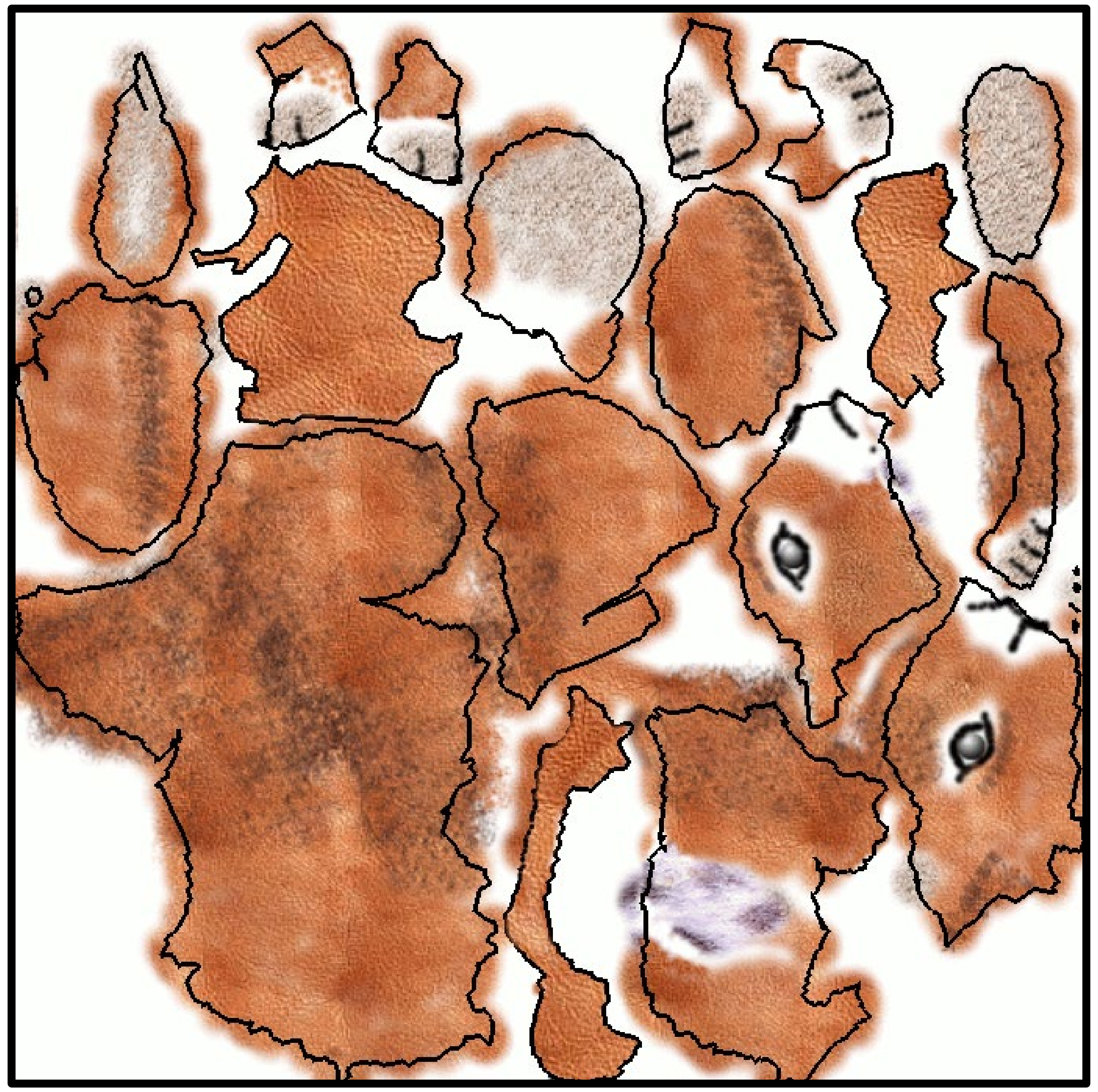
Texture mapping



15-468, 15-668, 15-868
Physics-based Rendering
Spring 2022, Lecture 4

http://graphics.cs.cmu.edu/courses/15-468

Course announcements

- Programming assignment 0 available on Canvas.
  - Ungraded, no due date.
  - Used to set up rendering environment and github-based submission system.
  - Should take no more than 1-2 hours max.

- Programming assignment 1 will be posted on Friday 1/28, will be due two weeks later.

- Take-home quiz 1 posted, due Tuesday 2/1.

- Office hours *for the rest of the semester*:
  - Po, Mondays 4 – 6 pm ET.
  - Yannis, Wednesdays 5 – 7 pm ET.
  - Bailey, Thursdays 5 – 7 pm ET.
  - For this week only, office hours are on Zoom. See Canvas for links.
  - Starting next week, office hours will be in-person at the Smith Hall graphics lounge.
  - Yannis may schedule additional office hours on Fridays.

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

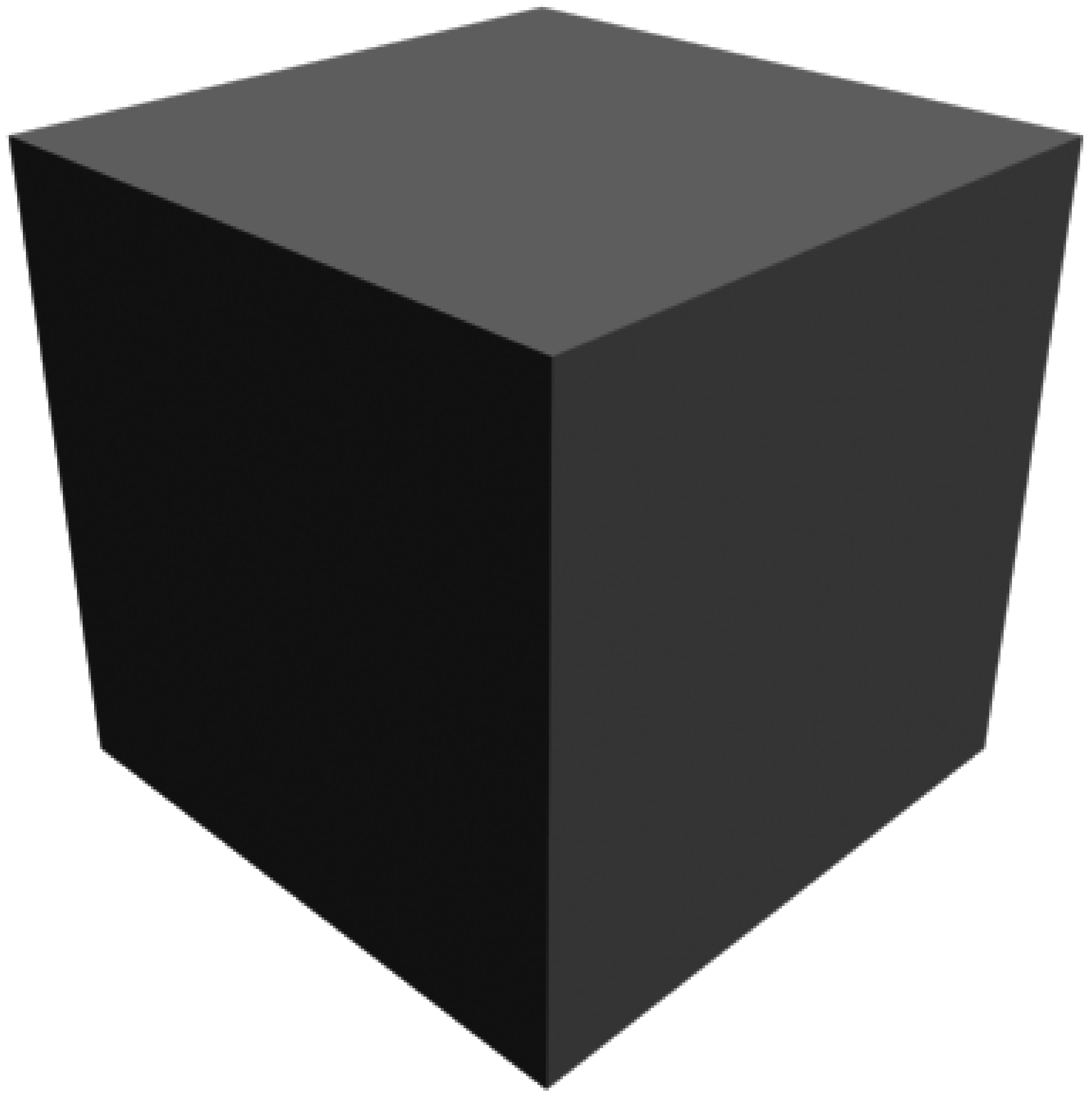
Why texture mapping?

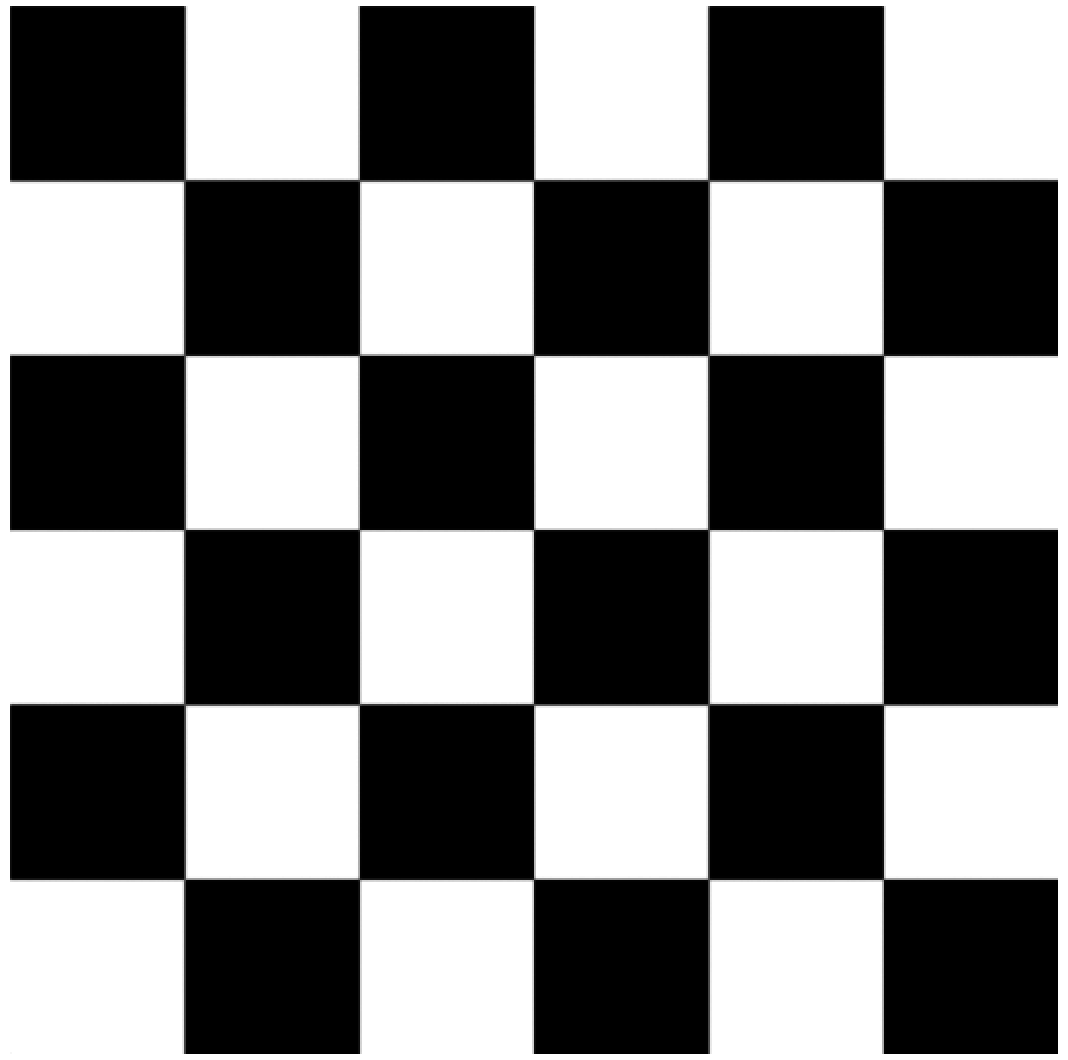
Use simple geometry

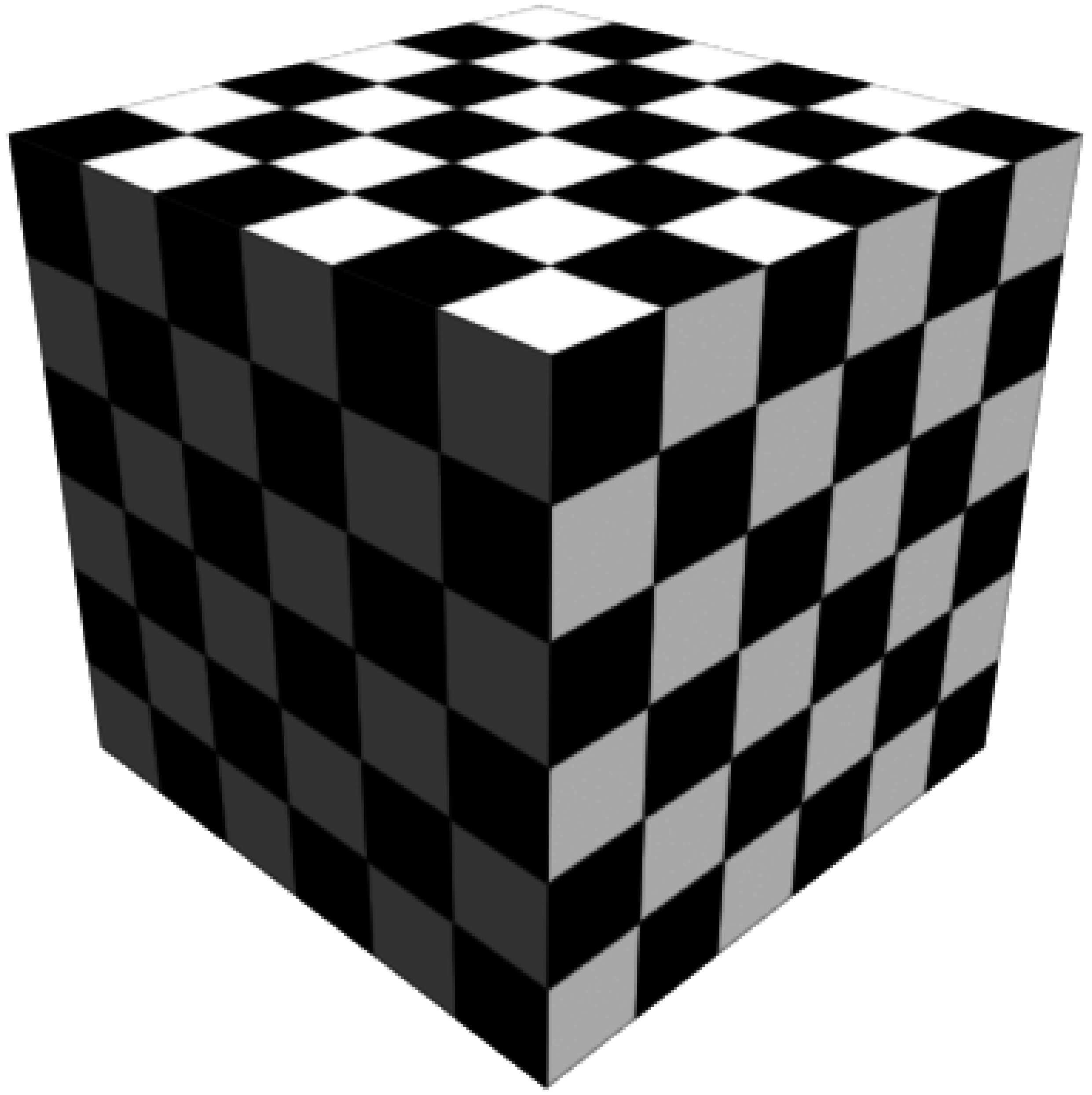
Store varying properties in images

Map to objects

Proposed by
Ed Catmull in the 70s



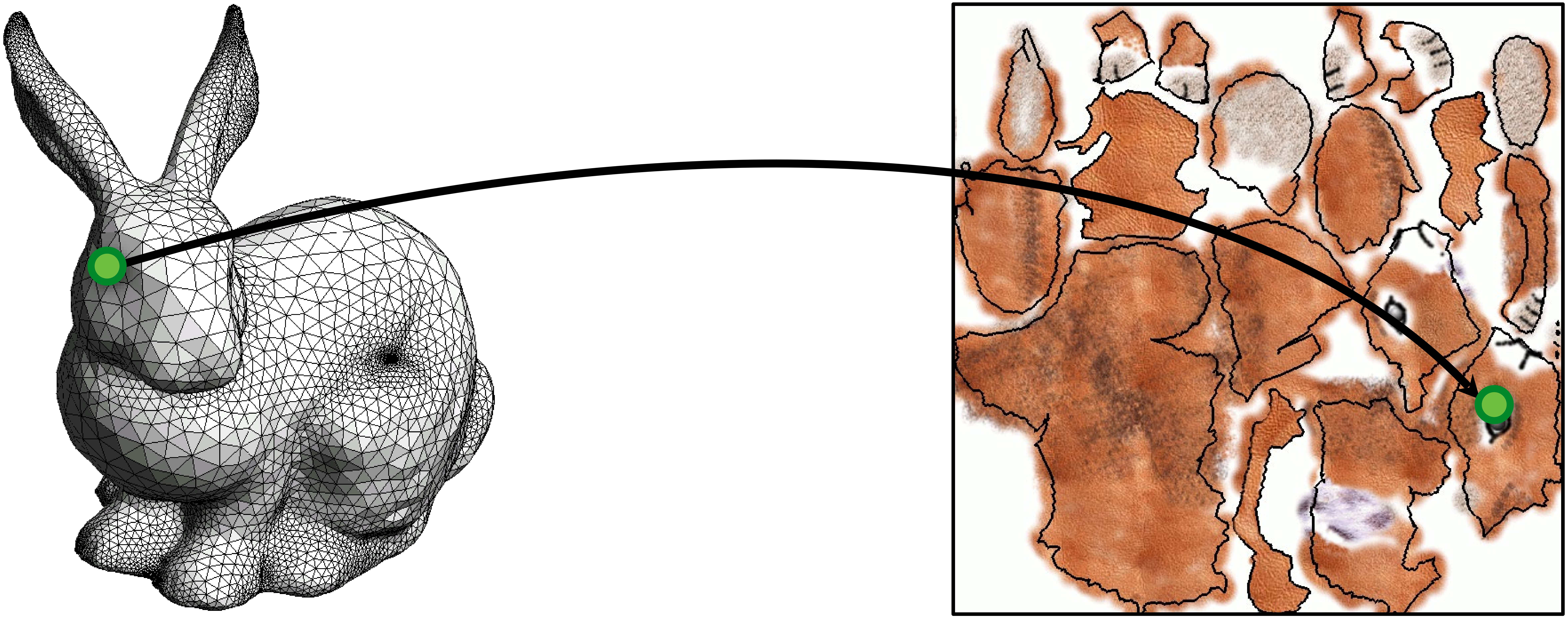




After a slide by Fabio Pellacini

Texture mapping – definition

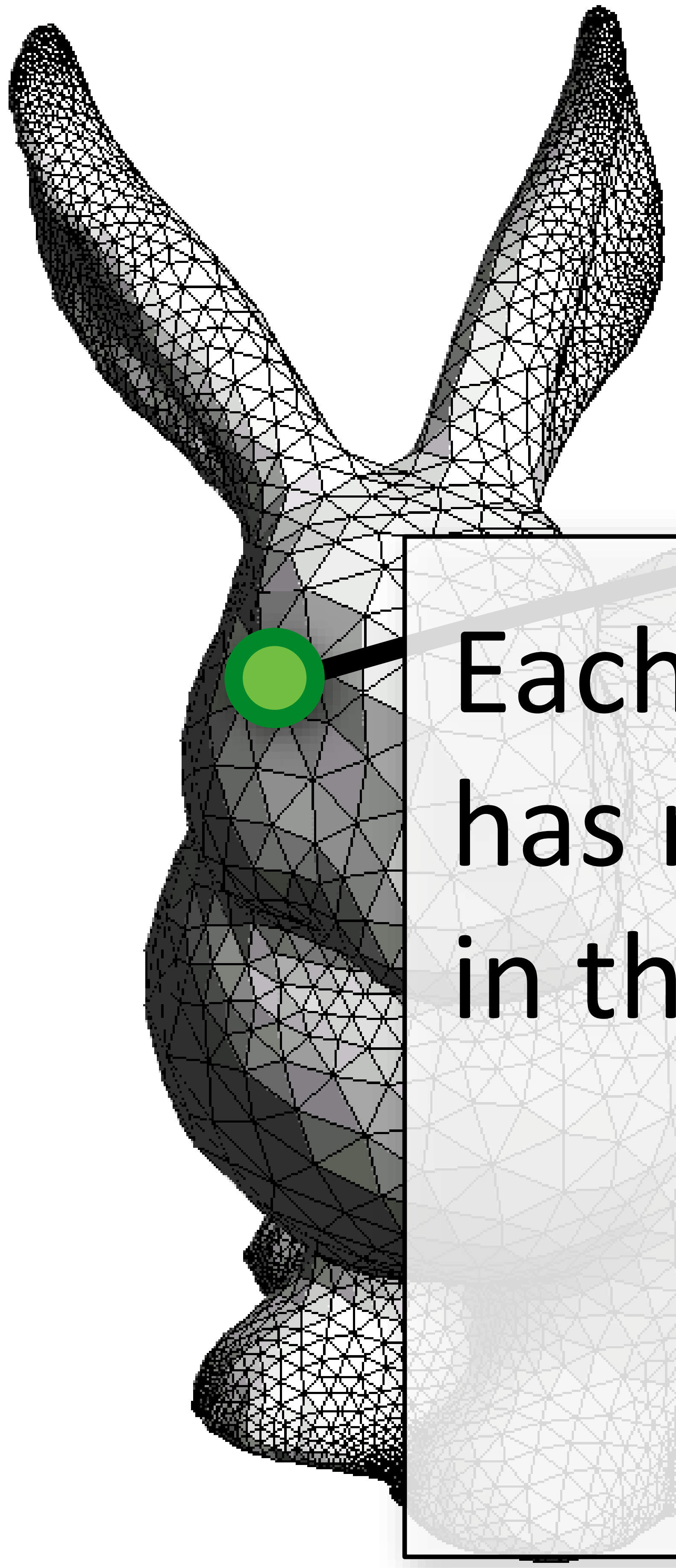
Mapping between the surface and the image



After a slide by Daniele Panozzo

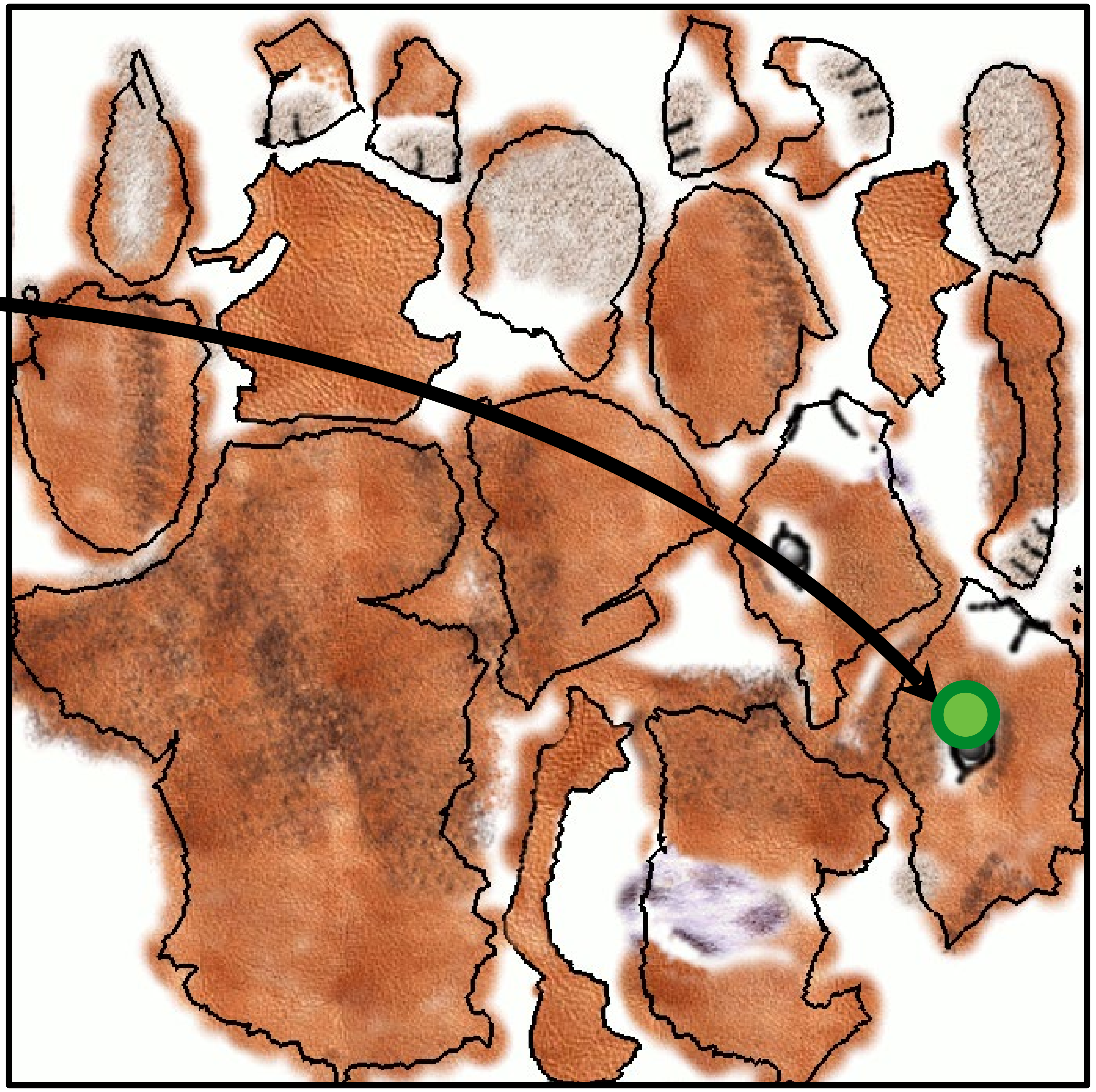
Texture mapping – definition

Mapping between the surface and the image



Each point (x,y,z) on the surface has mapped coordinates (u,v) in the texture image:

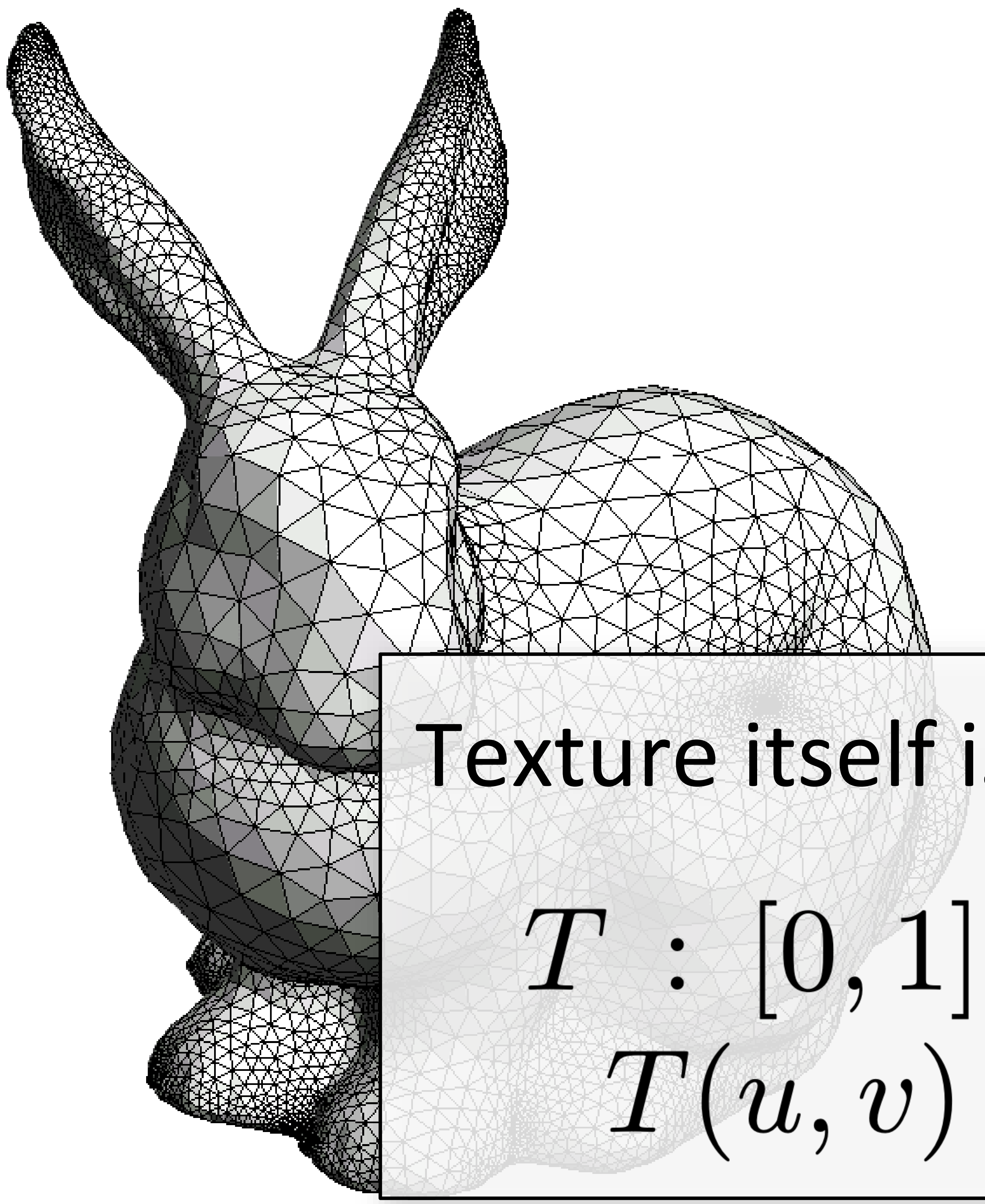
$$f : \mathbb{R}^3 \rightarrow [0,1]^2$$
$$f(x,y,z) = (u,v)$$



After a slide by Daniele Panozzo

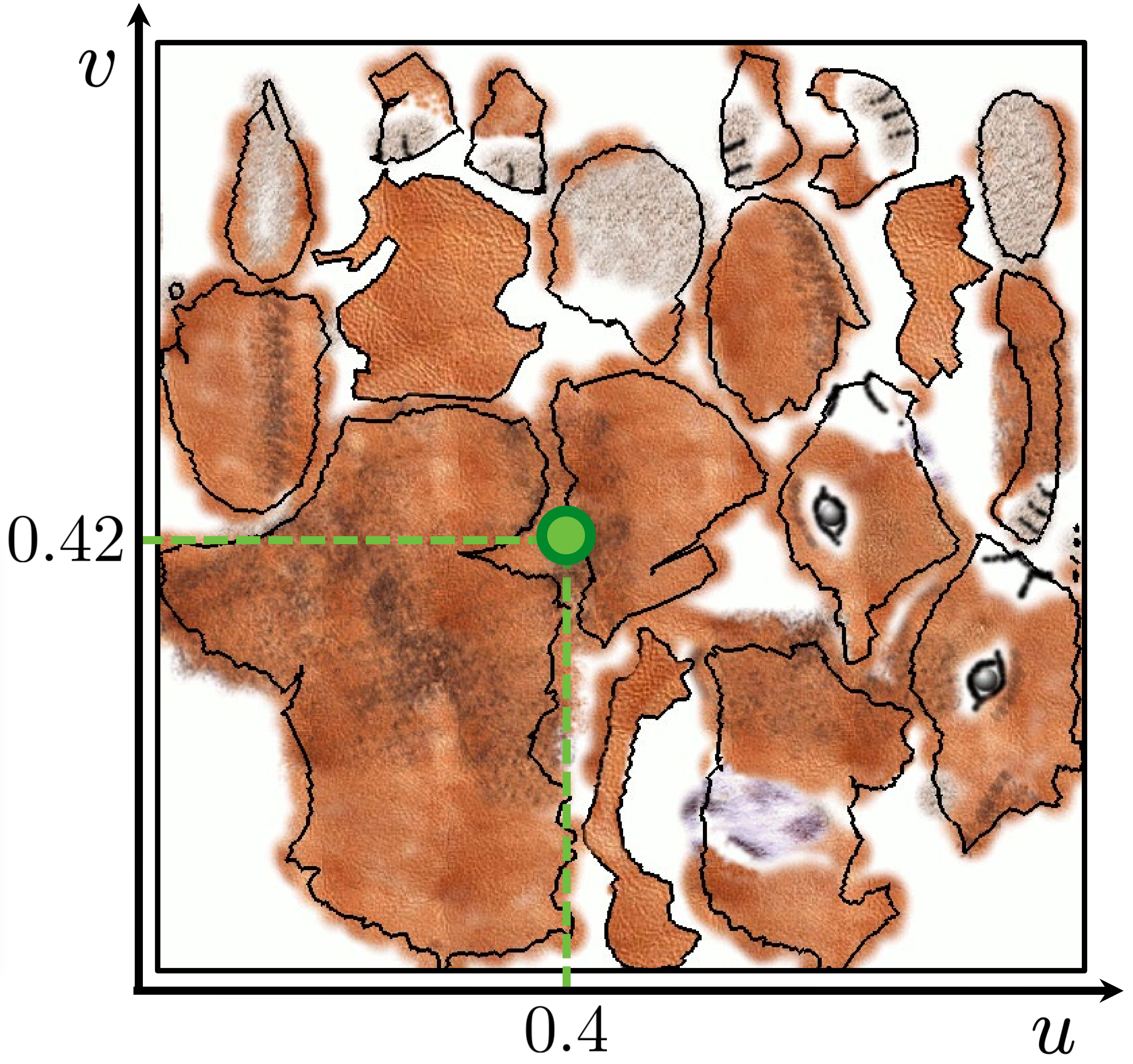
Texture mapping – definition

Mapping between the surface and the image



Texture itself is a function:

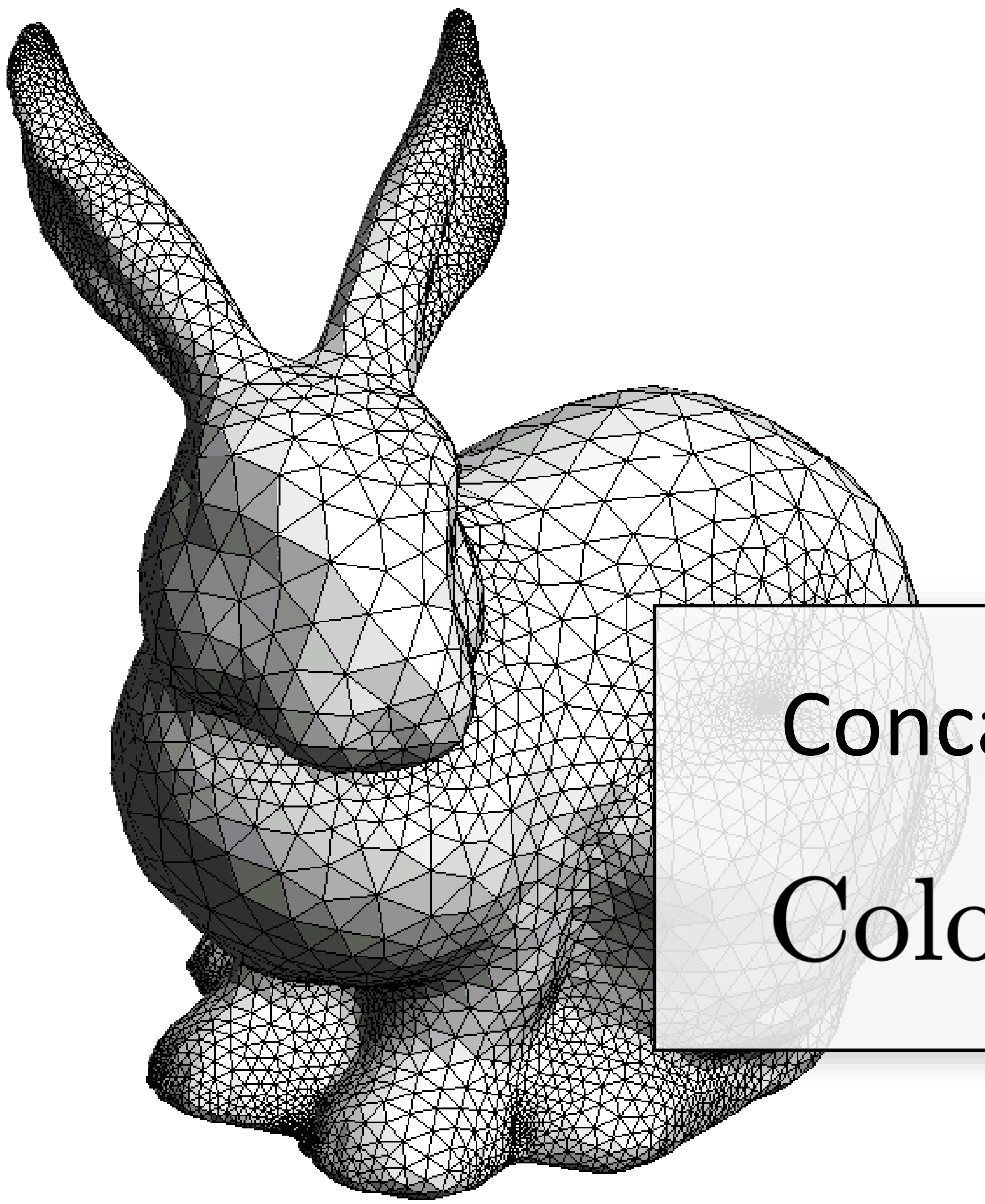
$$T : [0,1]^2 \to \mathrm{RGB}$$
$$T(u,v) = (r,g,b)$$

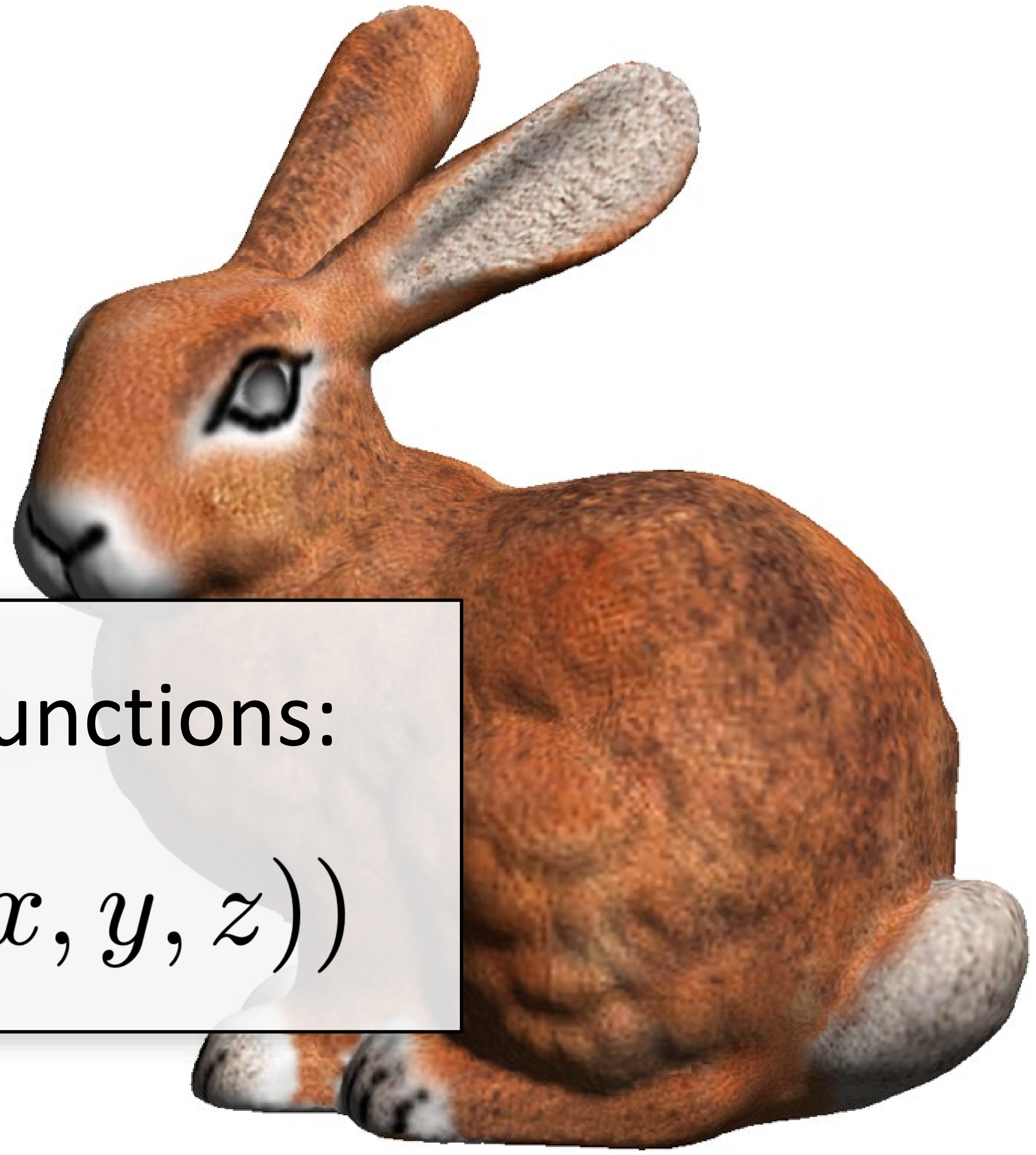


After a slide by Daniele Panozzo

Texture mapping – definition

Mapping between the surface and the image





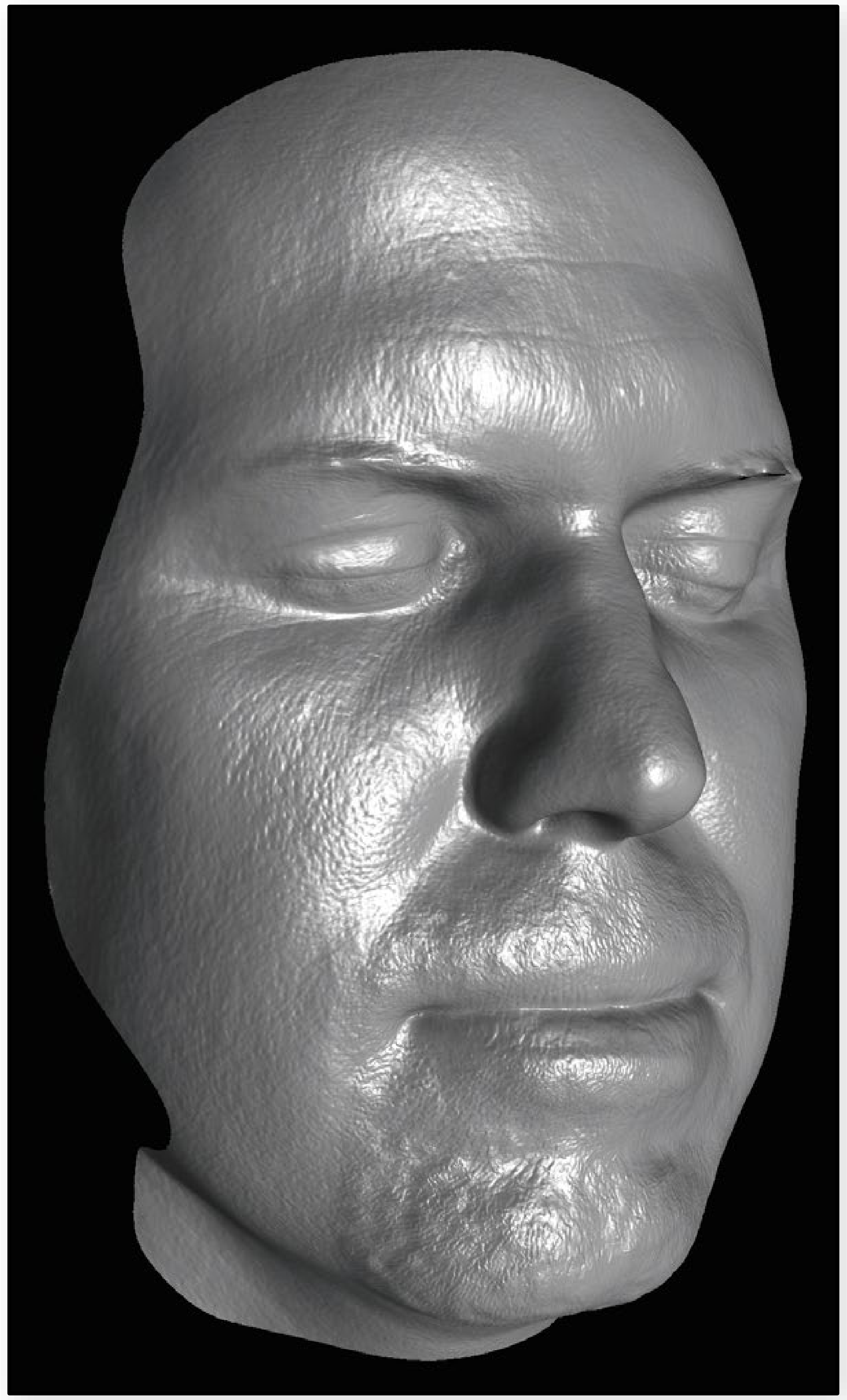
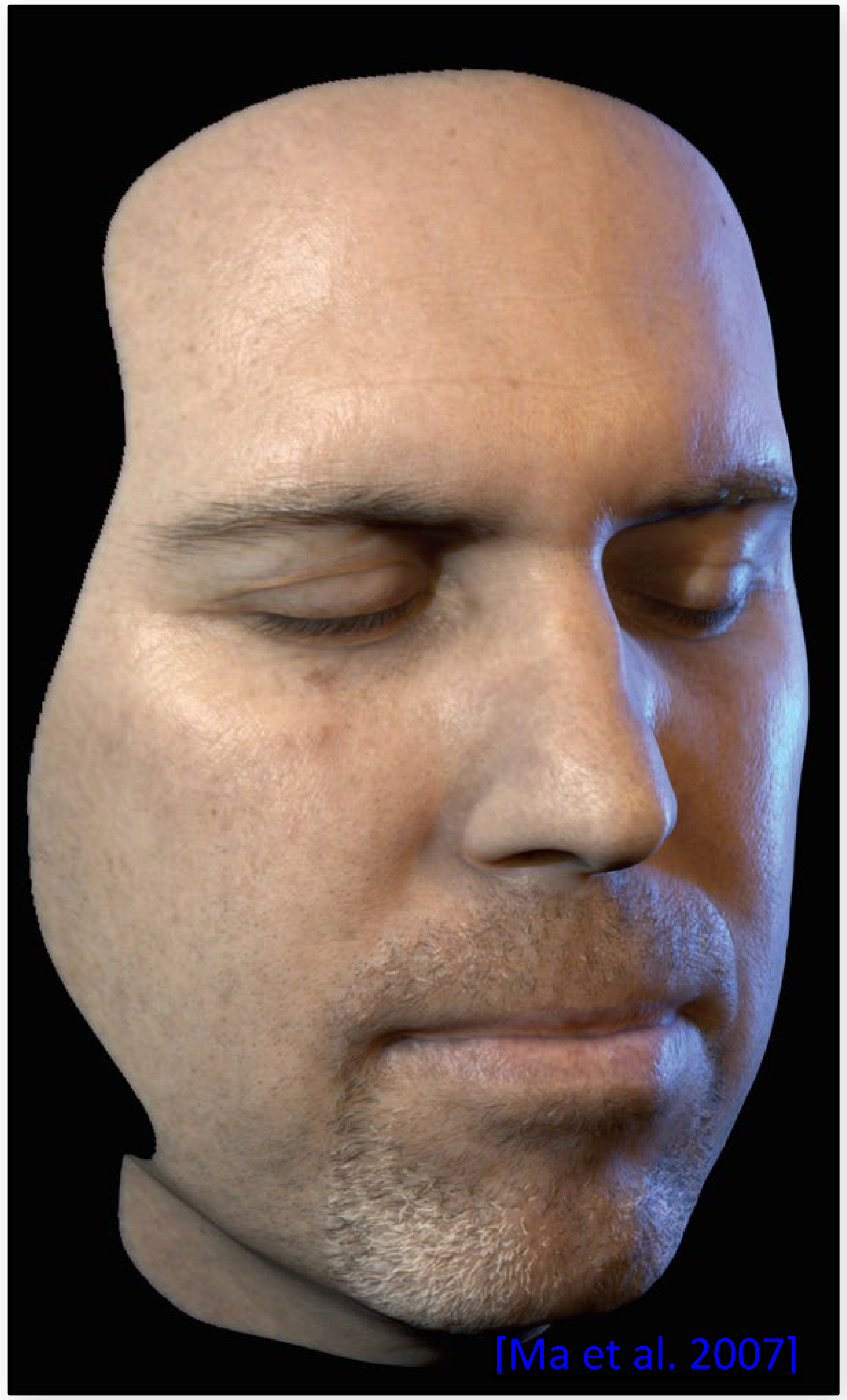
Concatenation of the two functions:

