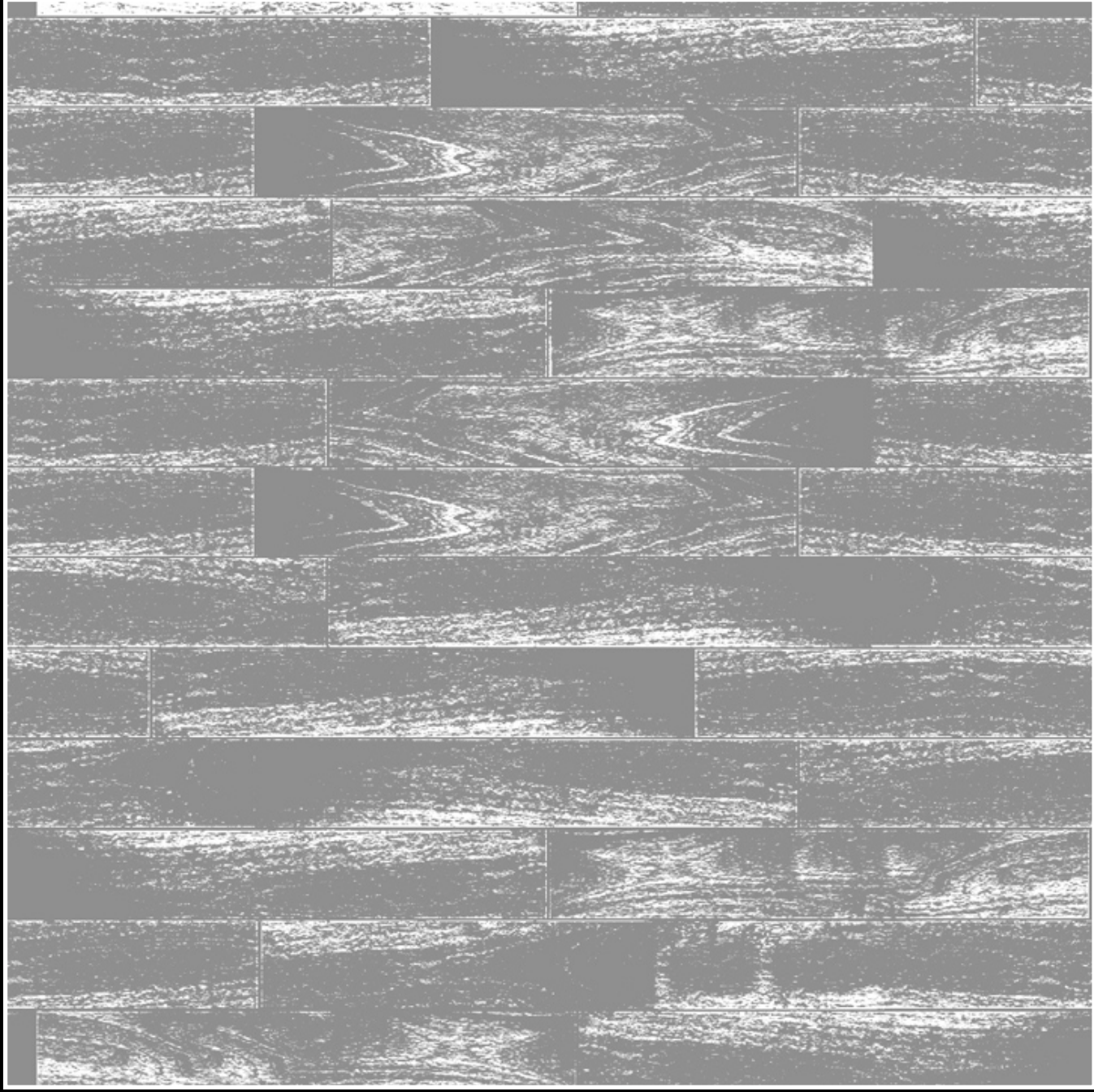
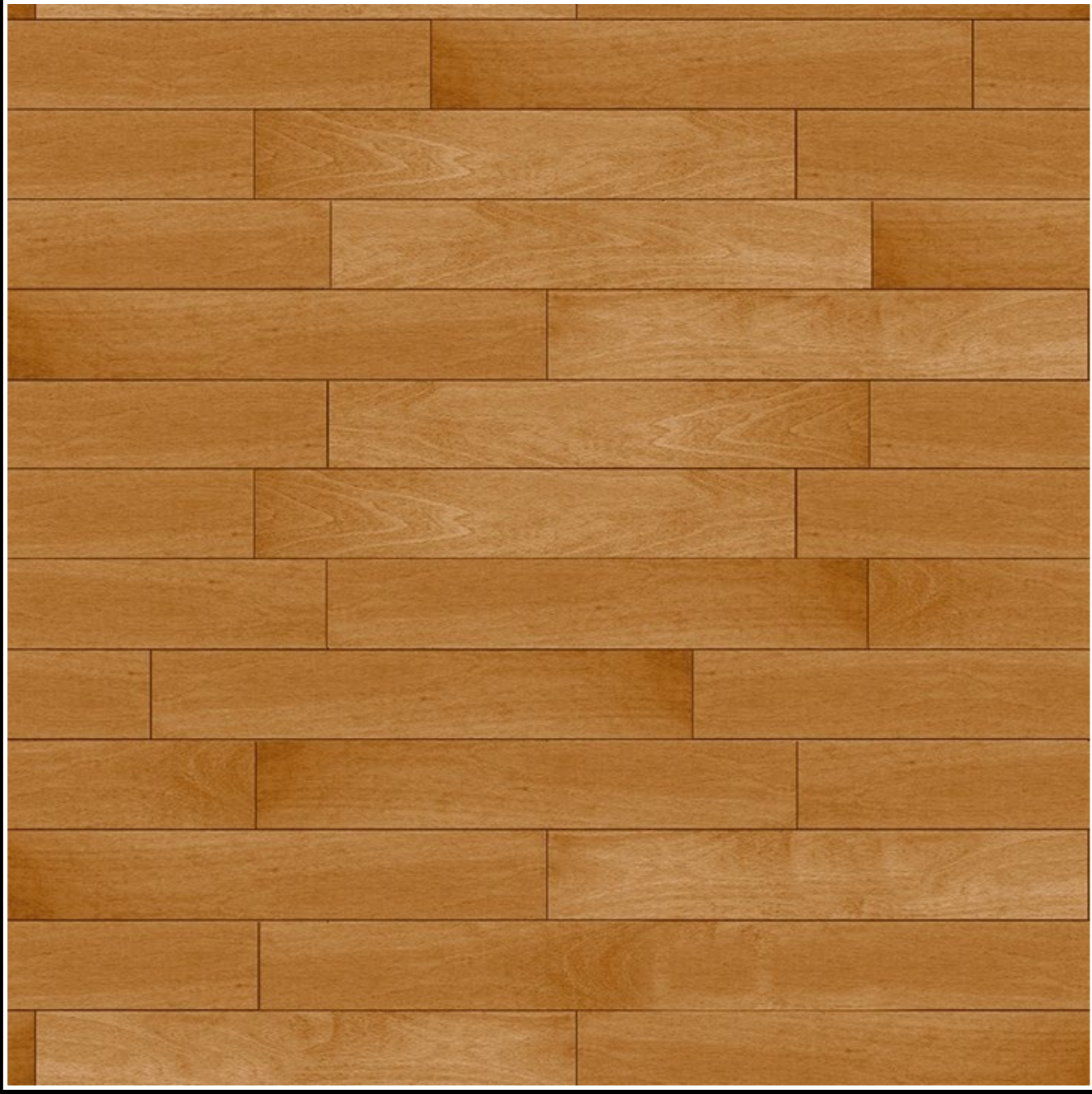
After a slide by Steve Marschner



After a slide by Steve Marschner



After a slide by Steve Marschner



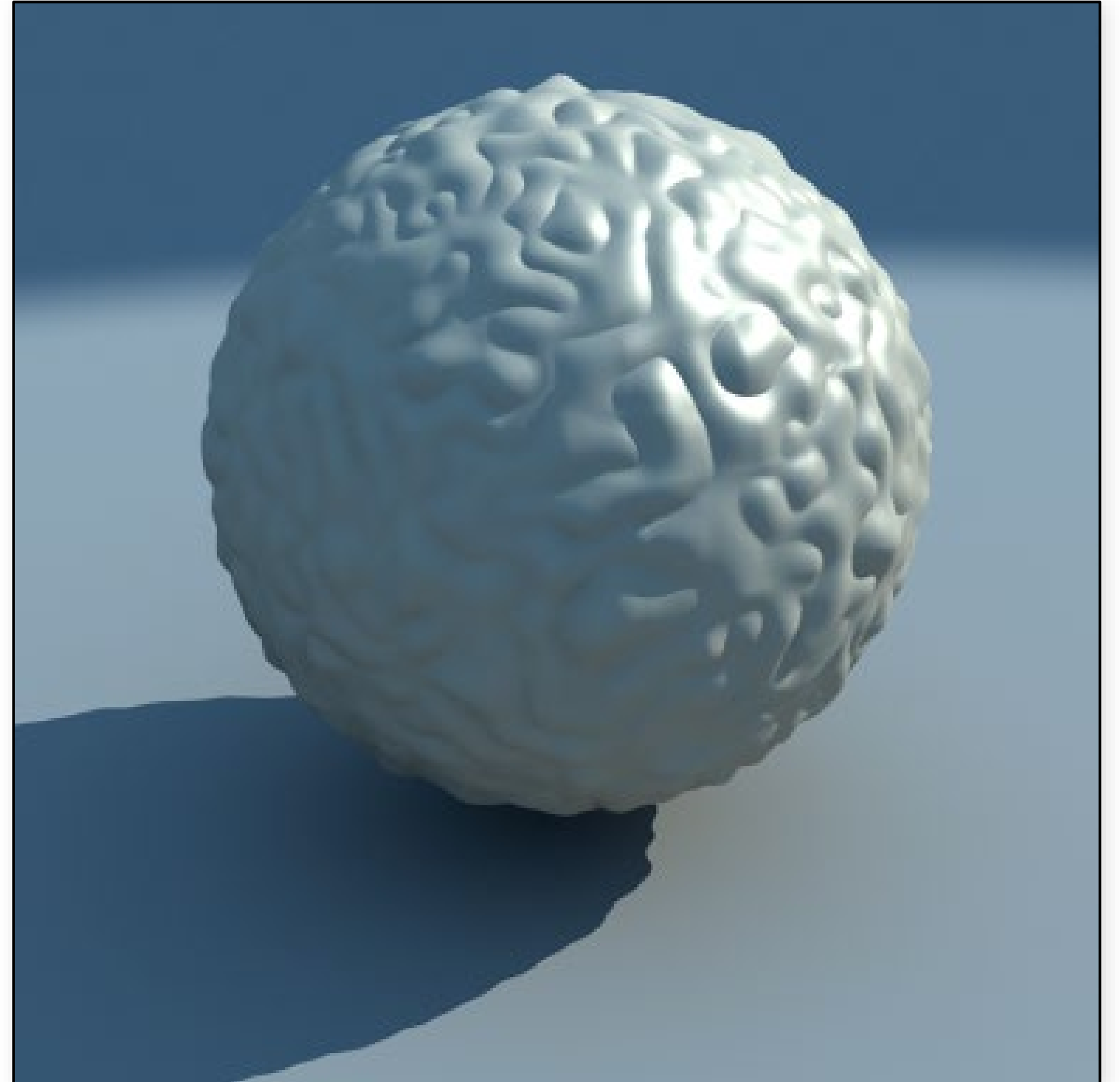
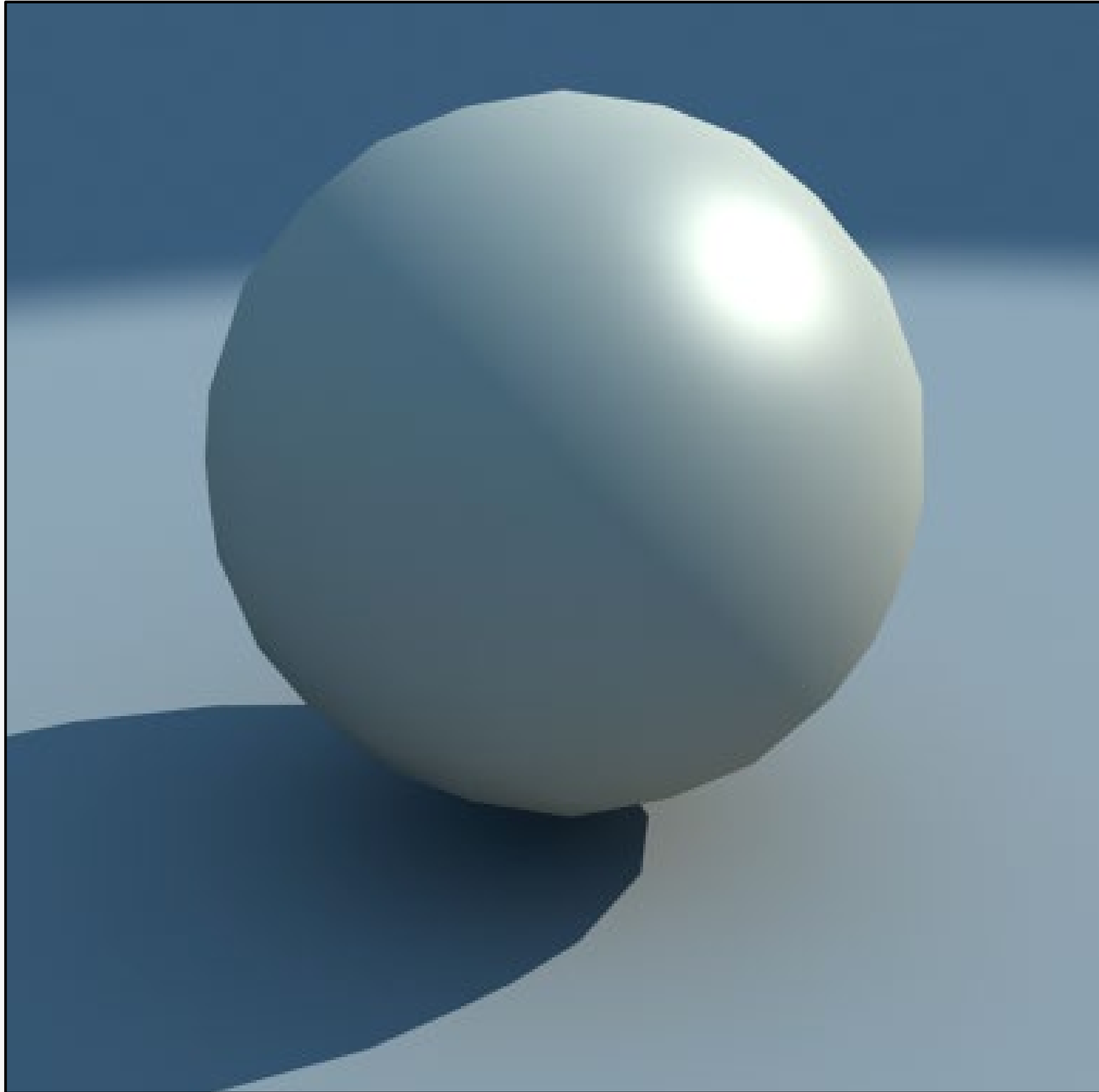
Texturing geometric
detail

Texturing geometric detail

Modify geometric properties based on a texture

- normals
- surface positions

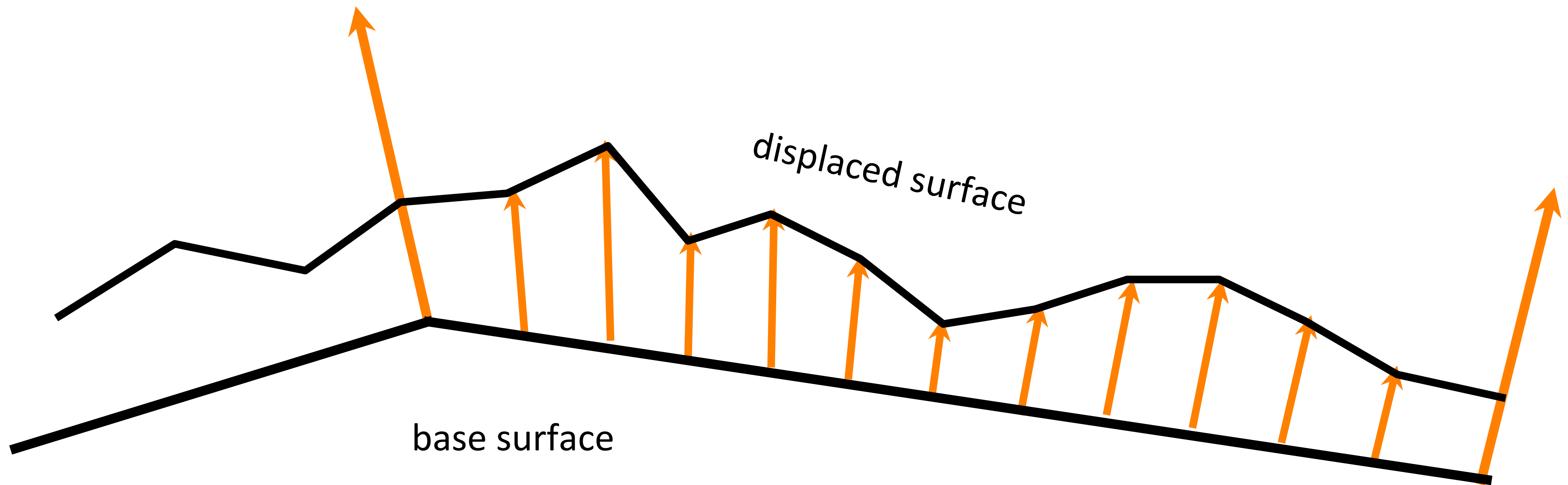
Displacement mapping



Displacement mapping

Encode a displacement distance in the texture map

- Measured, e.g., along interpolated normal



Displacement mapping

Update position by displacing points along normal

$$\mathbf{p}_d = \mathbf{p} + h\mathbf{n}$$

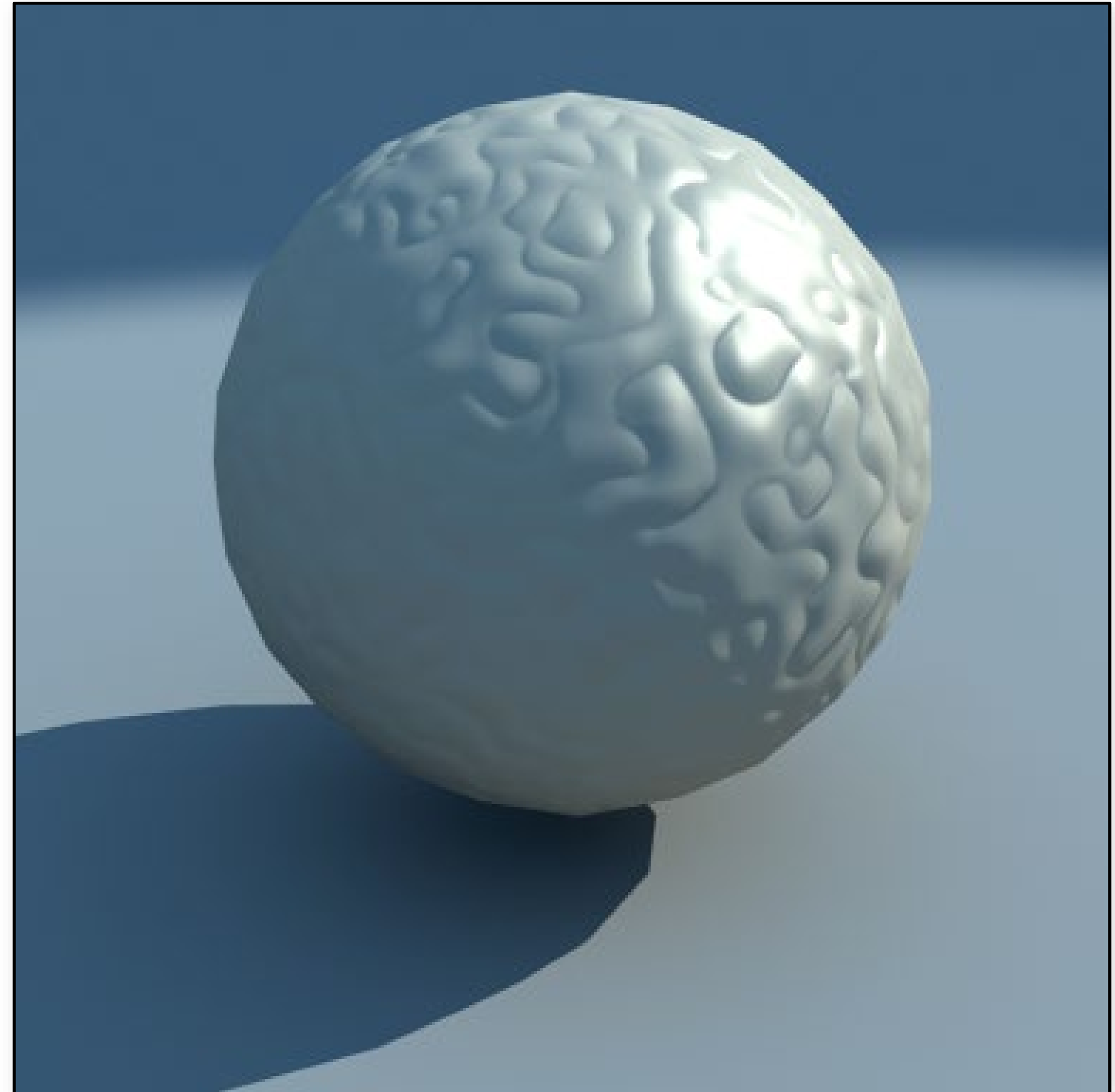
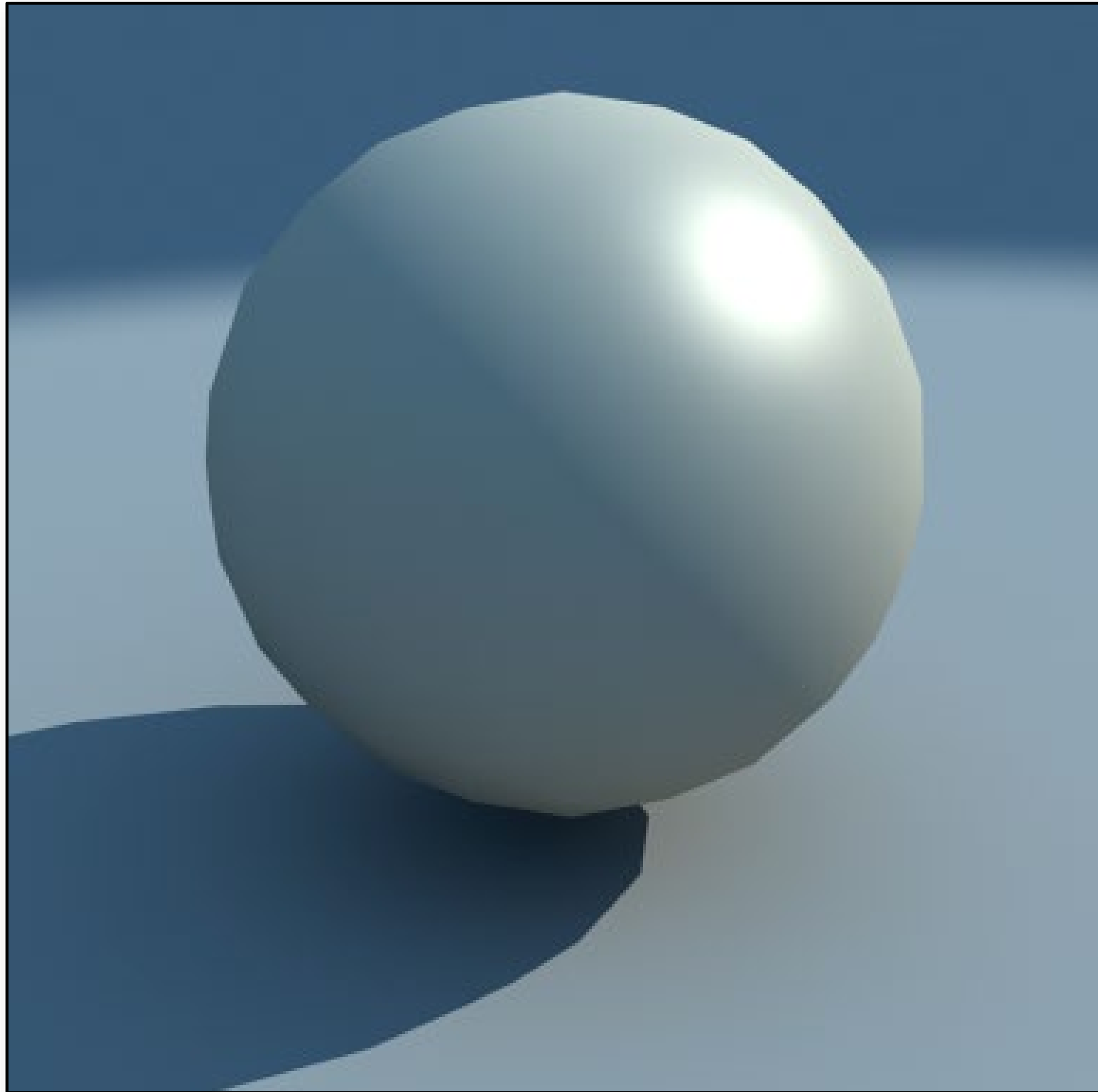
Recompute normals by evaluating derivatives

- often no closed form solution, so use finite differences

$$\mathbf{n}_d \propto \frac{\partial \mathbf{p}_d}{\partial u} \times \frac{\partial \mathbf{p}_d}{\partial v} \approx \frac{\Delta \mathbf{p}_d}{\Delta u} \times \frac{\Delta \mathbf{p}_d}{\Delta v}$$

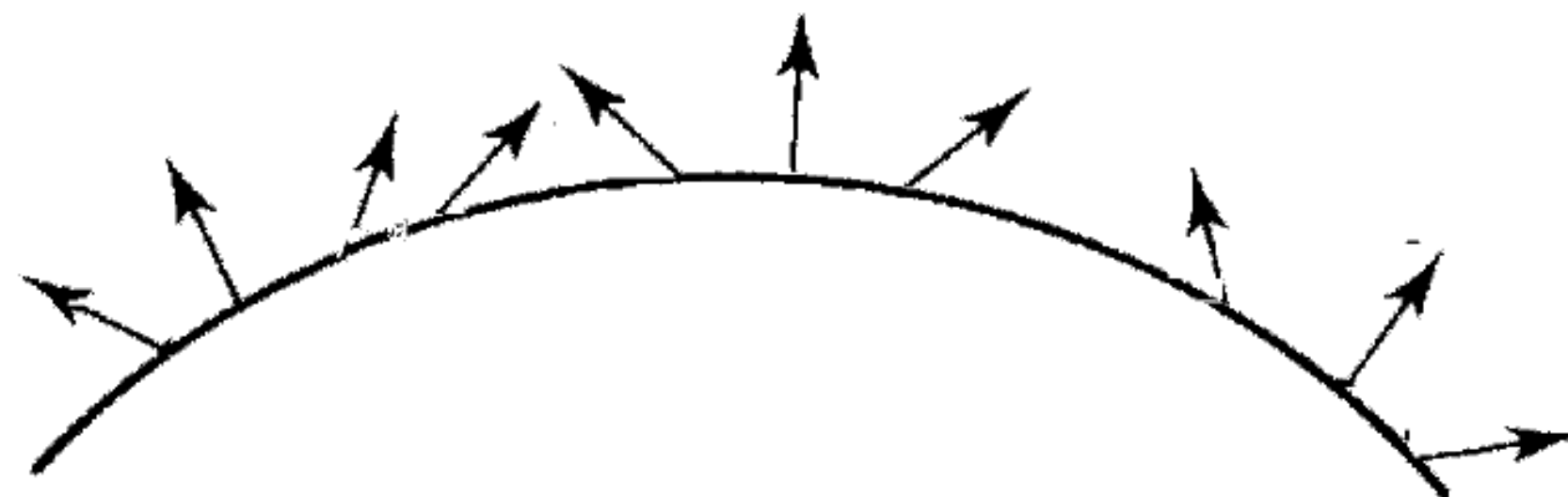
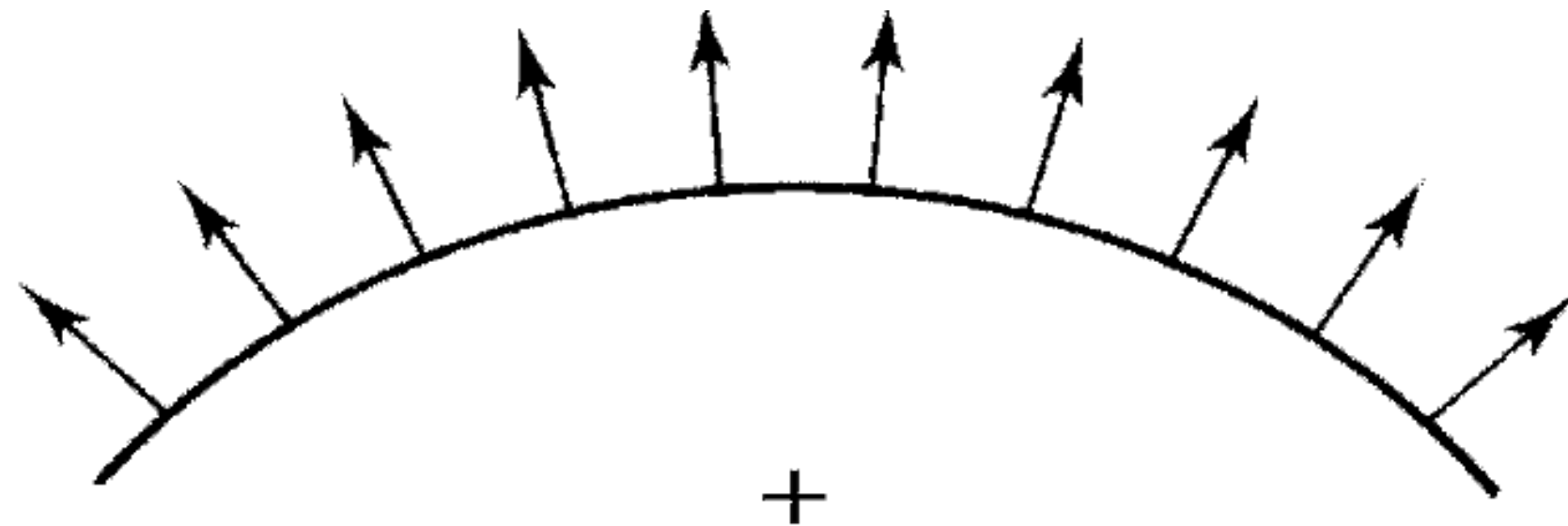
Bump mapping