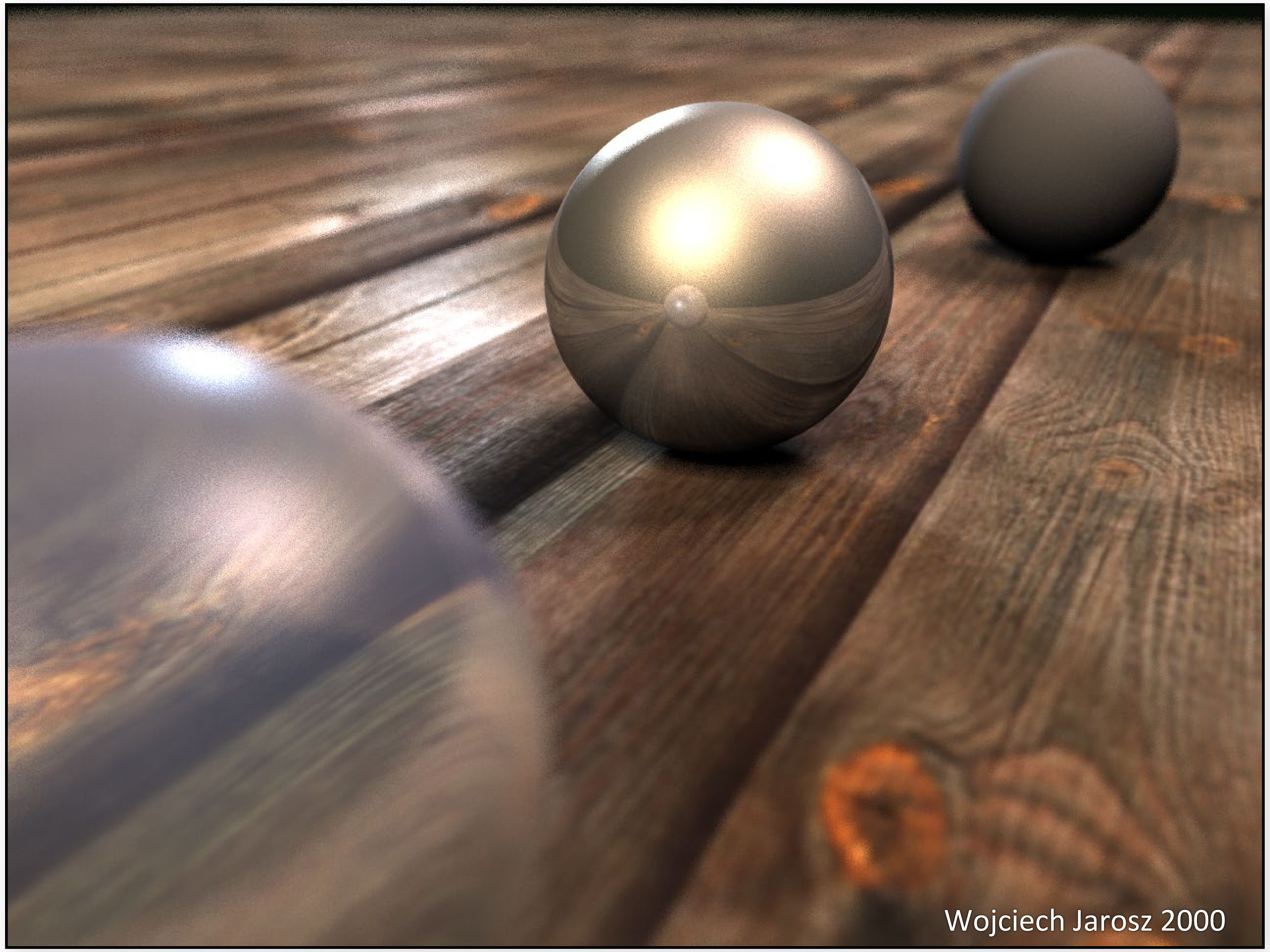
$$\mathrm{Color}(x, y, z) = T(f(x, y, z))$$

After a slide by Daniele Panozzo

Why texture mapping?

Produces compelling results



Agenda

How do we map between surface and texture?

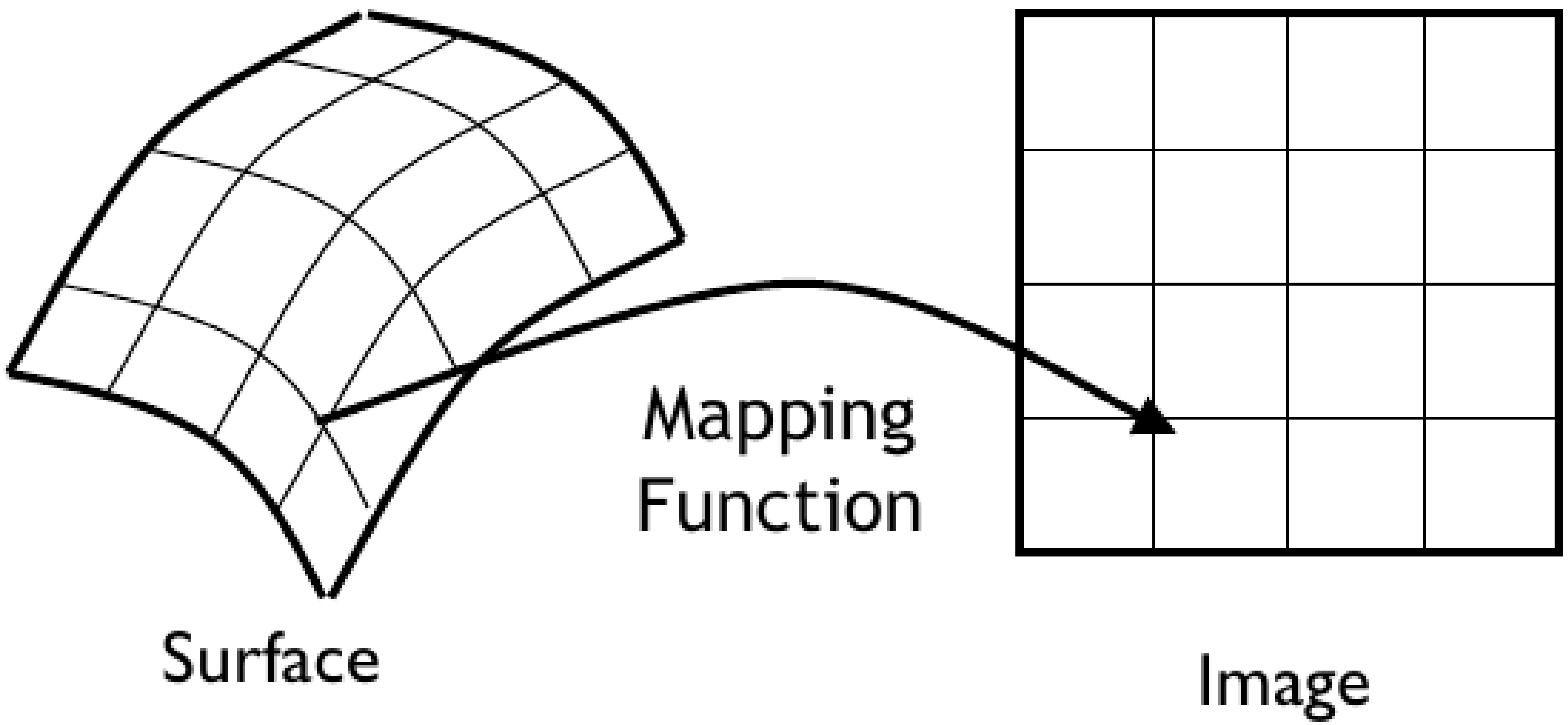
What do we map onto the surface?

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



After a slide by Fabio Pellacini

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

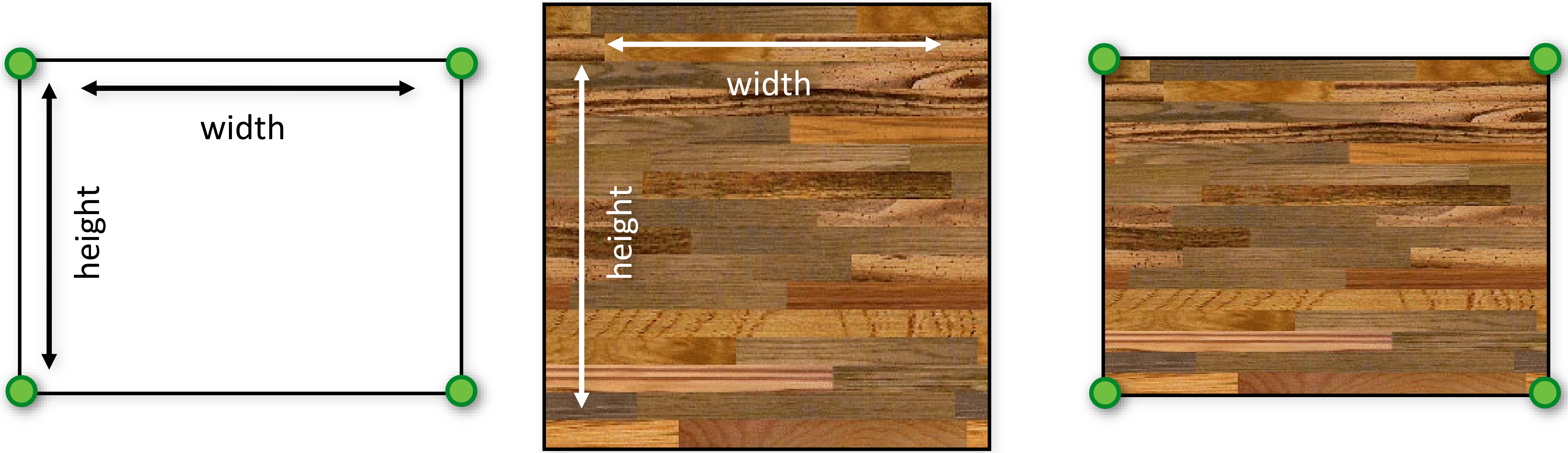
After a slide by Steve Marschner

A rectangle

Image can be mapped directly, unchanged

Texturing a rectangle

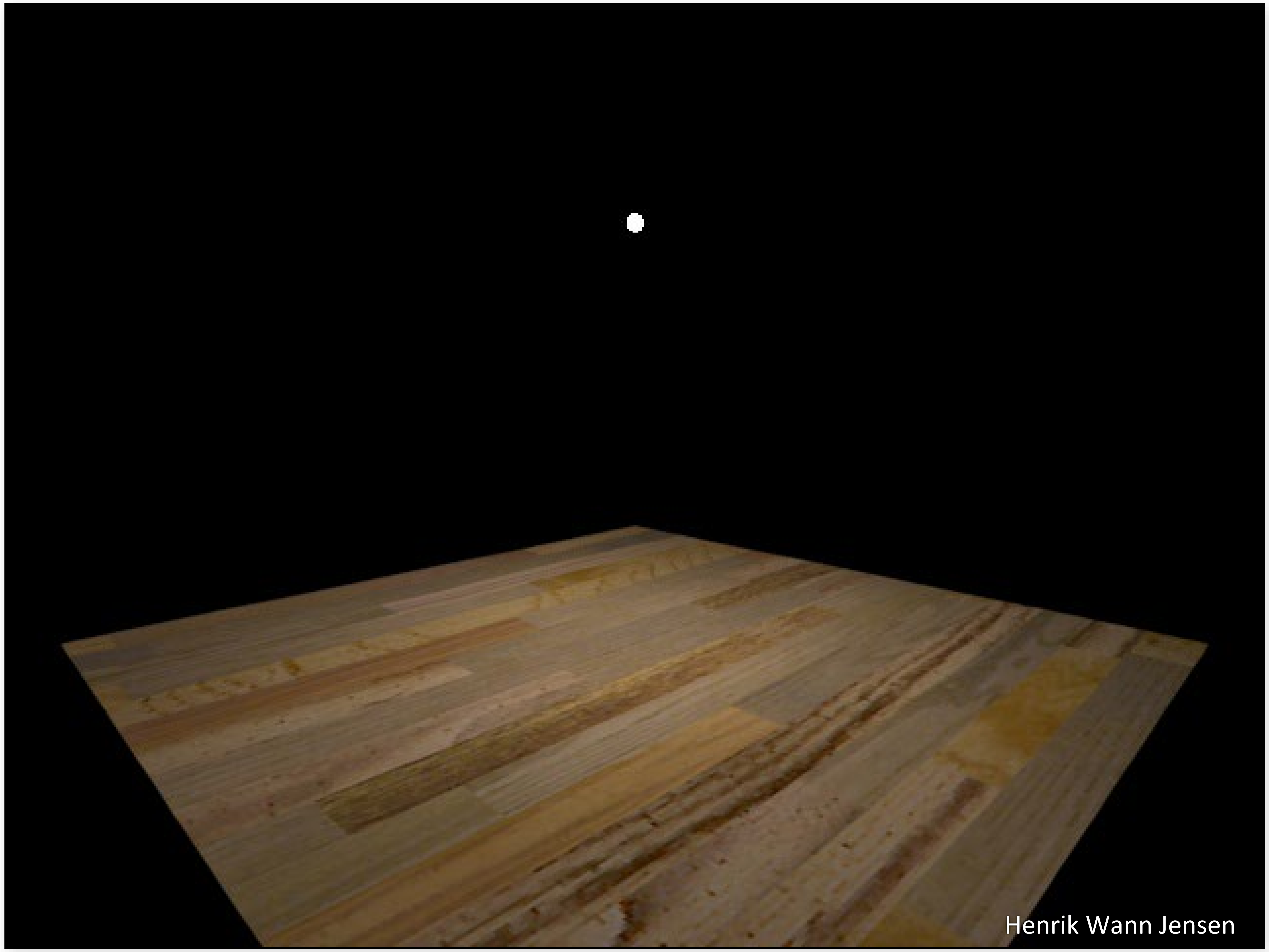
Image can be mapped directly, unchanged



Object Space Texture Space Object Space

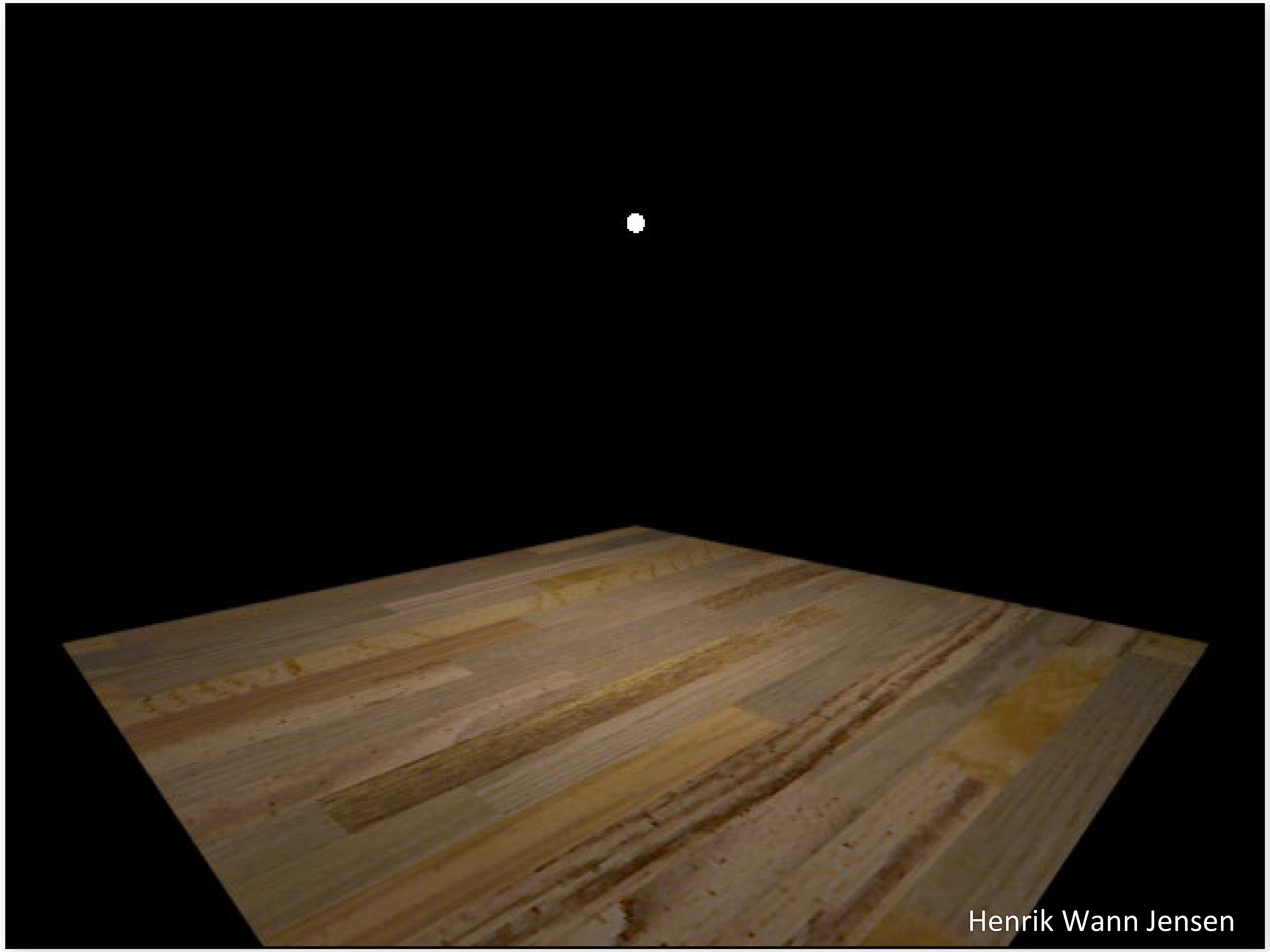
A textured rectangle

albedo = $T(u,v)$



Transformation of shape

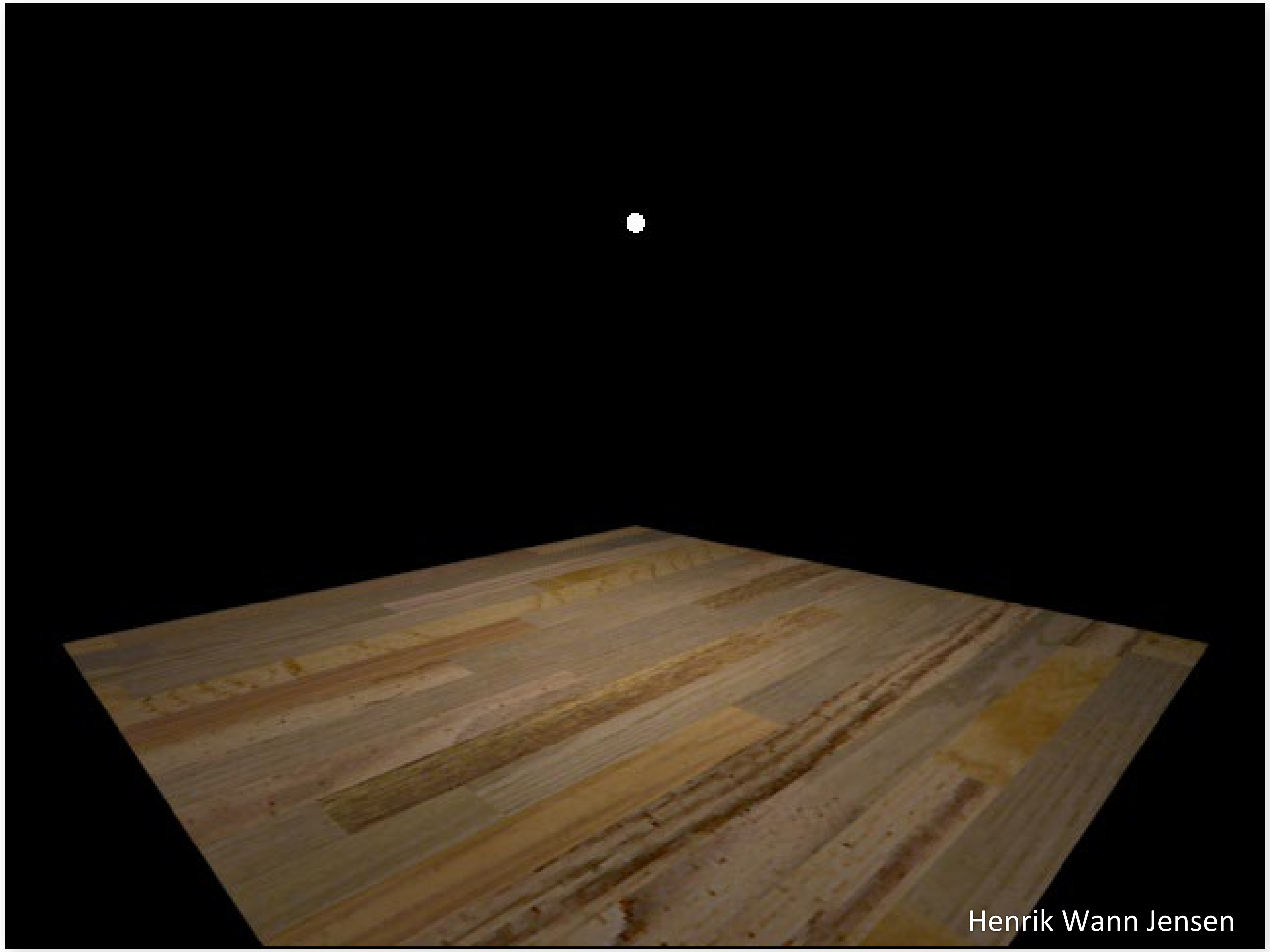
albedo=$T([u,v](x,y,z))$



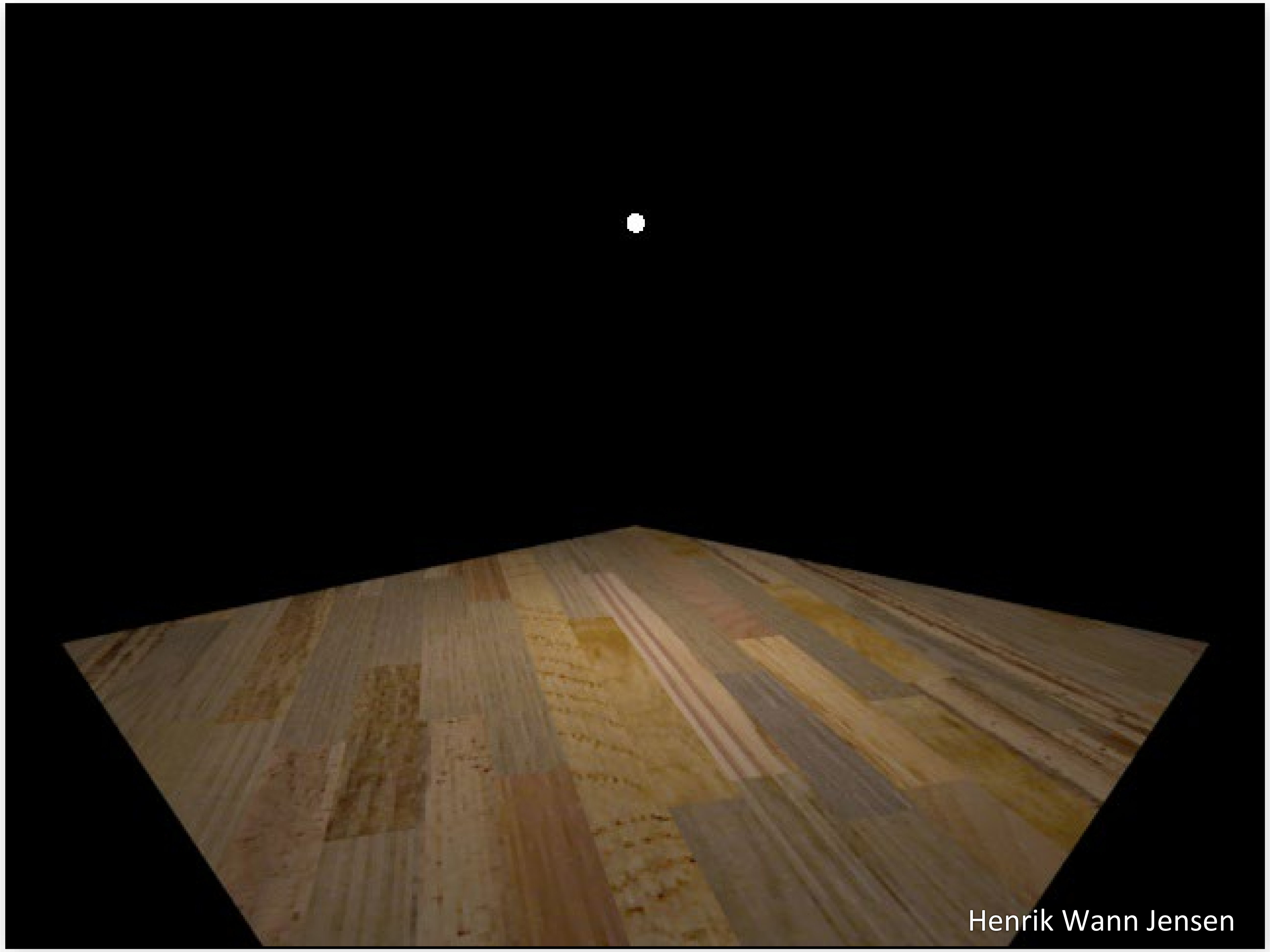
albedo=$T([u,v](\mathrm{transform}^{-1}(x,y,z)))$

Transformation of texture

albedo=$T([u,v](x,y,z))$



albedo=$T(transform[u,v](x,y,z))$



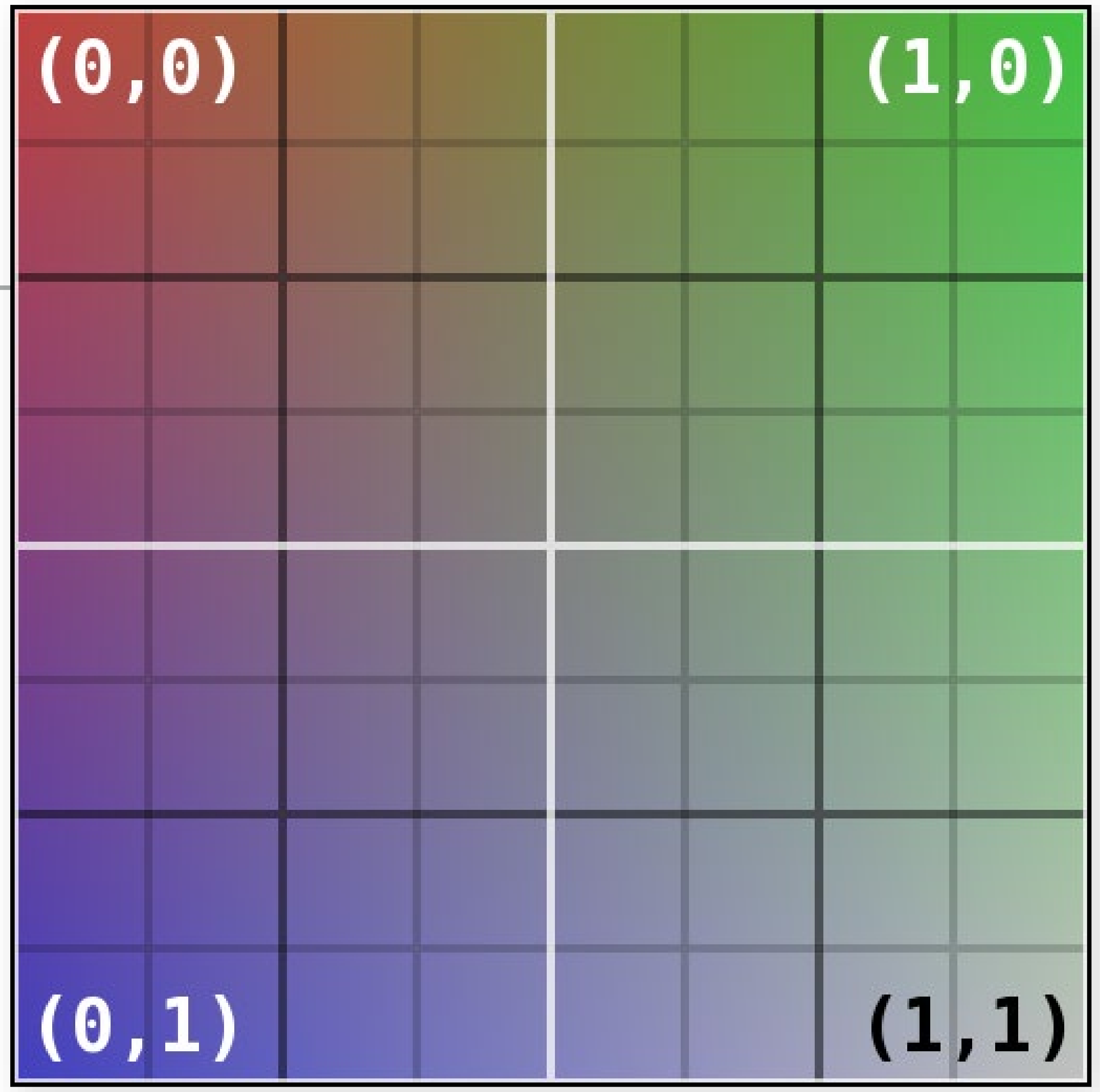
What if the object is not rectangular?

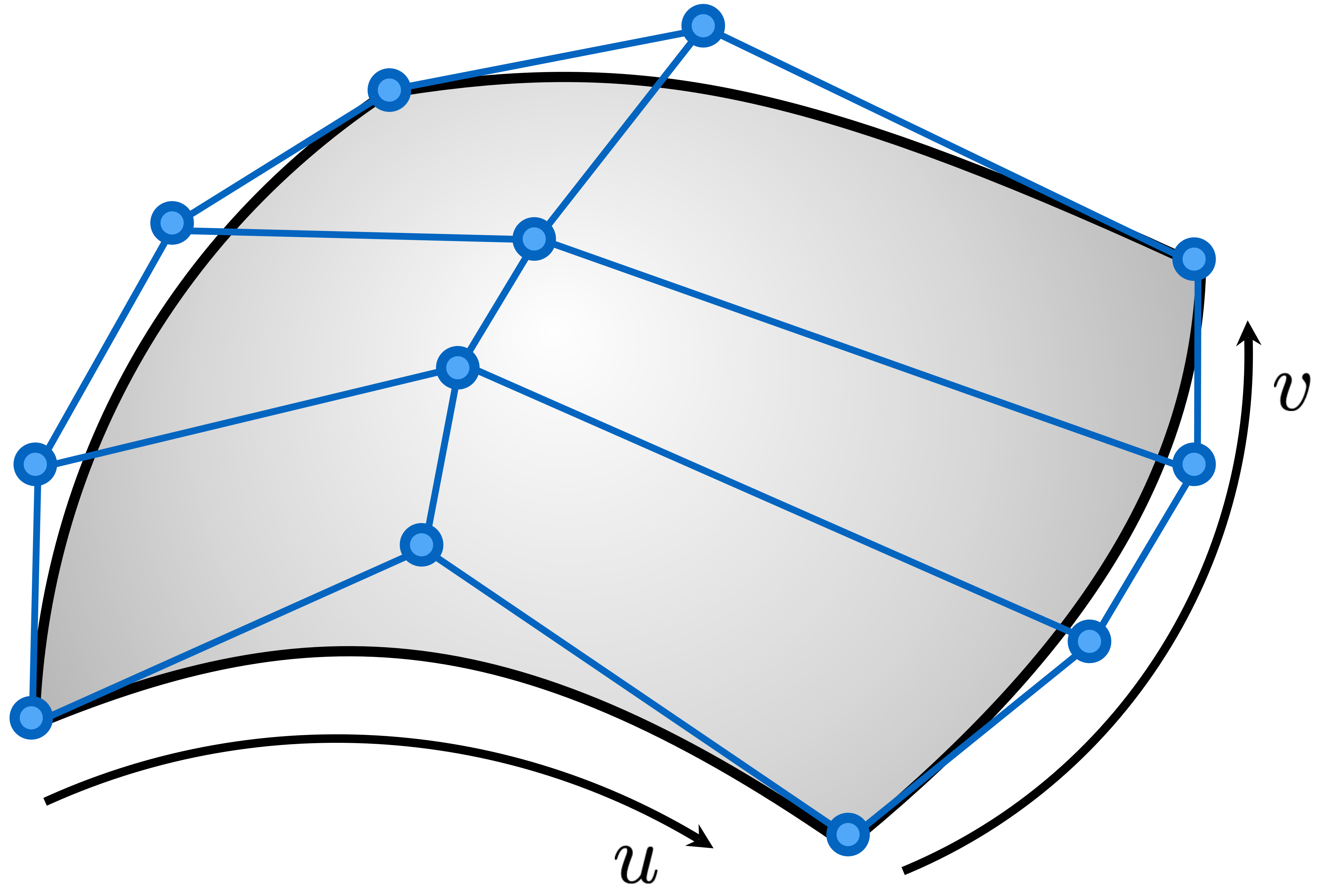
Need to determine mapping from 3D coordinates to 2D texture coordinates

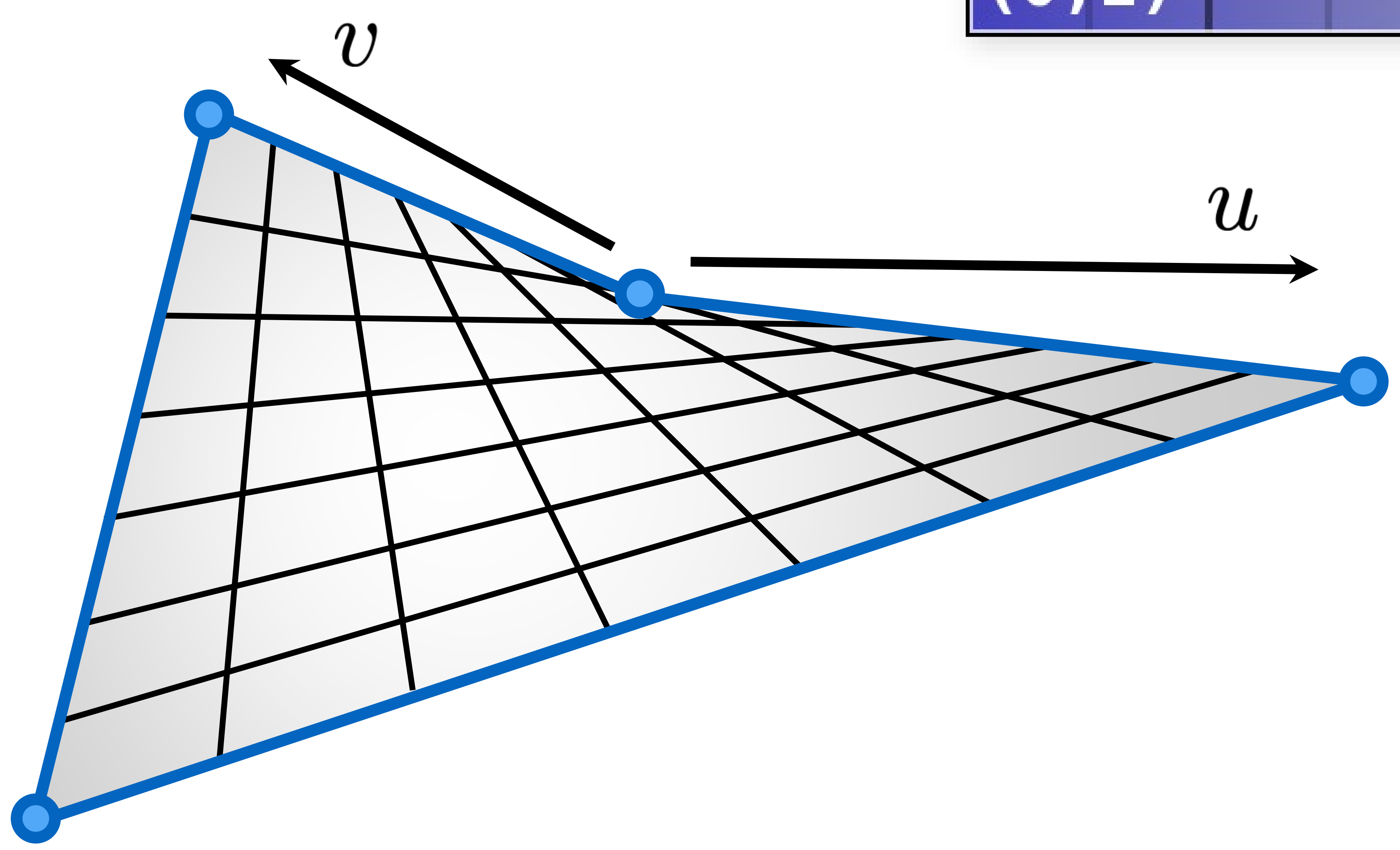
- Parametric surfaces
- Projection mapping
- UV mapping

Parametric surfaces

Parametric surfaces have a natural 2D coordinate system.