Apply normal perturbation without updating positions

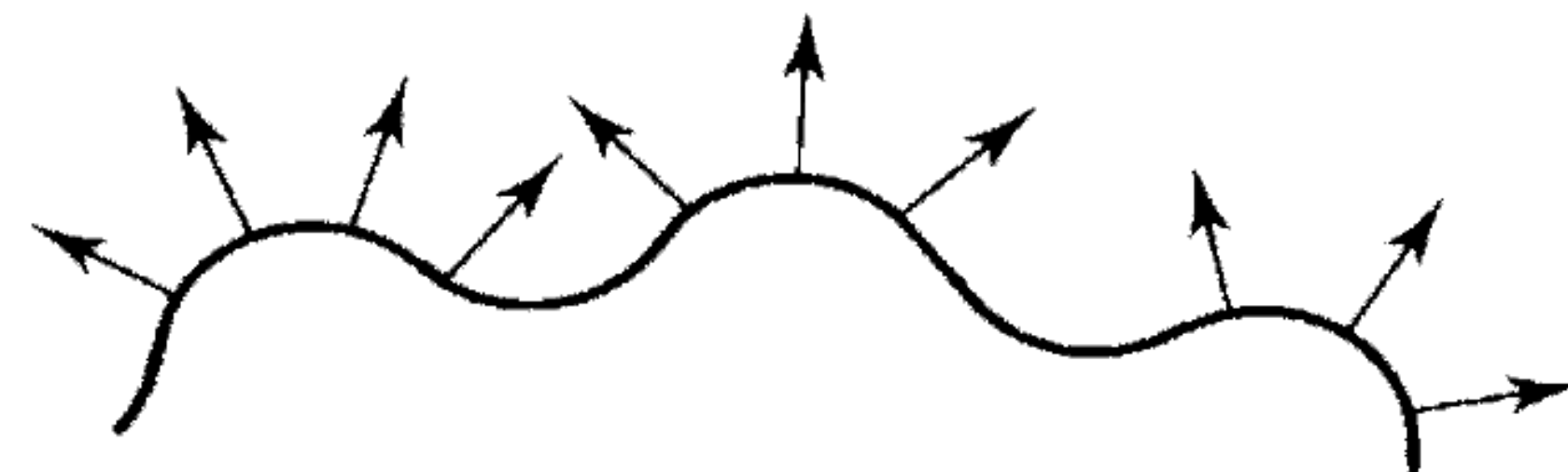
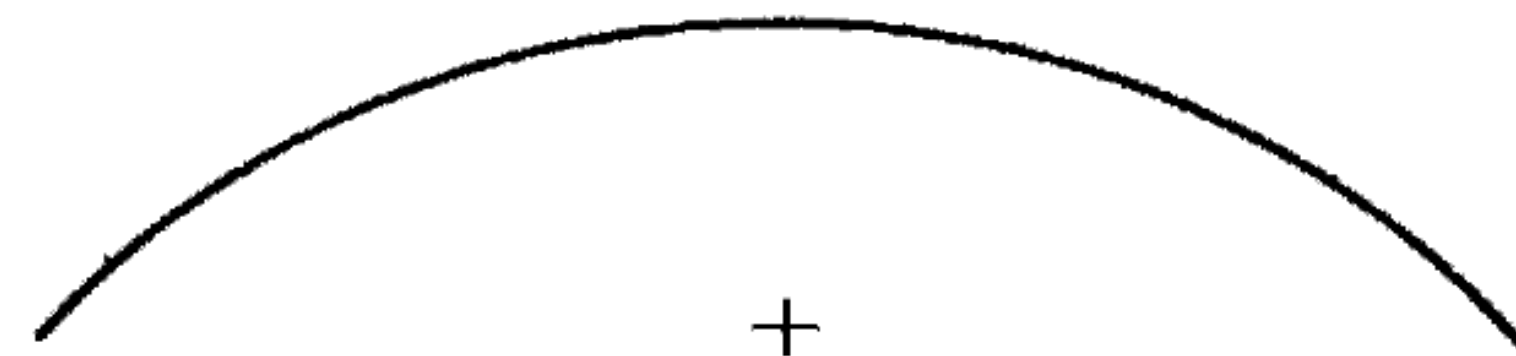