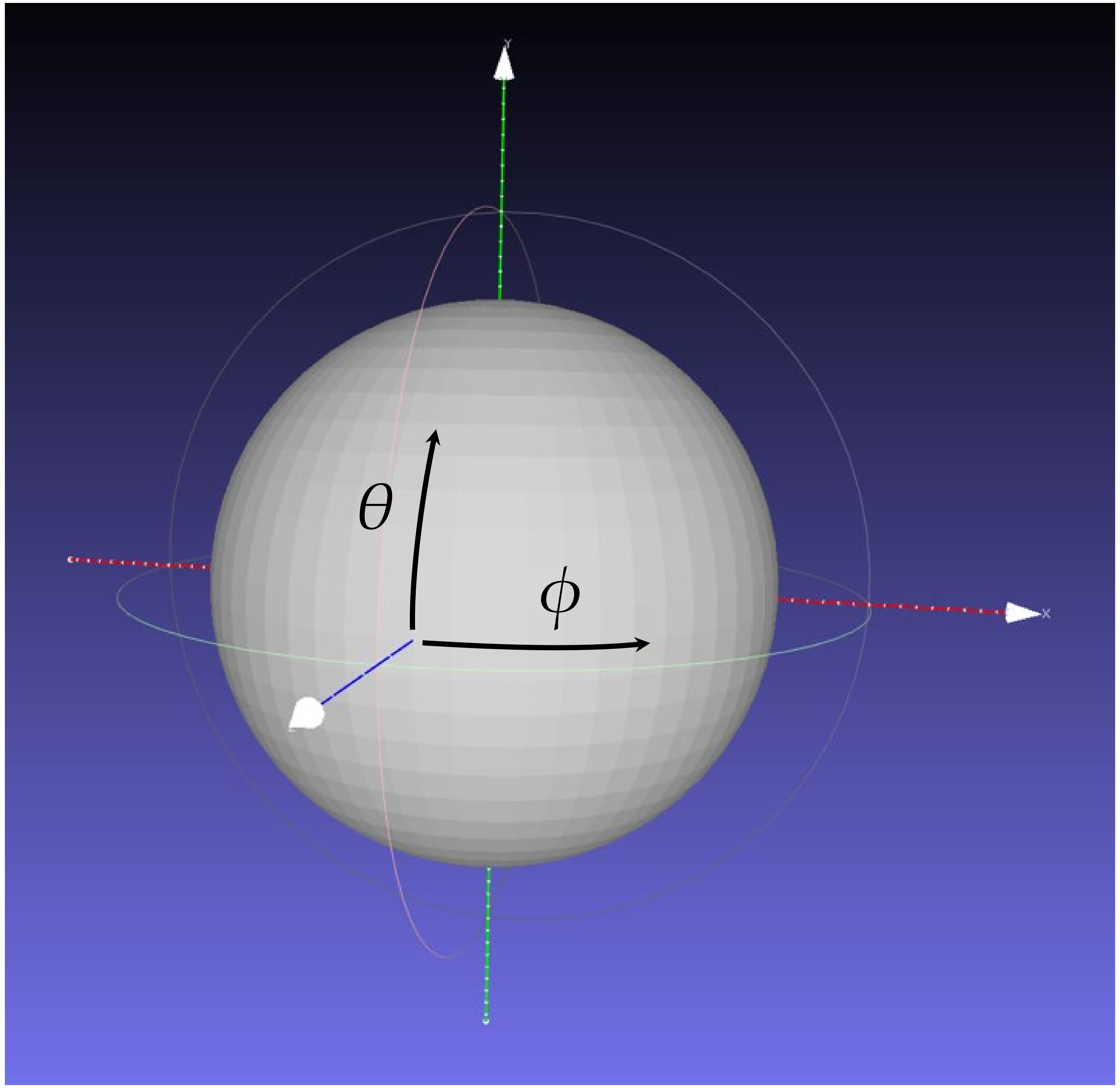
Can also define a sphere parametrically

Position:

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

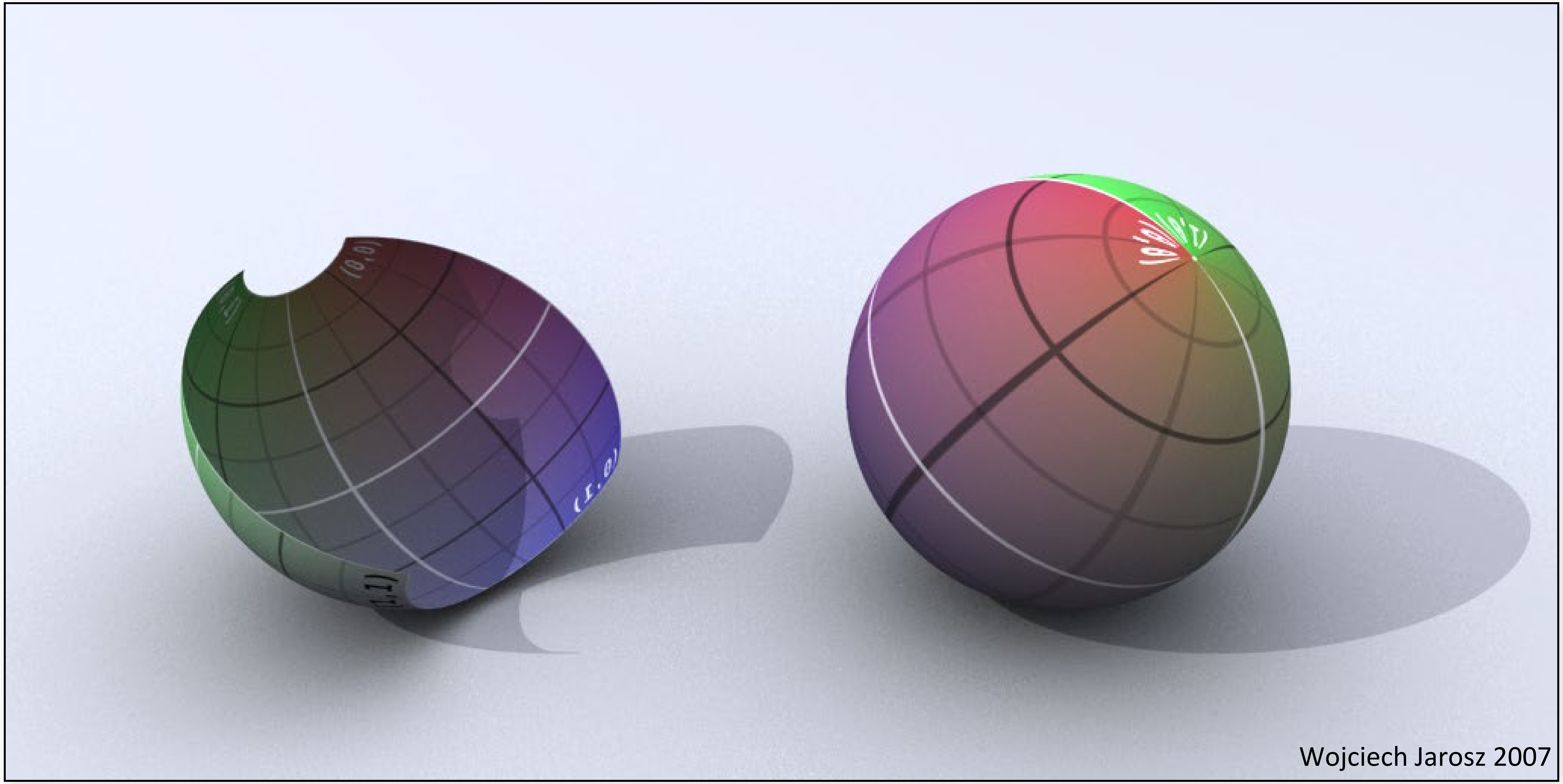
$$y = \sin\theta$$

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



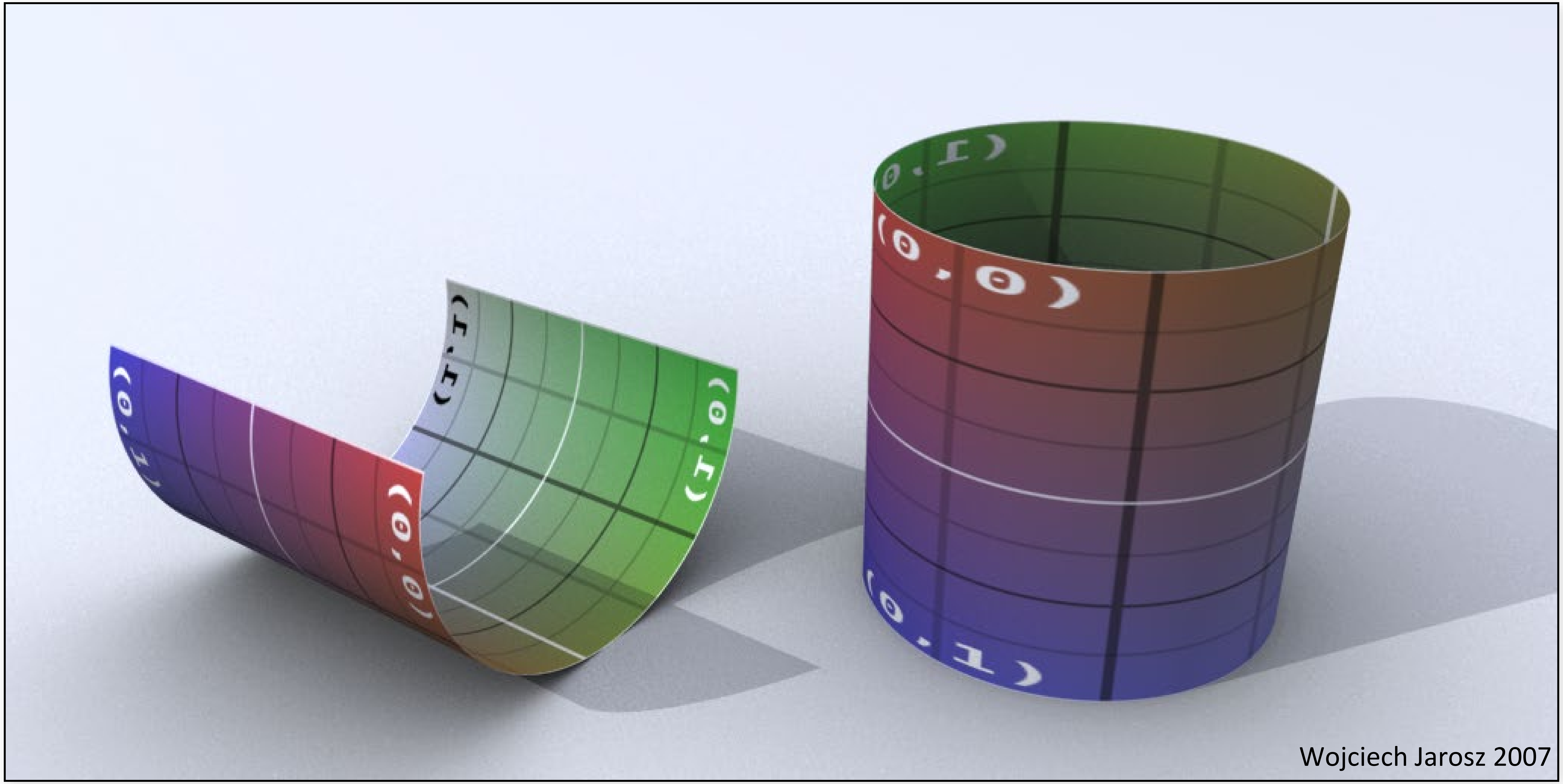
After a slide by Steve Marschner

Parametric spheres



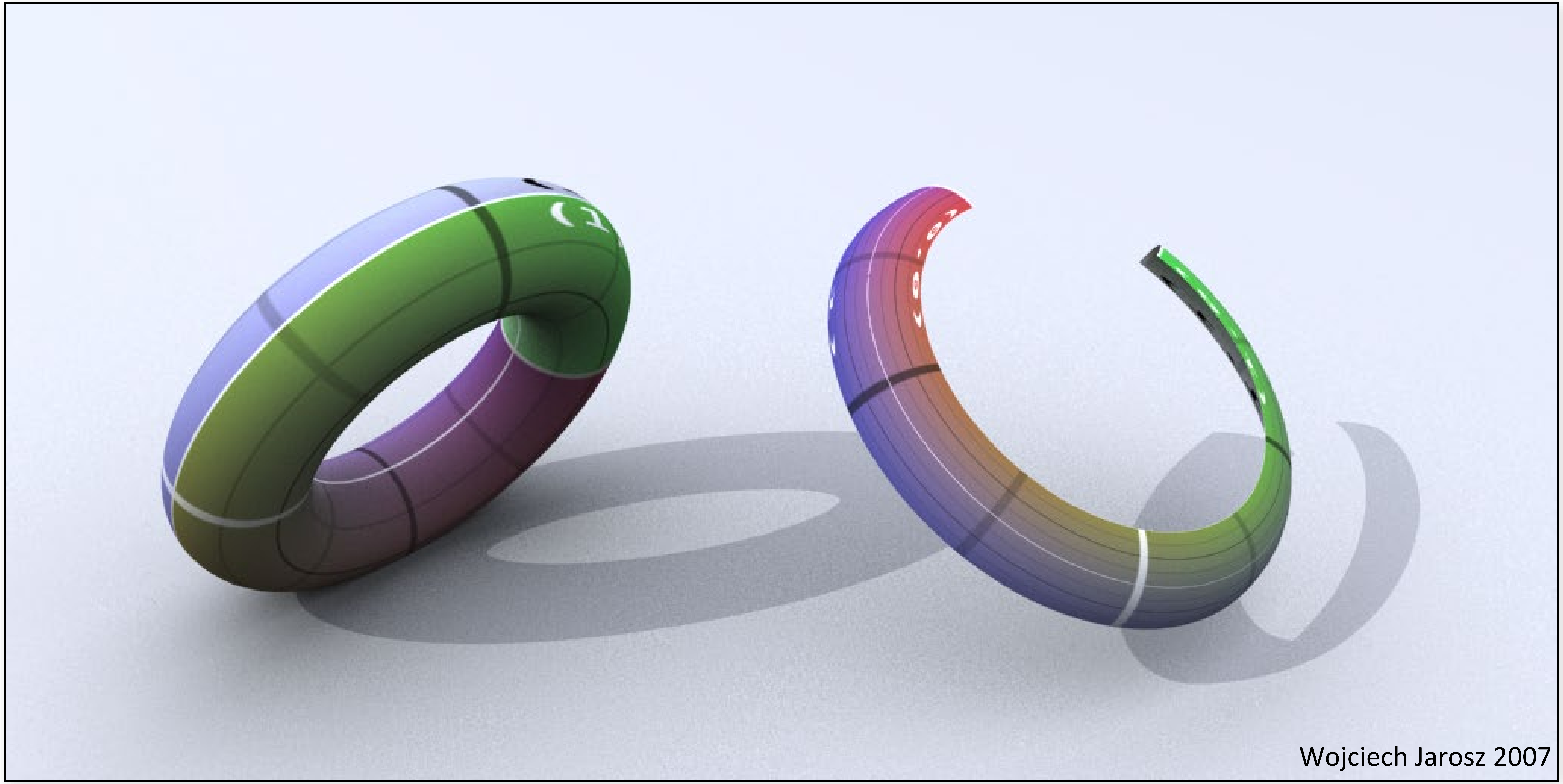
Wojciech Jarosz 2007

Parametric cylinders



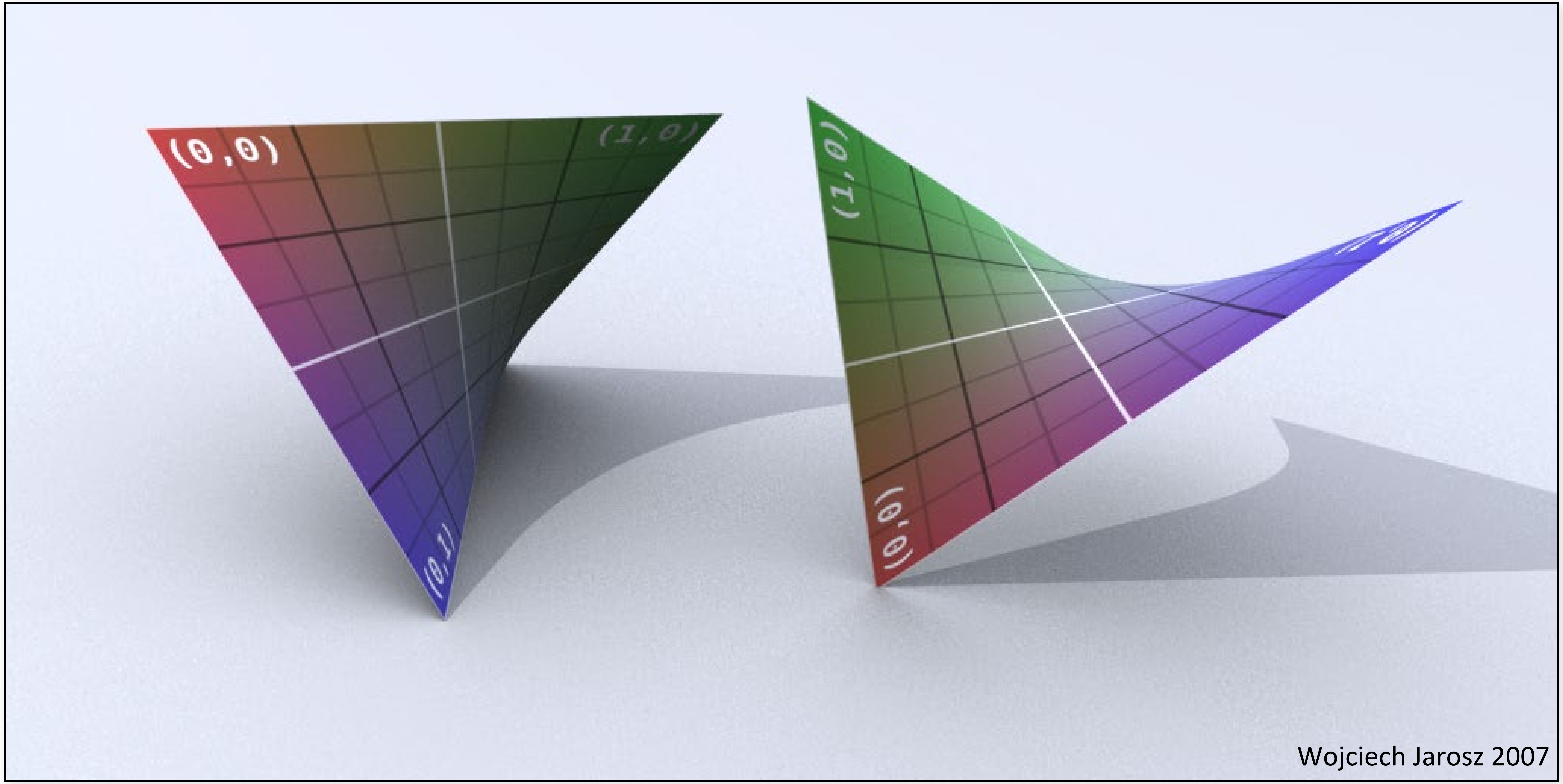
Wojciech Jarosz 2007

Parametric toruses

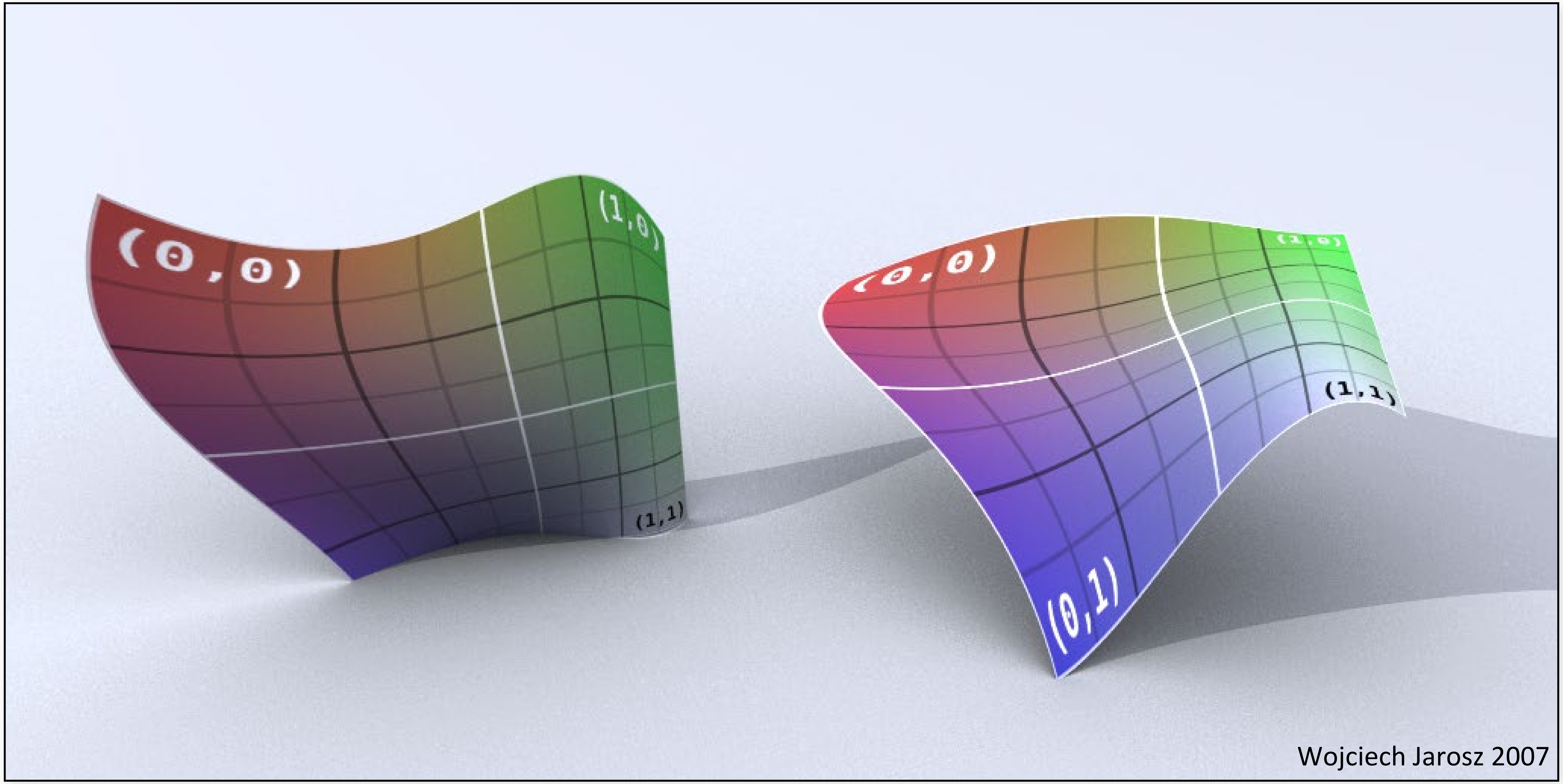


Wojciech Jarosz 2007

Bilinear patches



Bicubic patches



Creating parameterizations

For non-parametric surfaces it is trickier

Need to create a parametrization!

- Projection mapping
- UV mapping

[Wolfe / SG97]

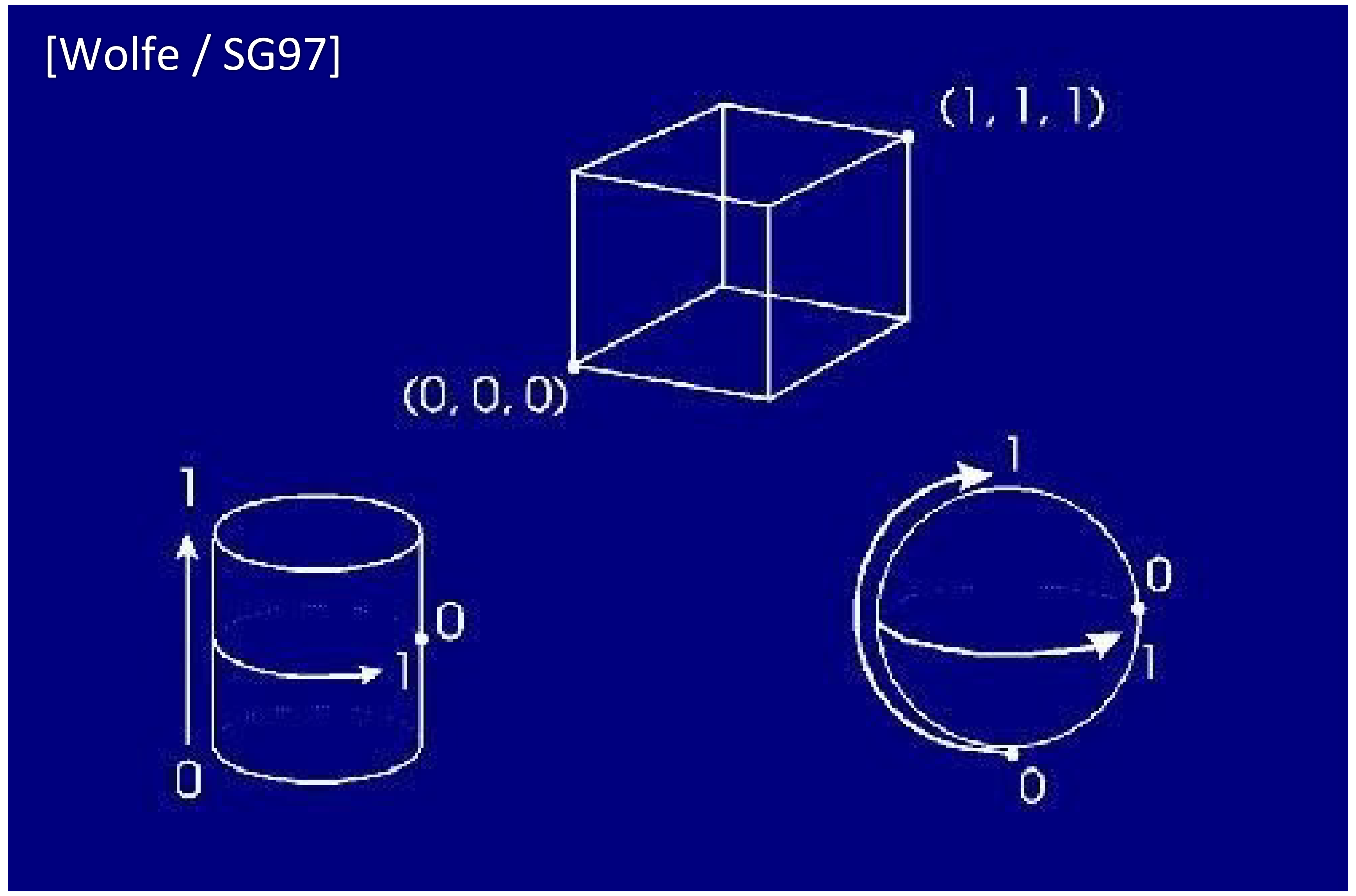
Projection mapping

Maps 3D surface points to 2D image coordinates

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

Different types of projections

- often corresponding to simple shapes
- useful for simple objects

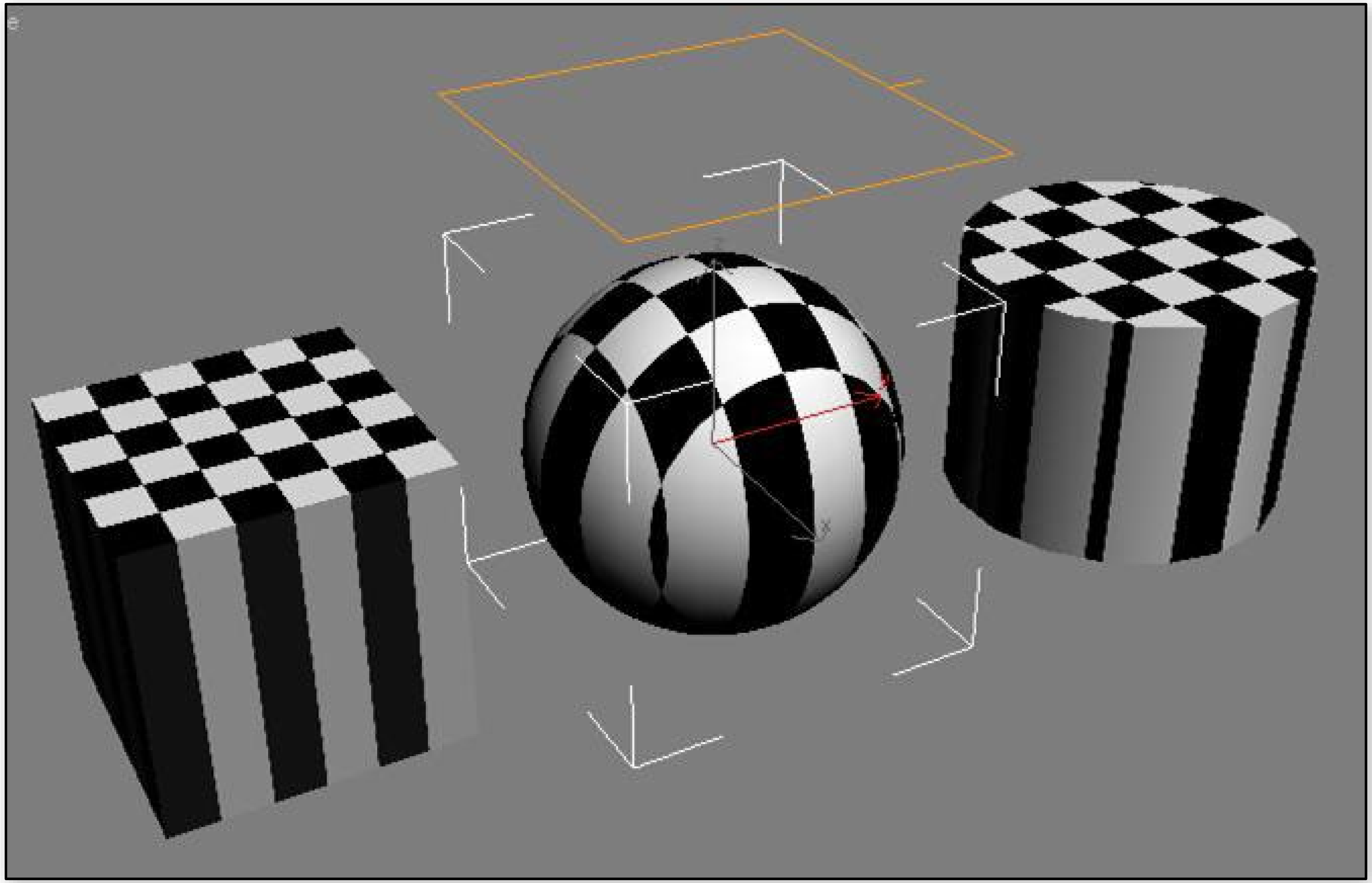


After a slide by Fabio Pellacini

Projections — planar

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

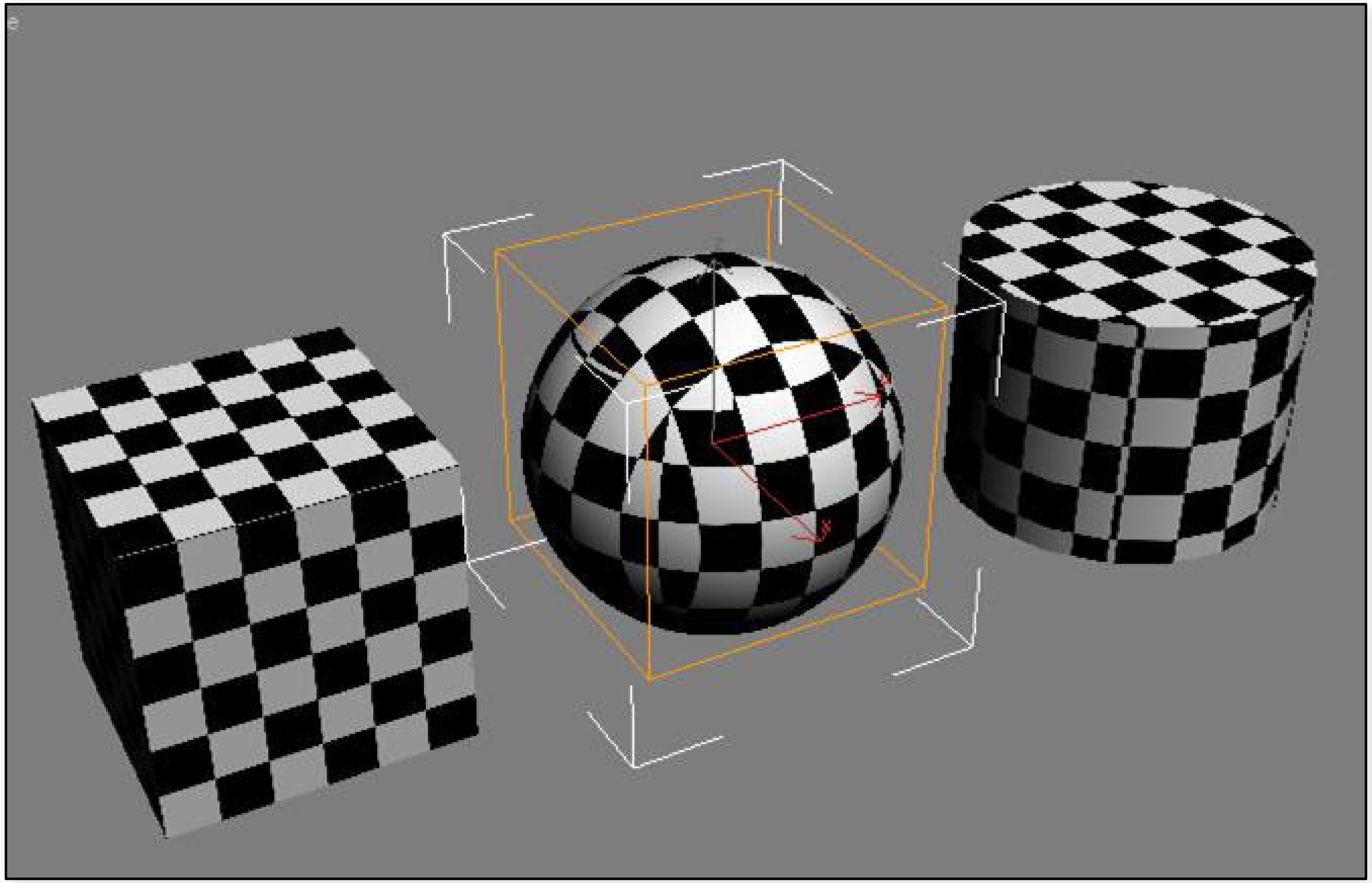
$$f(\mathbf{p}) = \begin{bmatrix} \frac{p_x}{w} \\ \frac{p_y}{h} \end{bmatrix}$$



After a slide by Fabio Pellacini

Projections — cubical

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



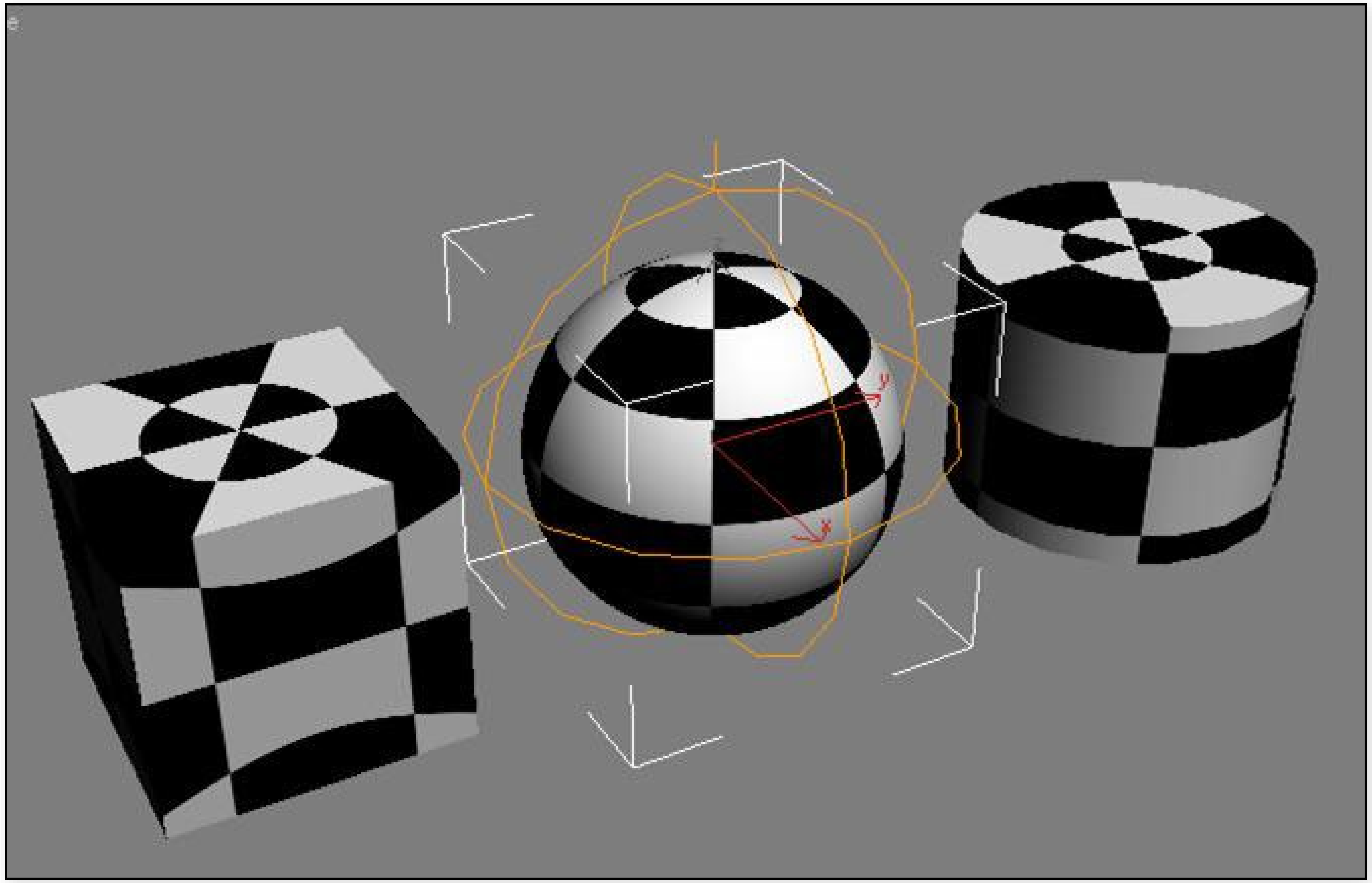
After a slide by Fabio Pellacini

Projections — spherical

Project point onto unit sphere

- compute spherical coordinates

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



After a slide by Fabio Pellacini

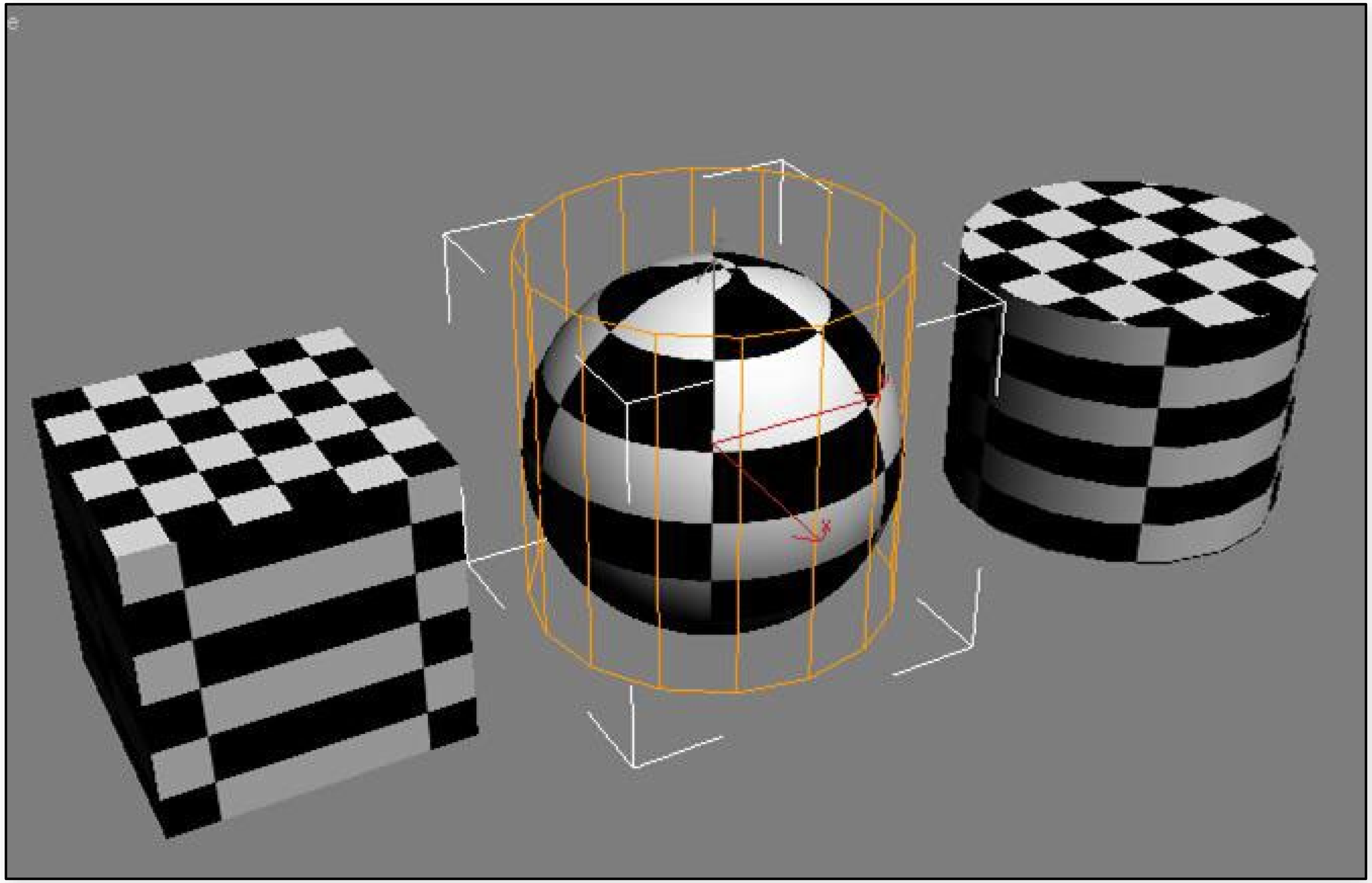
Projections — cylindrical

Project point onto cylinder of height h

- compute cylindrical coordinates

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

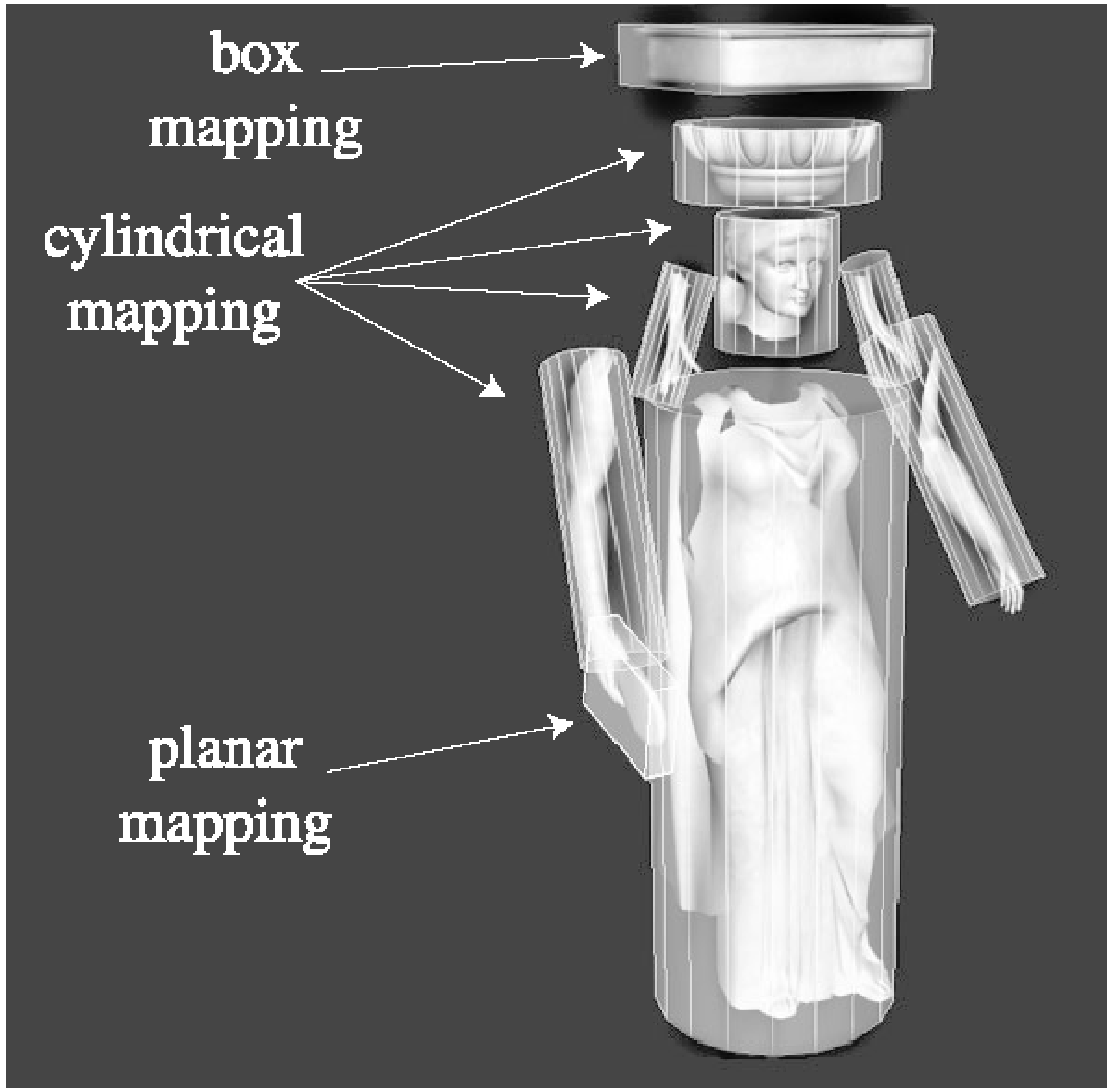
- treat caps separately



After a slide by Fabio Pellacini

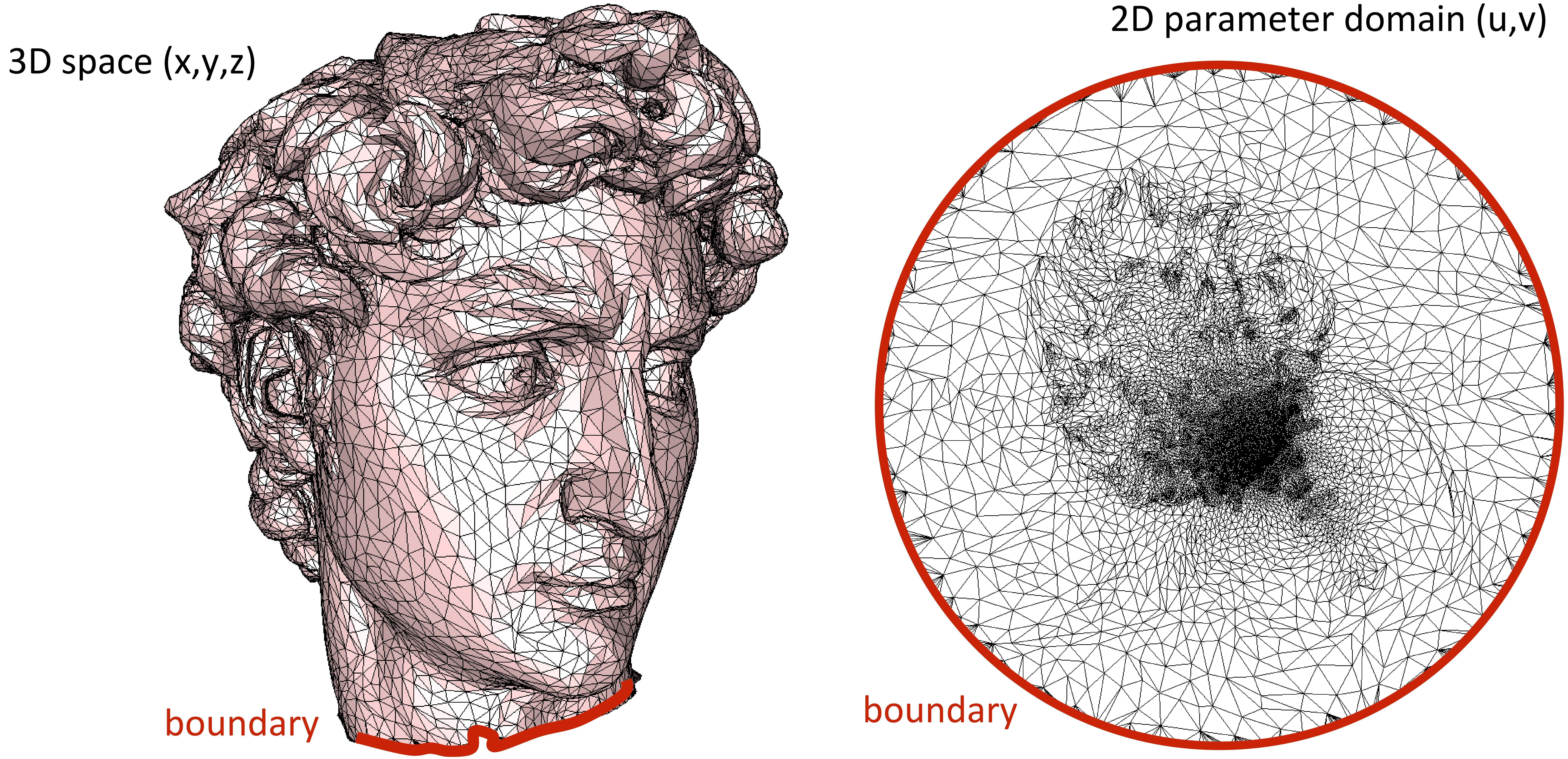
Combining projections

Non-parametric surfaces: project pieces to parametric surface



[Moller & Haines 2002]