Bump vs. displacement mapping

bump map

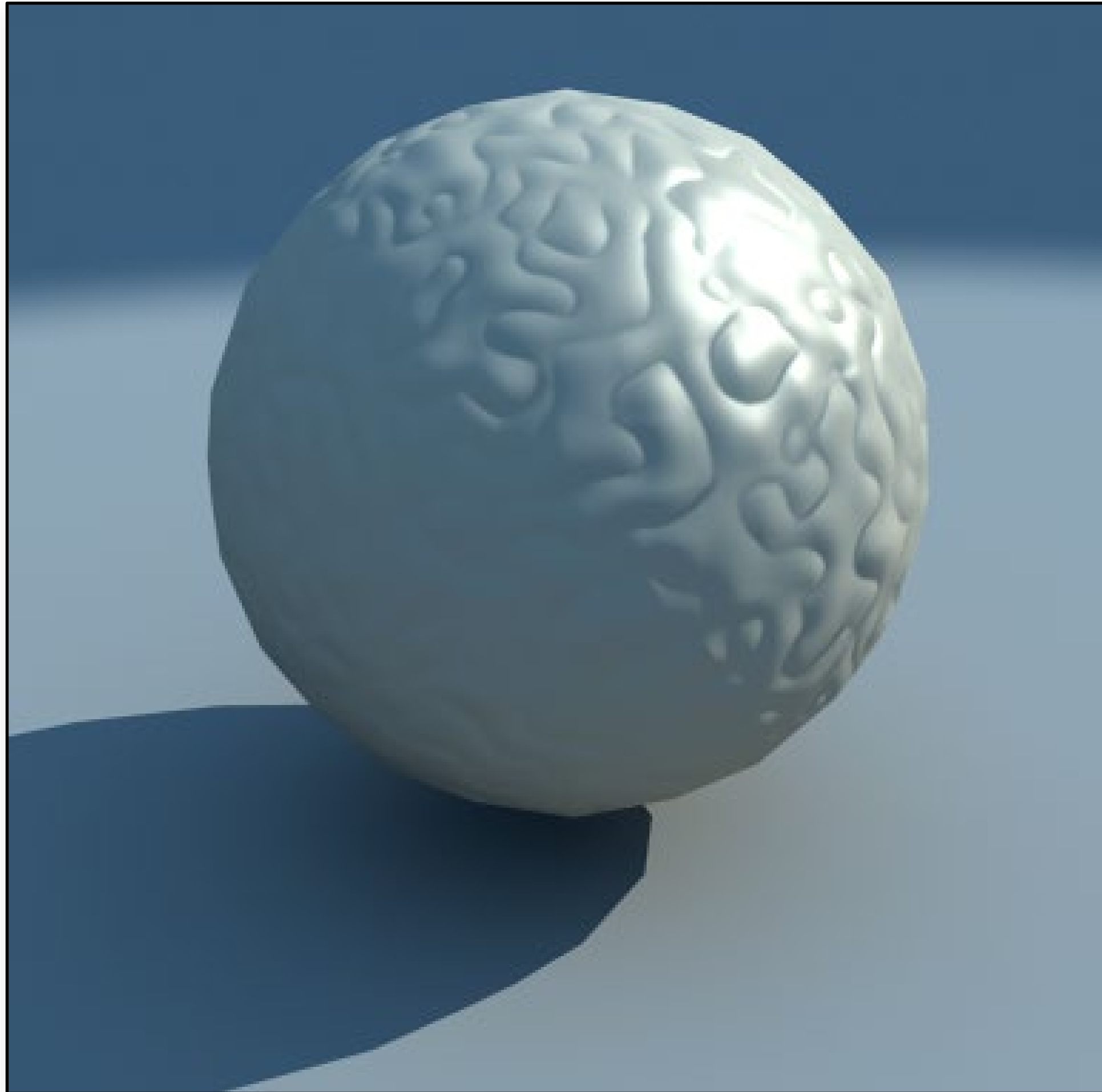


displacement map

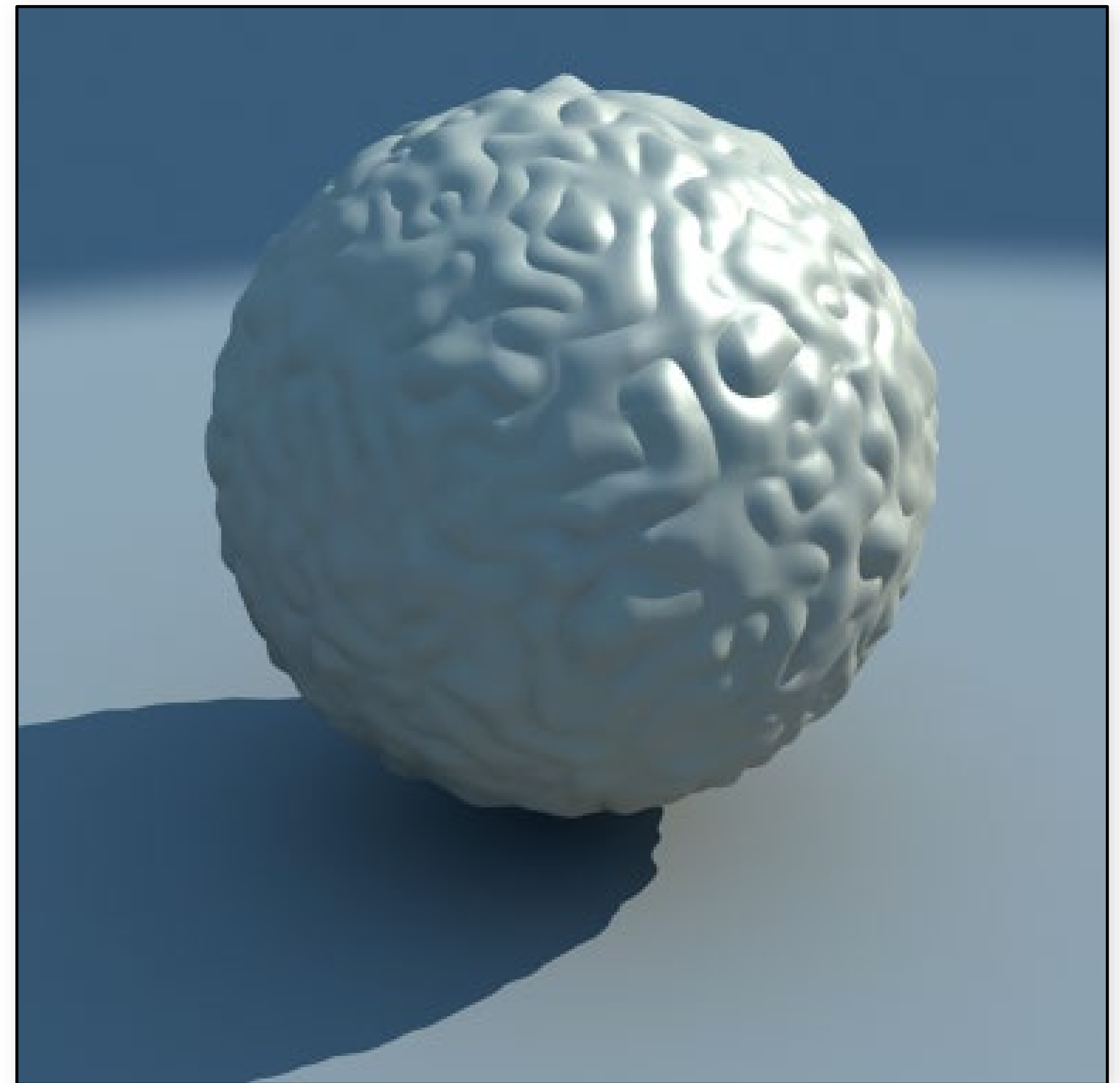


Bump vs. displacement mapping

bump map



displacement map



Displacement vs. bump mapping



Max Displace 1.5Mil

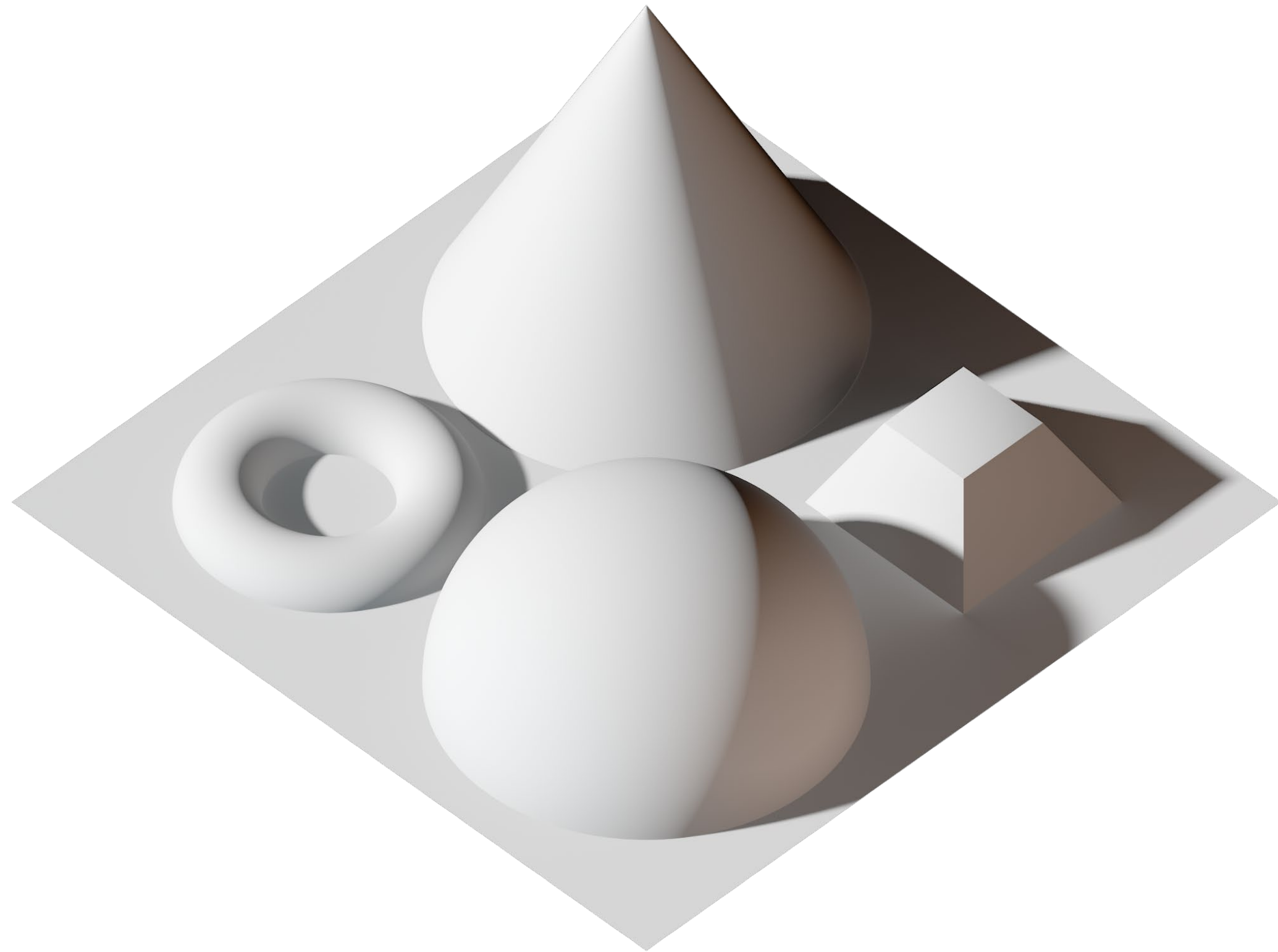


Normal Map 2900Tris

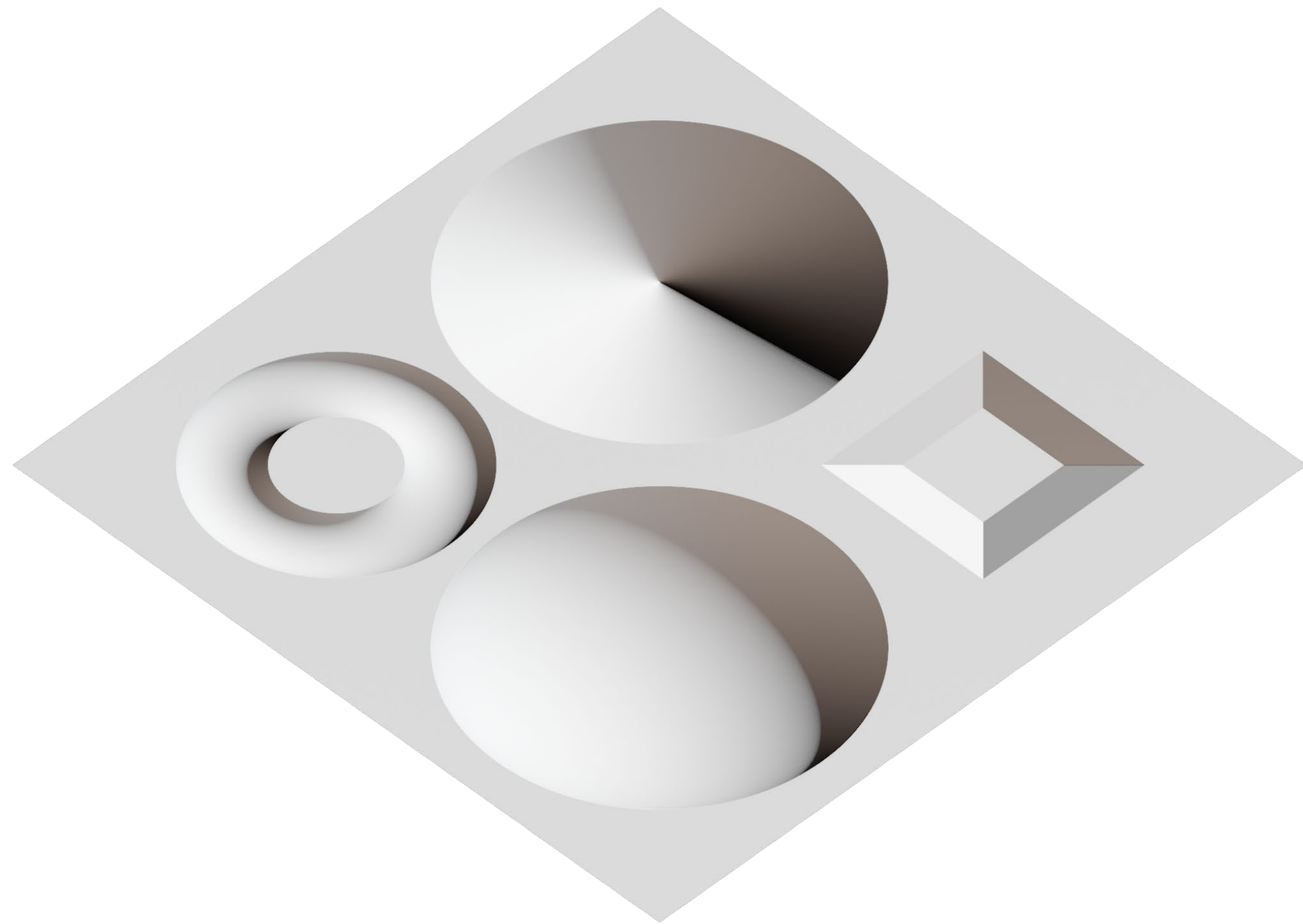


Wire

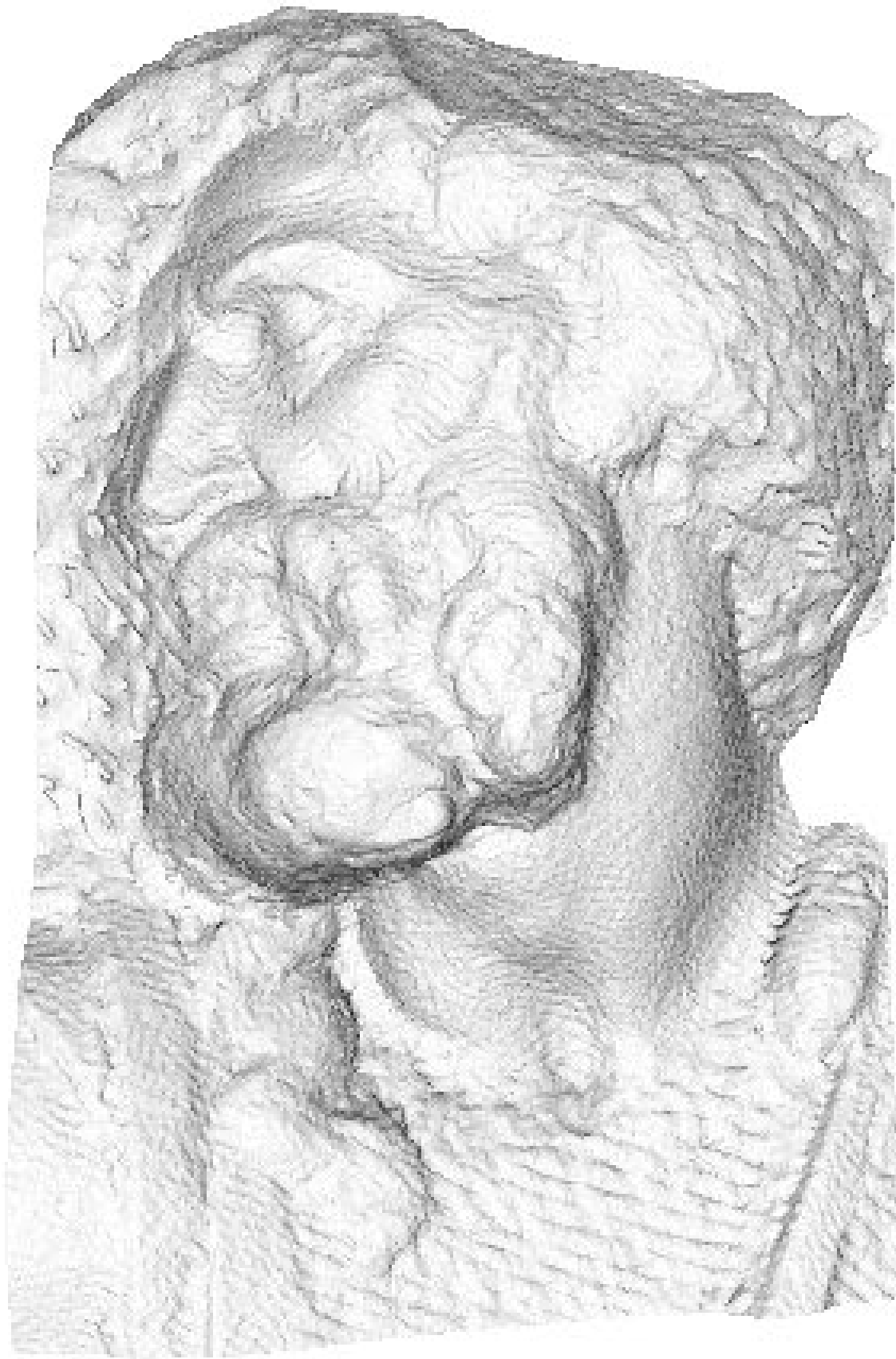
Surface normals



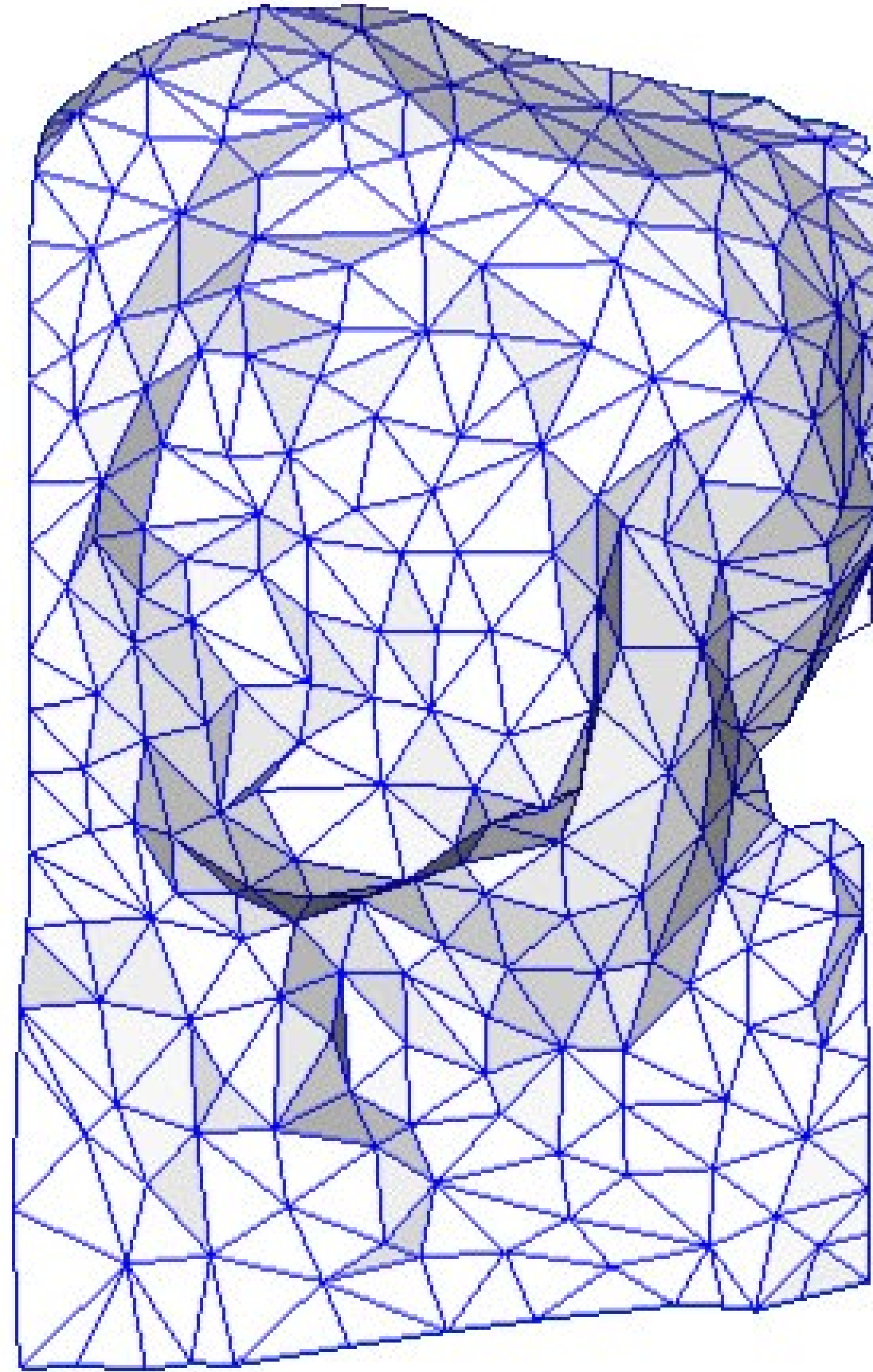
Normal mapping



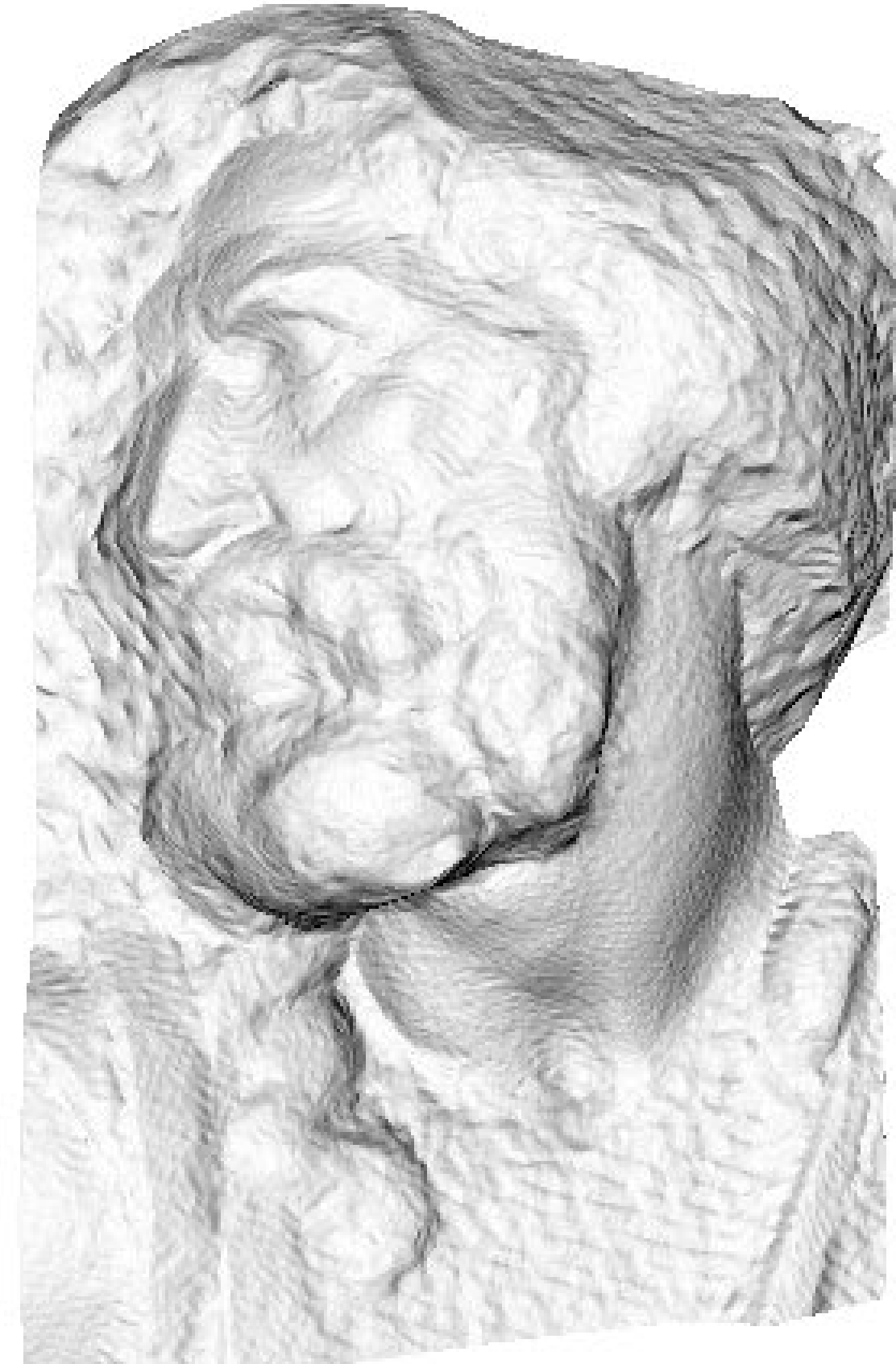
Normal mapping



original mesh
4M triangles



simplified mesh
500 triangles



simplified mesh
and normal mapping
500 triangles

Environment & Reflection mapping

Shiny objects

The key to creating a realistic shiny-looking material is providing something for it to reflect.



Environment/reflection mapping

Sidesteps tedious modeling of the environment by representing it using one or more images

The image “wraps” around the virtual scene, serving as a source for reflections

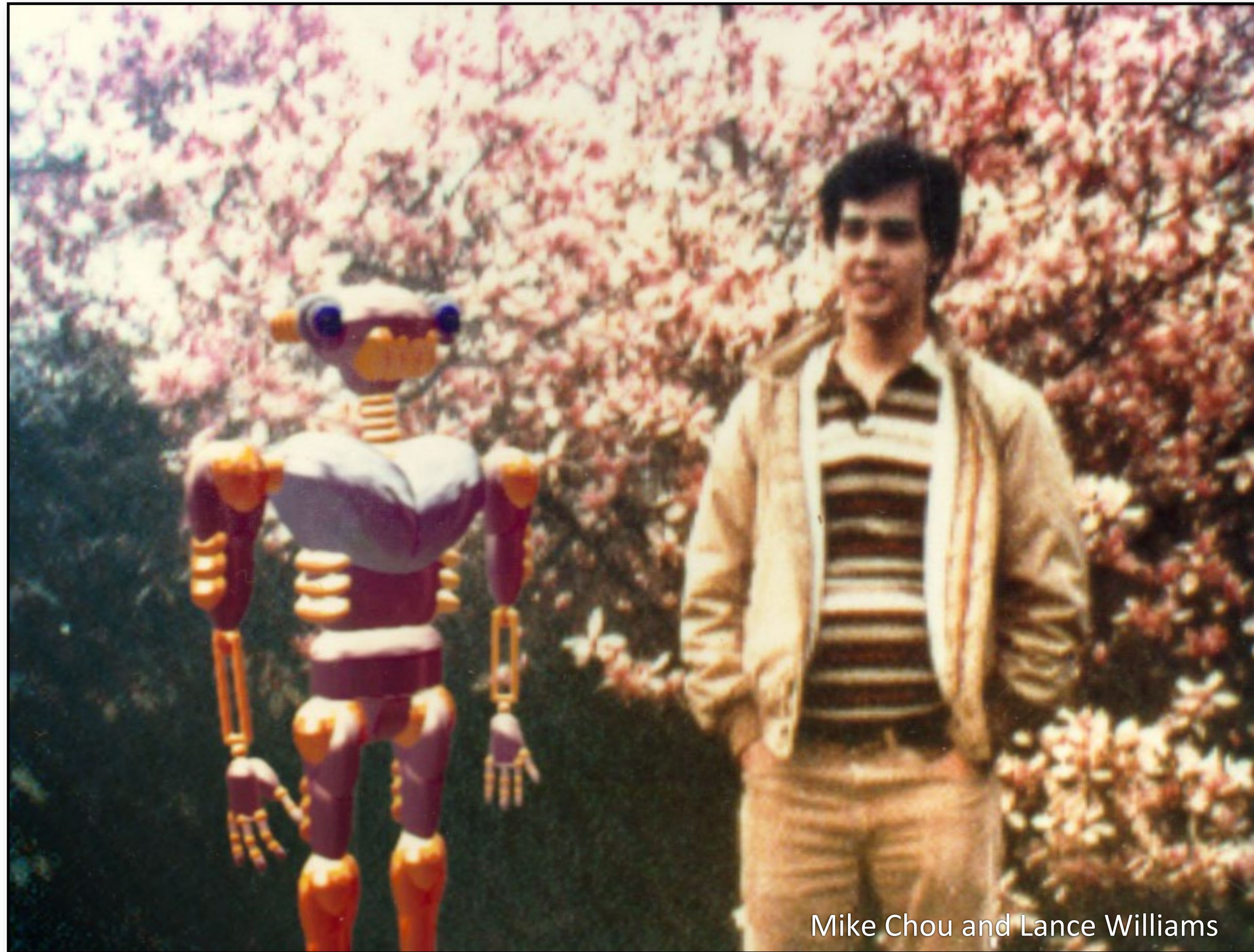


Environment map

A function from the sphere to colors, stored as a texture.

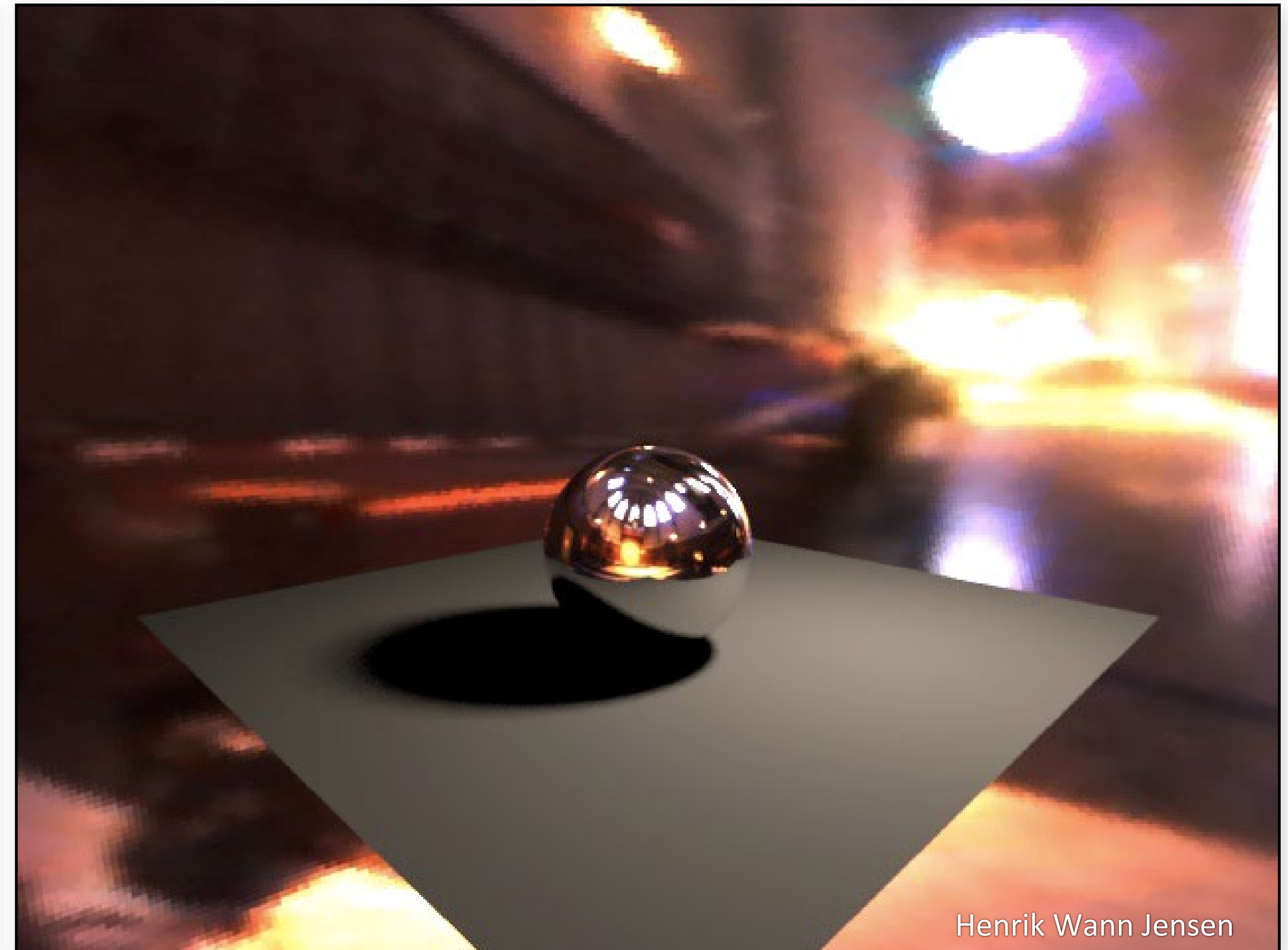


Reflection mapping (1982)



Environment mapping

Ray tracing easy: rays that hit nothing look up in envmap



Acquiring environment maps

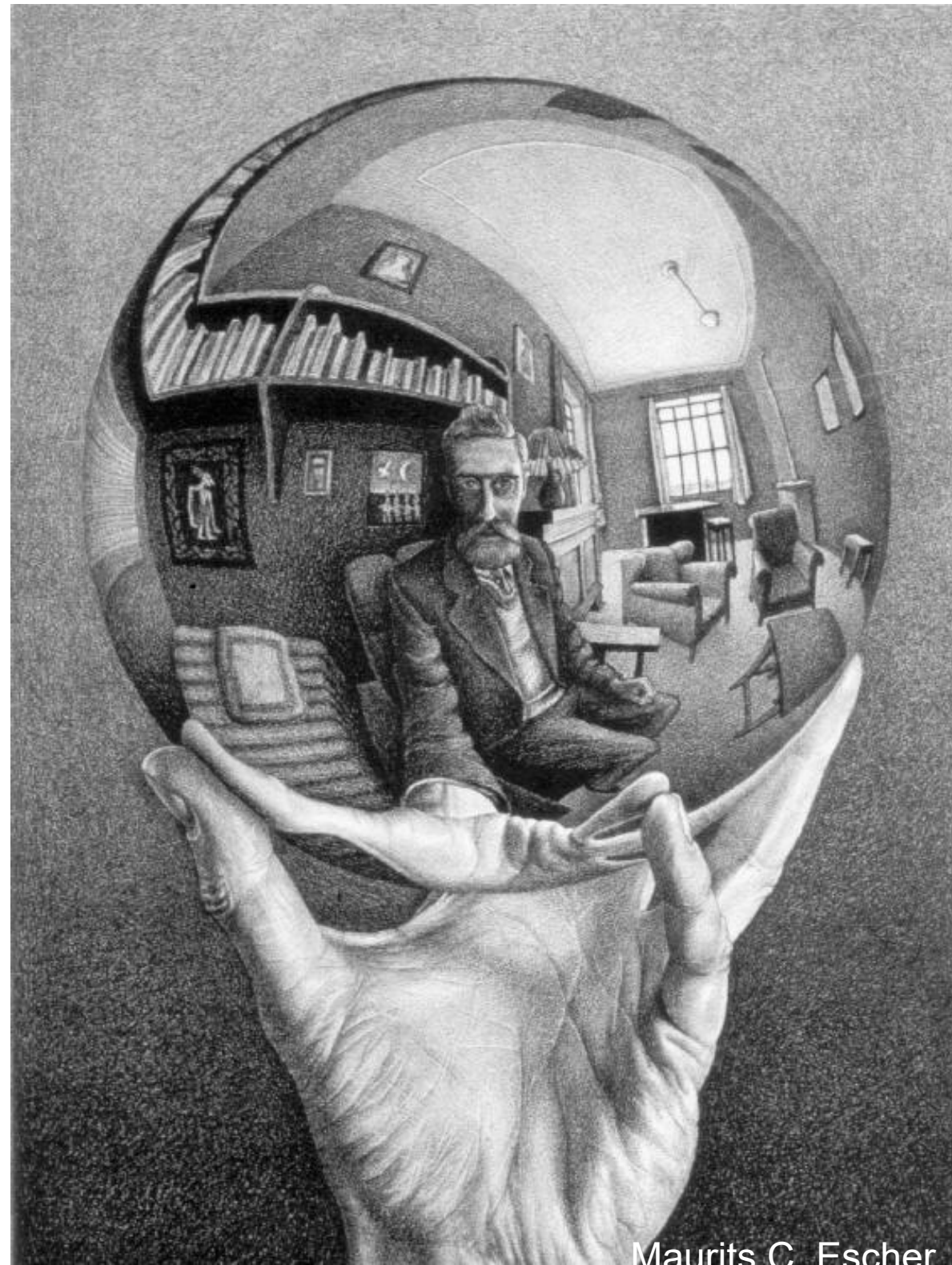
Mirrored ball + camera

Fisheye lens images

Stitching images together

Panoramic camera

Acquisition - Low Tech.



Lightprobe



omnidirectional,
360° panoramic,
HDR image

Acquisition - Even Lower Tech.



Arash Keshmirian & Wojciech Jarosz 2006

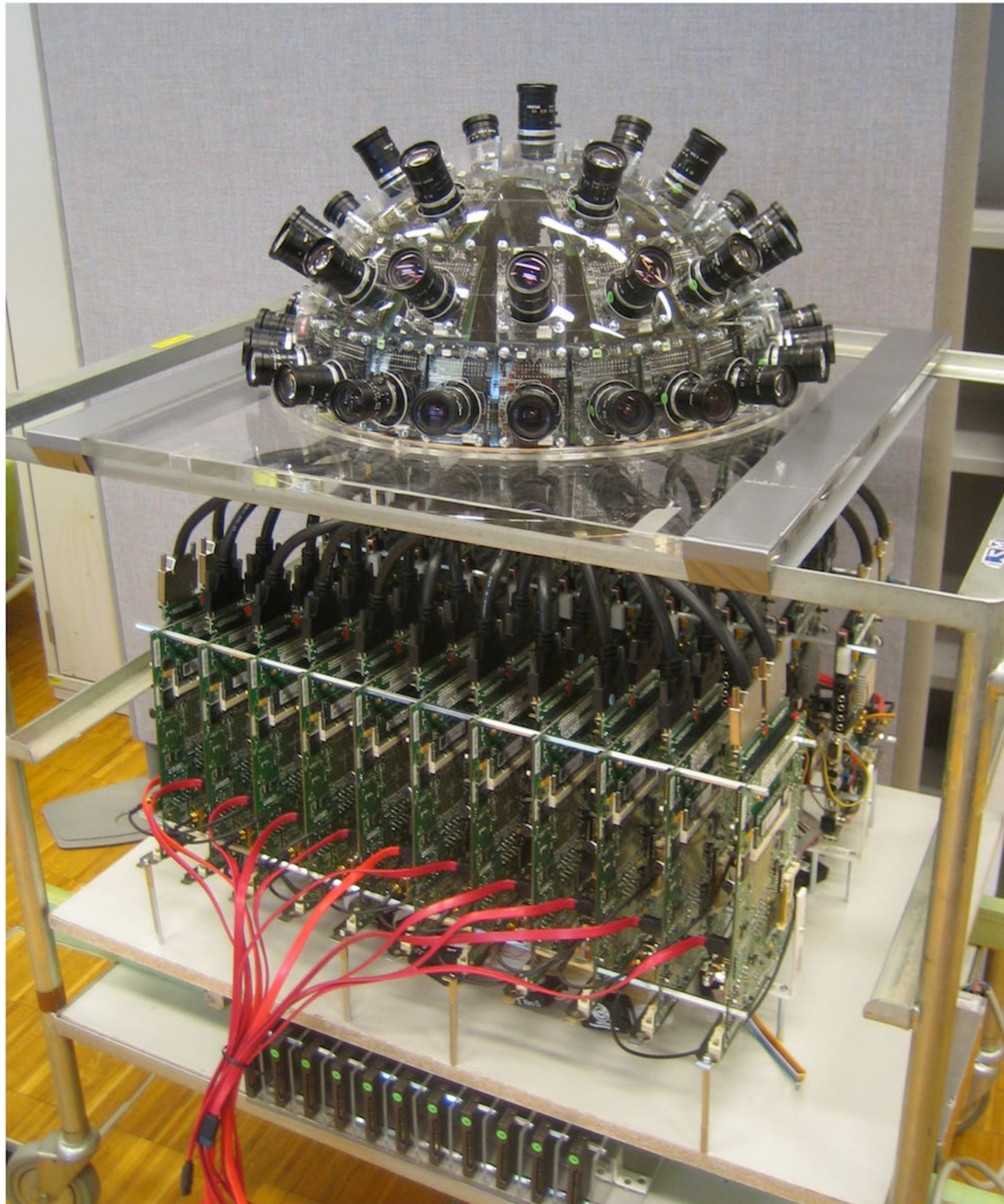


Wojciech Jarosz & Arash Keshmirian 2006

Acquisition - Stitched Panorama



Acquisition - High Tech.