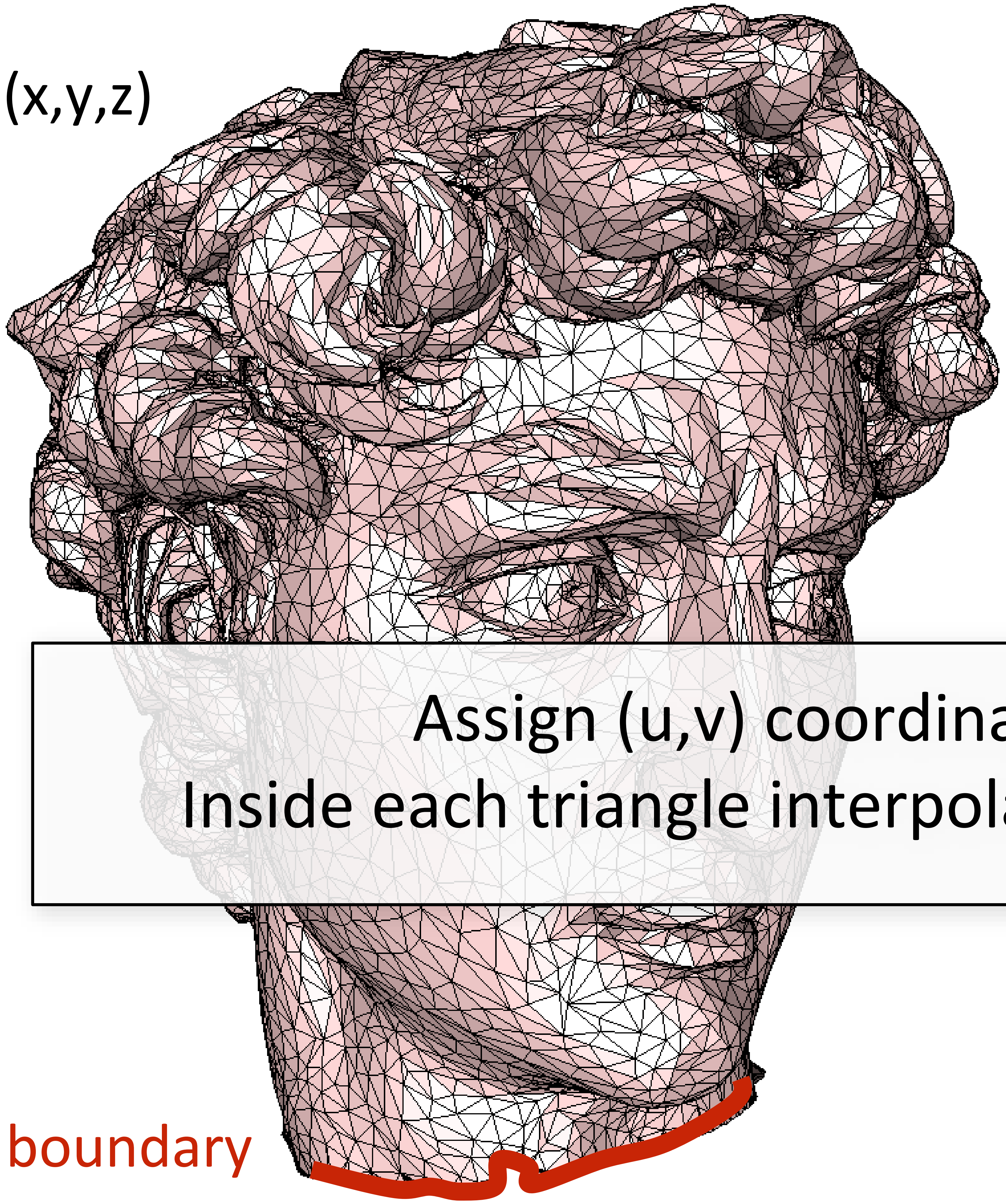
Creating UV parameterizations



After a slide by Daniele Panozzo

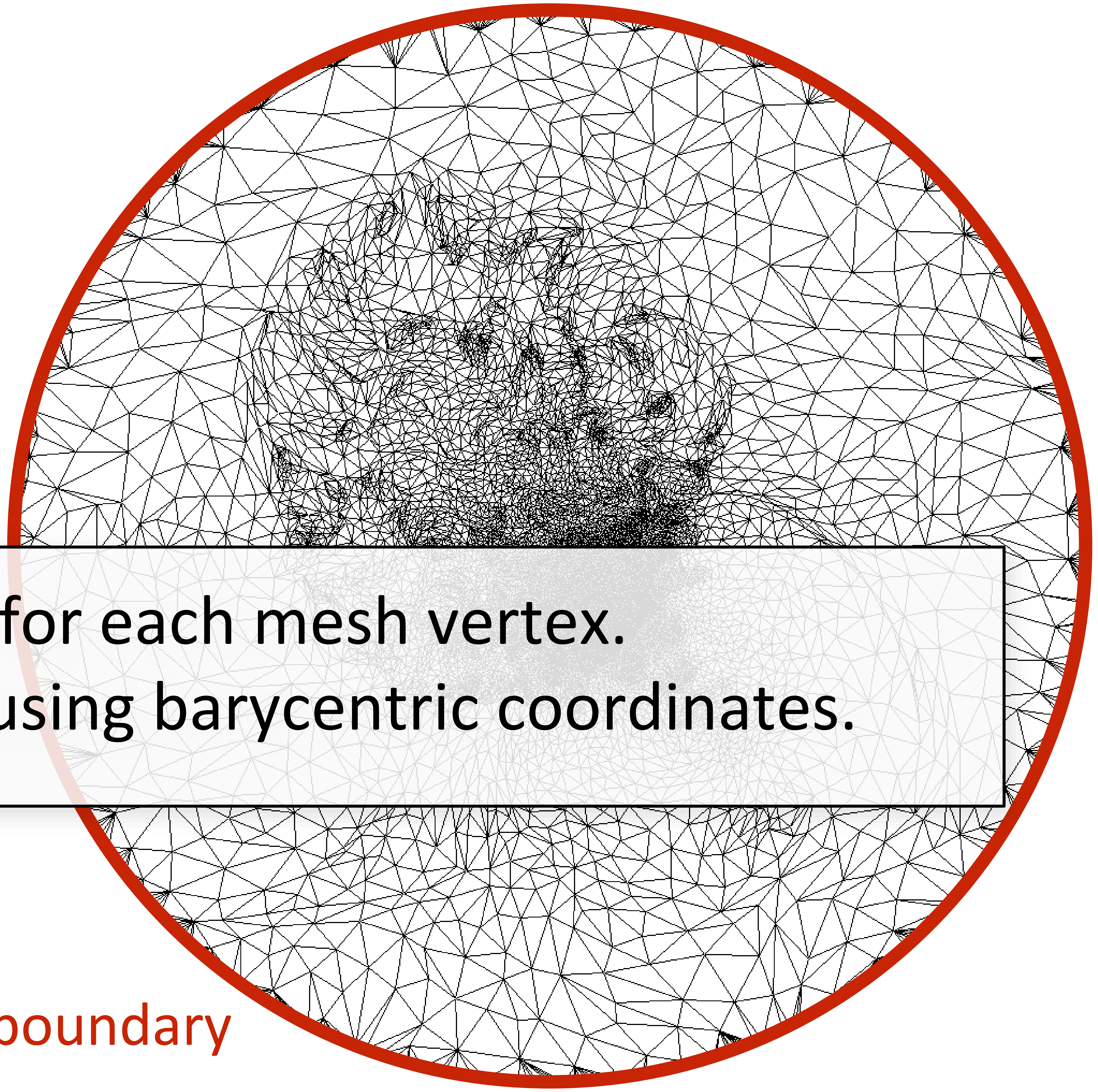
Creating UV parameterizations

3D space (x,y,z)



boundary

2D parameter domain (u,v)

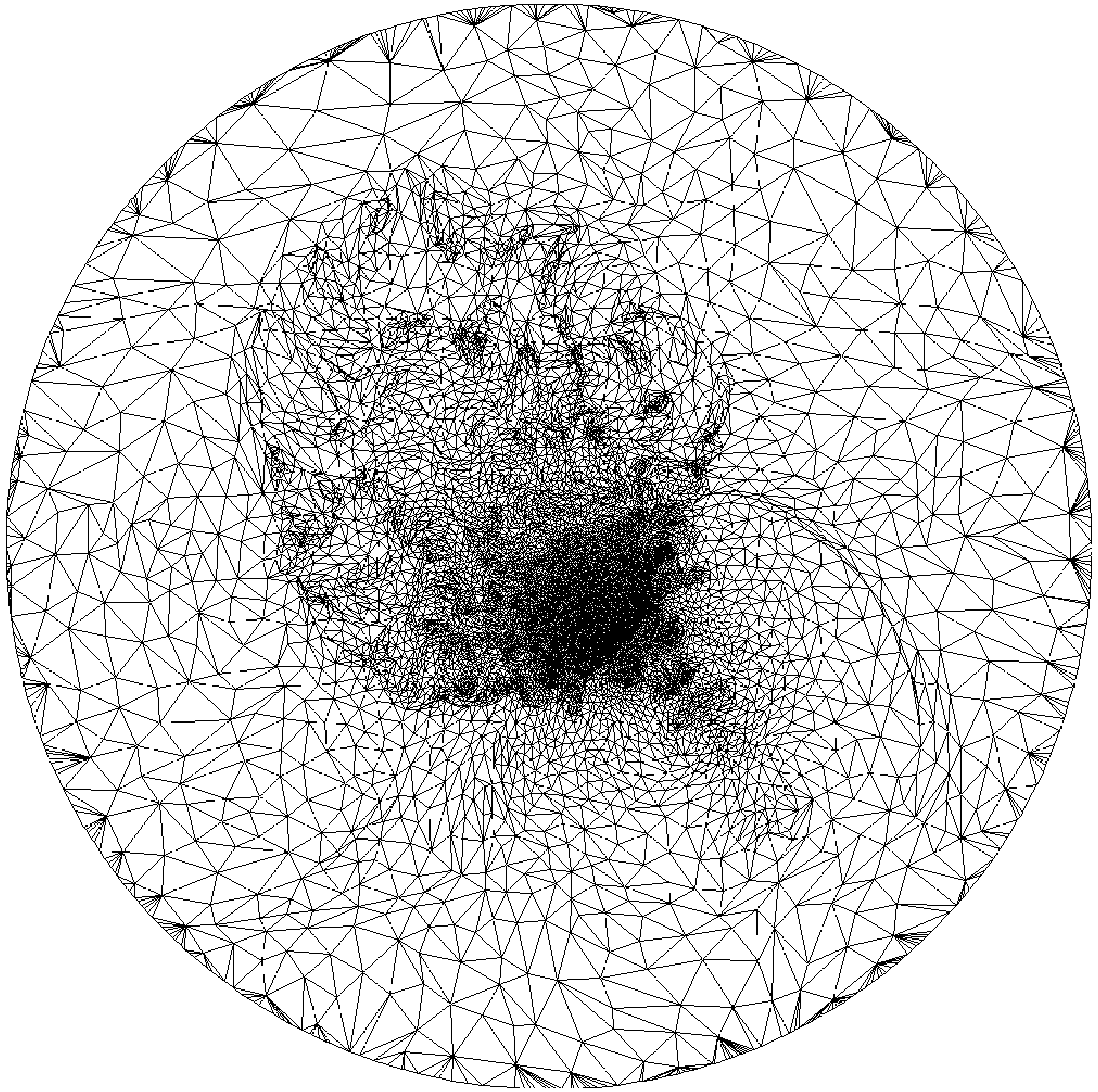


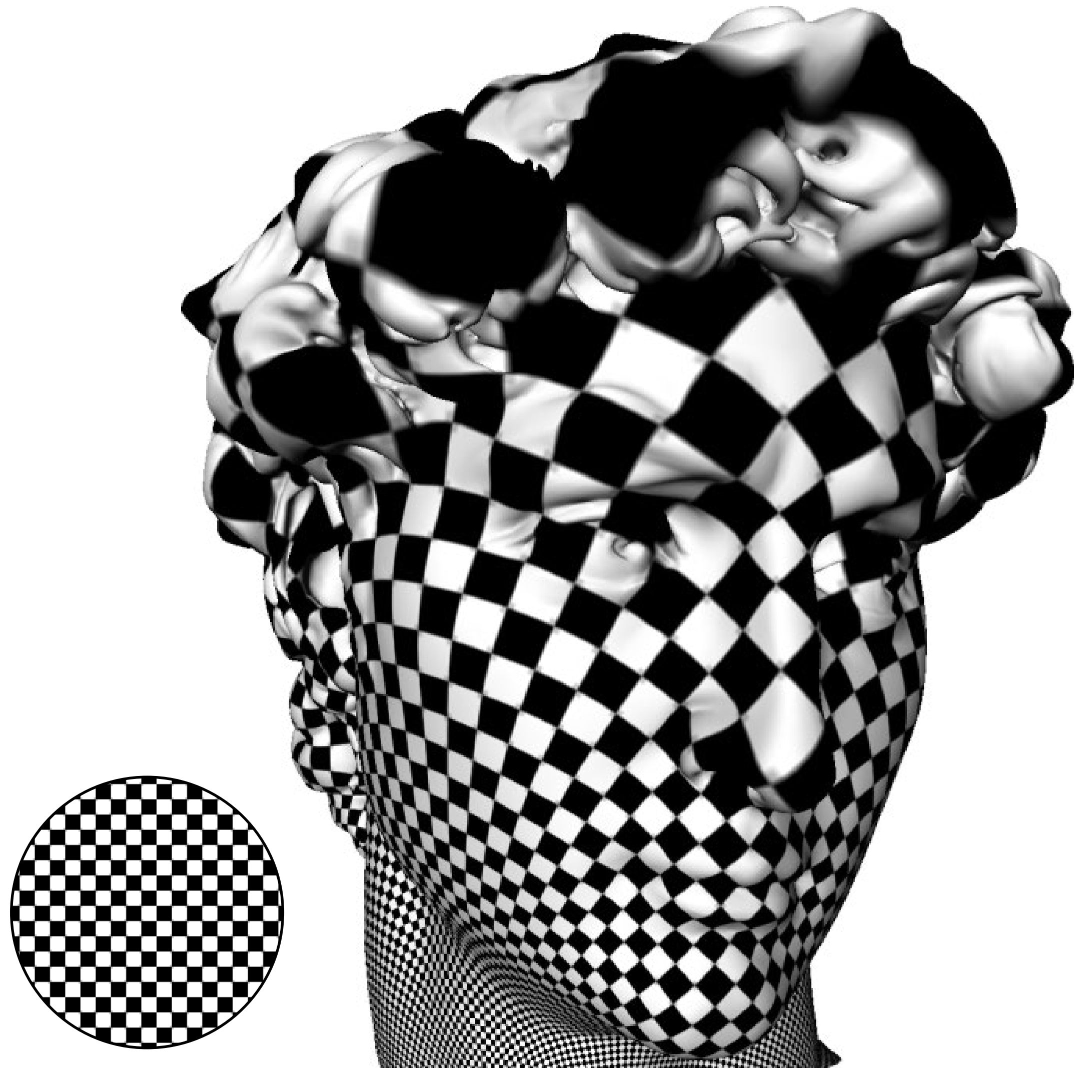
boundary

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

After a slide by Daniele Panozzo

Creating UV parameterizations

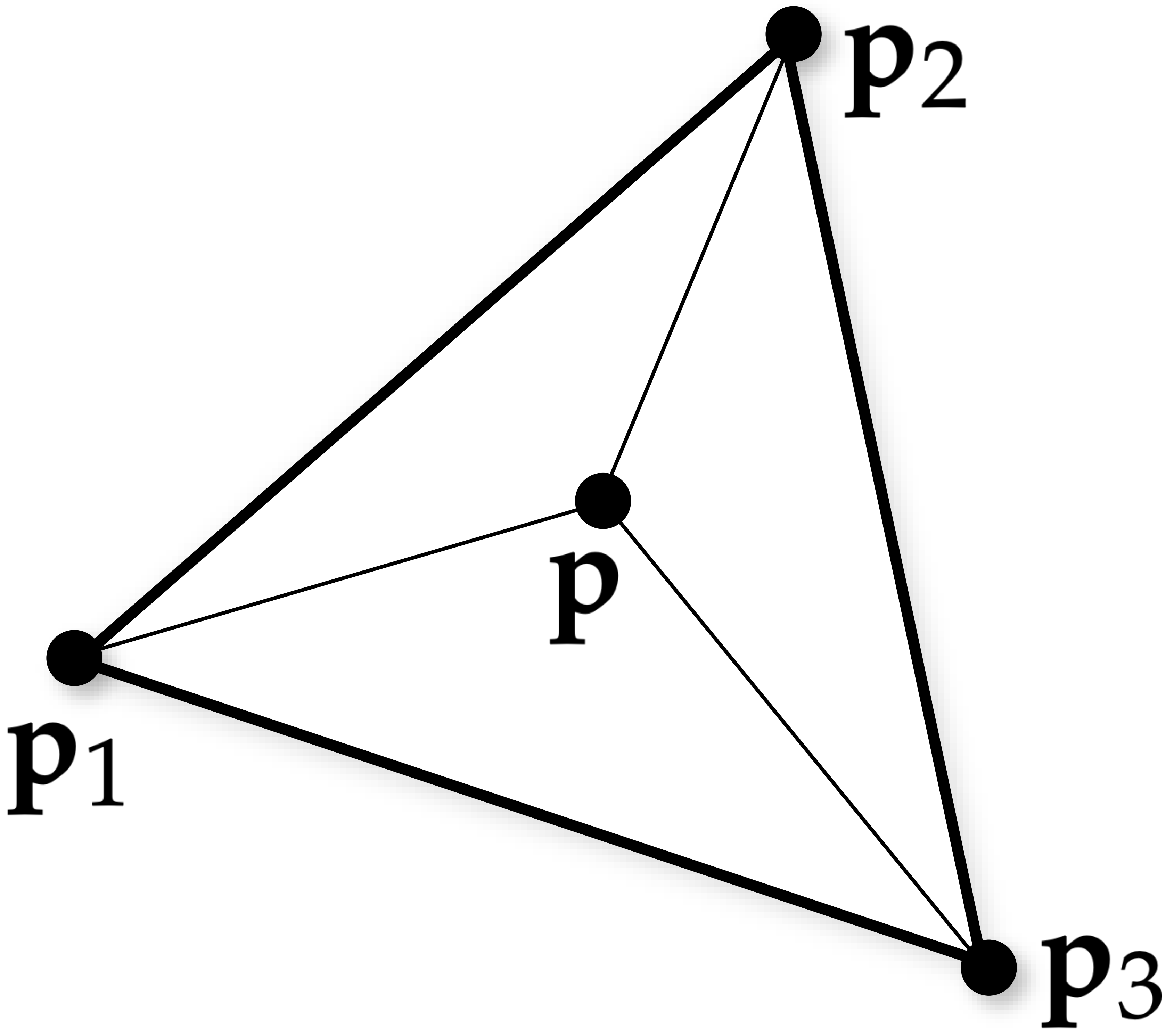




After a slide by Daniele Panozzo

Barycentric coordinates

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



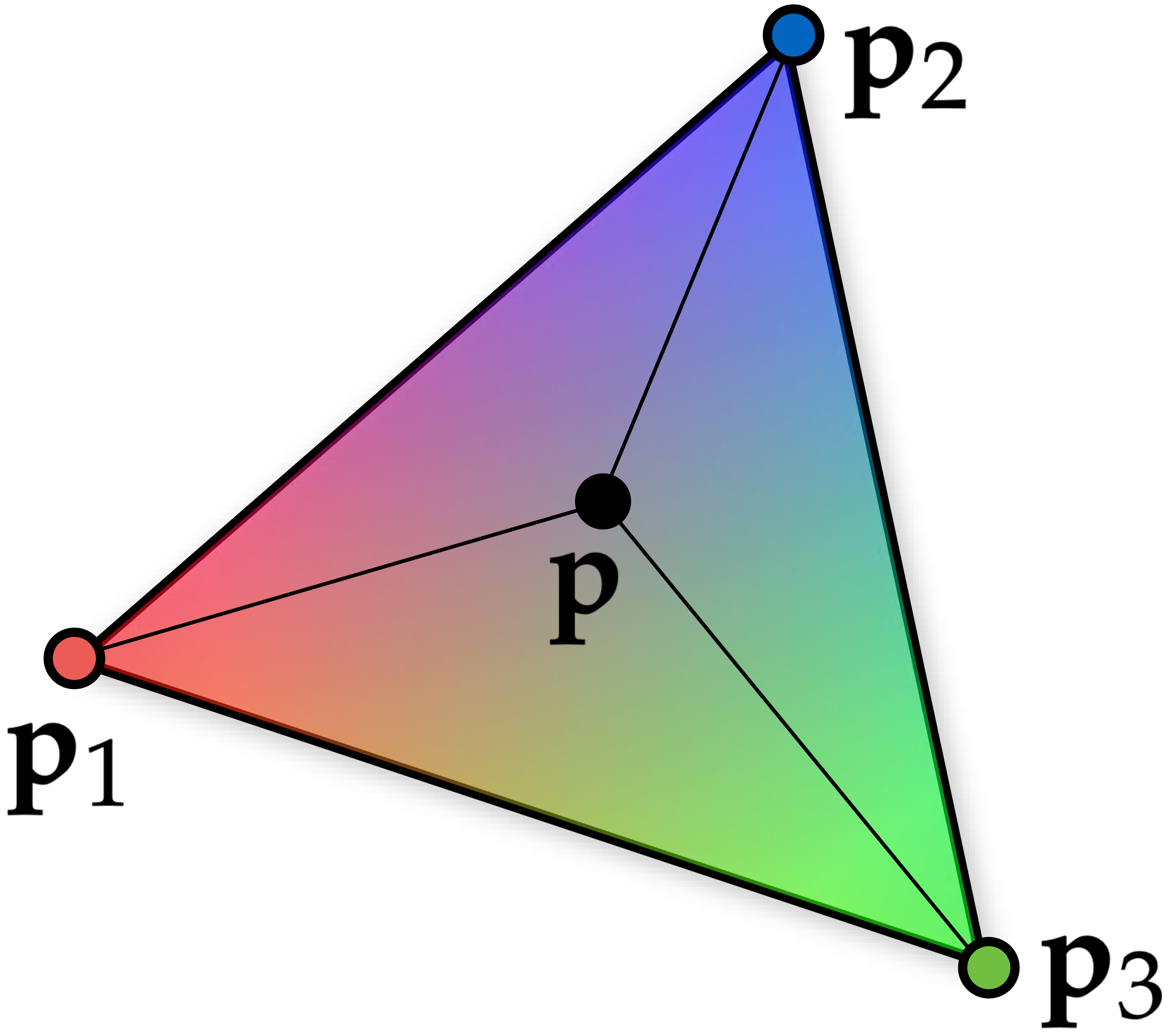
Can use this eqn. to interpolate any vertex quantity across triangle!

Barycentric coordinates

Barycentric interpolation:

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

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

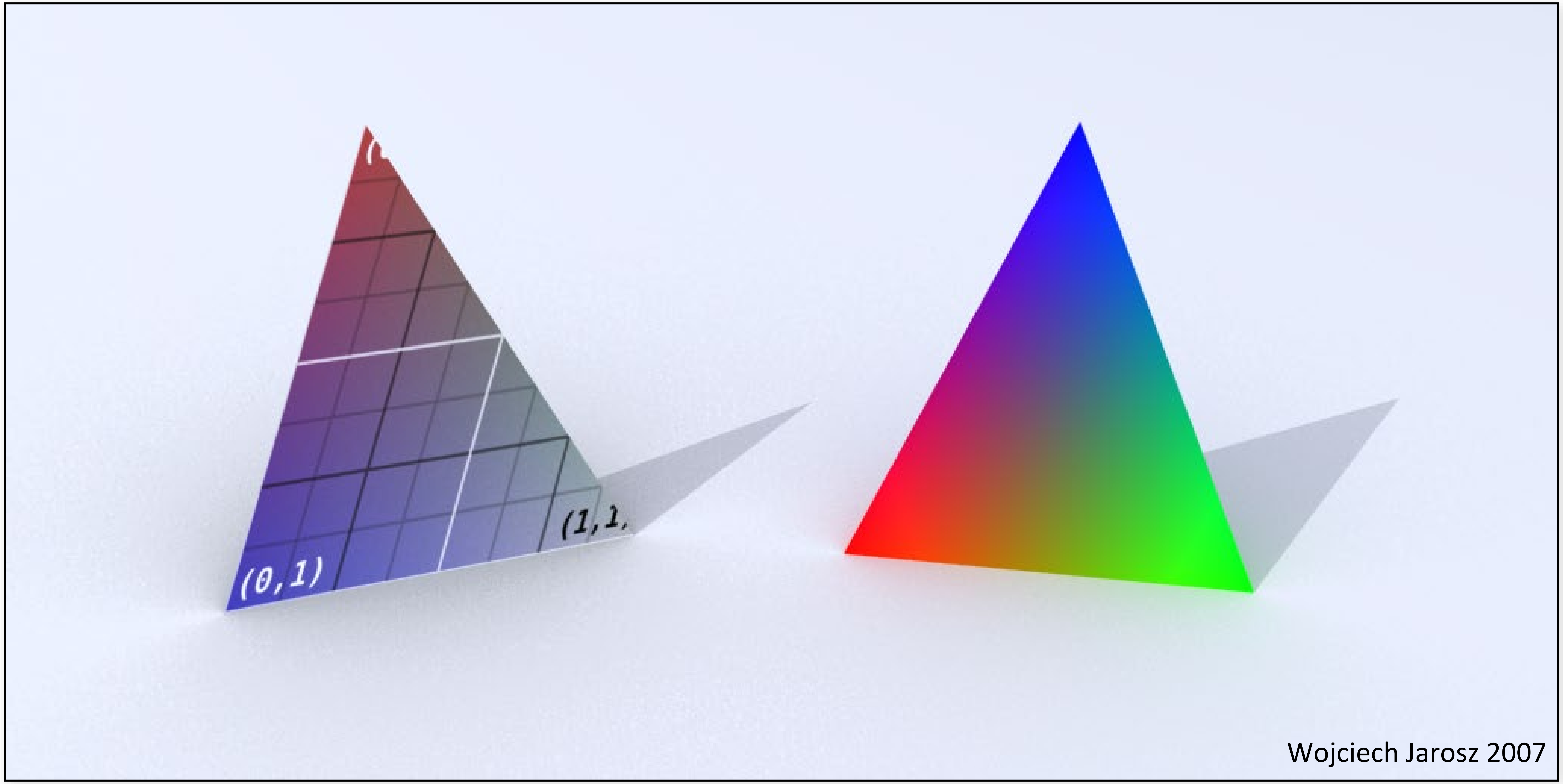


Can use this eqn. to interpolate any vertex quantity across triangle!

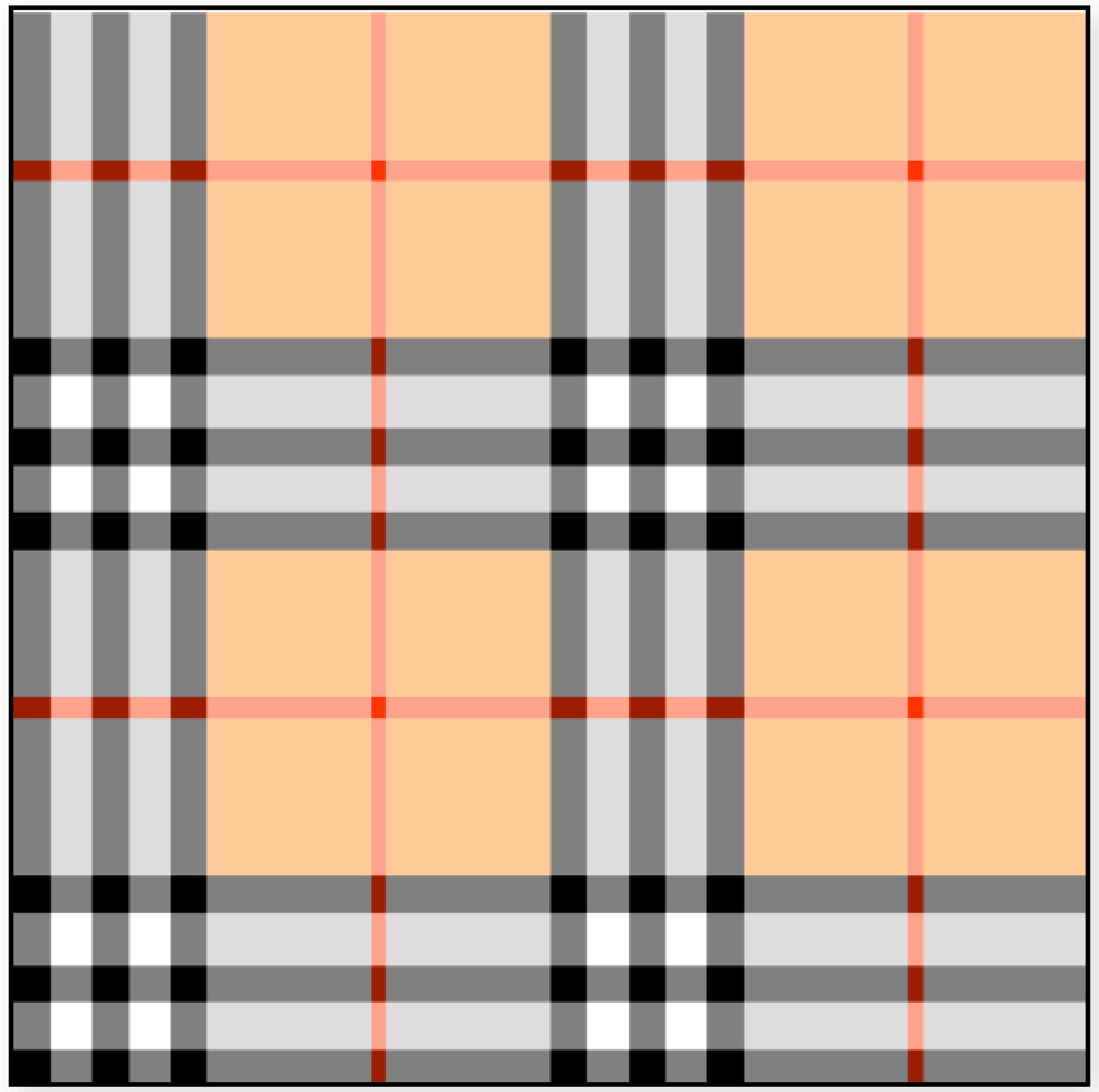
UV texturing

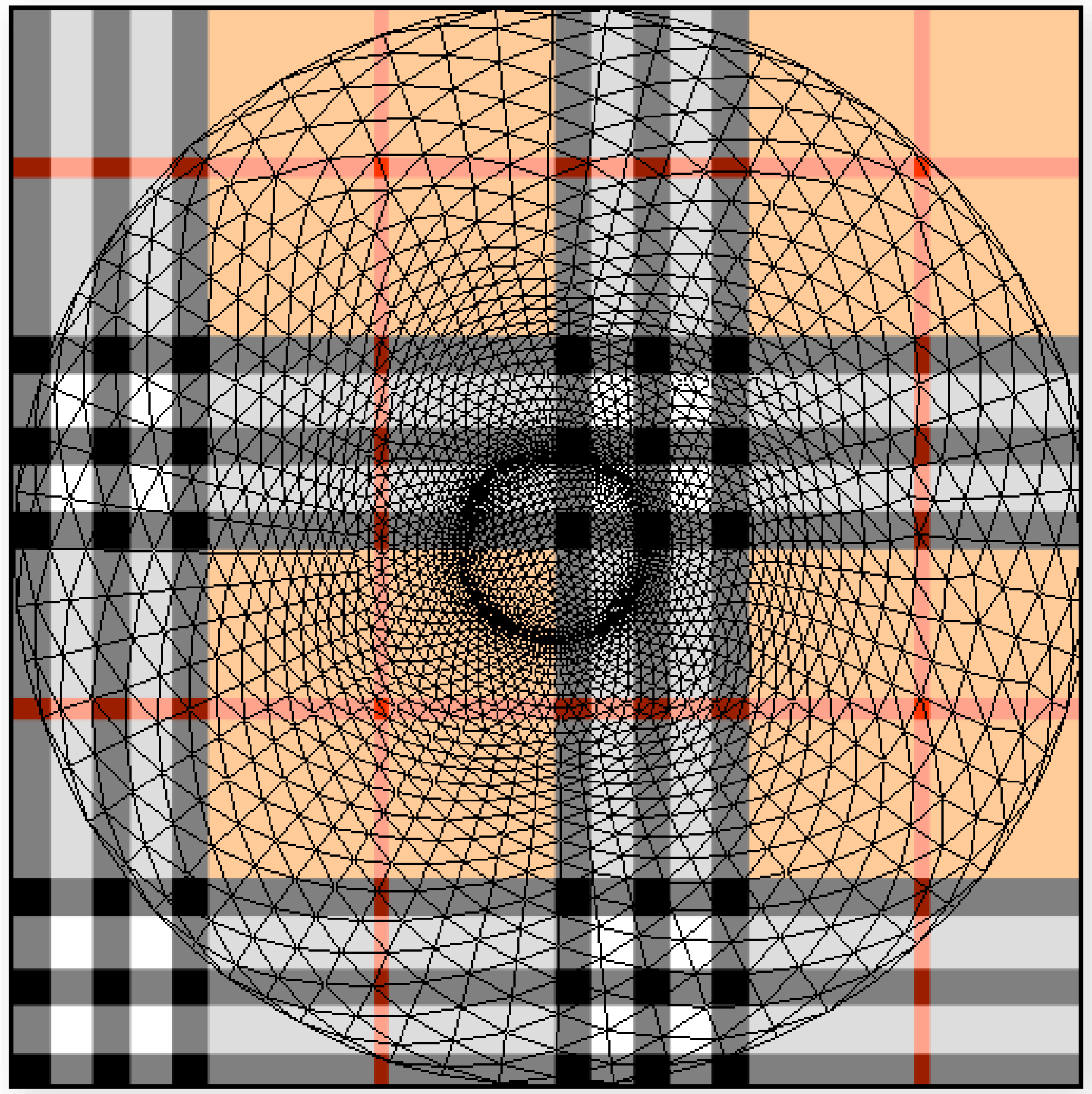
$$(u,v) = \alpha(u_1,v_1) + \beta(u_2,v_2) + \gamma(u_3,v_3)$$

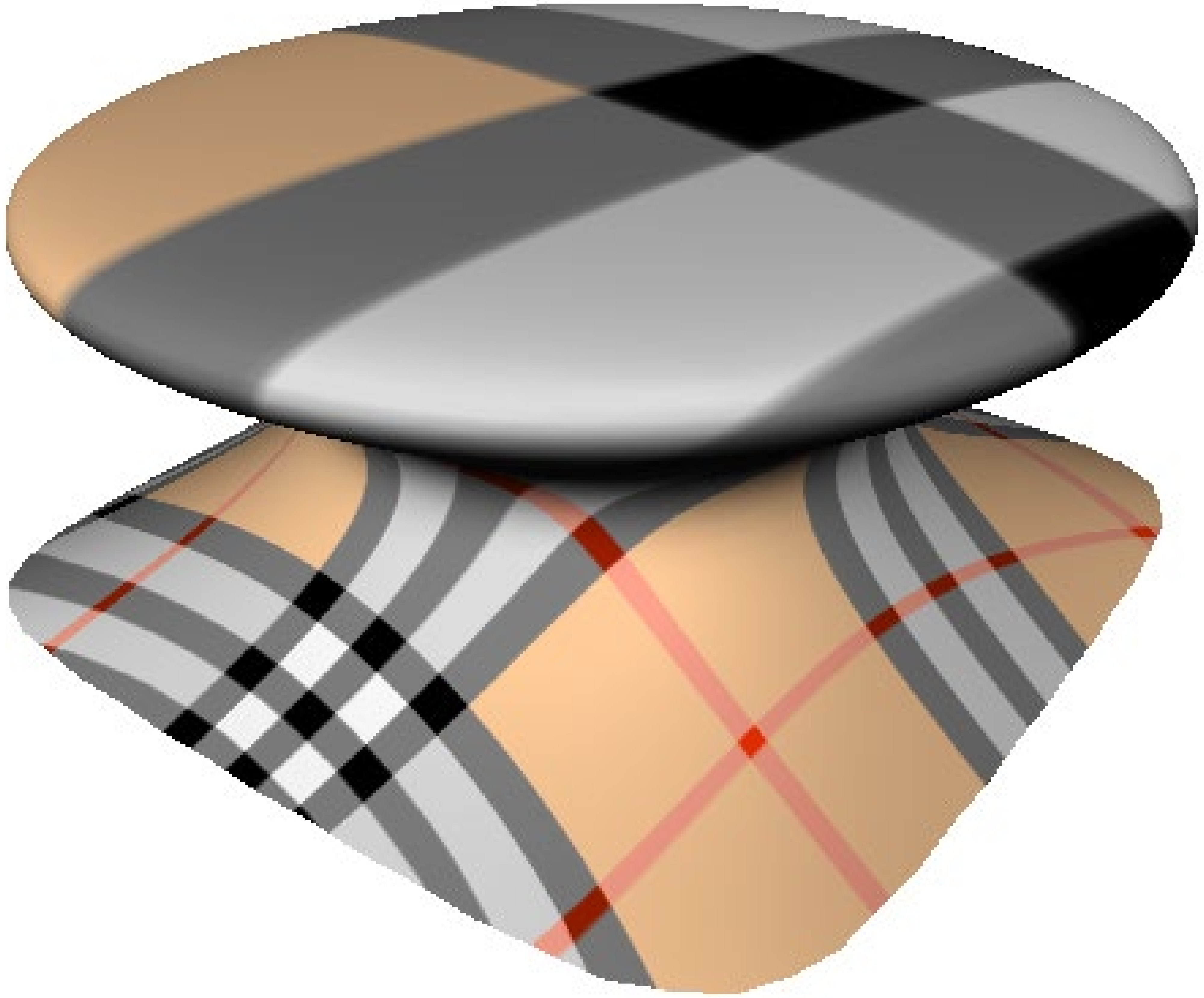
UV texturing



Area distortion vs. angle distortion



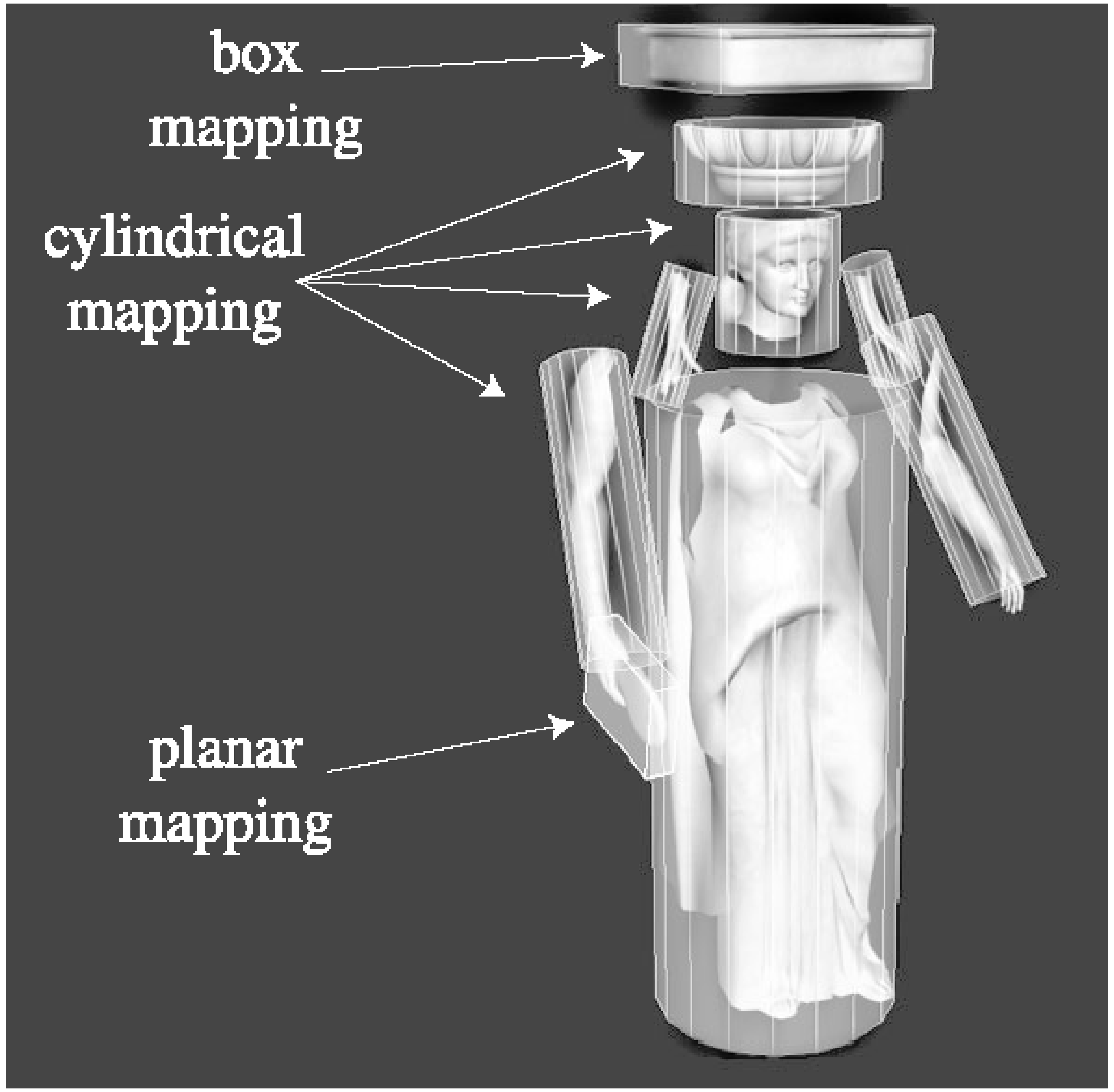




After a slide by Daniele Panozzo

Creating UV parameterizations

Can compute vertex UVs using projections



[Moller & Haines 2002]