lsm.epfl.ch



immersivemedia.com

Acquisition



Storing environment maps

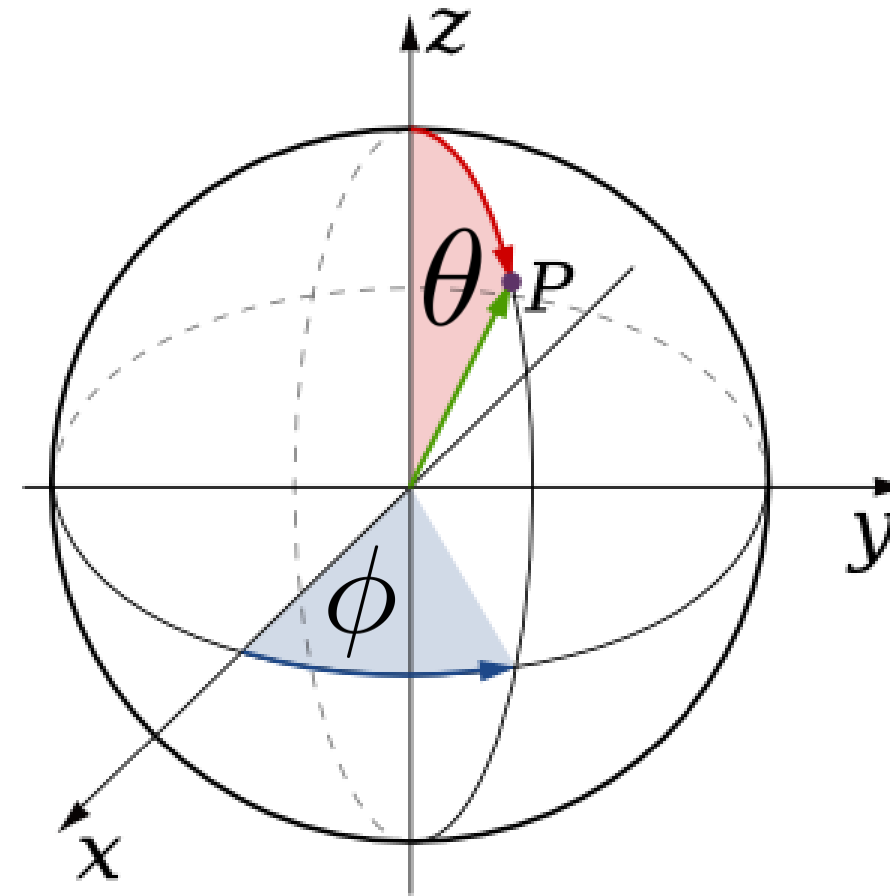
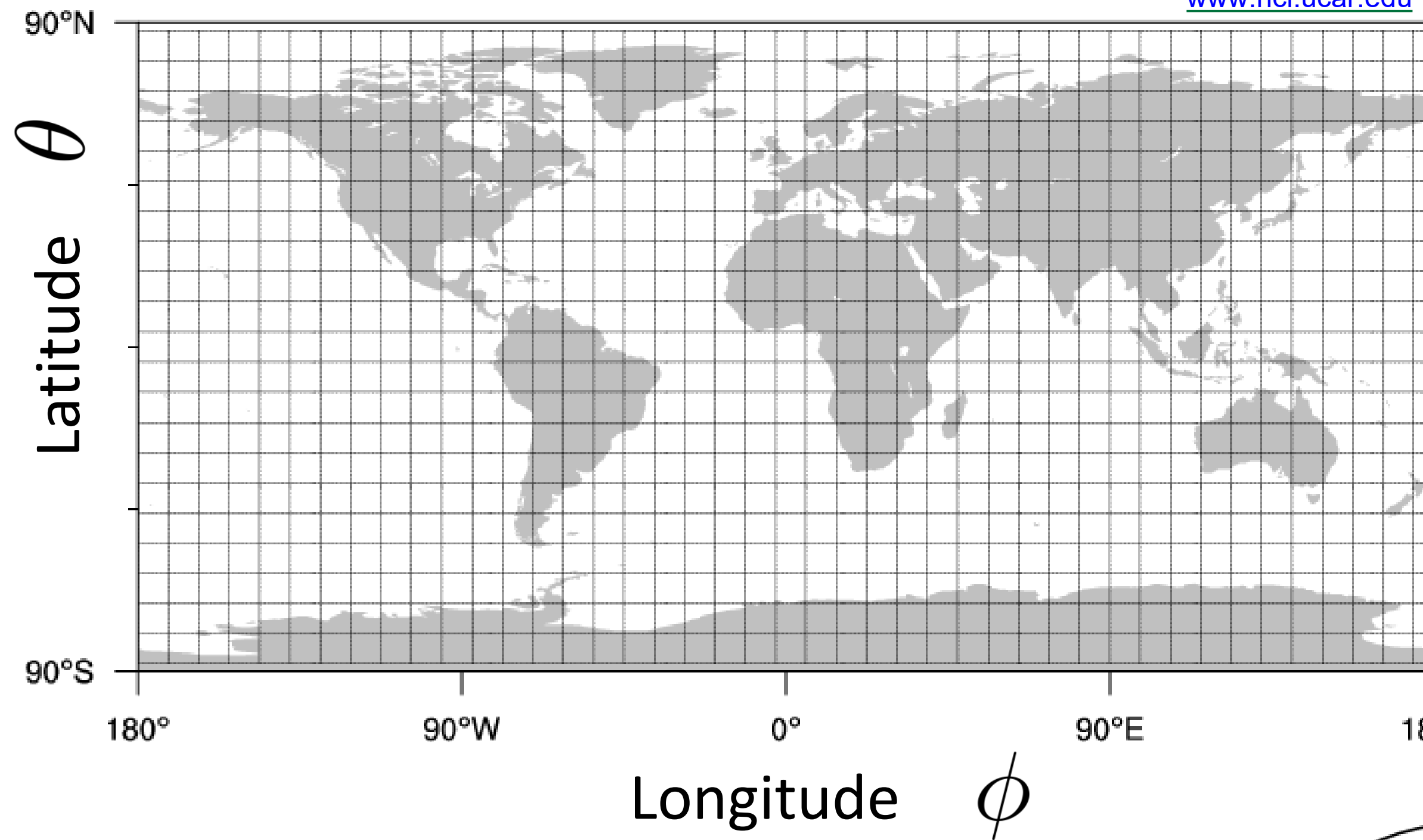
Various ways to parametrize environment maps

Related to cartography

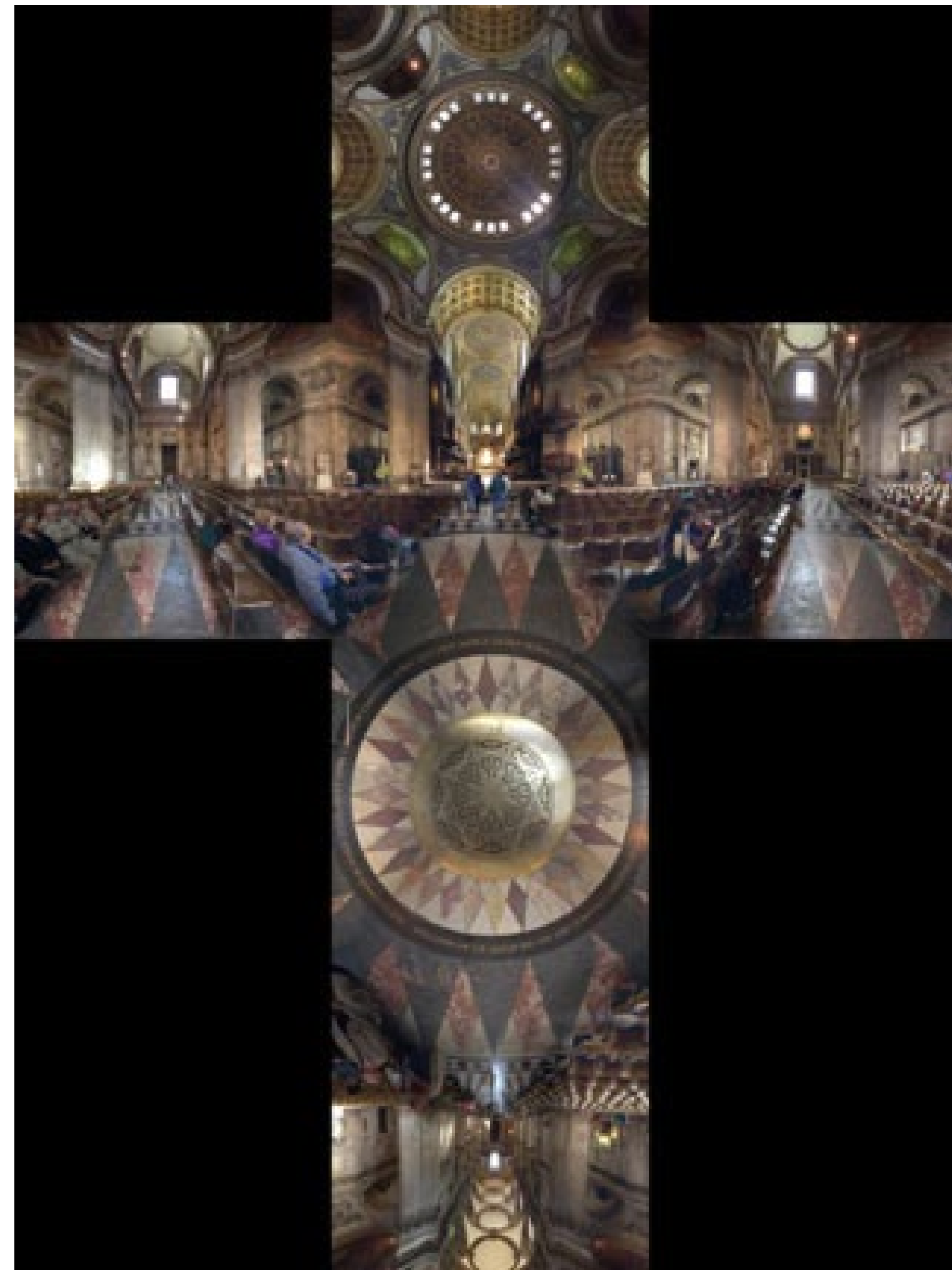
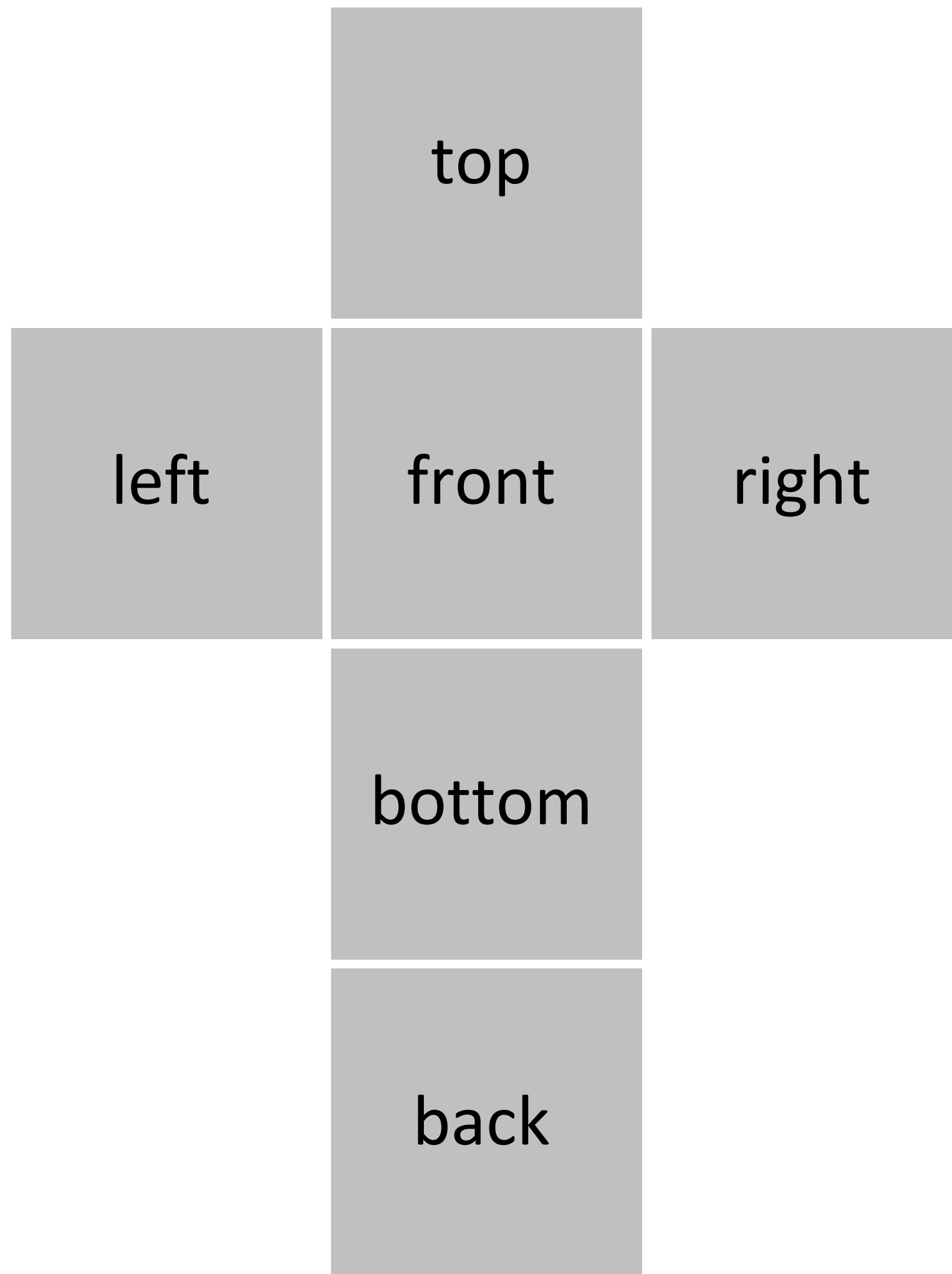
- Projecting the earth (sphere) onto a plane

Latitude/Longitude Map

www.ncl.ucar.edu

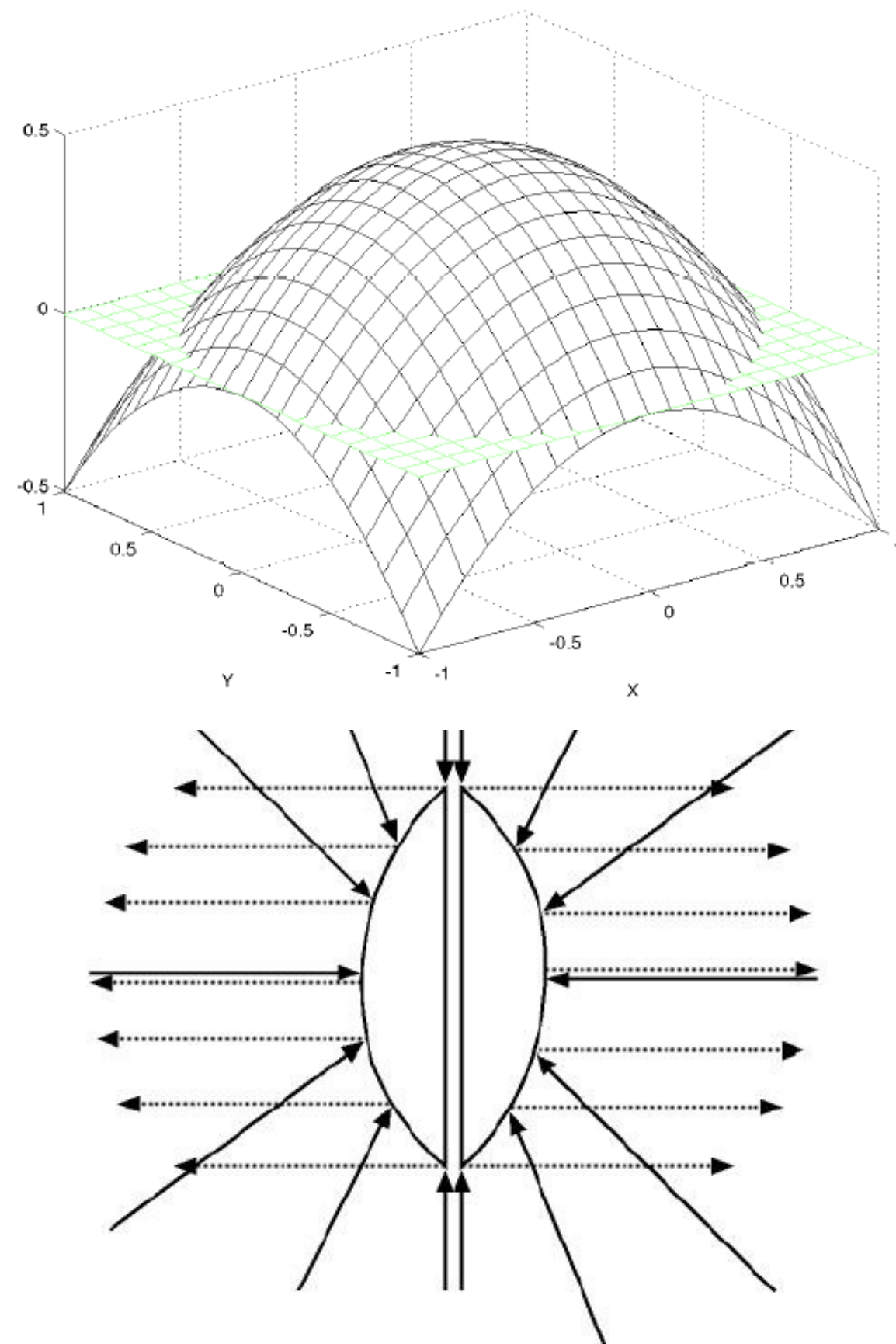


Cube Map (Skybox)