Creating UV parameterizations

"Atlas" - break up model into single texture

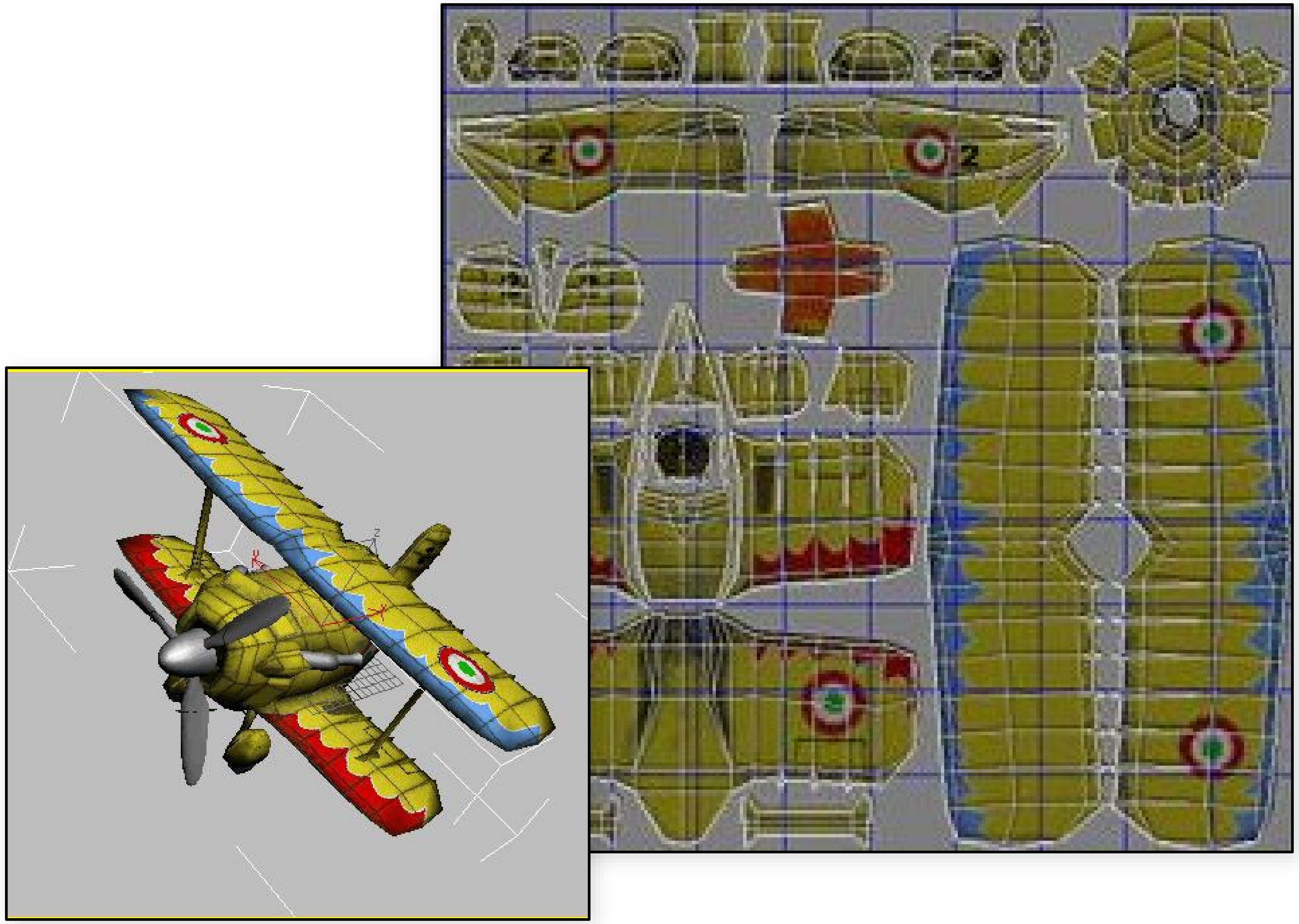


Image from Vallet and Levy, techreport INRIA

Texture lookup

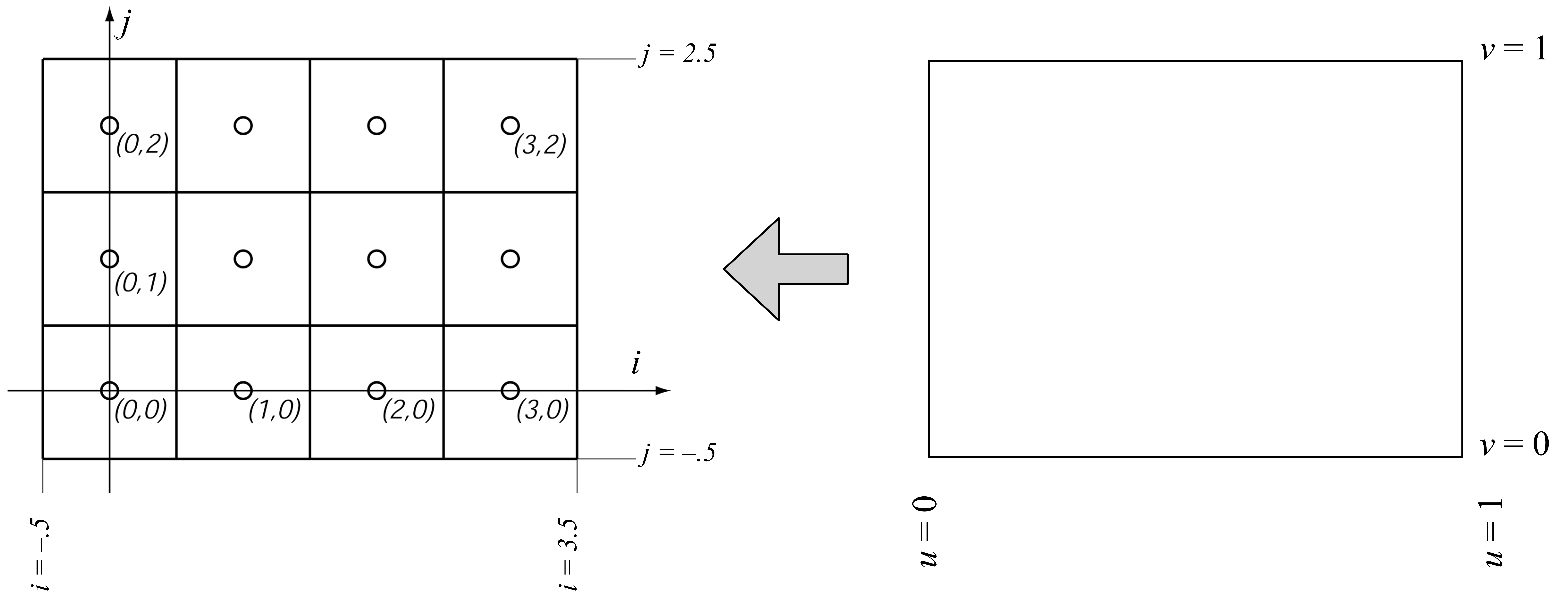
Texture lookups and wrapping

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

After a slide by Steve Marschner

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

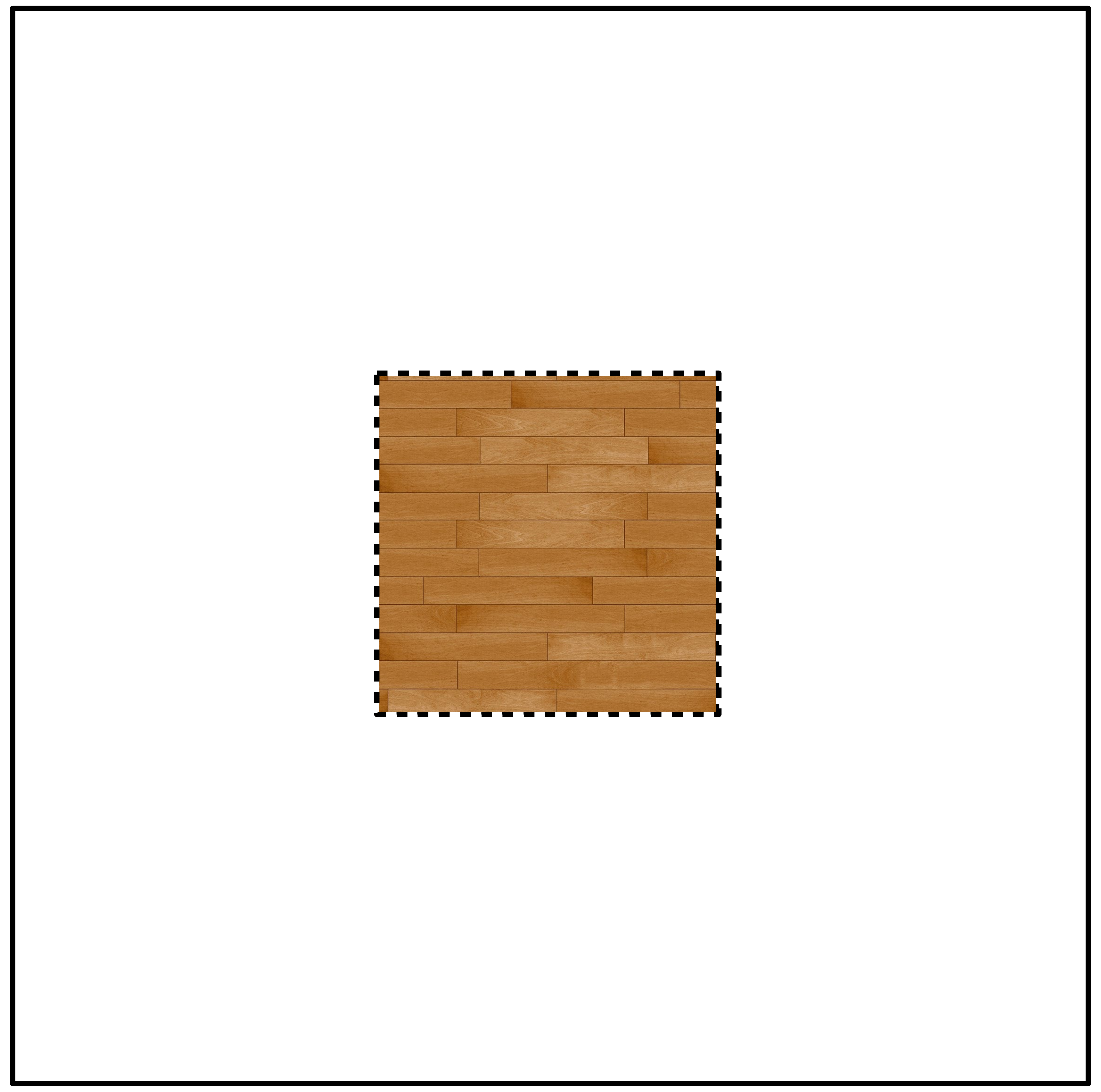
- simplest: round to nearest (nearest neighbor lookup)
- various ways to be smarter and get smoother results

Texture lookups and wrapping

What if `i` and `j` are out of range?

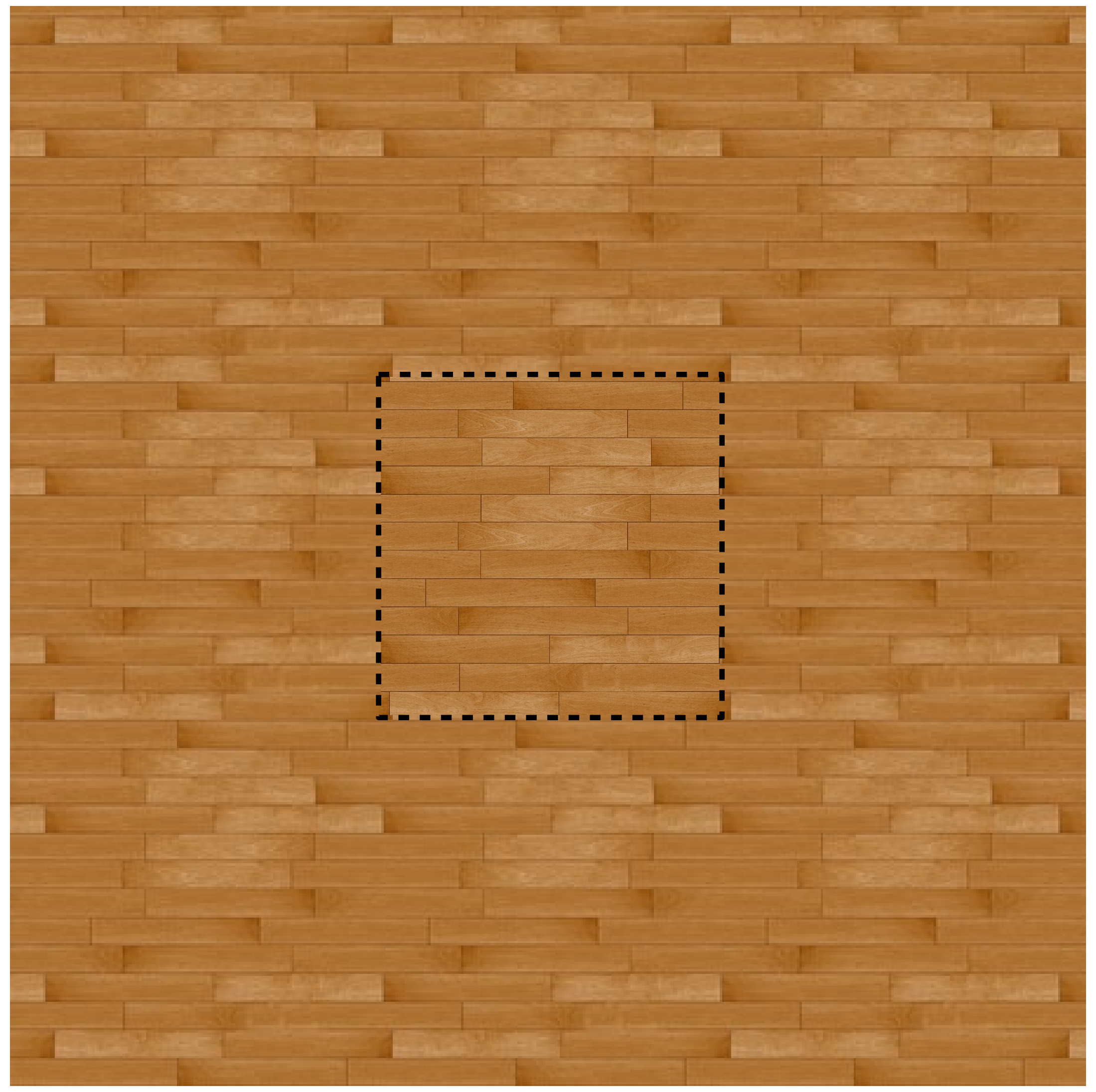
Option 1, clamp: take the nearest pixel that is in the image

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



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

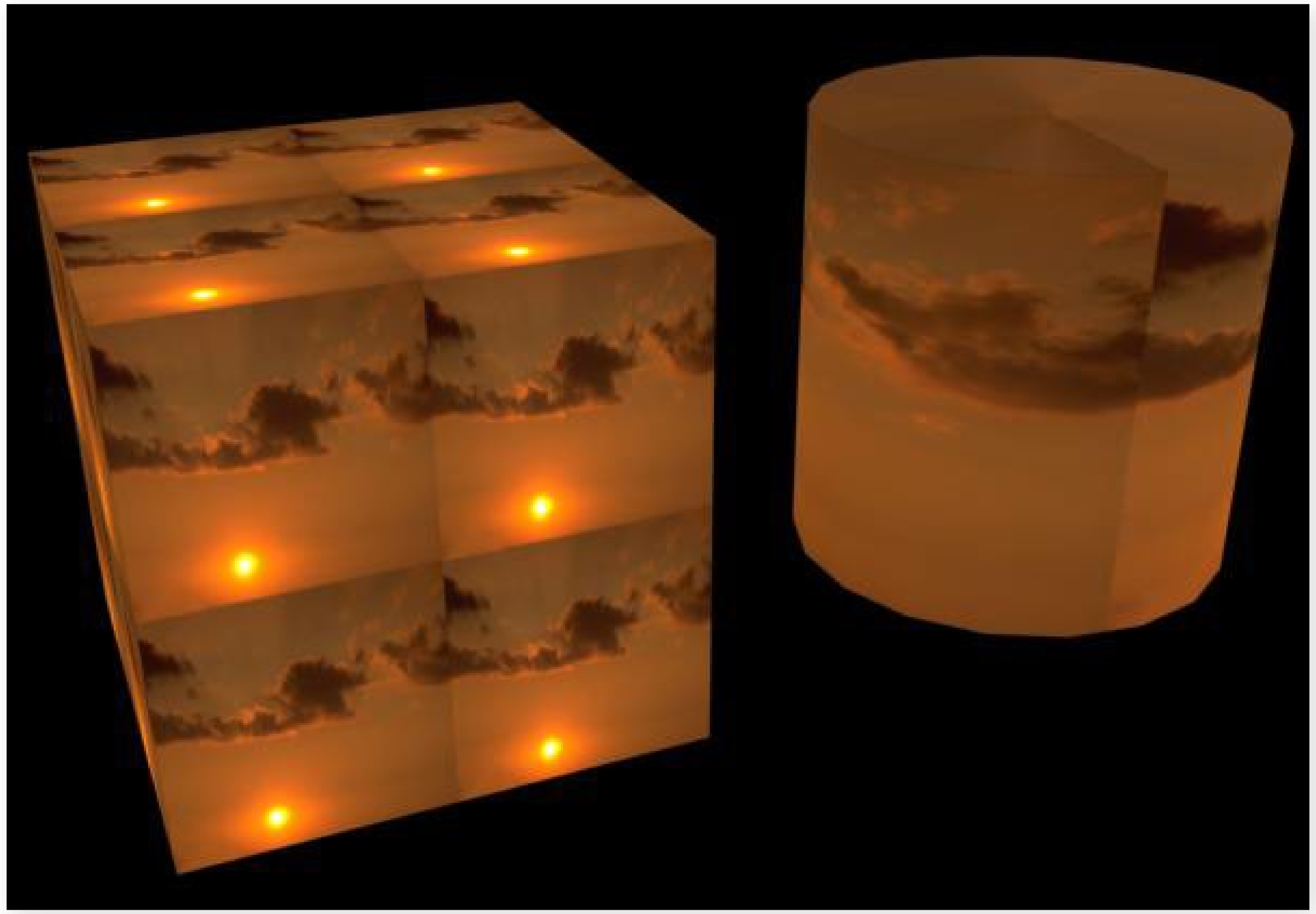
$$i_{\text{pixel}} = \text{remainder}(i_{\text{lookup}}, n_x)$$



Texture mapping artifacts

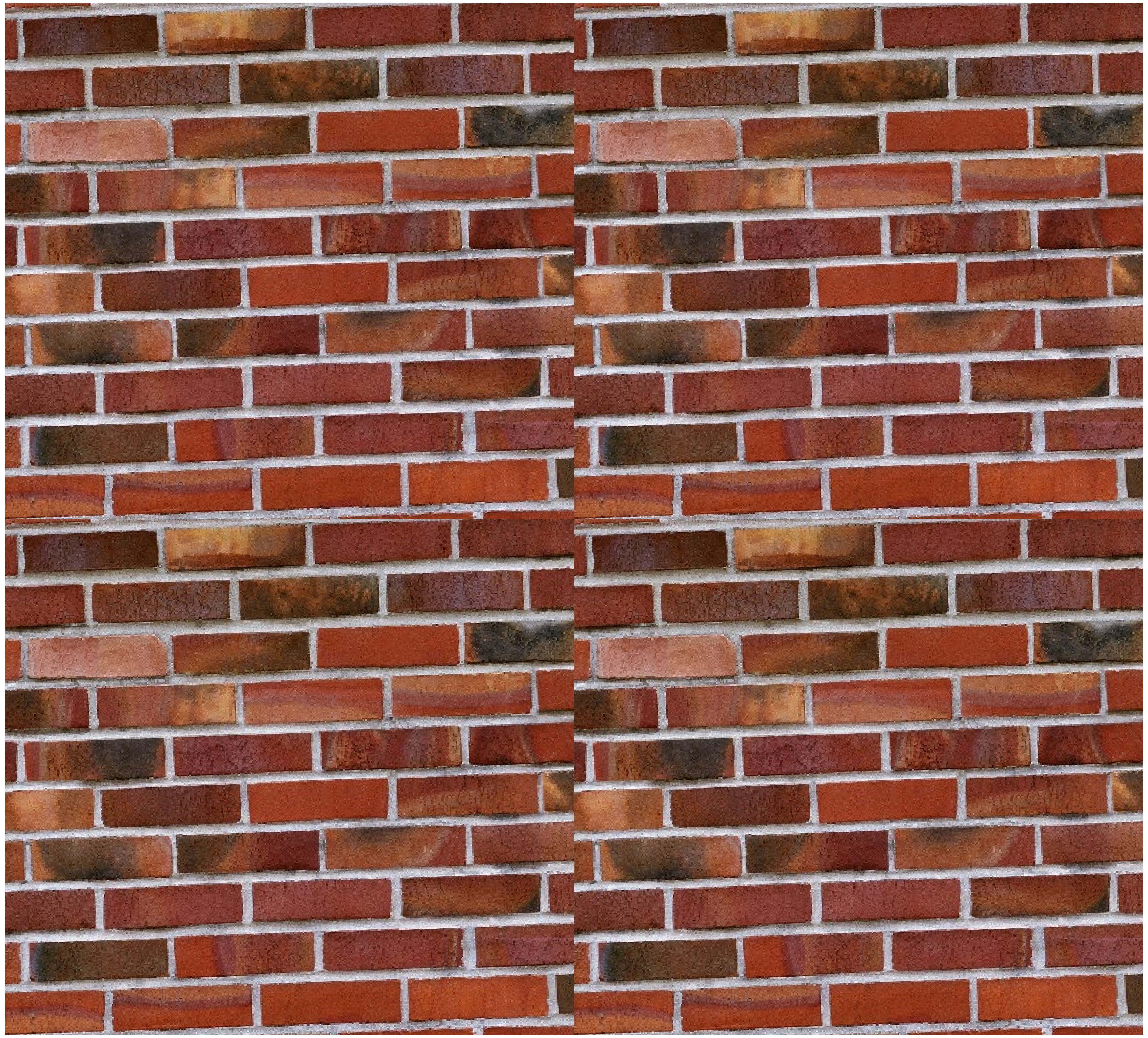
Tiling textures might introduce seams

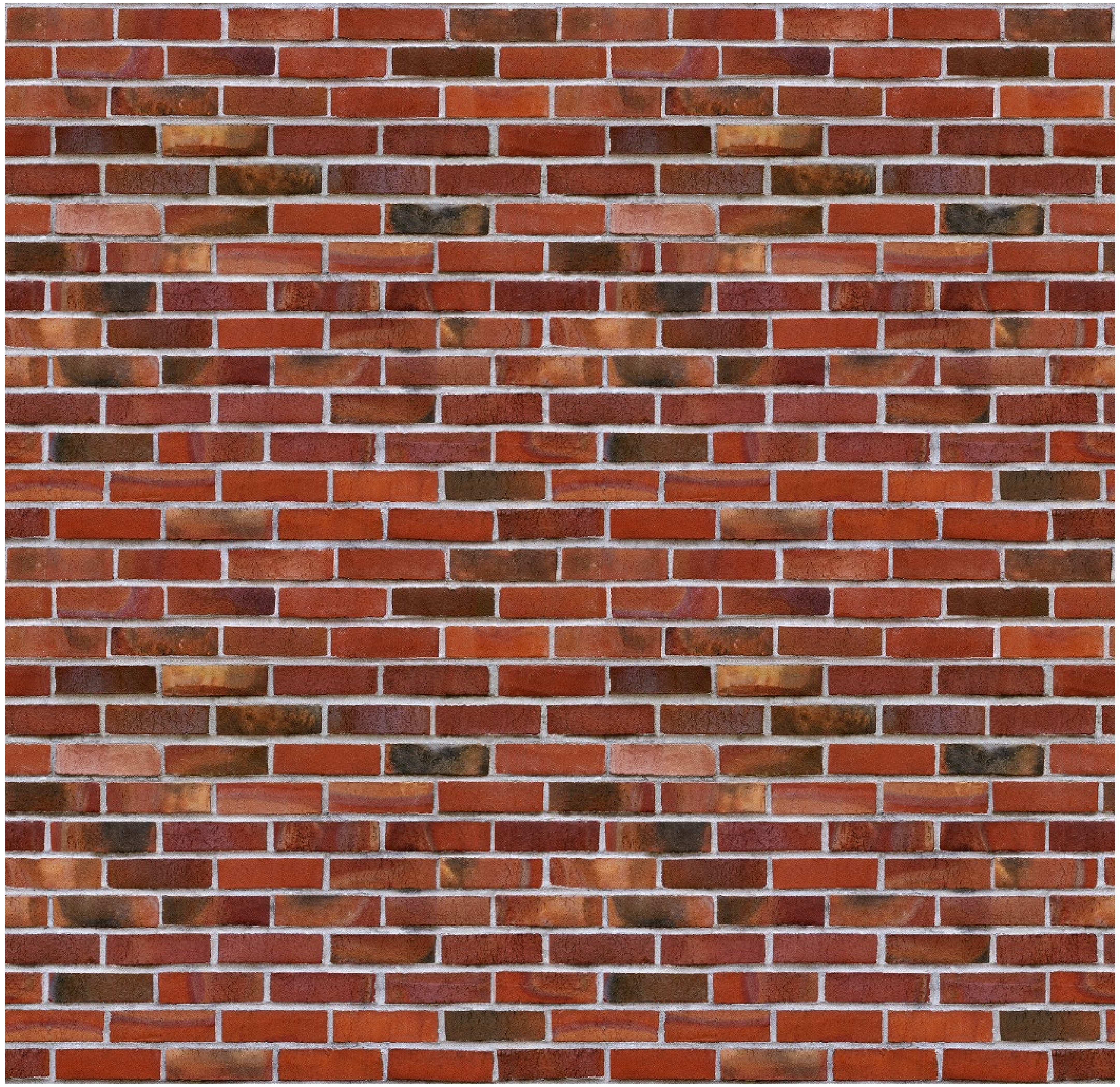
- discontinuities in the mapping function
- change textures to be "tileable" when possible



After a slide by Fabio Pellacini

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)

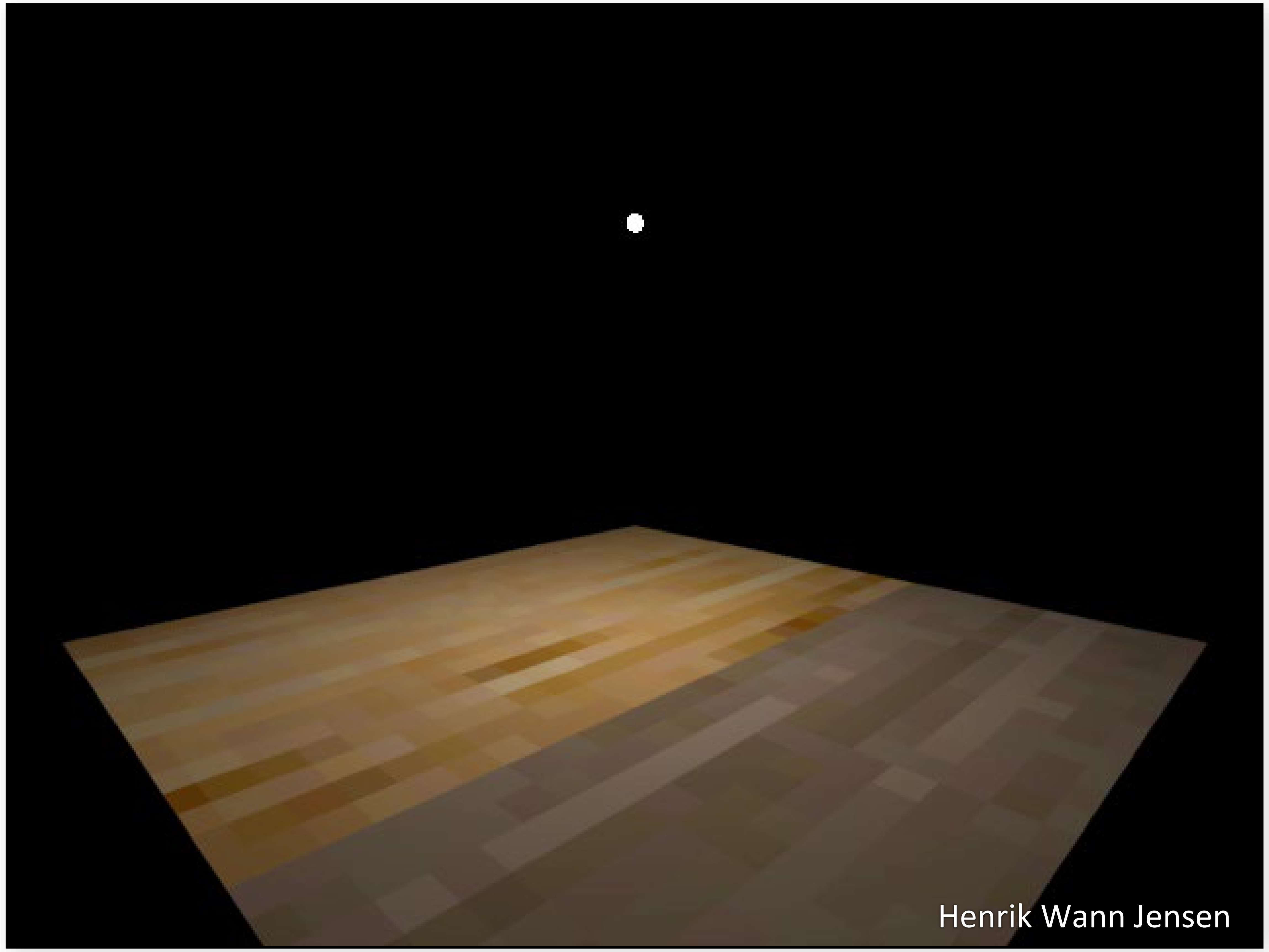
Texture Filtering - Magnification

Nearest Neighbor

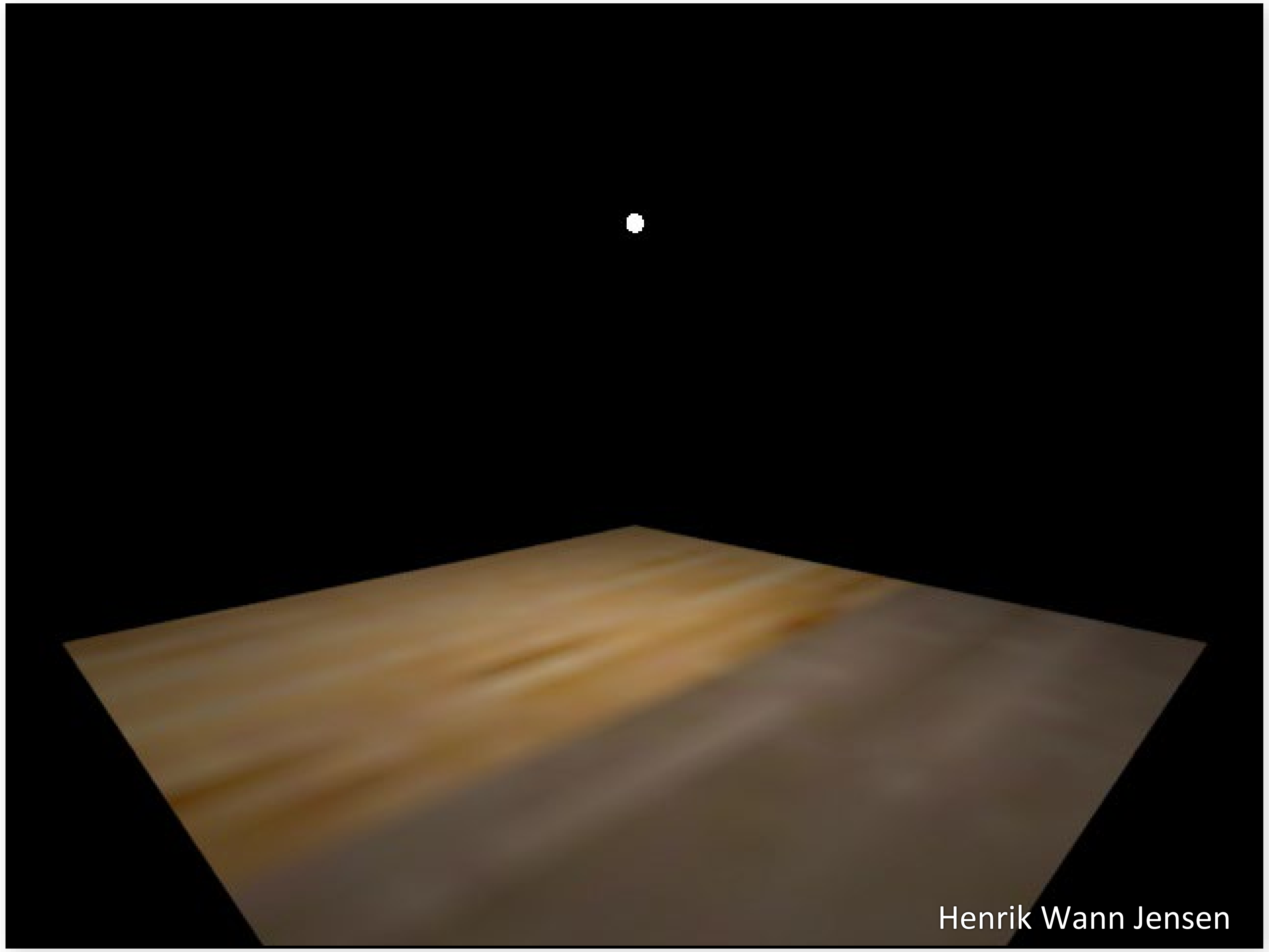
Bilinear

Bicubic

Texture Filtering - Magnification



Nearest Neighbor

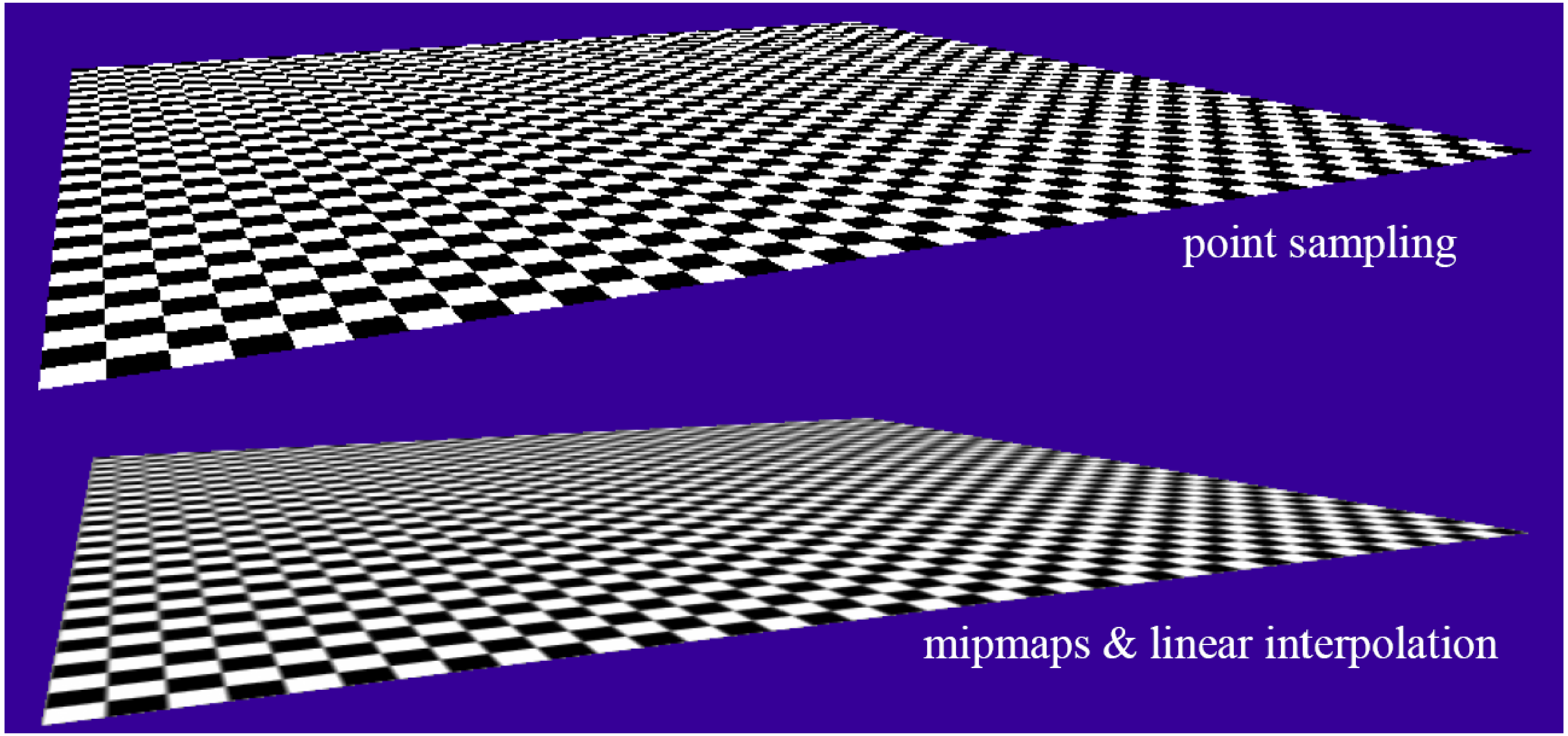


Bilinear

Texture minification (aliasing)

Point-sampling introduces artifacts (aliasing)

- need average of texture within area of a pixel



After a slide by Fabio Pellacini

Texture minification

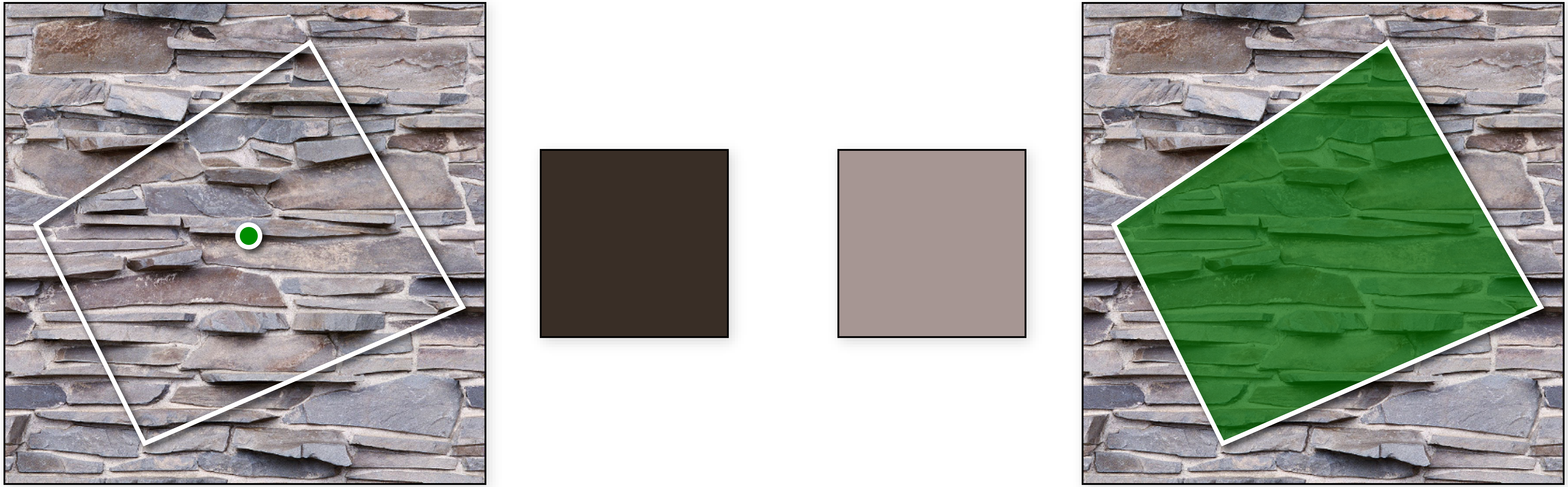
In ray tracing, in theory, you're already sending many rays randomly through area of pixel