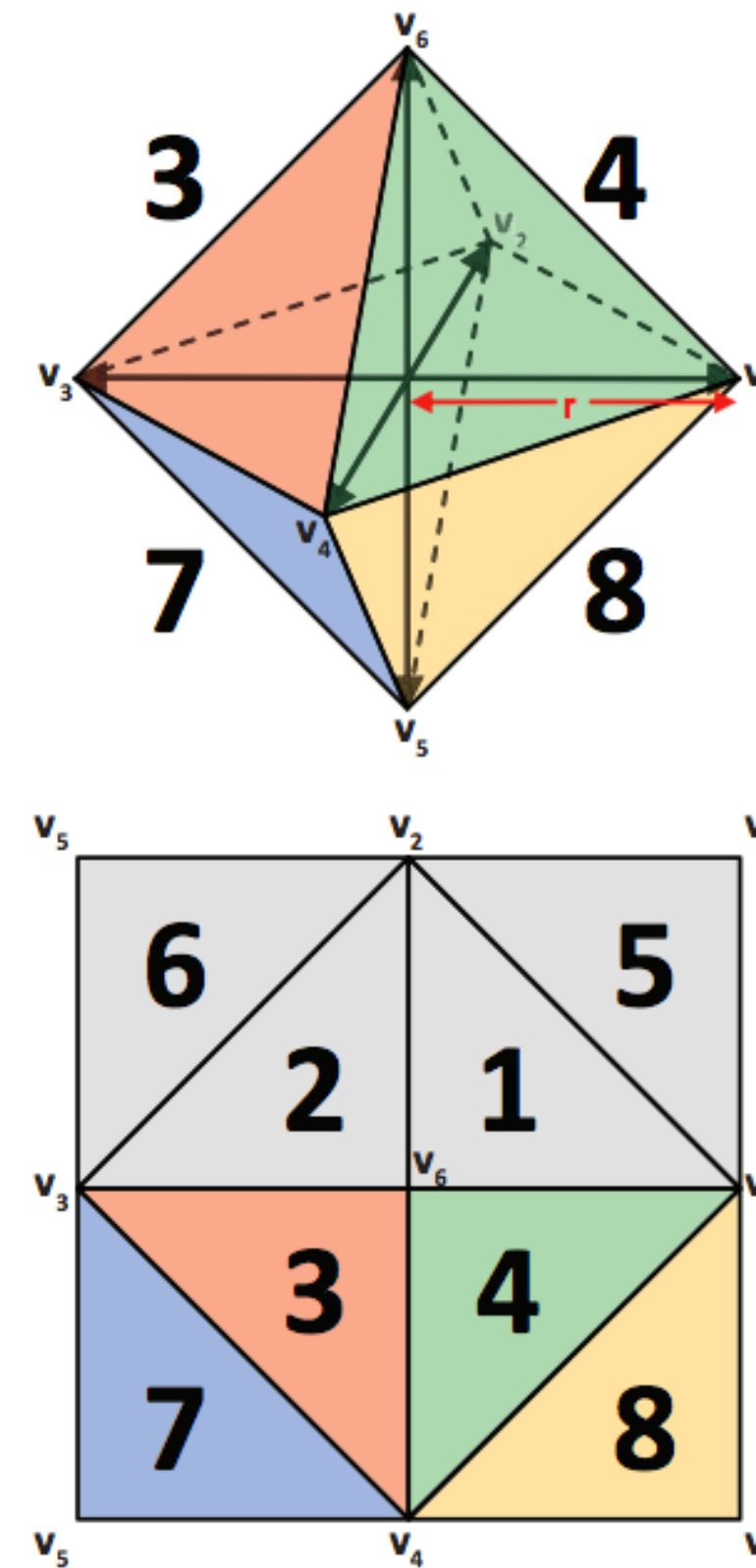
Other Parameterizations

Dual-paraboloid



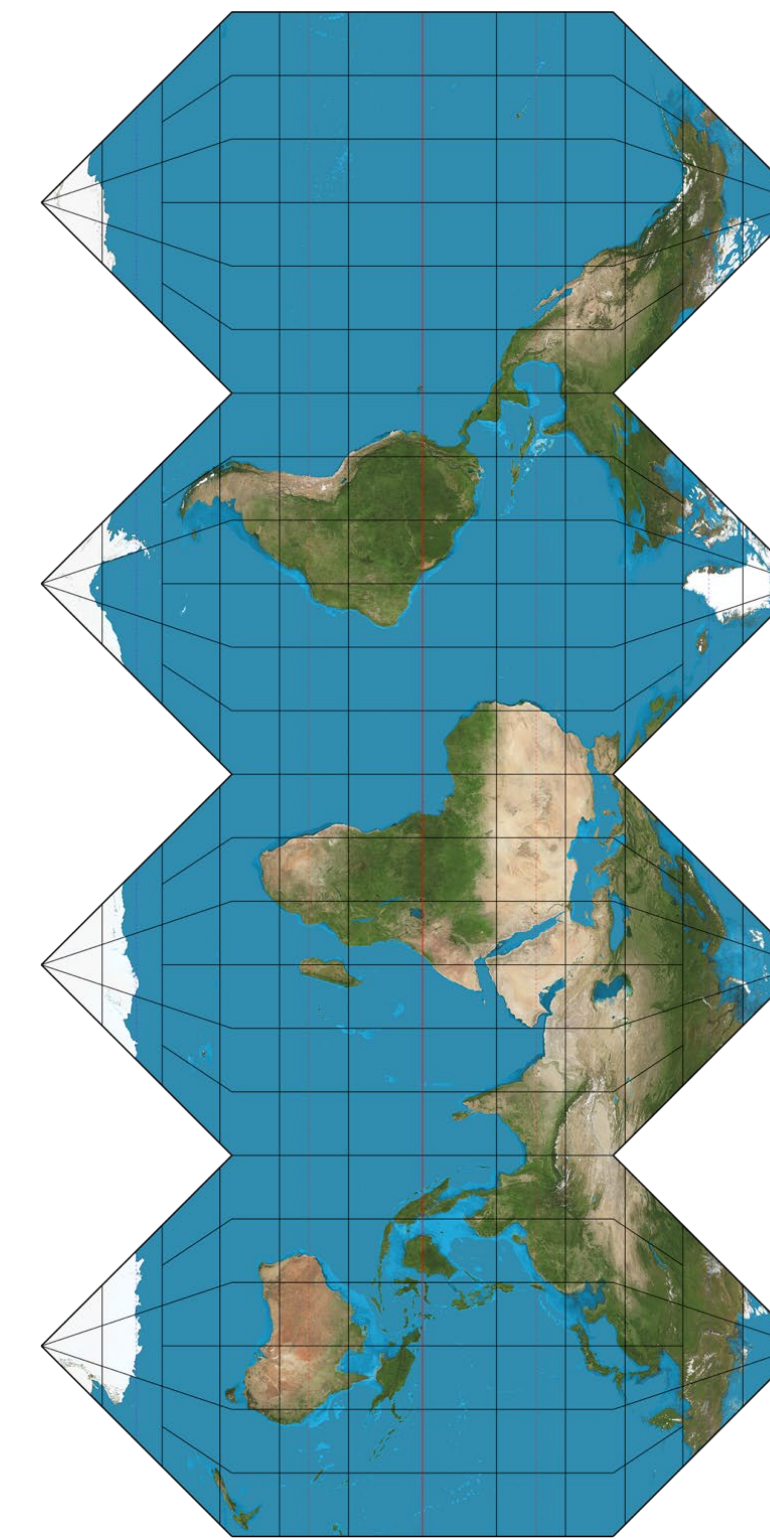
Courtesy of Brabec

Octahedron map



Courtesy of Engelhardt and Dachsbacher

HEALPix



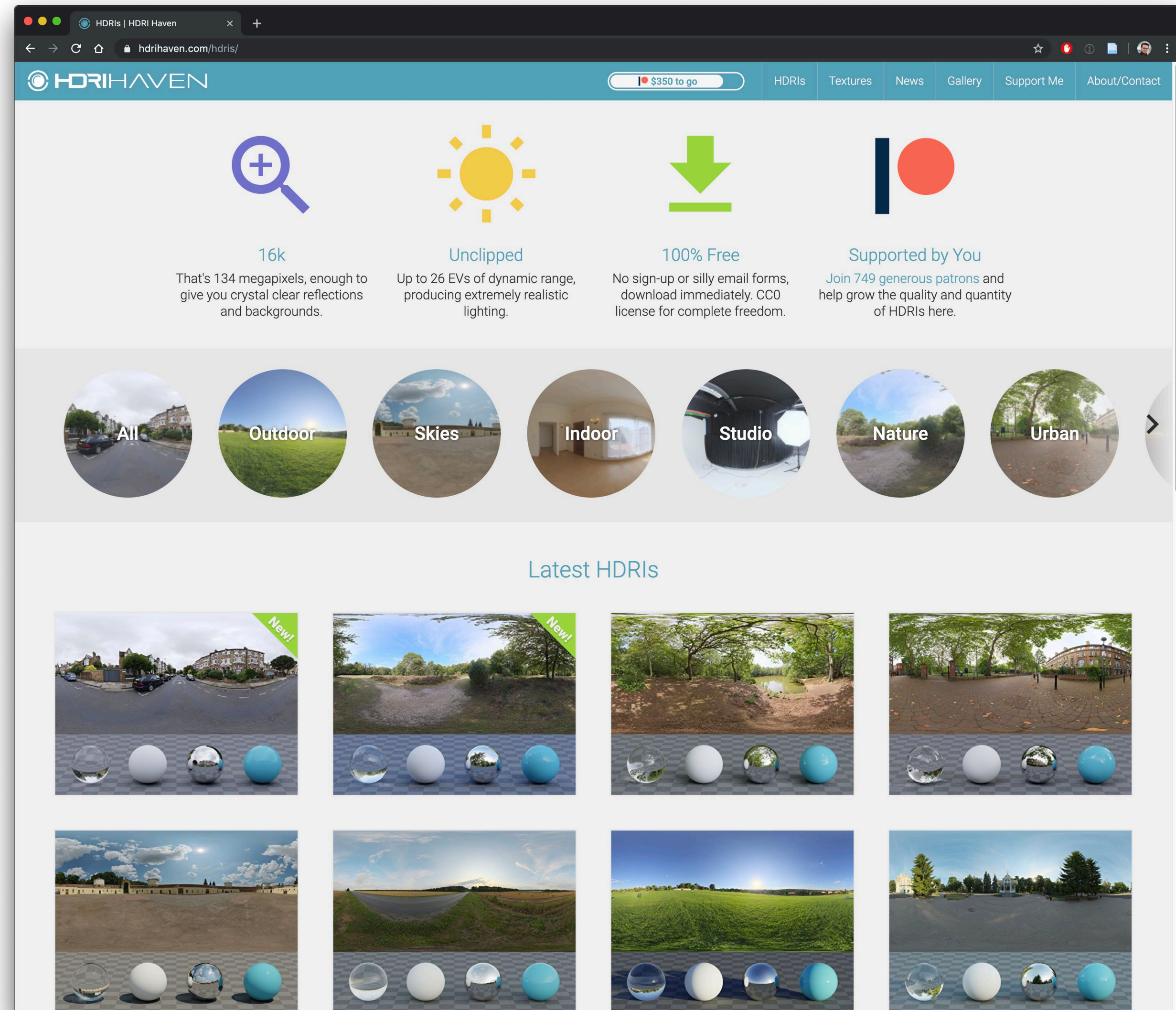
Courtesy of Ryazanov

Get them online

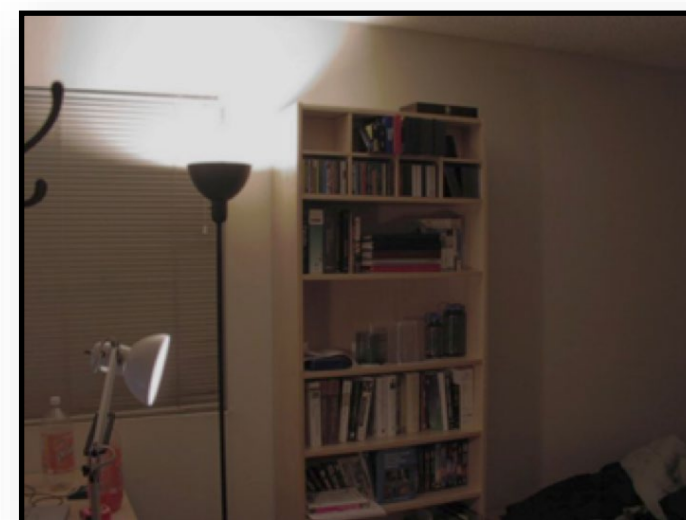
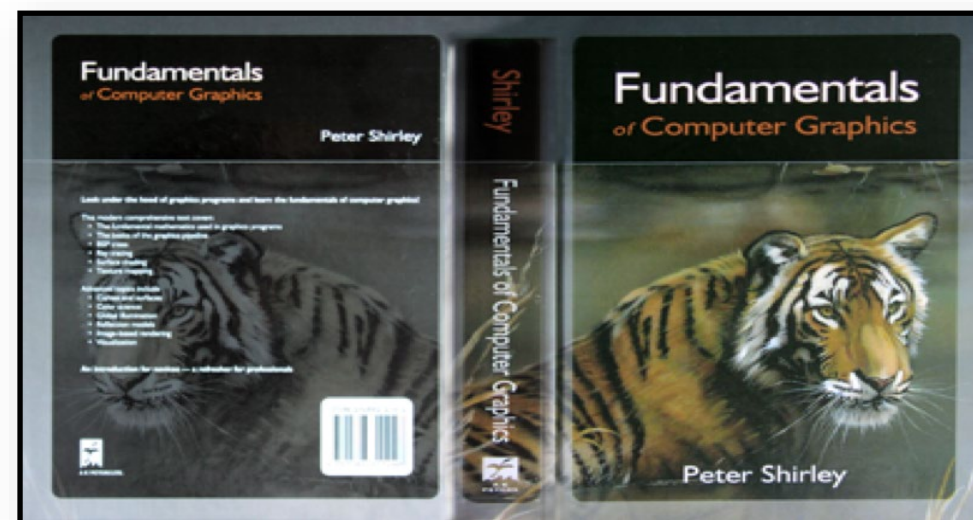
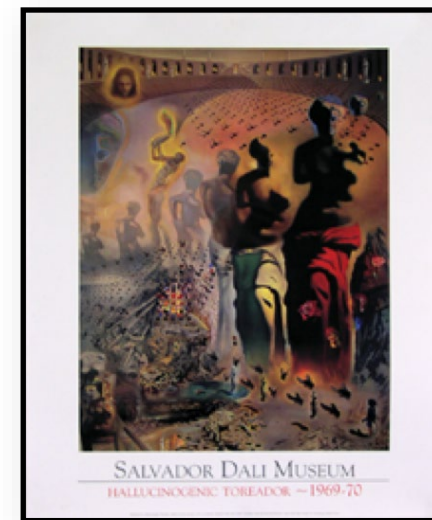
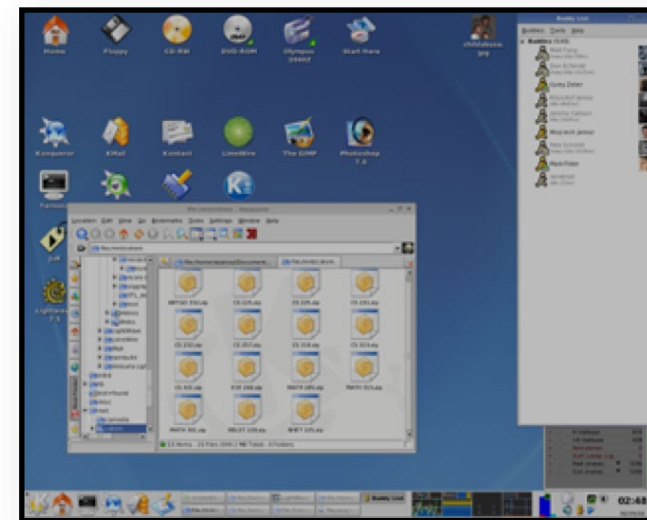
hdrihaven.com

The original:

- www.pauldebevec.com/Probes
- gl.ict.usc.edu/Data/HighResProbes



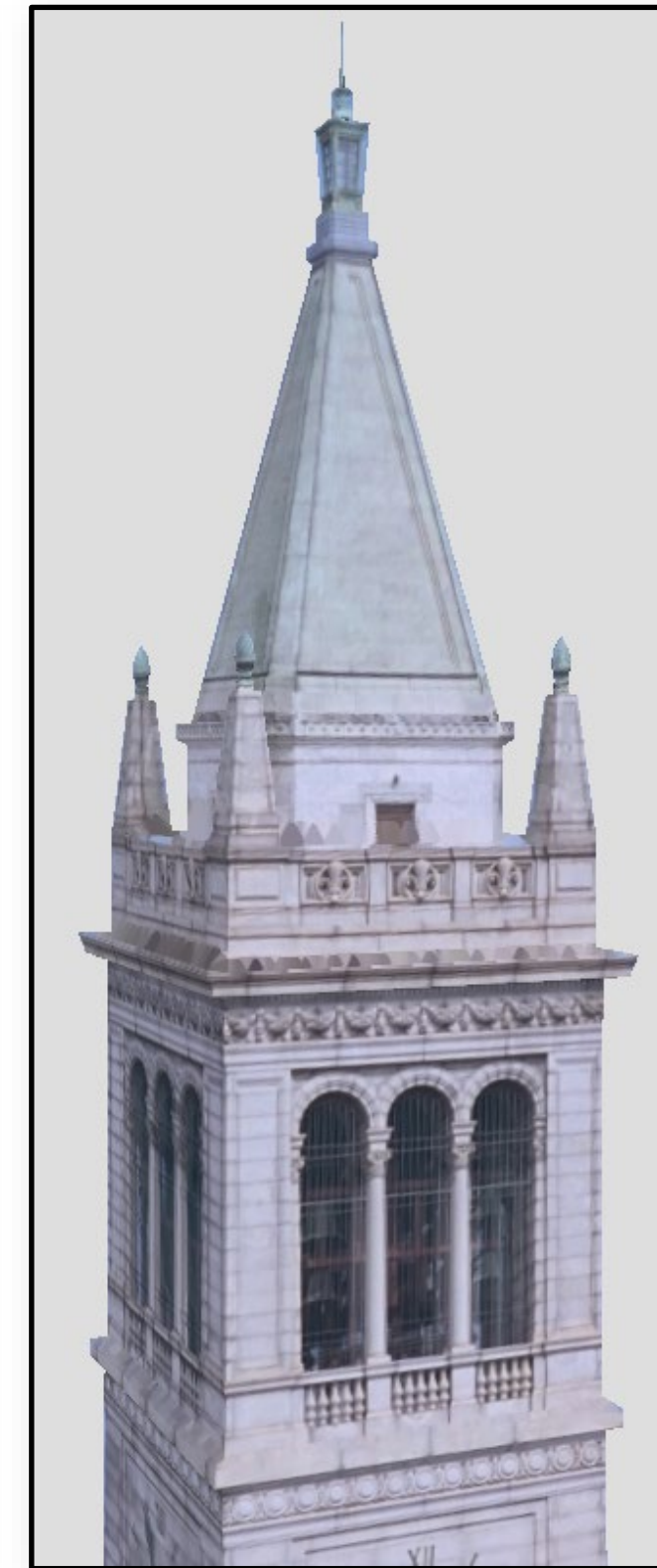
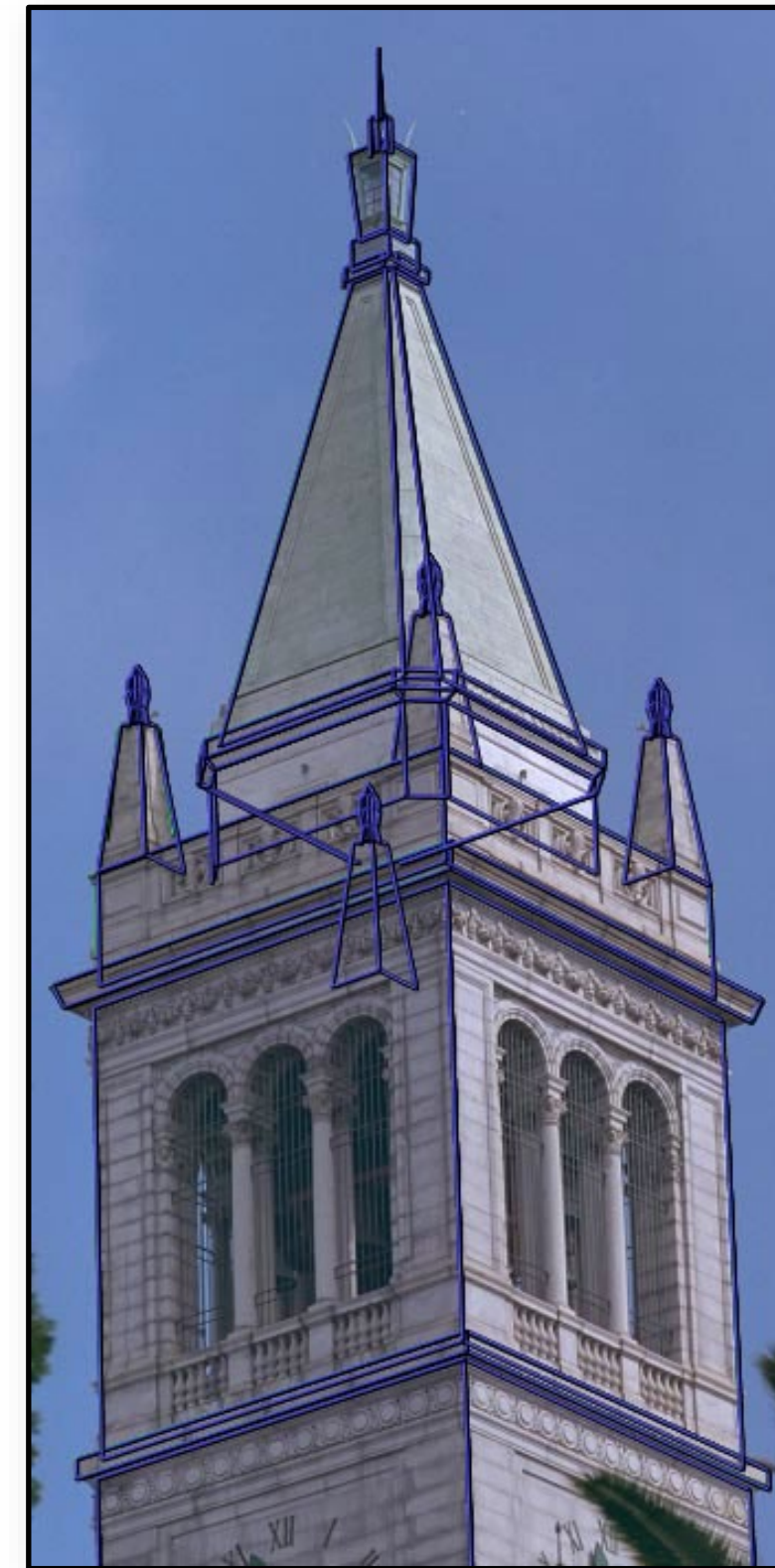
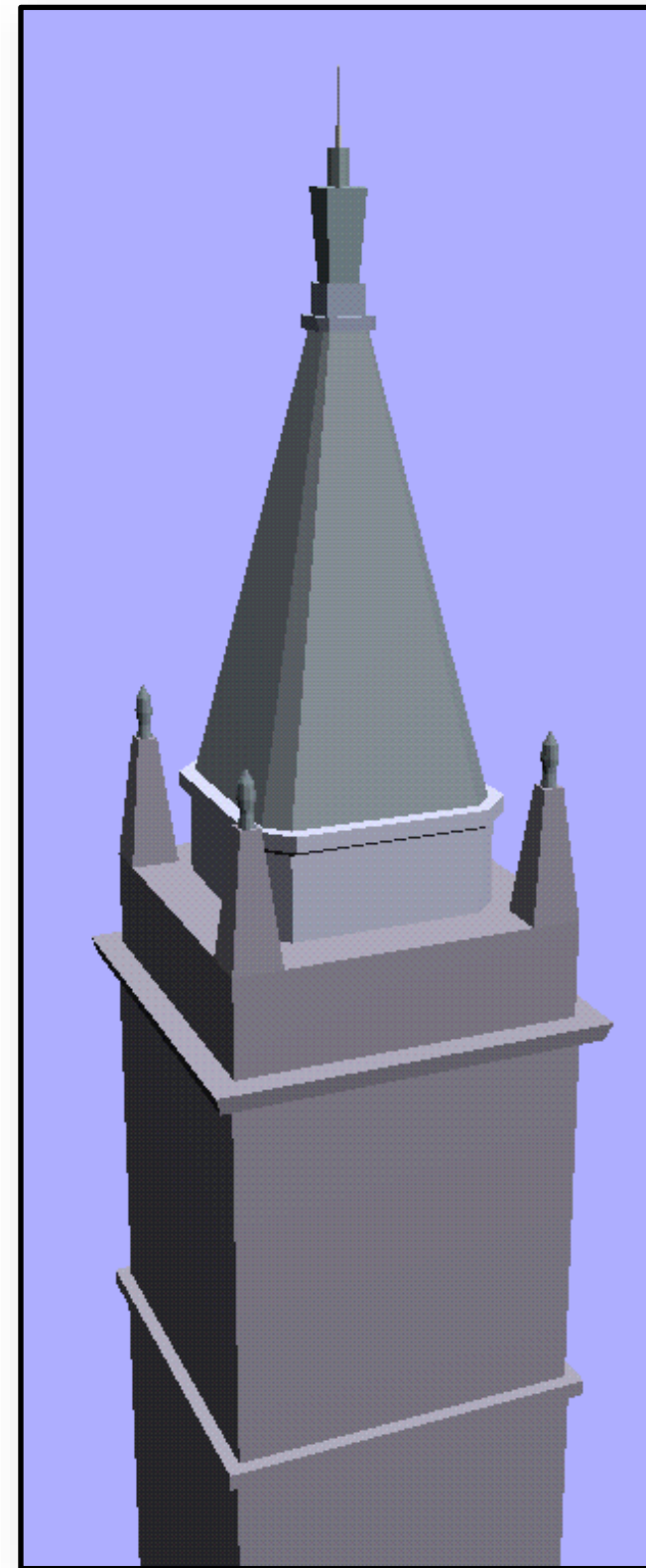
Exploit texturing!



Projective texture example

Modeling from photographs

Using input photos as textures





More details

“Rendering with Natural Light”

- <http://www.pauldebevec.com/RNL/>

“Fiat Lux”

- <http://www.pauldebevec.com/FiatLux/>

History of reflection mapping

- <http://www.pauldebevec.com/ReflectionMapping/>