Minification artifacts will go away, eventually

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

- mipmapping
- ripmapping
- summed area tables

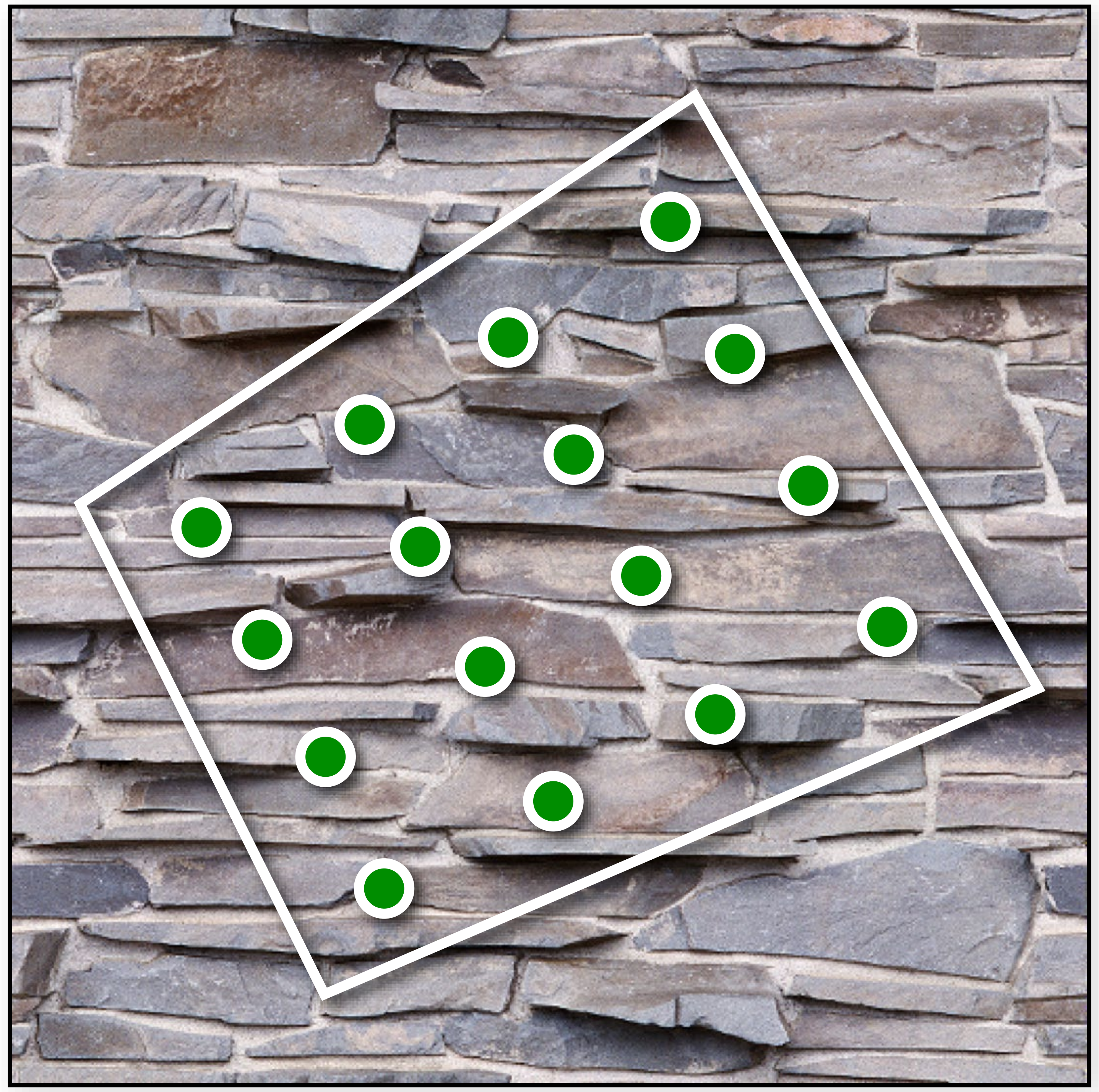
Texture filtering - minification

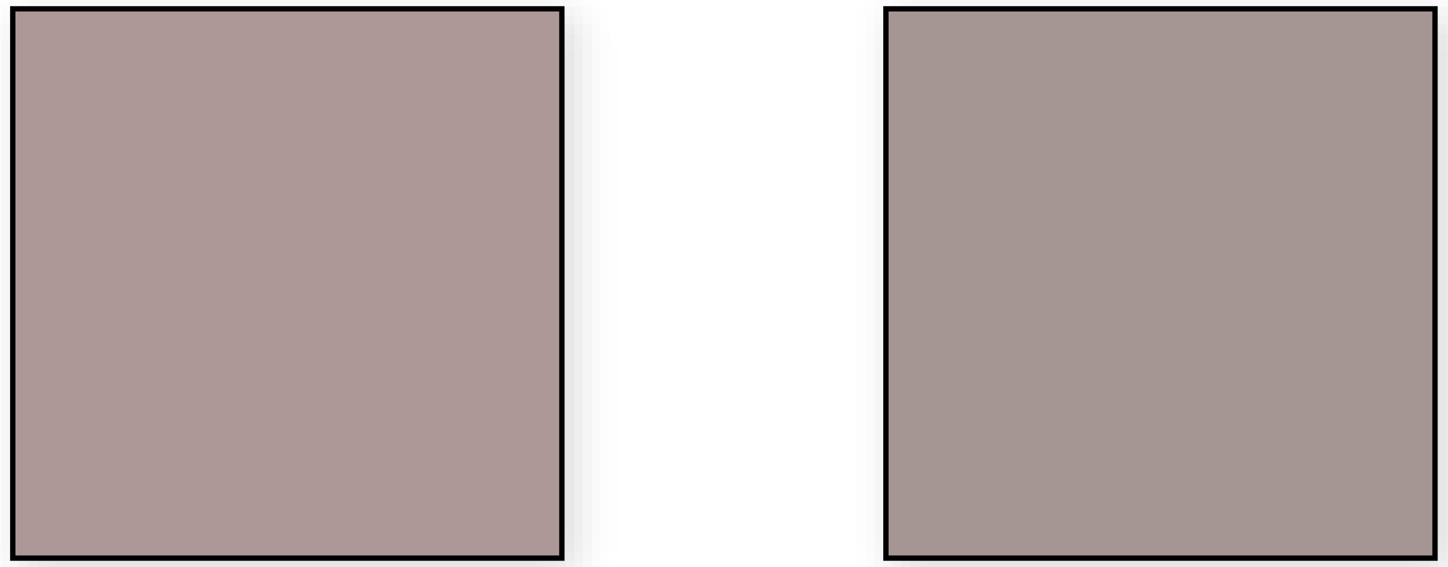


point sample

area average

Texture filtering - minification





4×4 supersampling

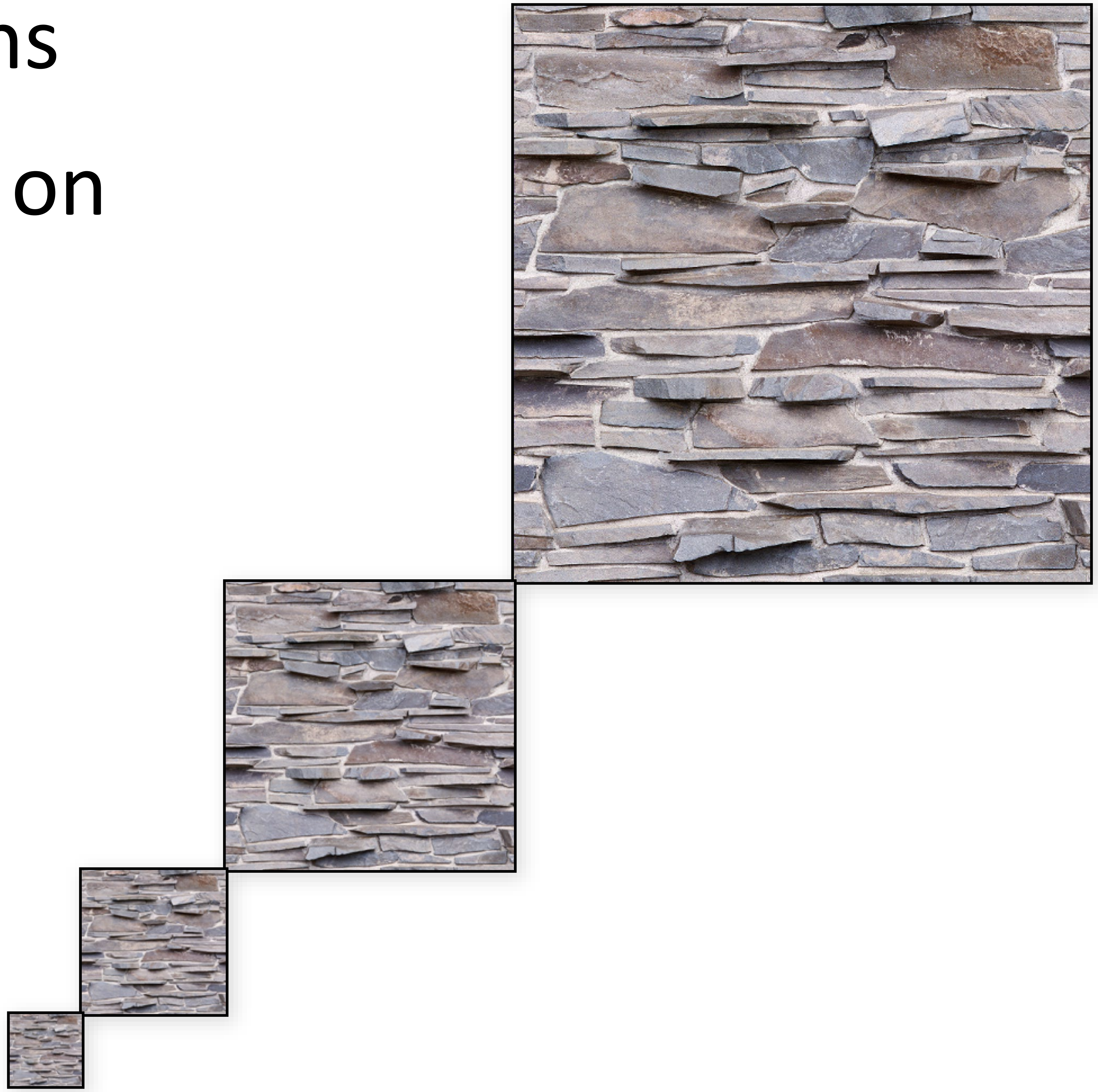
expensive!

area average

Mipmapping

Store textures at different resolutions

Look up in appropriate image based on projected pixel size



Mipmapping

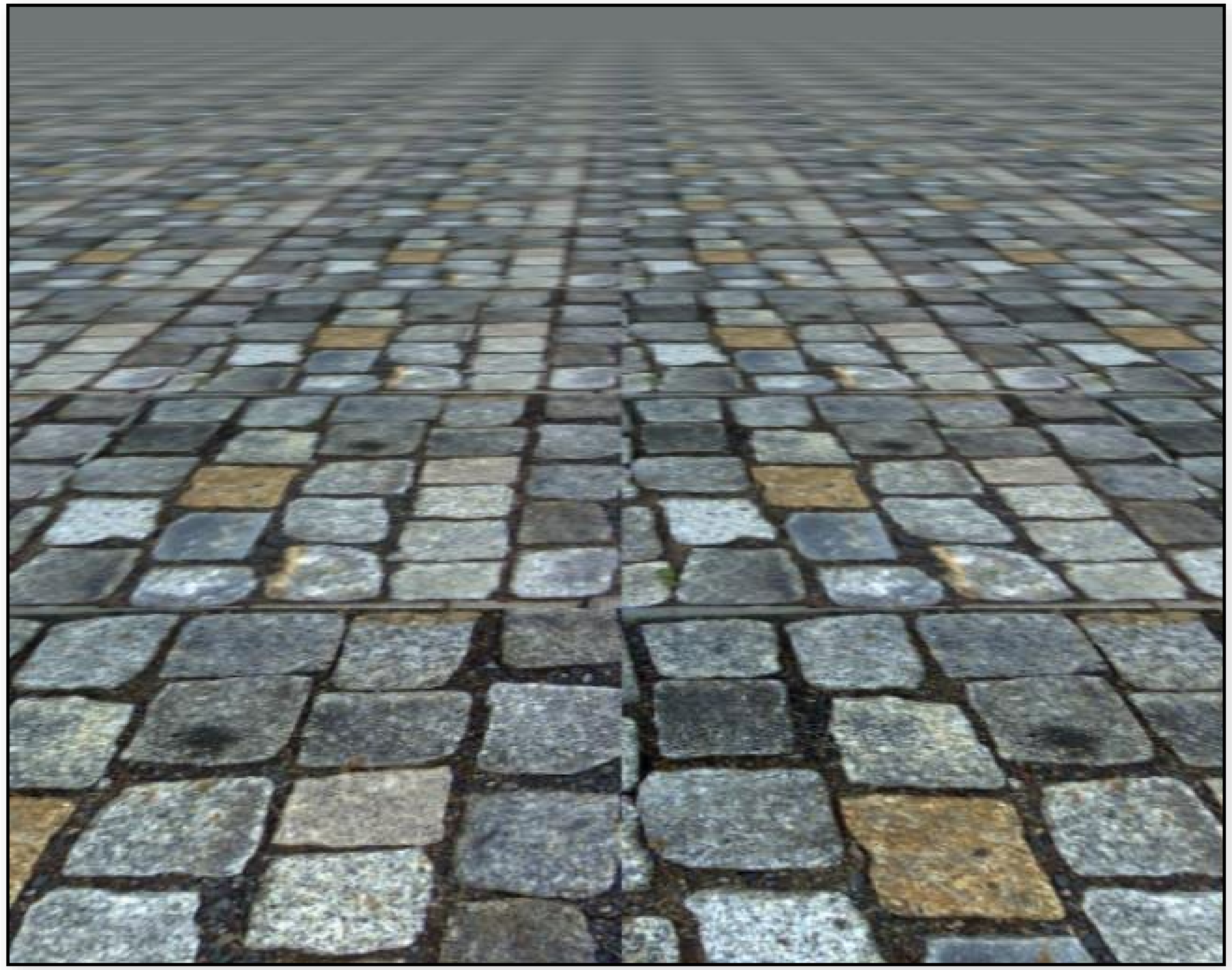
Bilinear: Look up in closest resolution and bilinearly interpolate

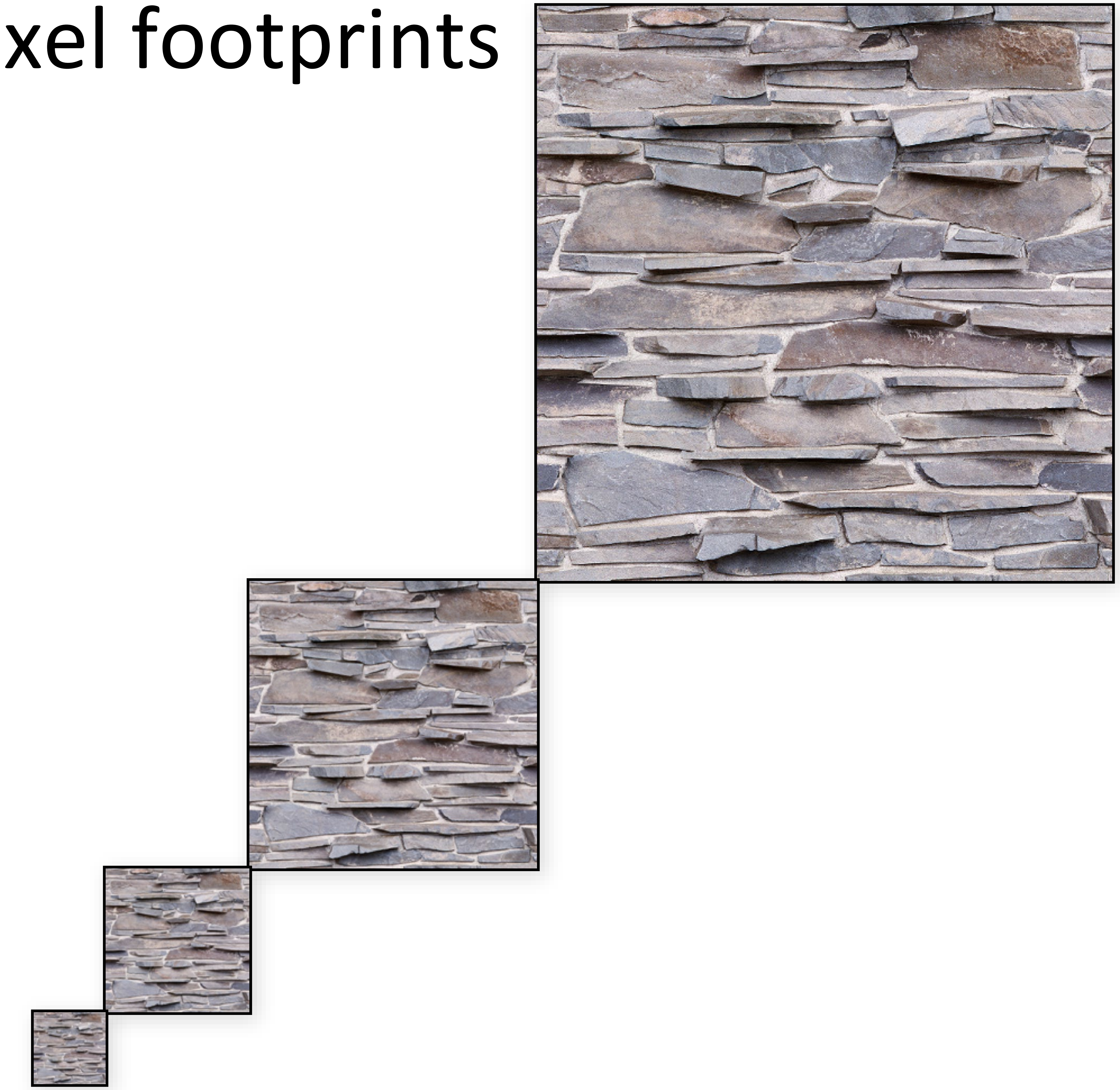
- transition artifacts

Trilinear: linearly interpolate between two closest levels

Mipmapping

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

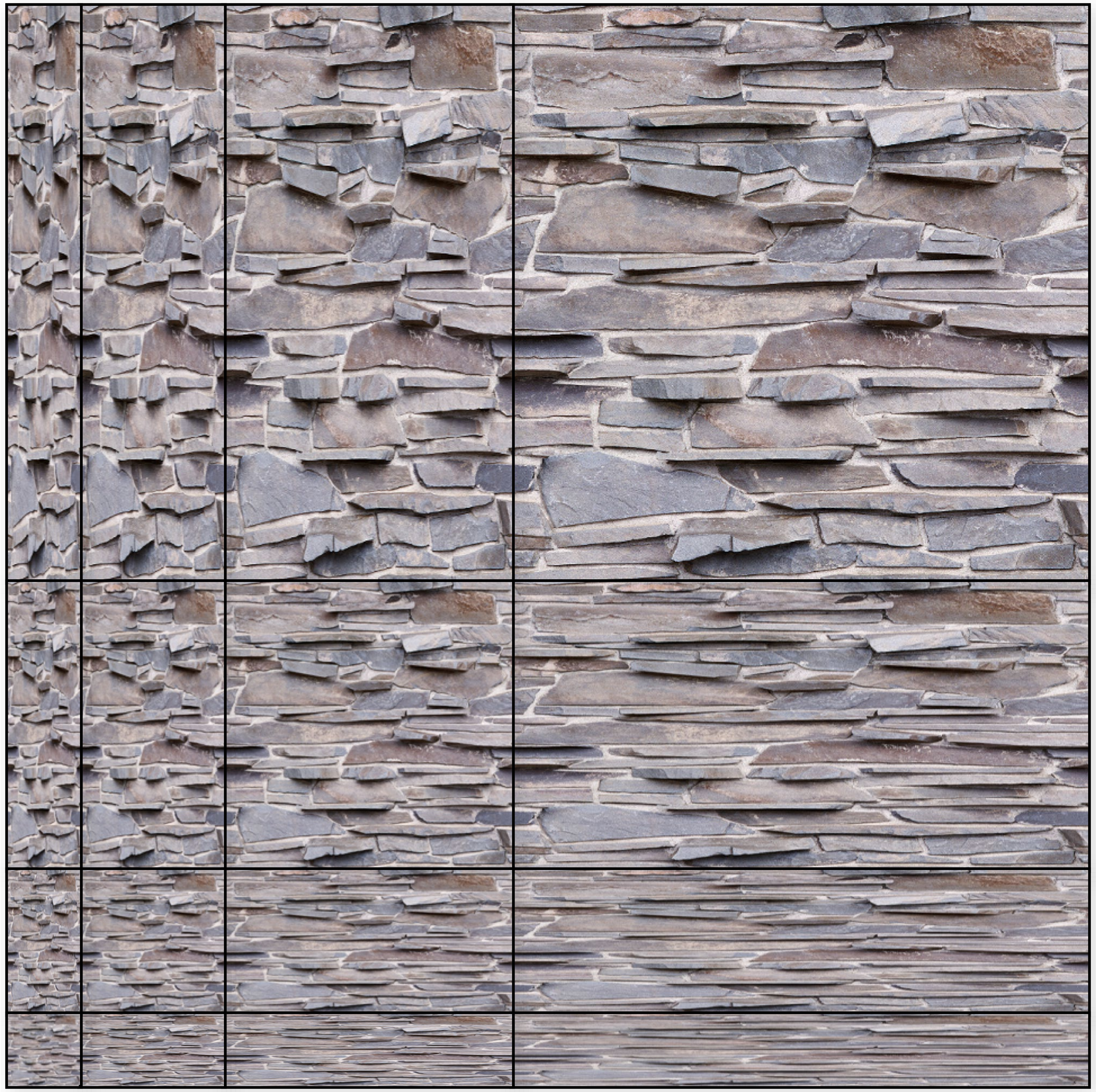




Ripmap

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)

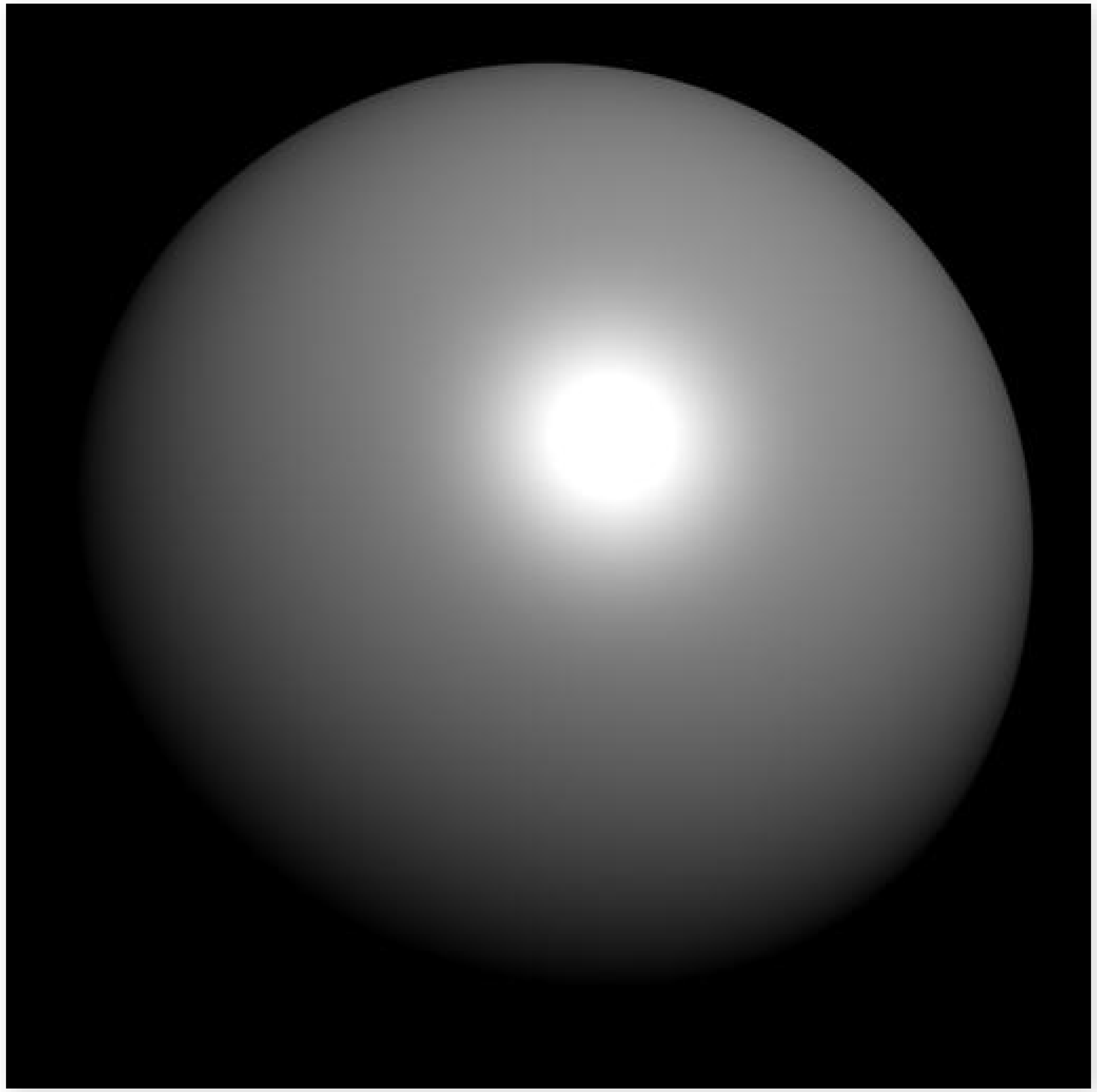
Texturing reflectance properties

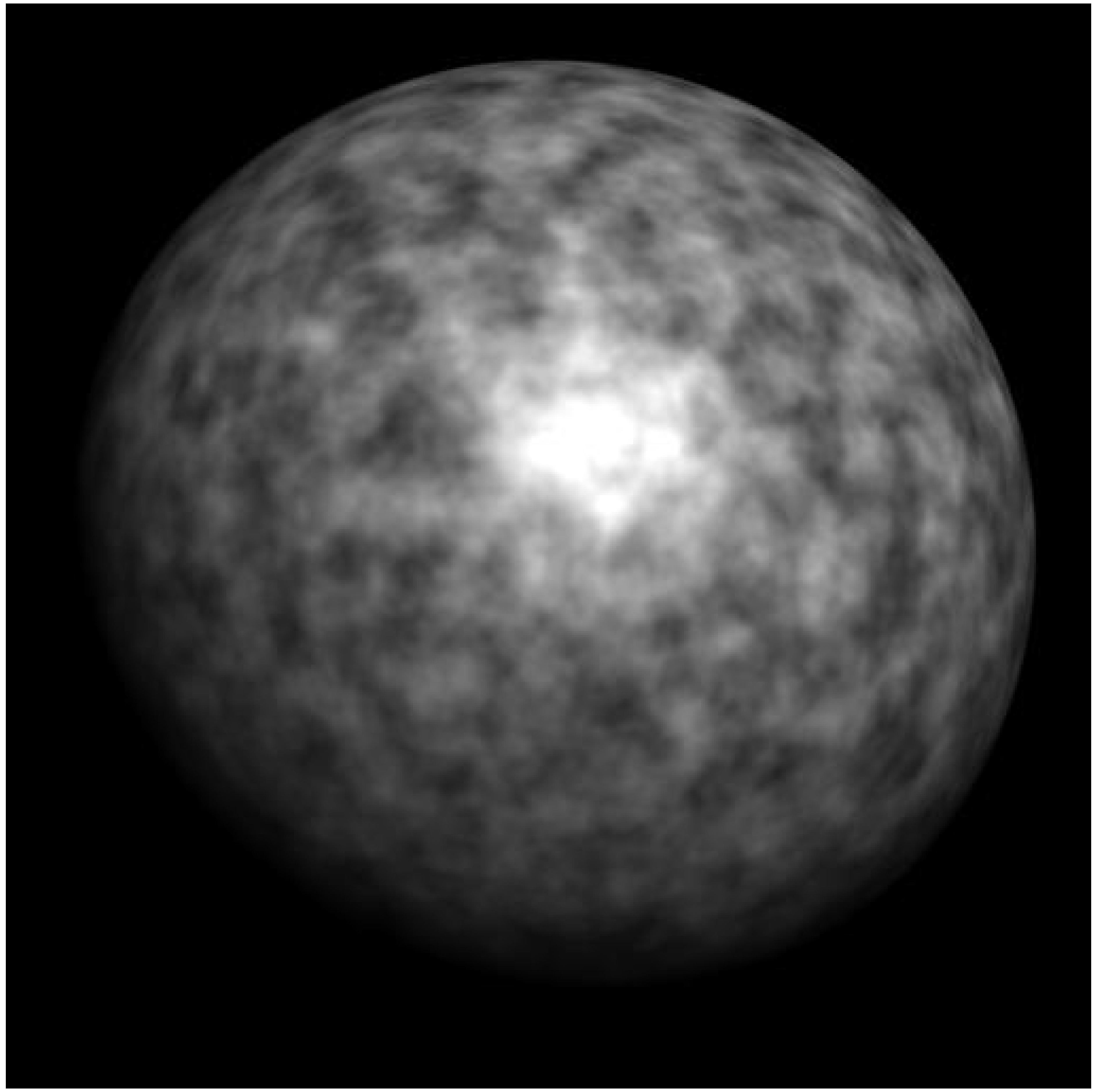
Texture mapping

Surface properties are not the same everywhere

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

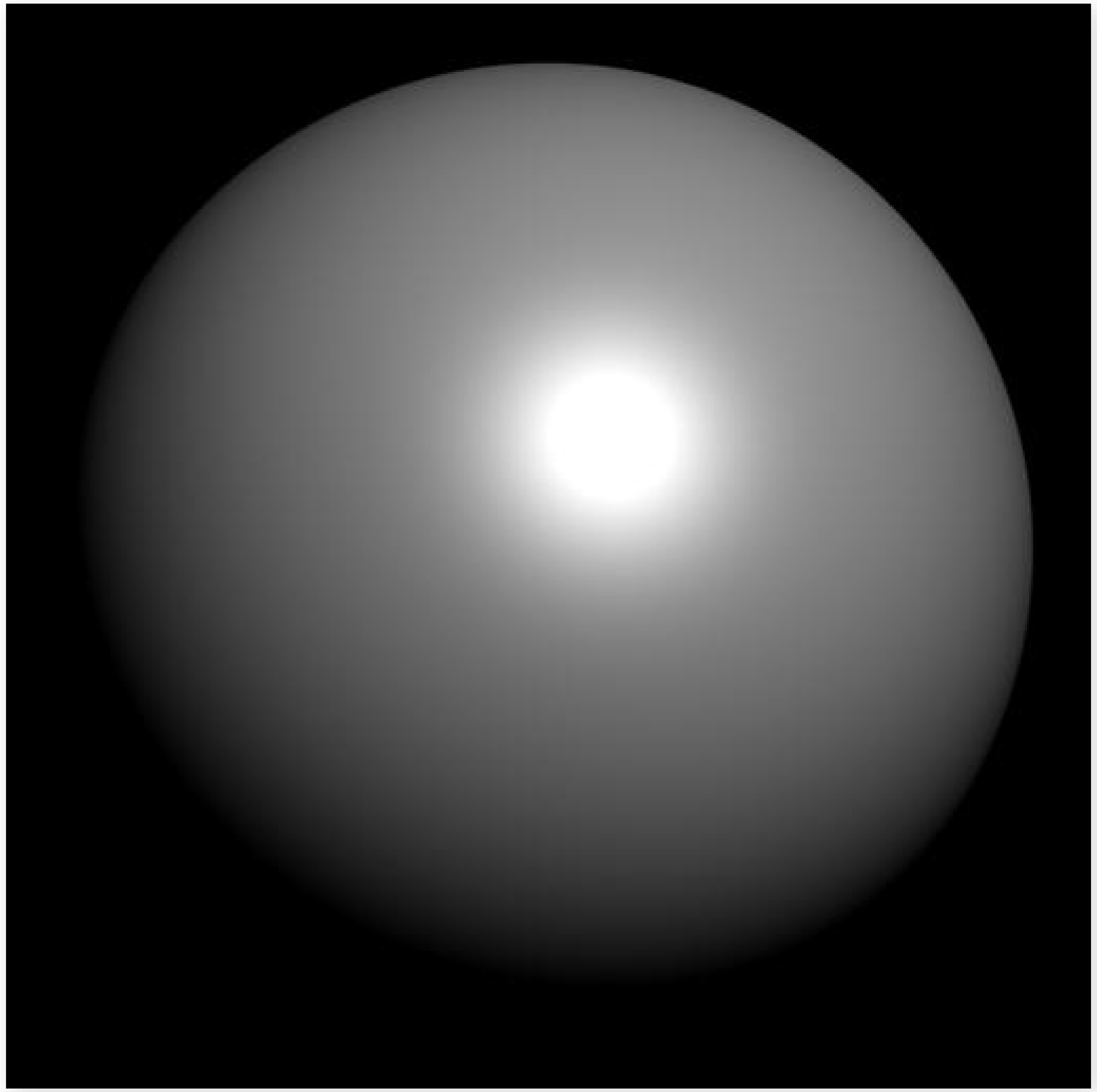
Diffuse coefficient

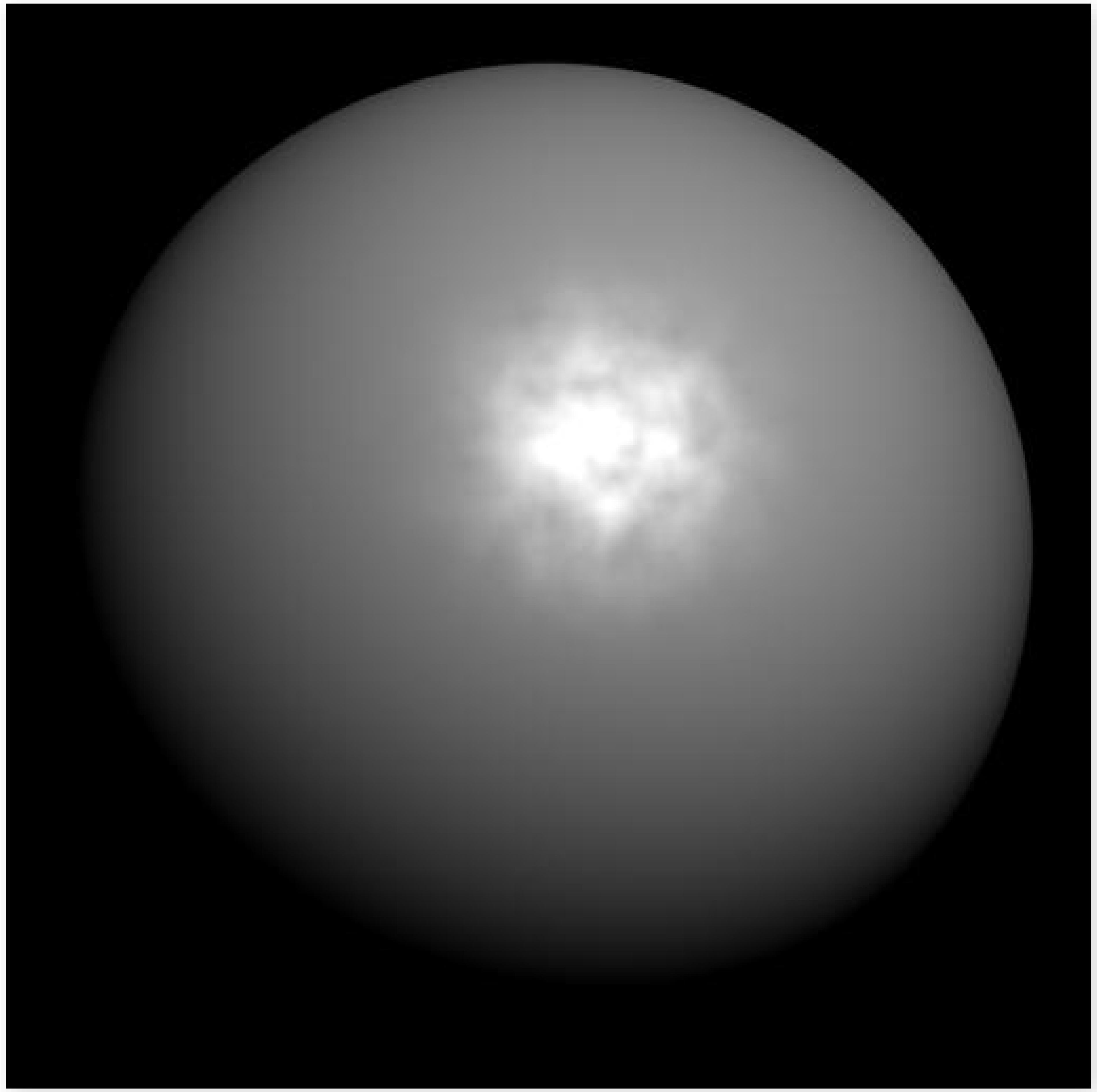




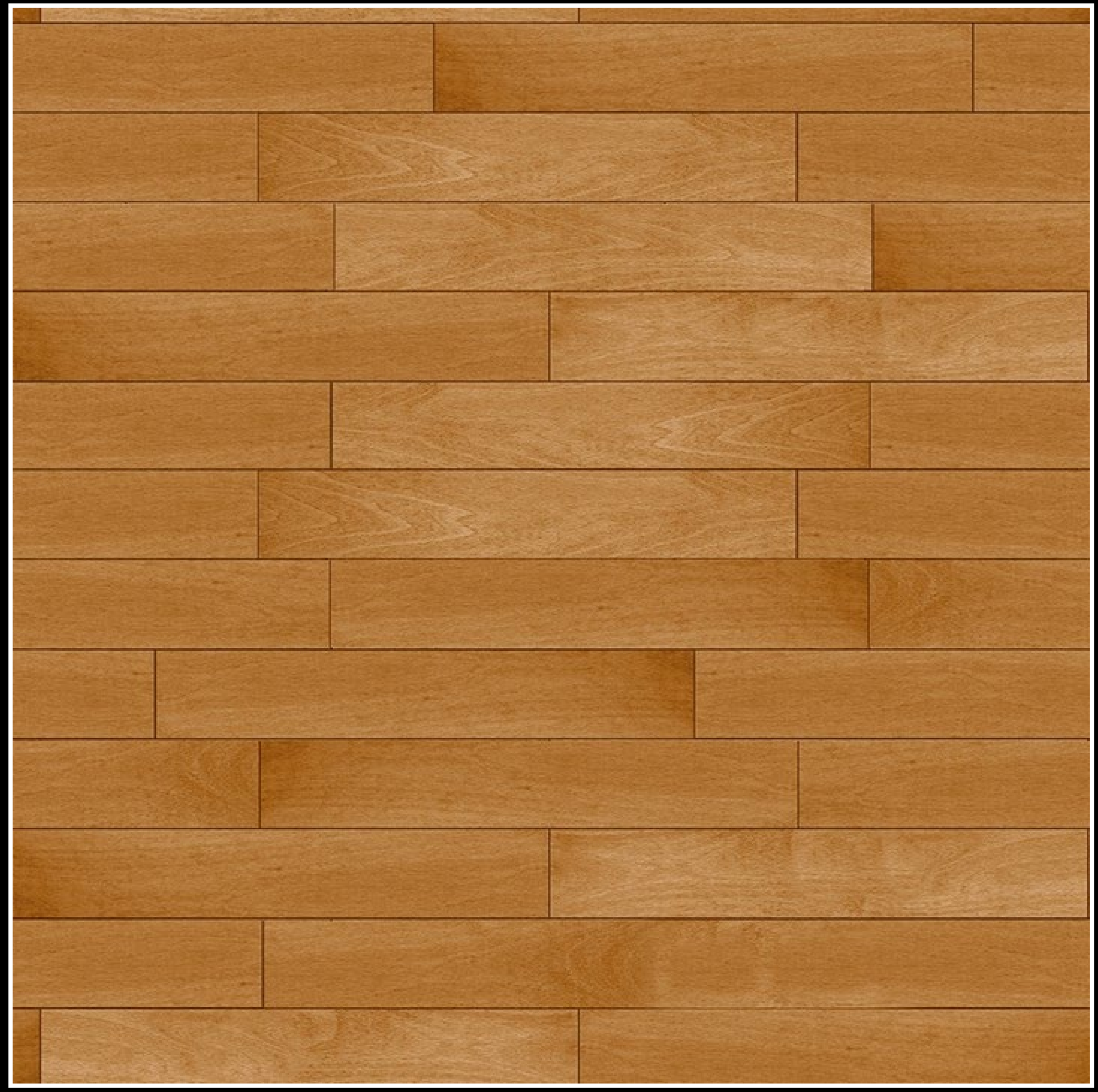
After a slide by Fabio Pellacini

Specular “shininess” coefficient

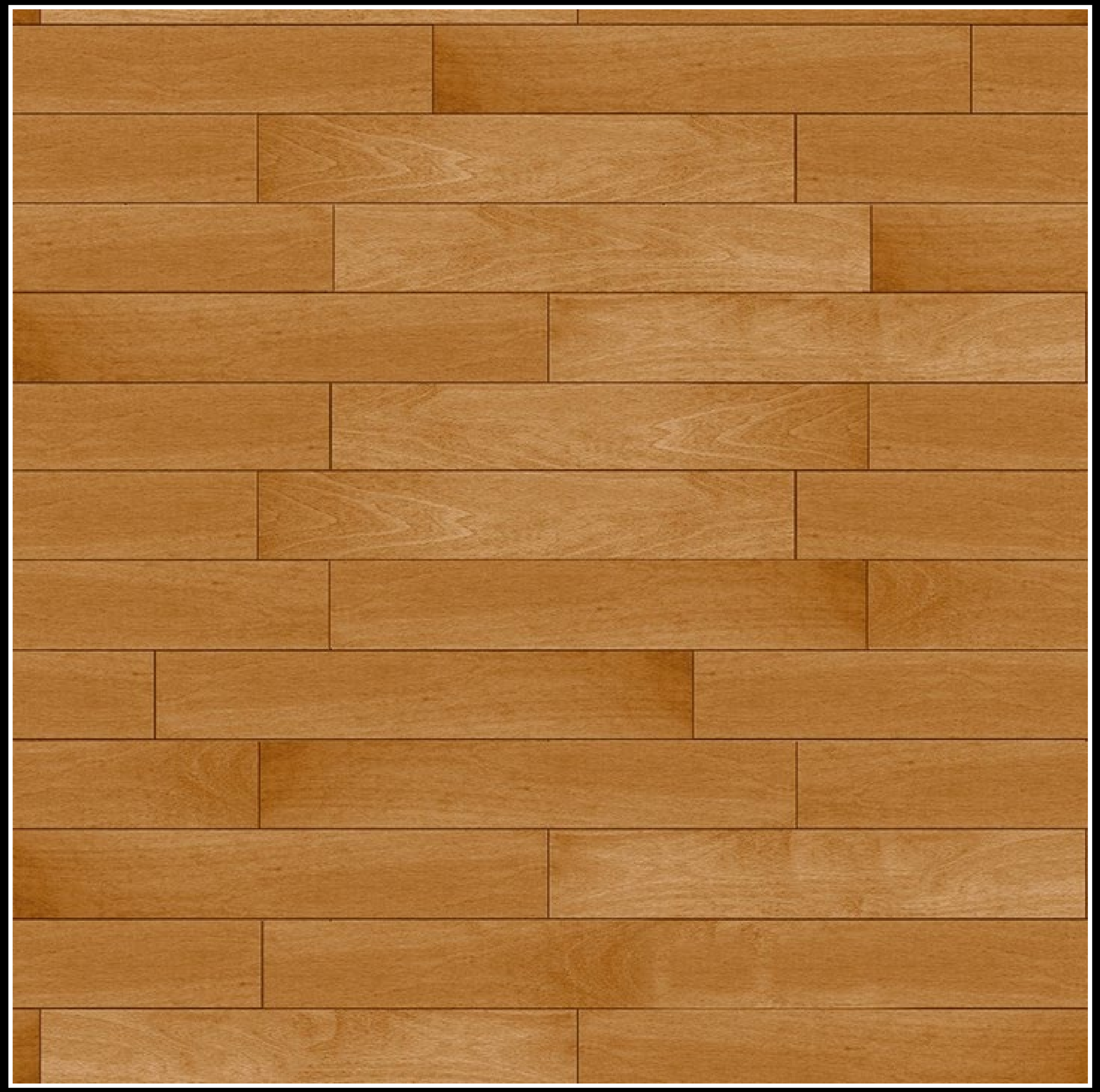




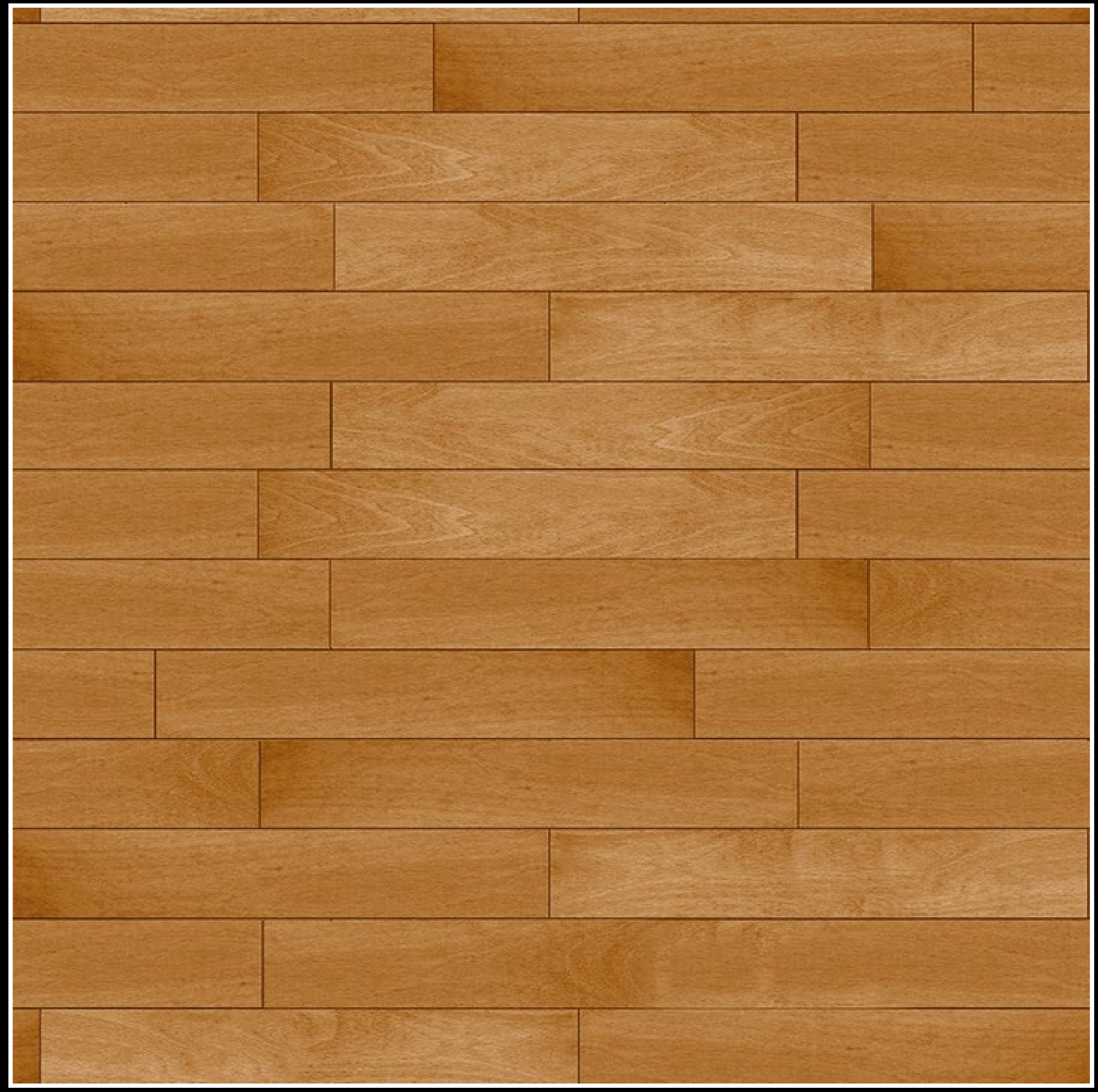
After a slide by Fabio Pellacini

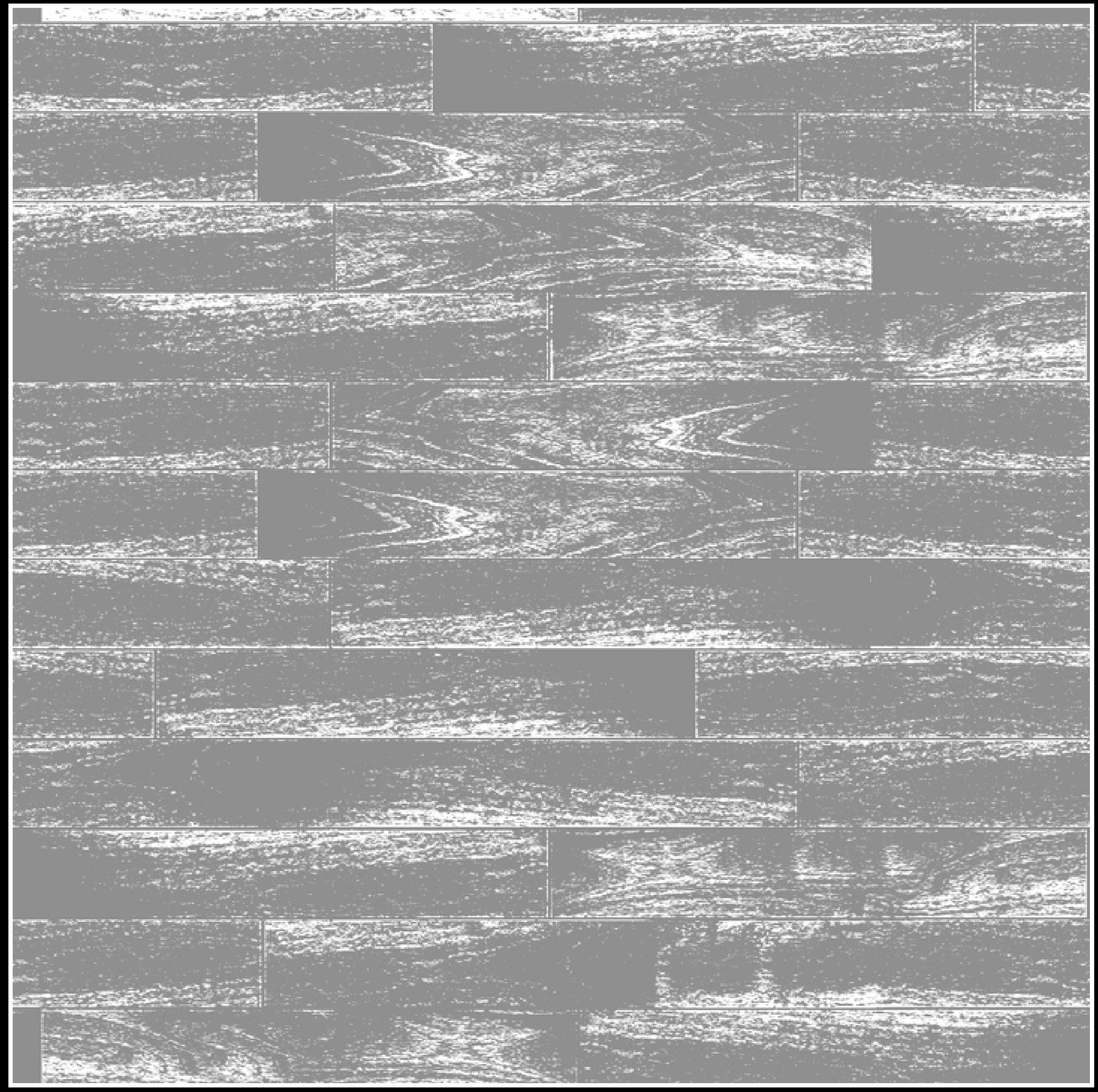


After a slide by Steve Marschner



After a slide by Steve Marschner





After a slide by Steve Marschner

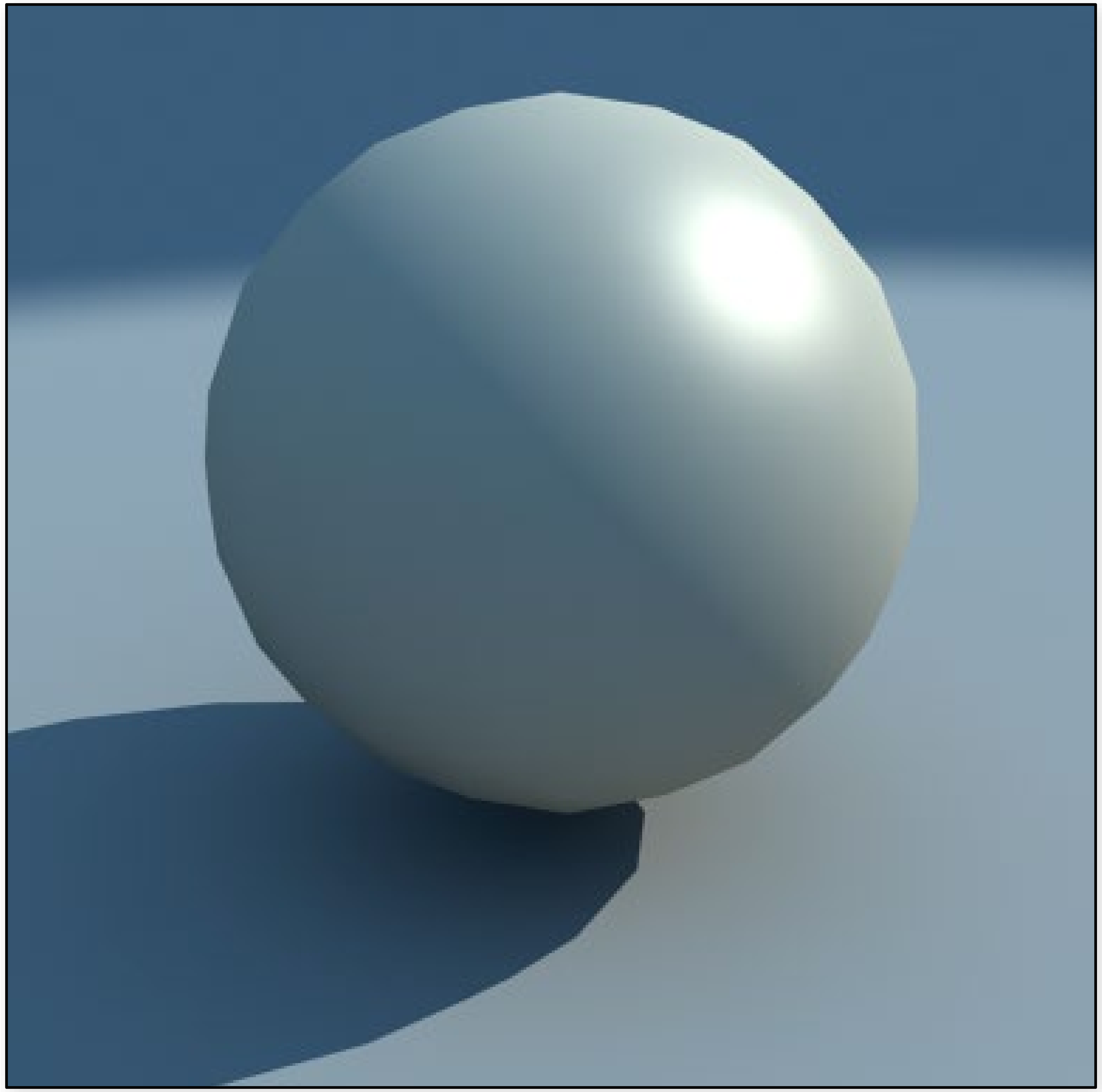
Texturing geometric detail

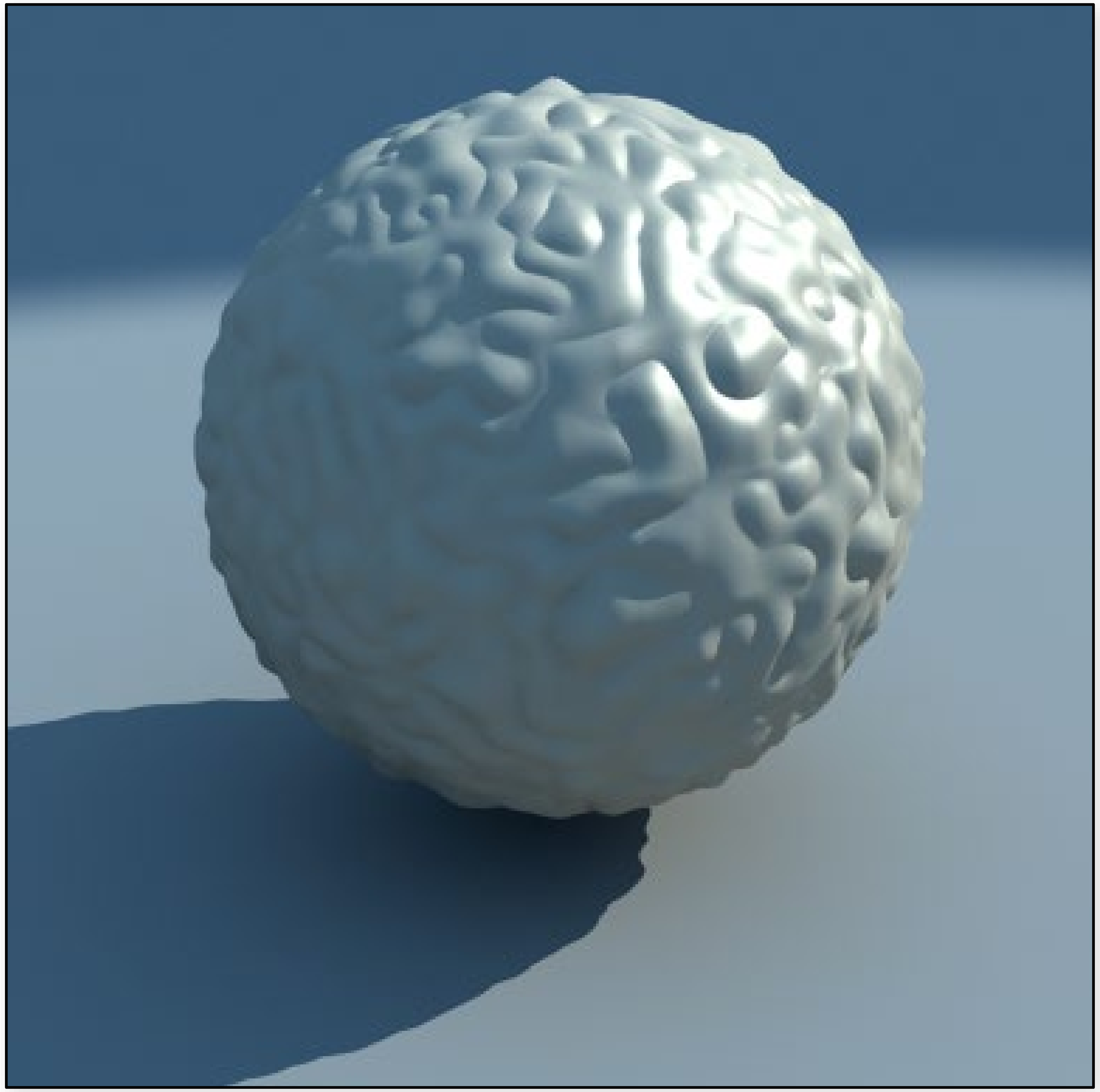
Texturing geometric detail

Modify geometric properties based on a texture

- normals
- surface positions

Displacement mapping



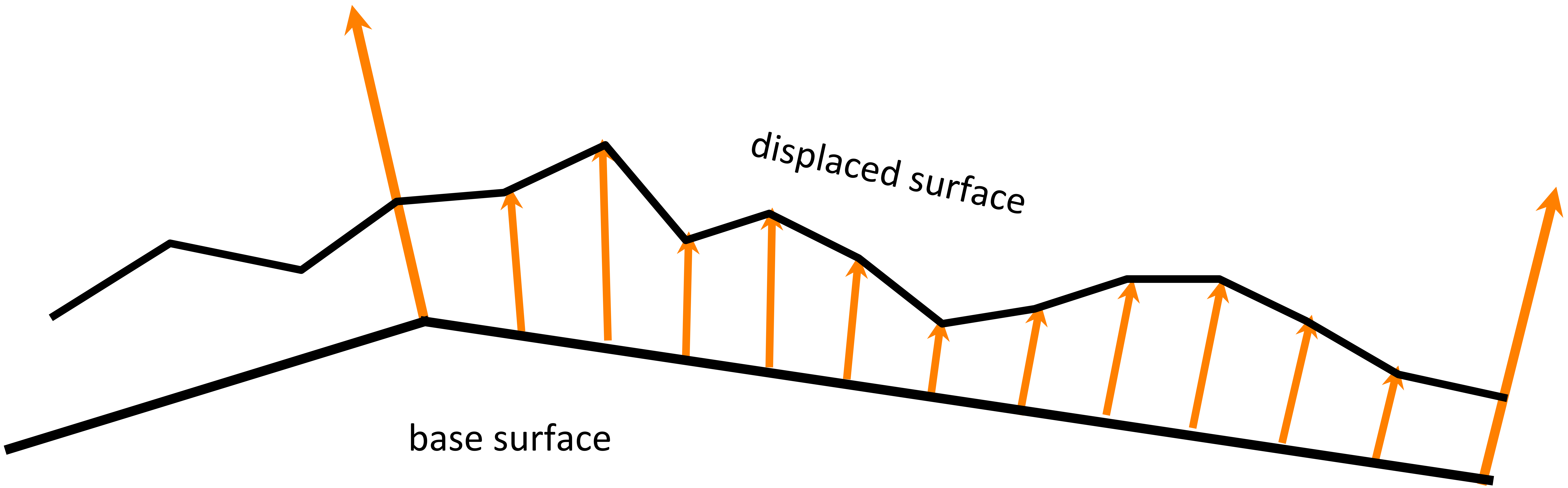


from: www.spot3d.com

Displacement mapping

Encode a displacement distance in the texture map

- Measured, e.g., along interpolated normal



After a slide by Daniele Panozzo

Displacement mapping

Update position by displacing points along normal

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

Recompute normals by evaluating derivatives

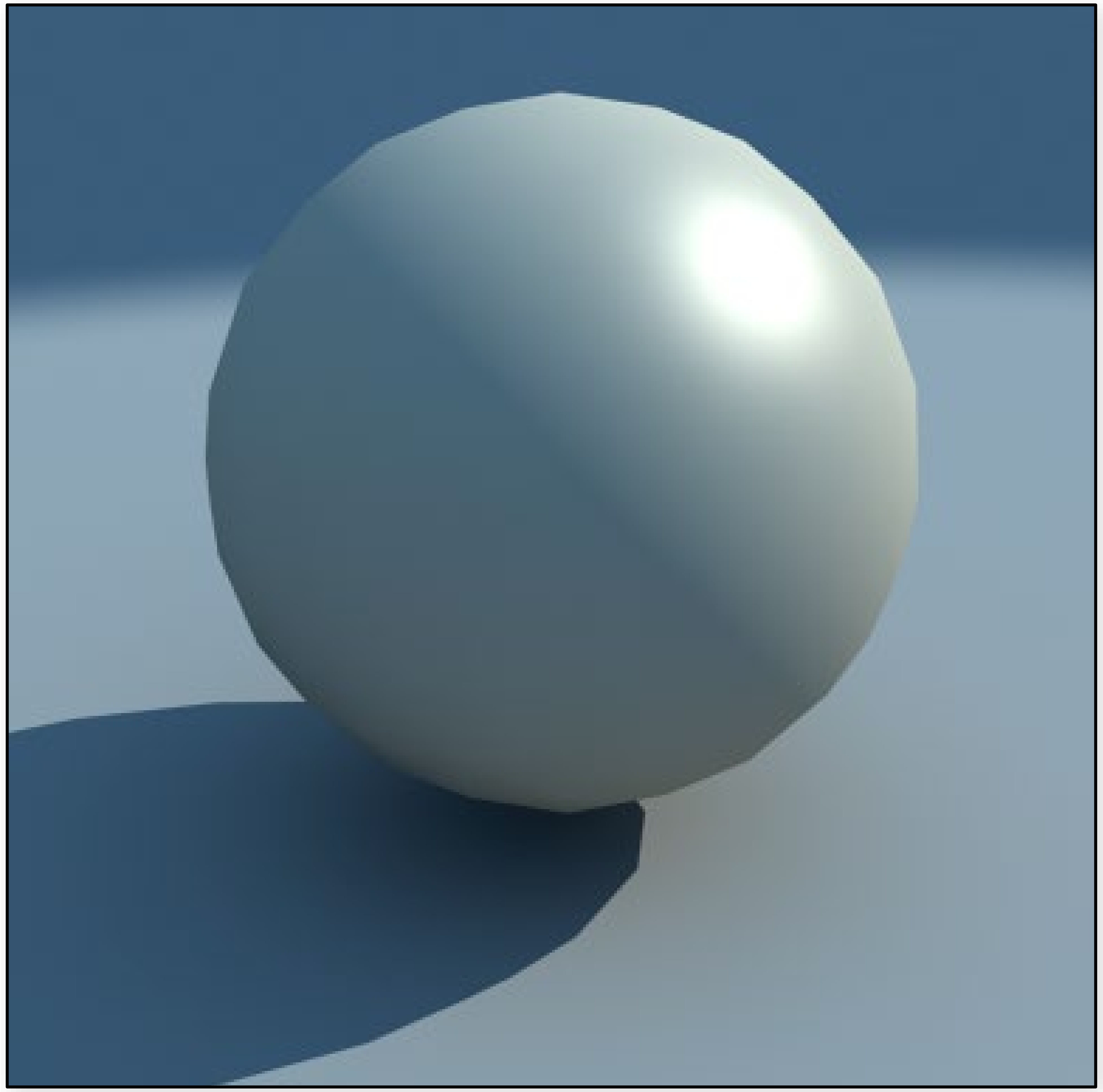
- often no closed form solution, so use finite differences

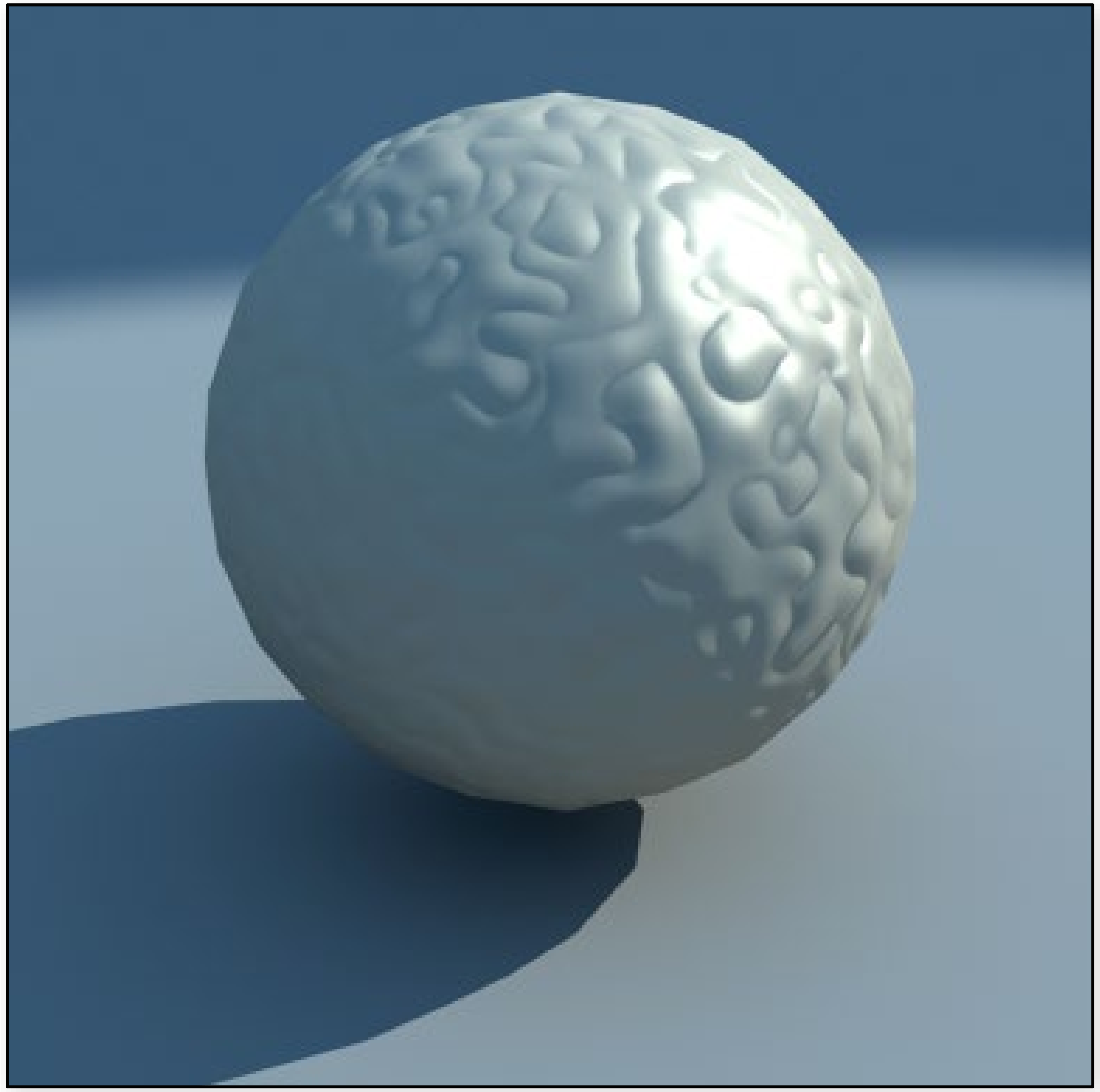
$$\mathbf{n}_d \propto \frac{\partial \mathbf{p}_d}{\partial u} \times \frac{\partial \mathbf{p}_d}{\partial v} \approx \frac{\Delta \mathbf{p}_d}{\Delta u} \times \frac{\Delta \mathbf{p}_d}{\Delta v}$$

After a slide by Fabio Pellacini

Bump mapping

Apply normal perturbation without updating positions

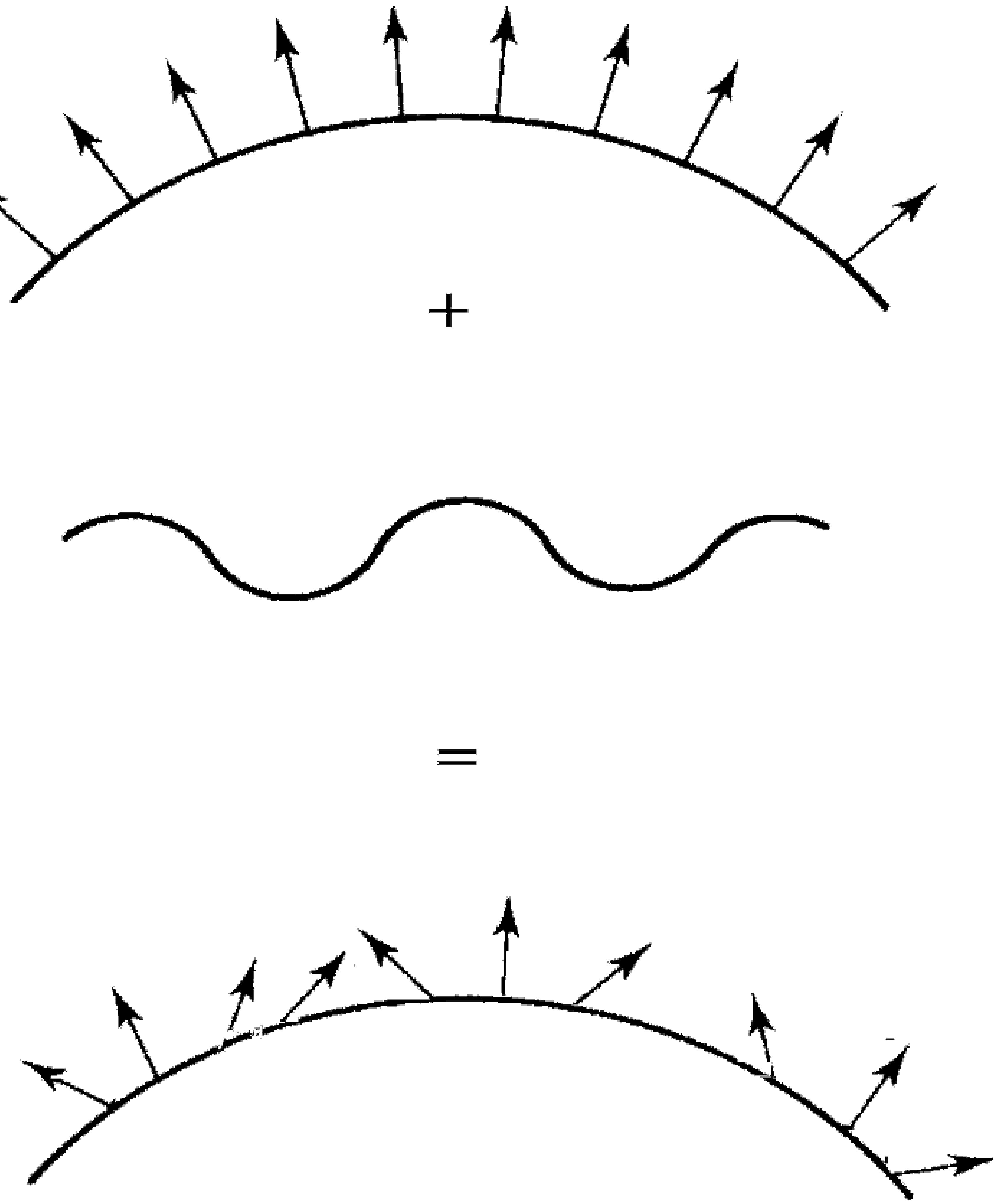




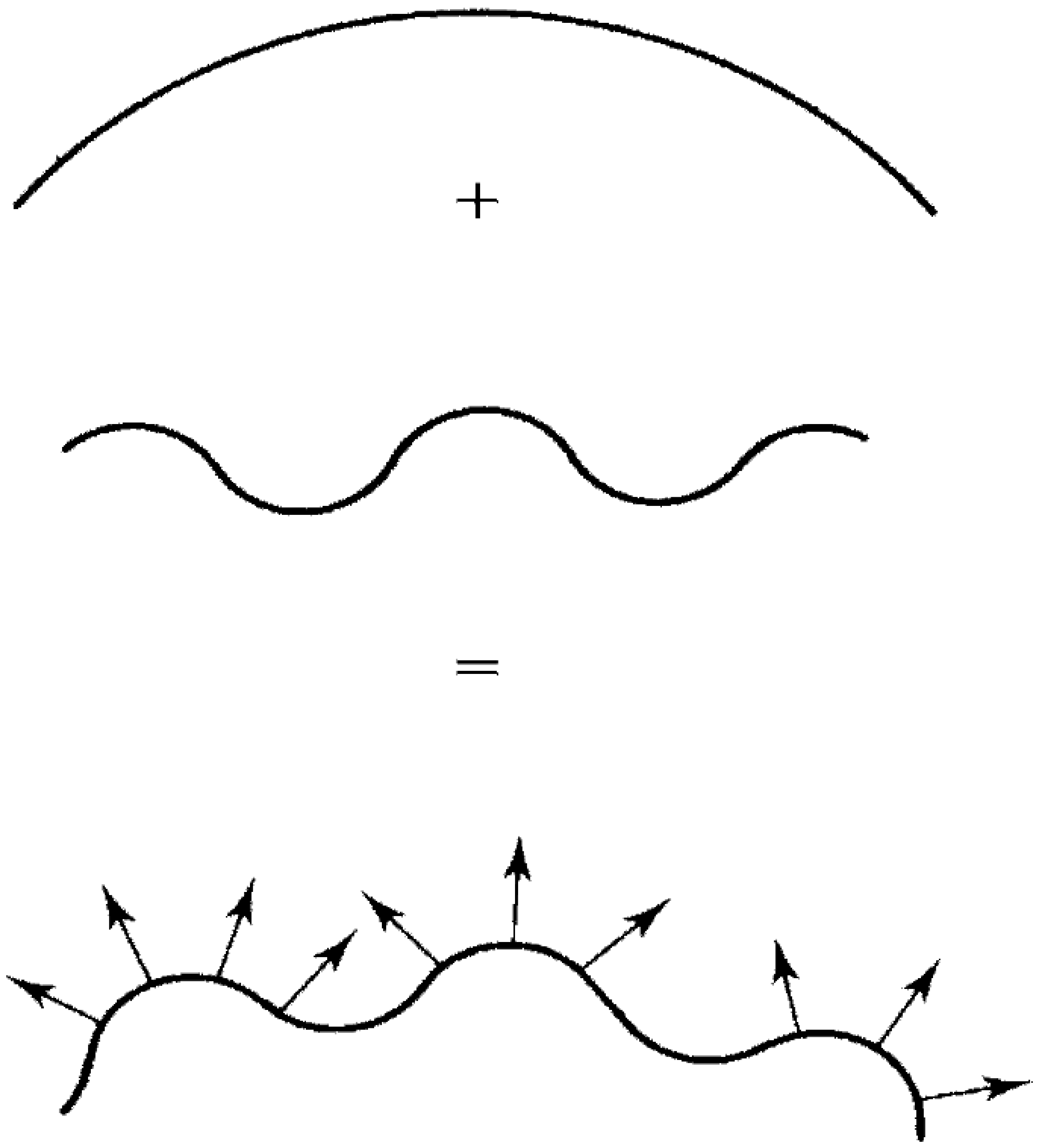
from: www.spot3d.com

Bump vs. displacement mapping

bump map



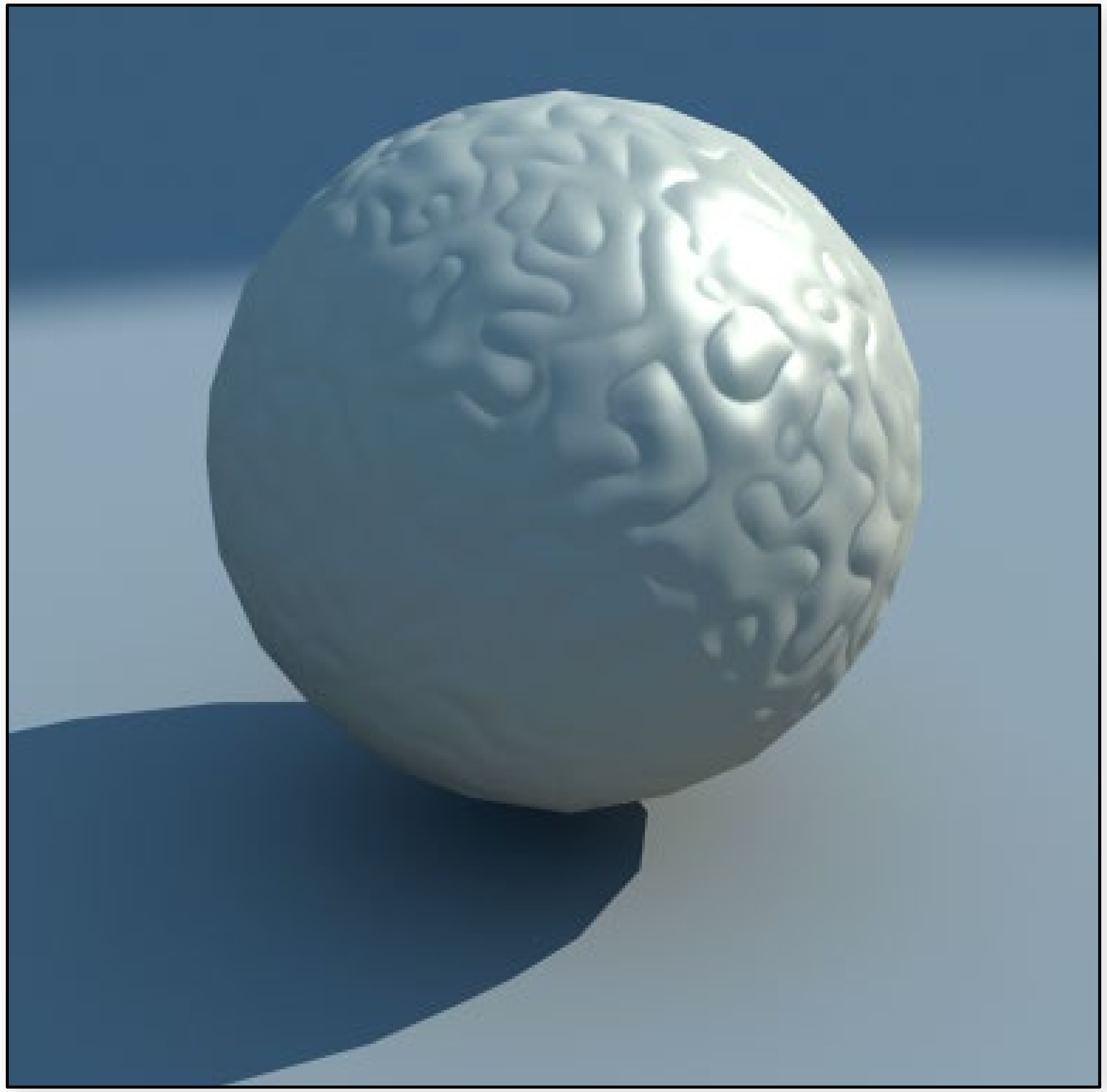
displacement map



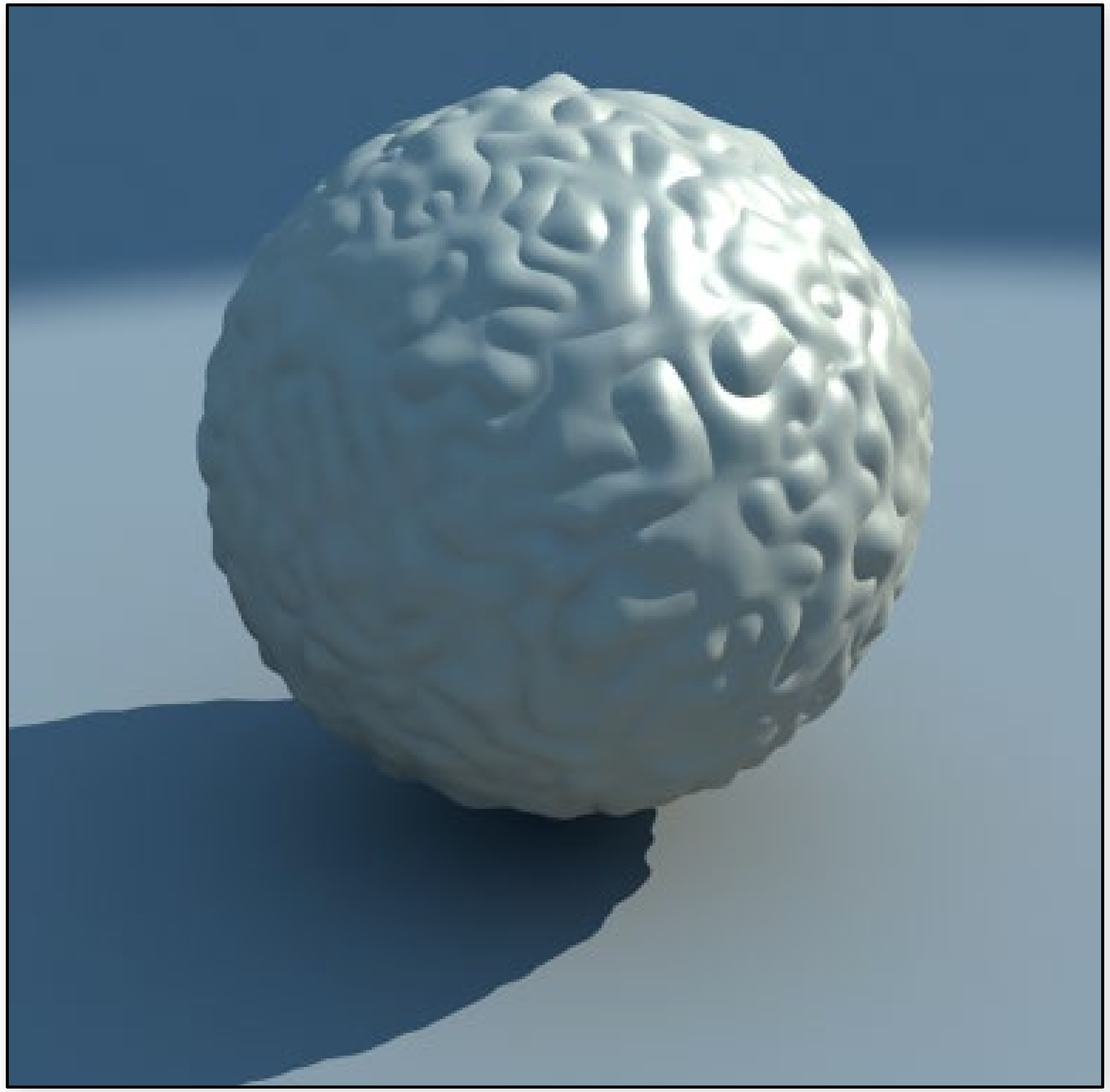
After a slide by Fabio Pellacini

Bump vs. displacement mapping

bump map

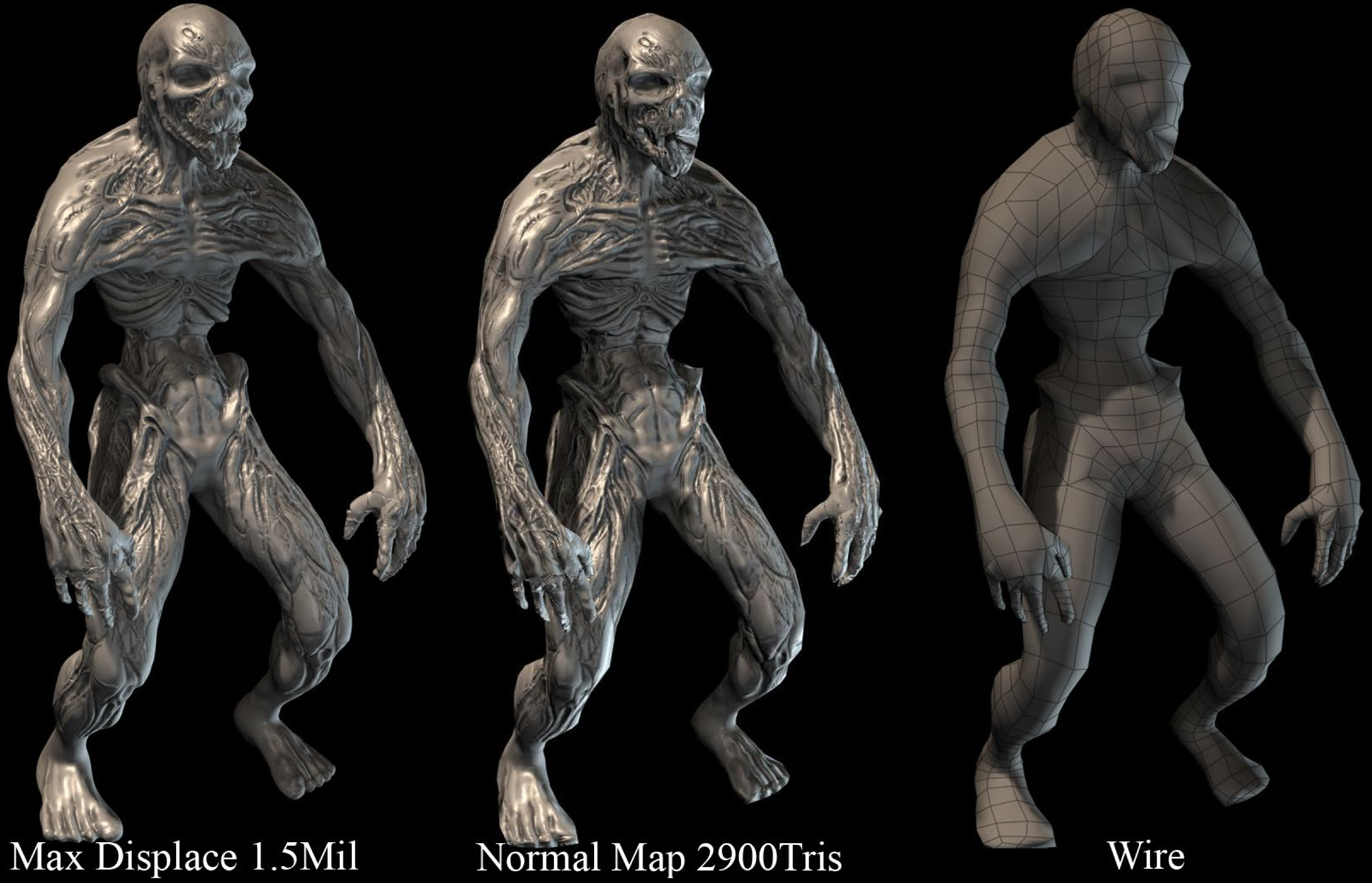


displacement map



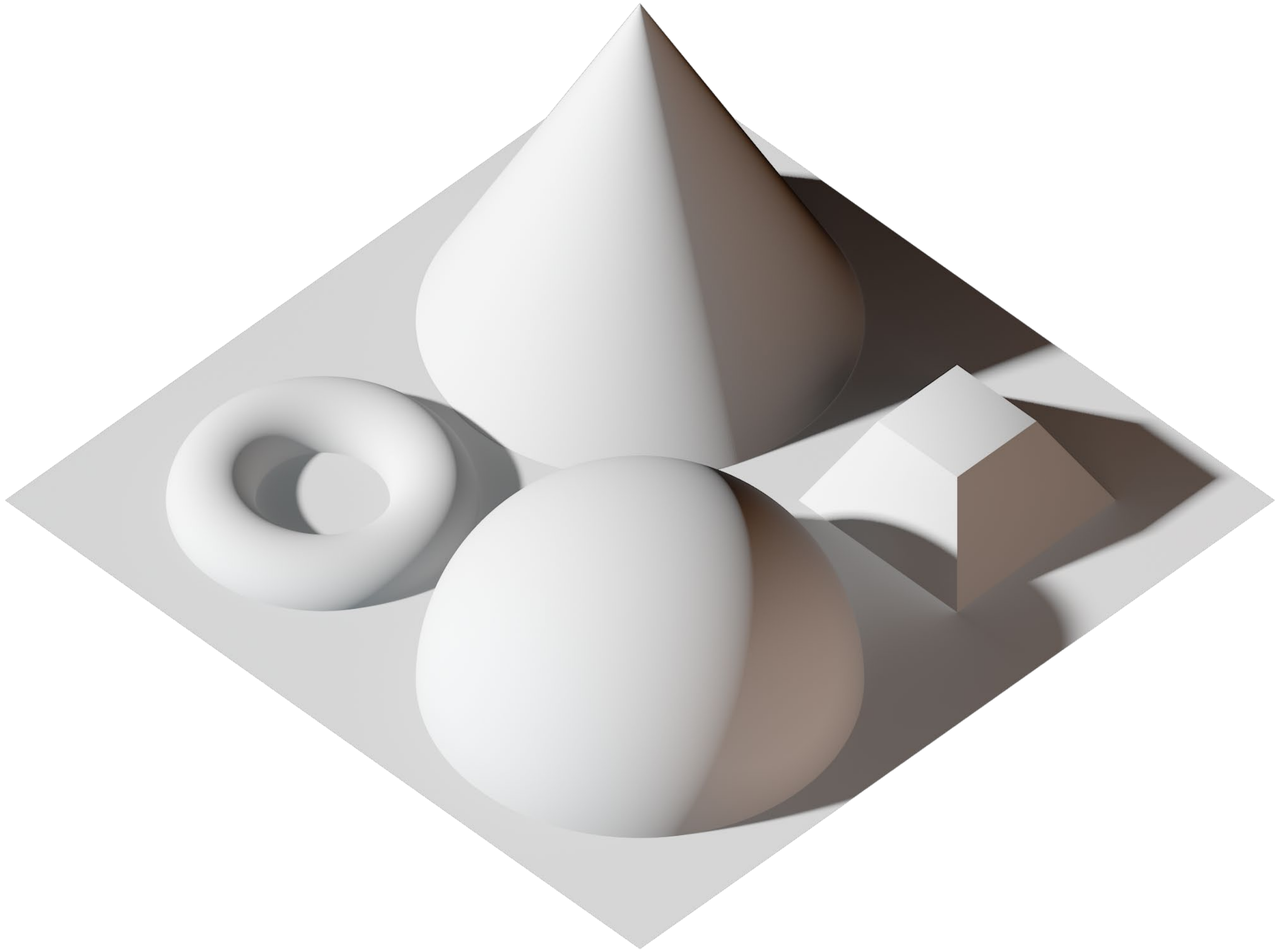
from: www.spot3d.com

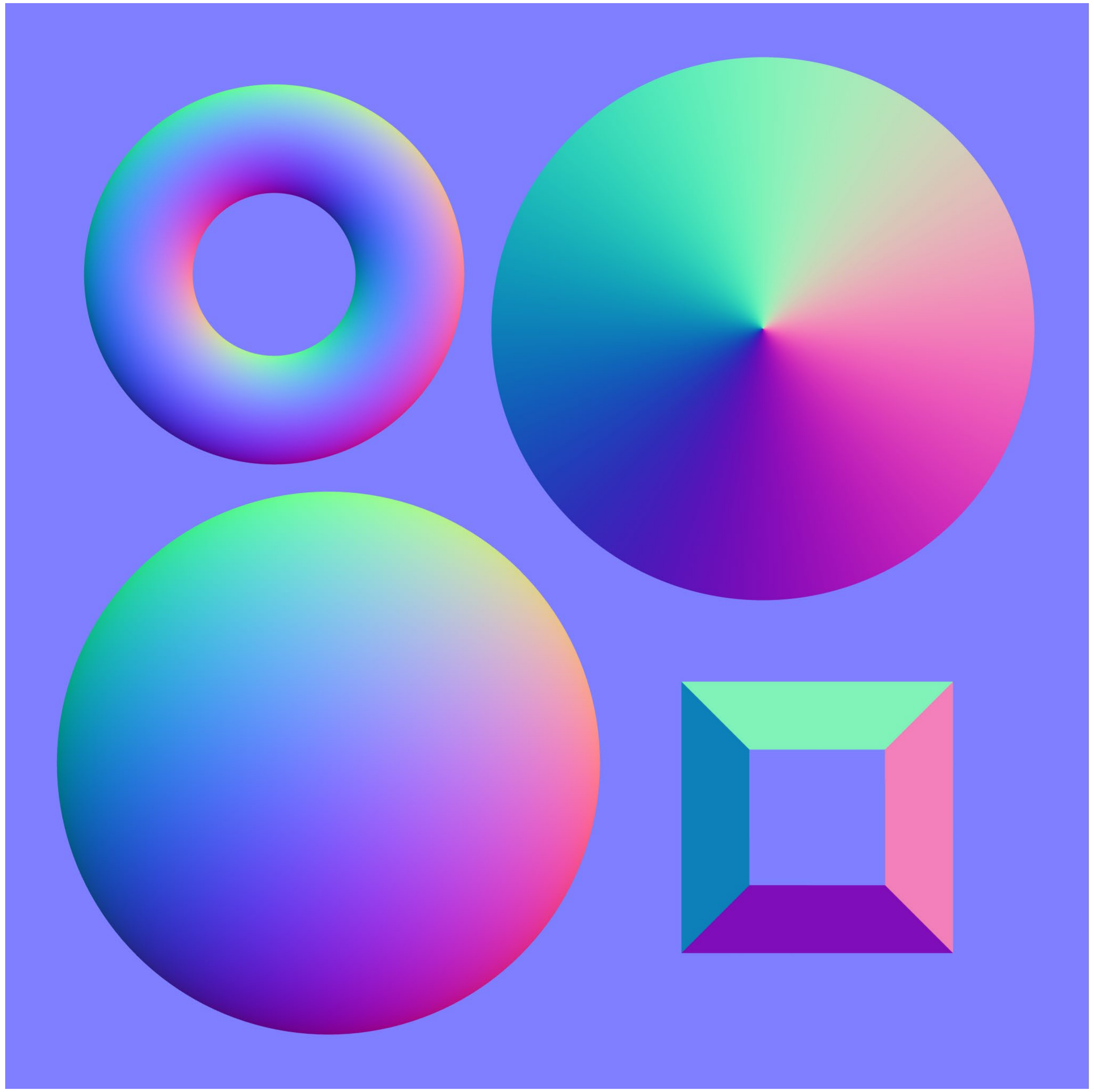
Displacement vs. bump mapping



After a slide by Kavita Bala

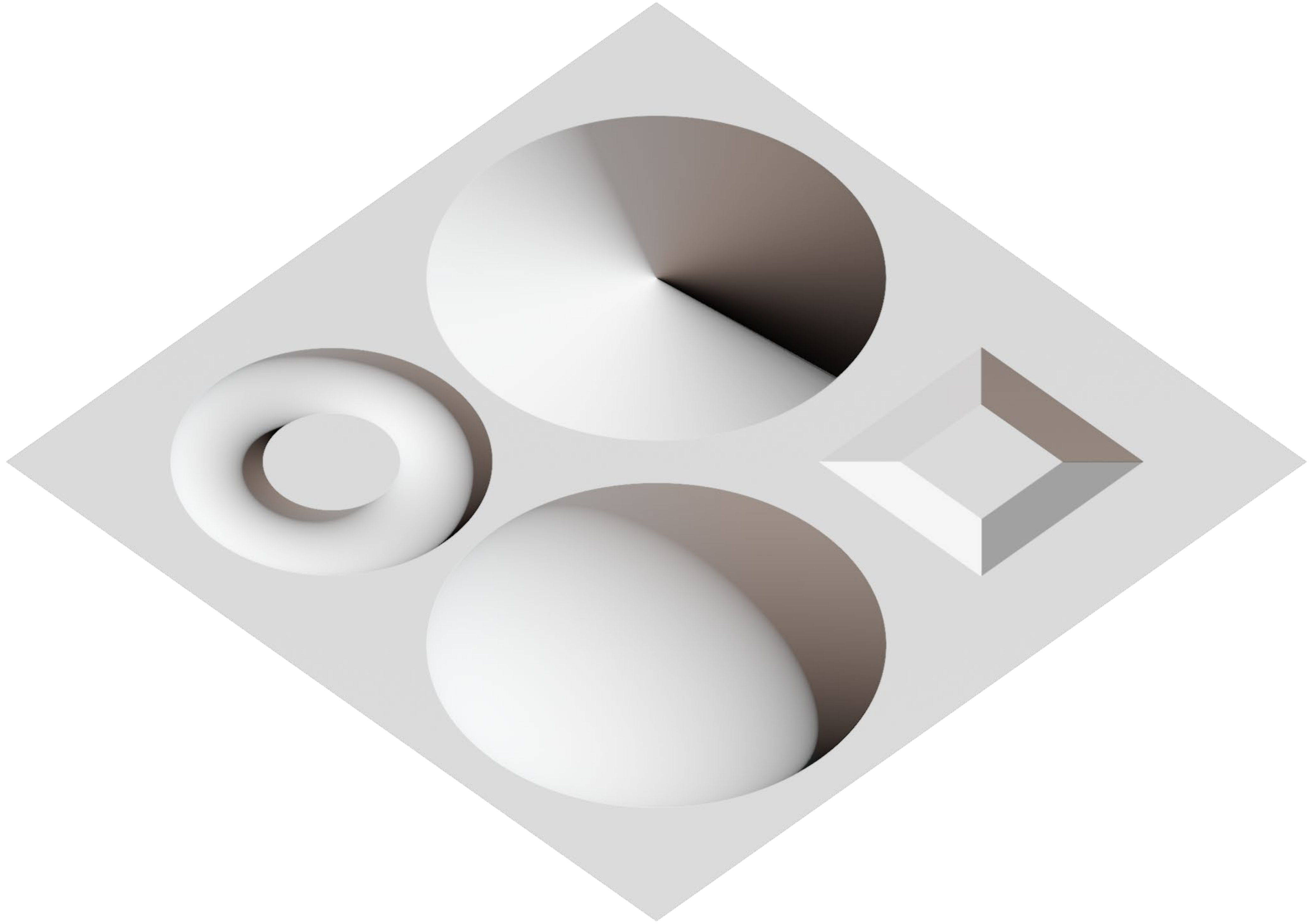
Surface normals

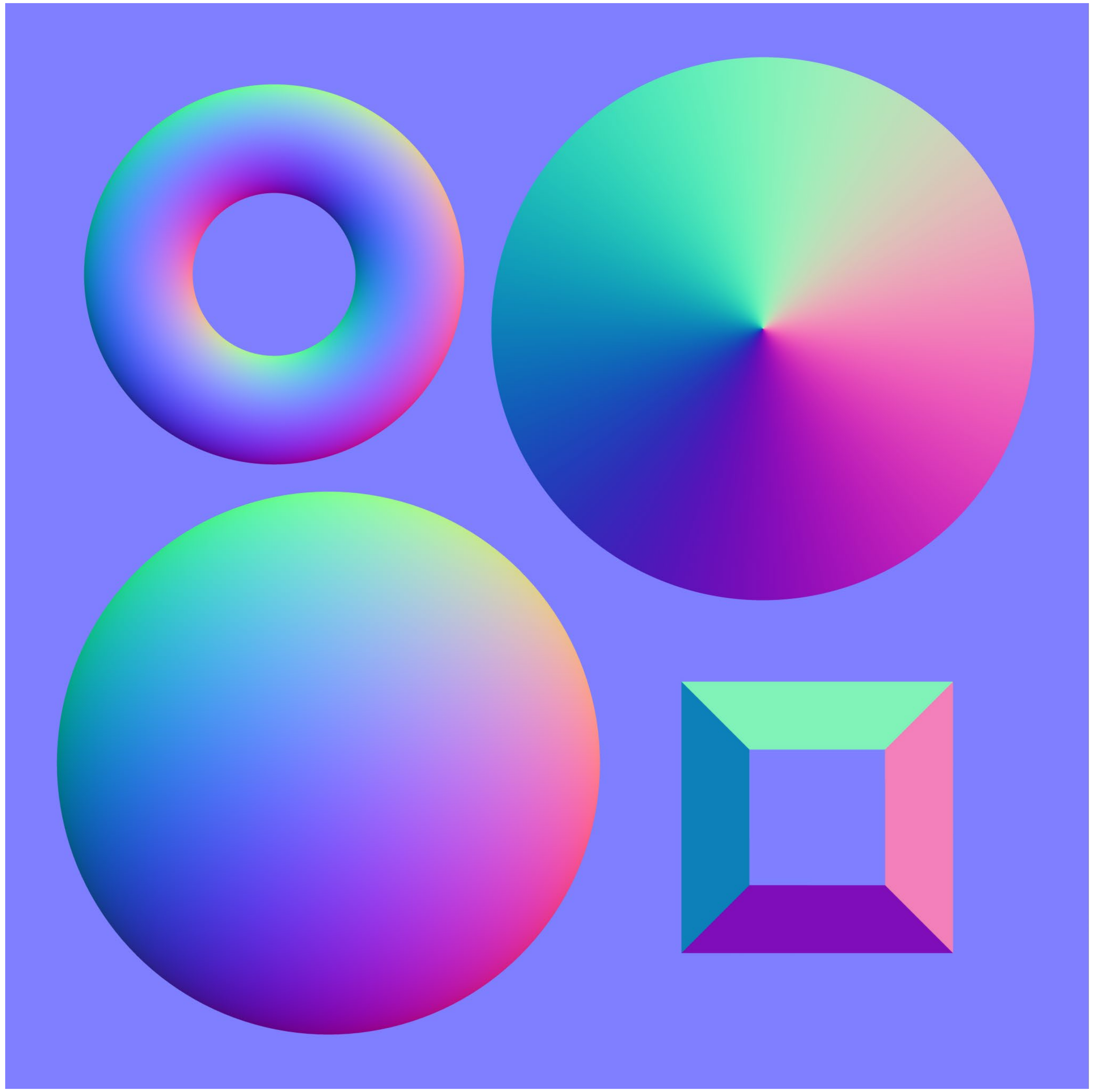




[Wikipedia]

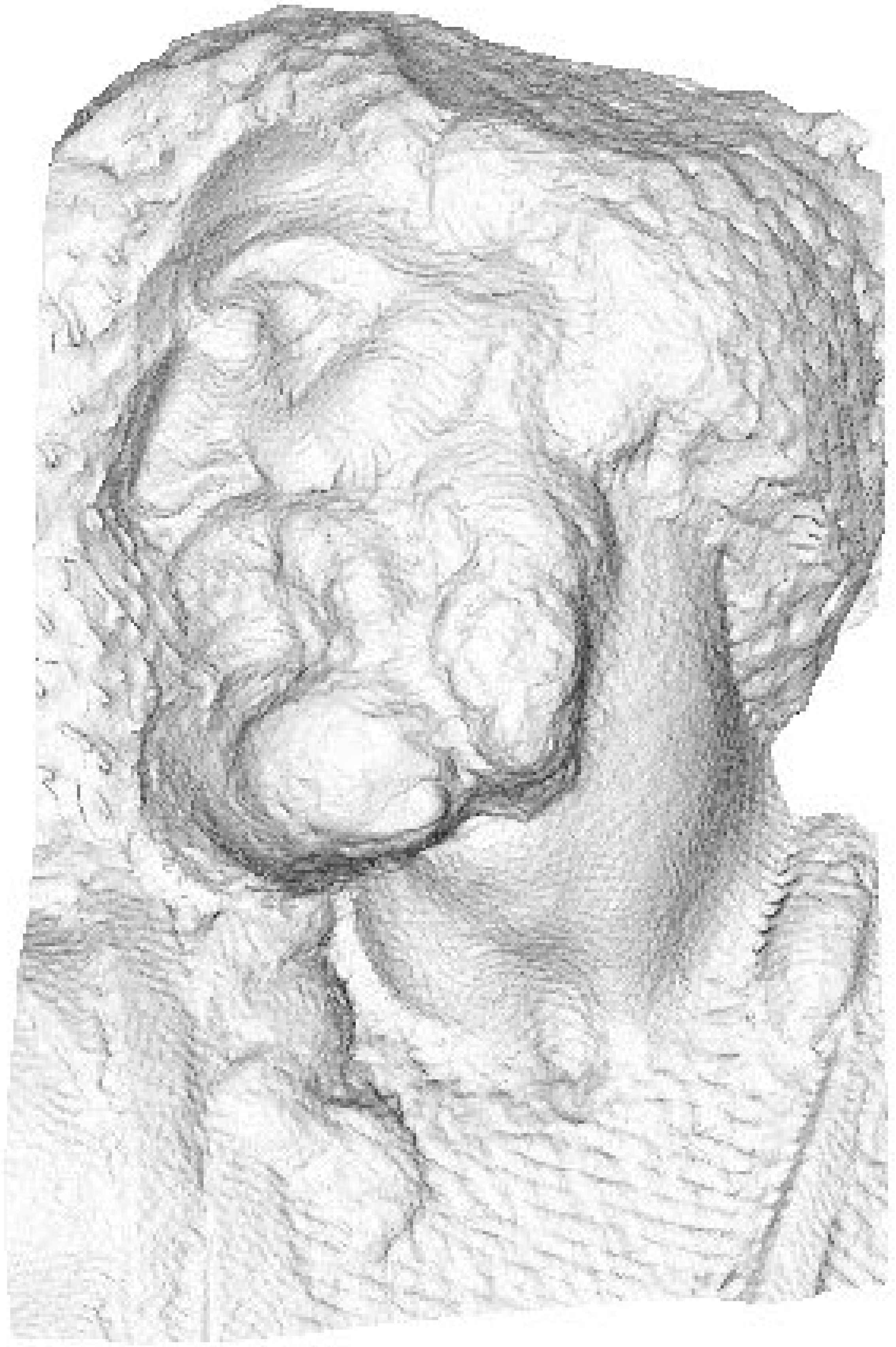
Normal mapping



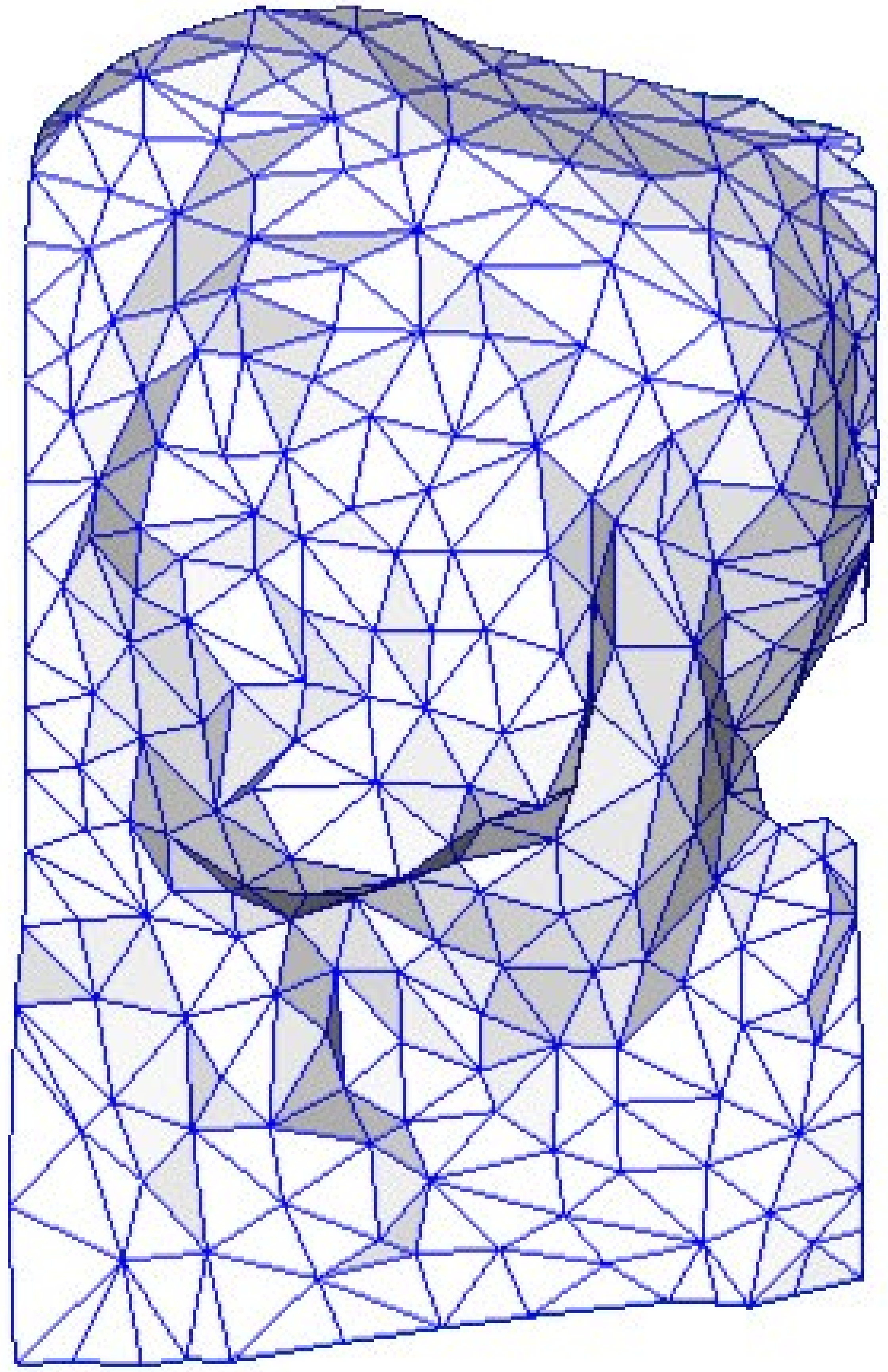


[Wikipedia]

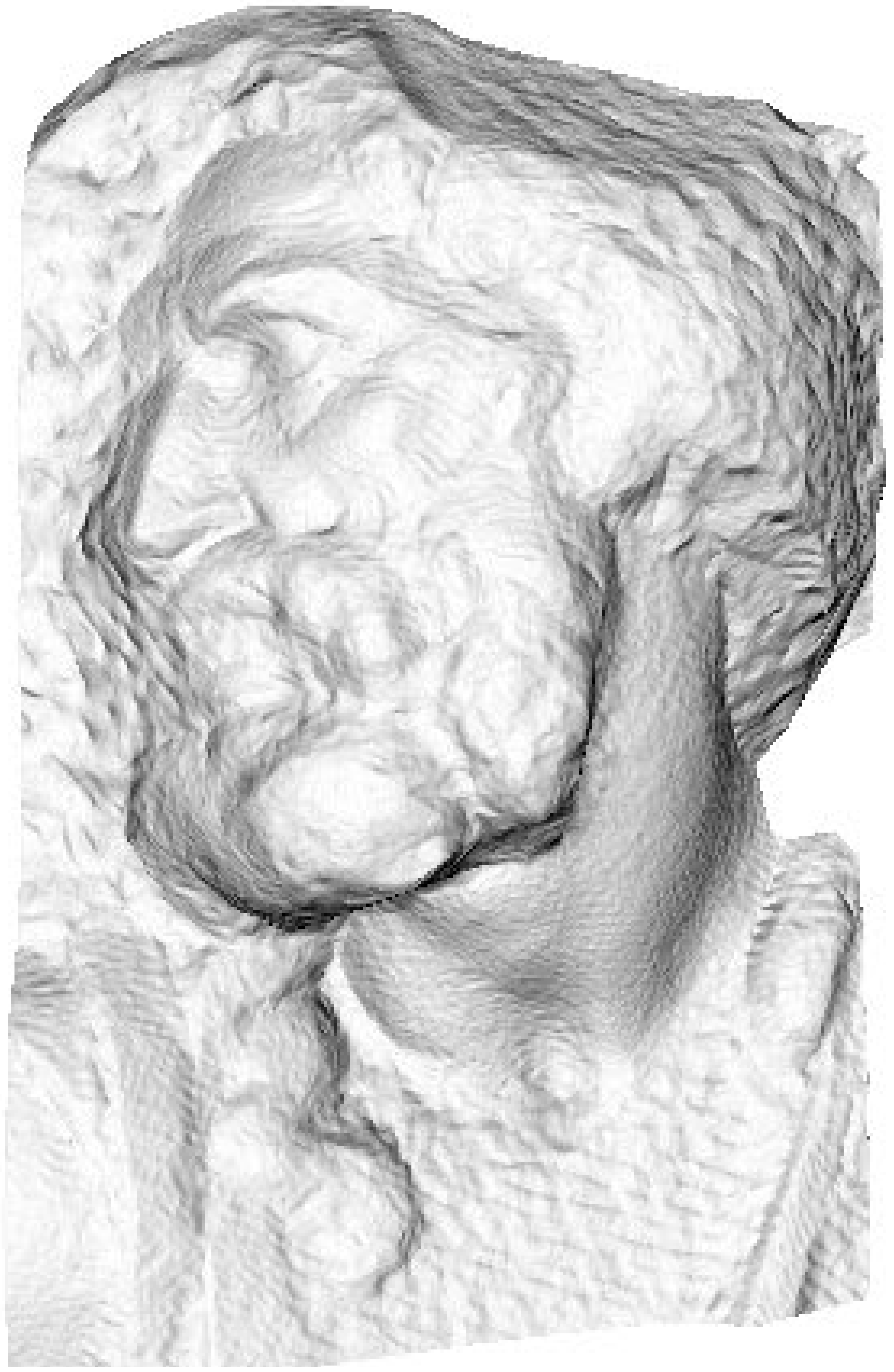
Normal mapping



original mesh
4M triangles



simplified mesh
500 triangles



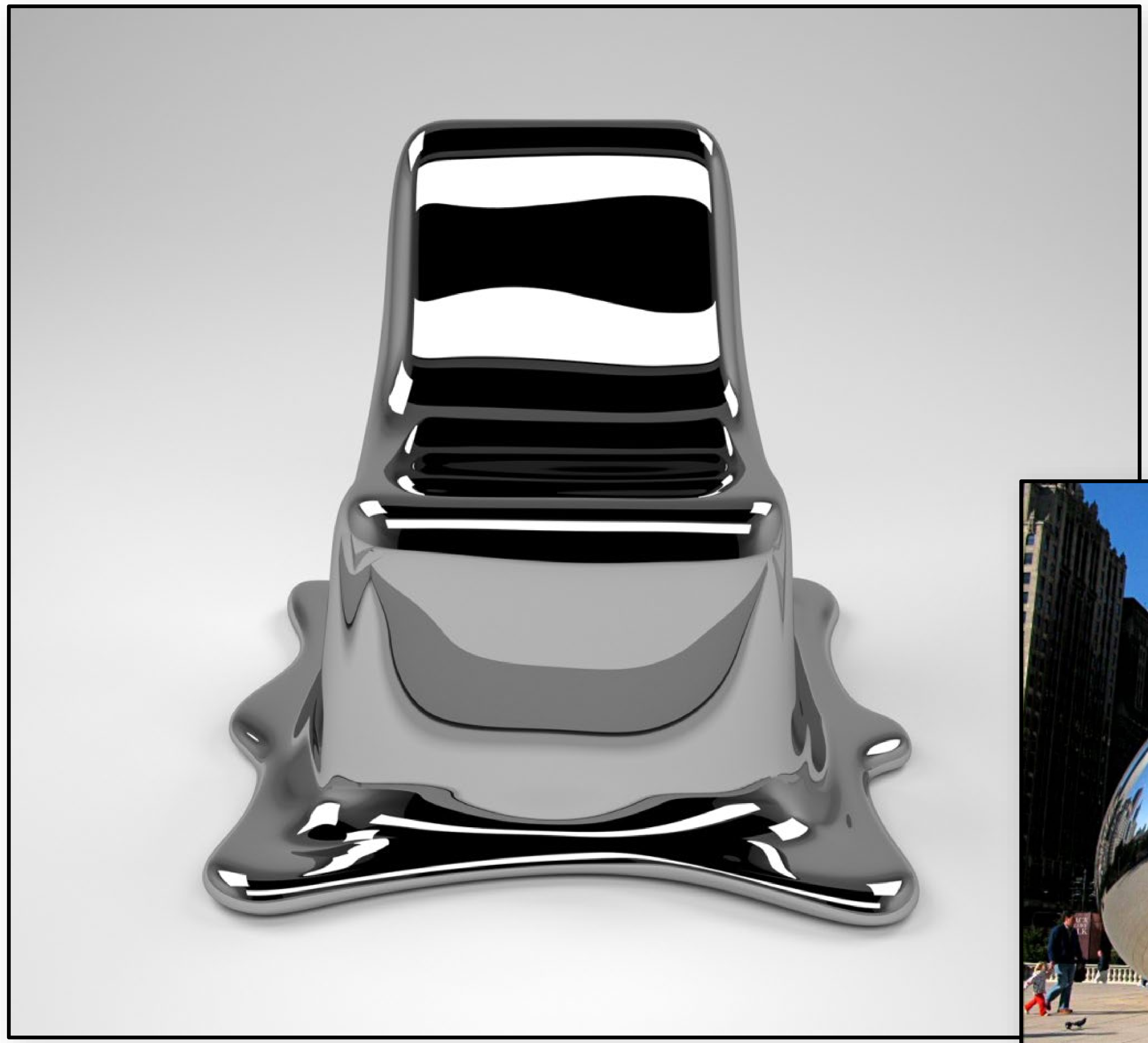
simplified mesh
and normal mapping
500 triangles

After a slide by Daniele Panozzo

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

source: 3delight

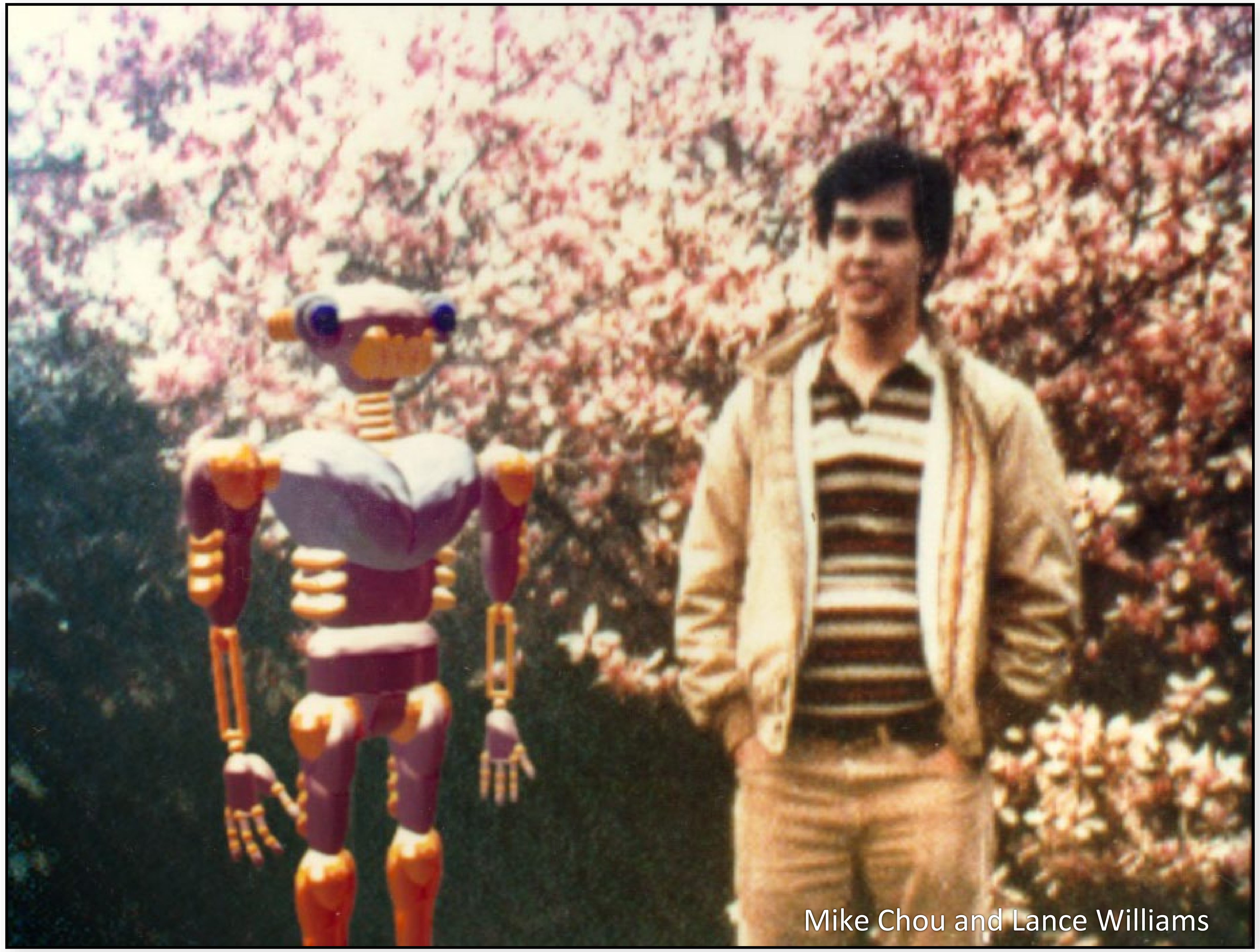
Environment map

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

After a slide by Steve Marschner

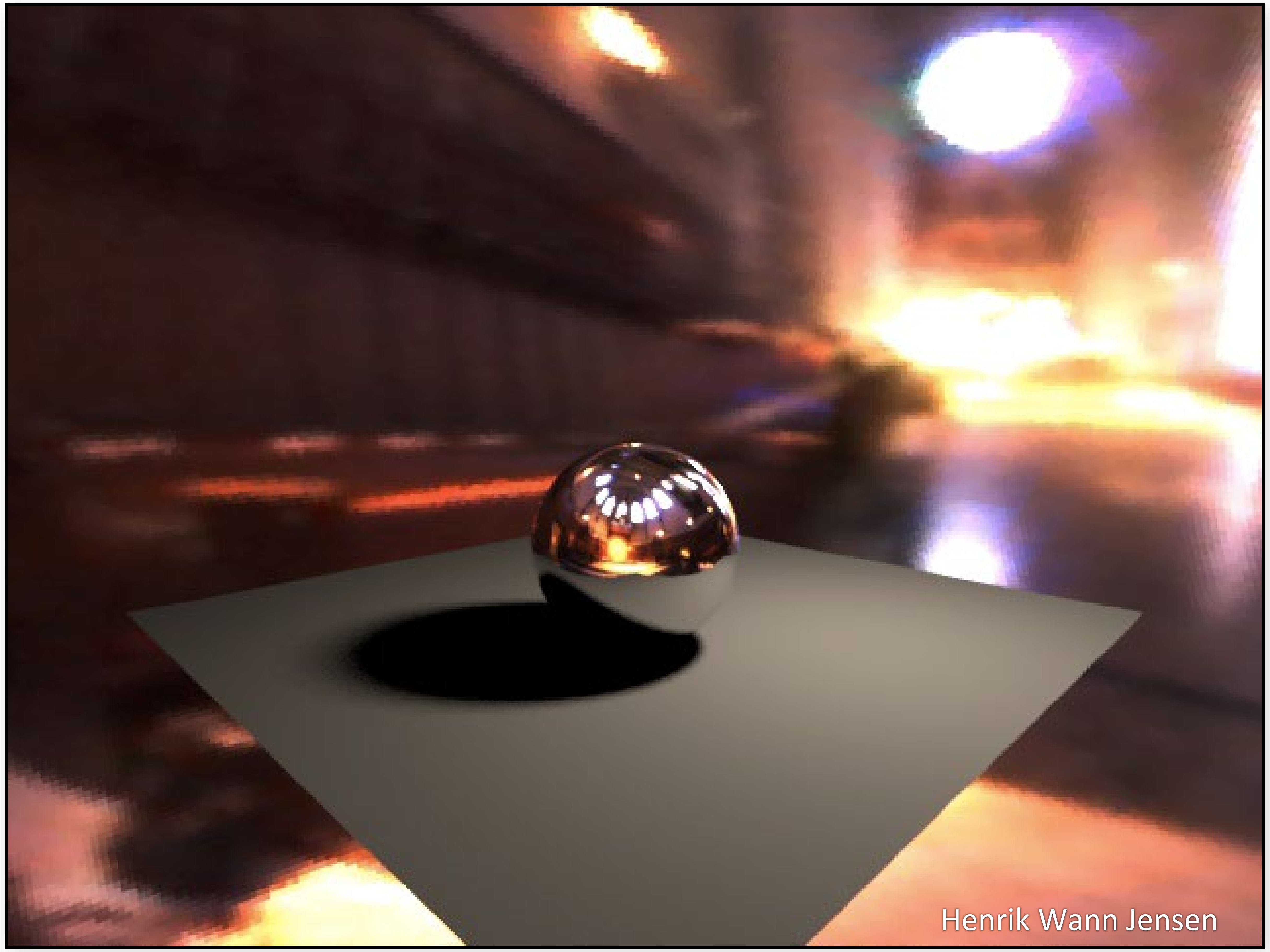
[Blinn & Newell 1976]

Reflection mapping (1982)



Environment mapping

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



Acquiring environment maps

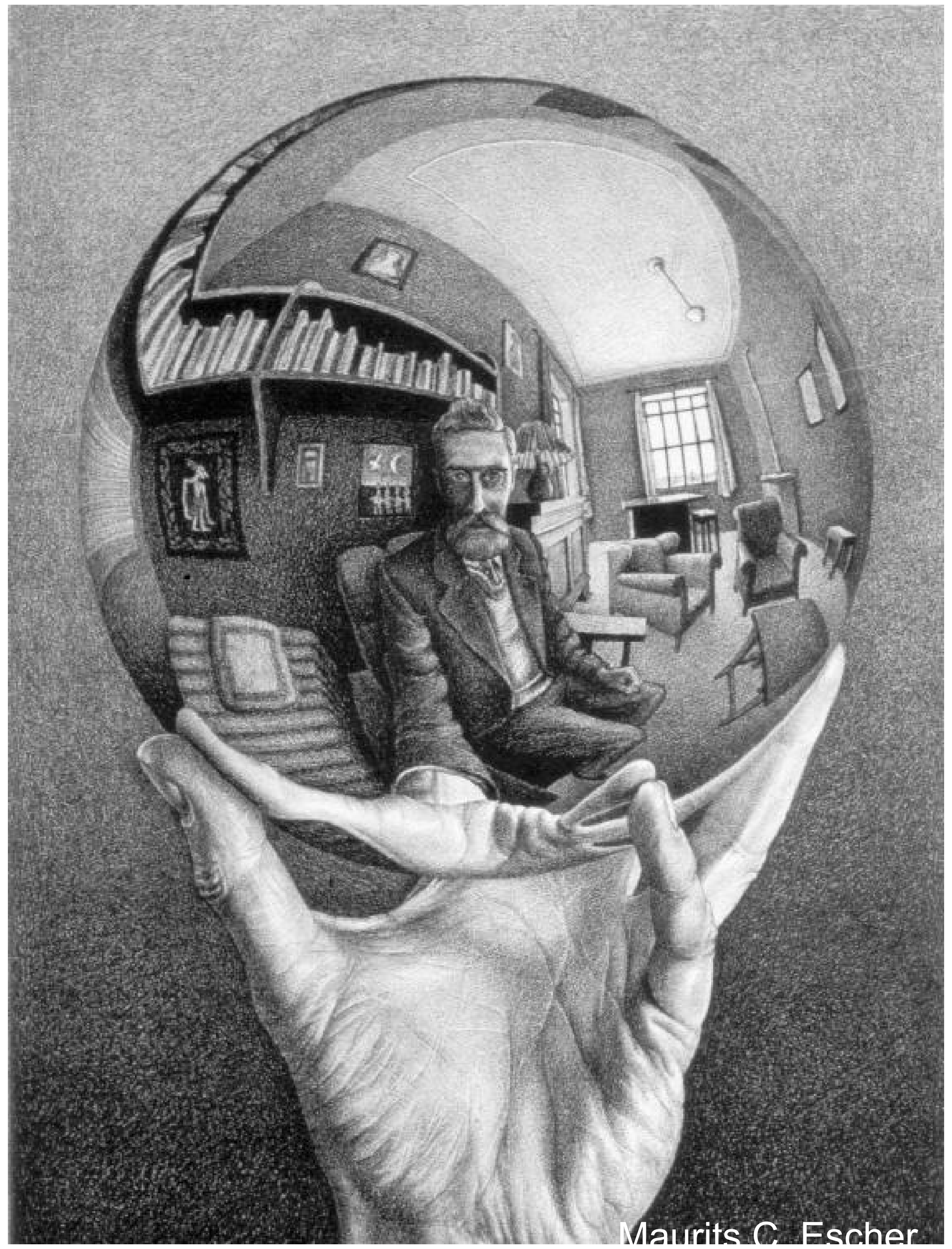
Mirrored ball + camera

Fisheye lens images

Stitching images together

Panoramic camera

Acquisition - Low Tech.

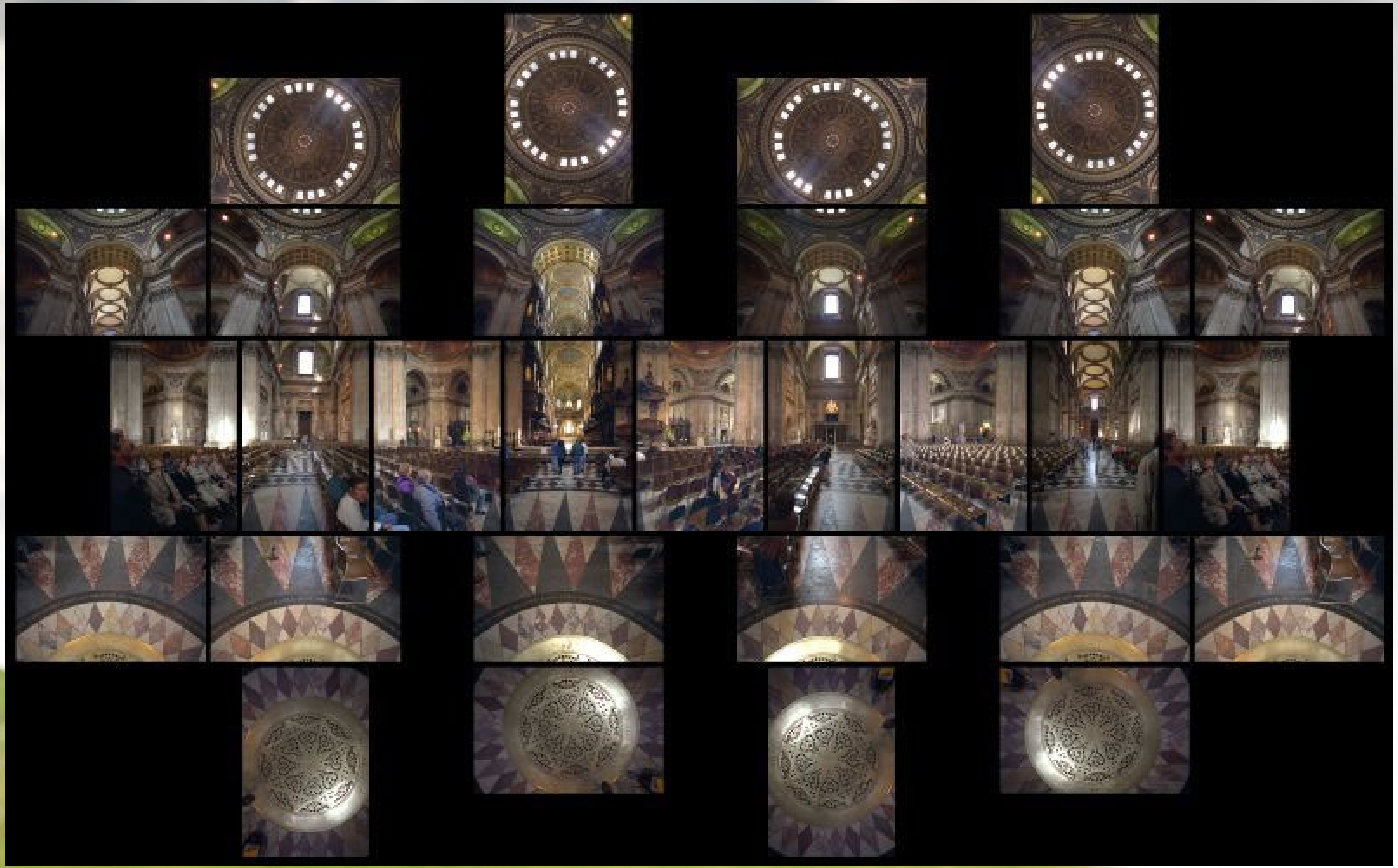


Lightprobe

omnidirectional,
360° panoramic,
HDR image

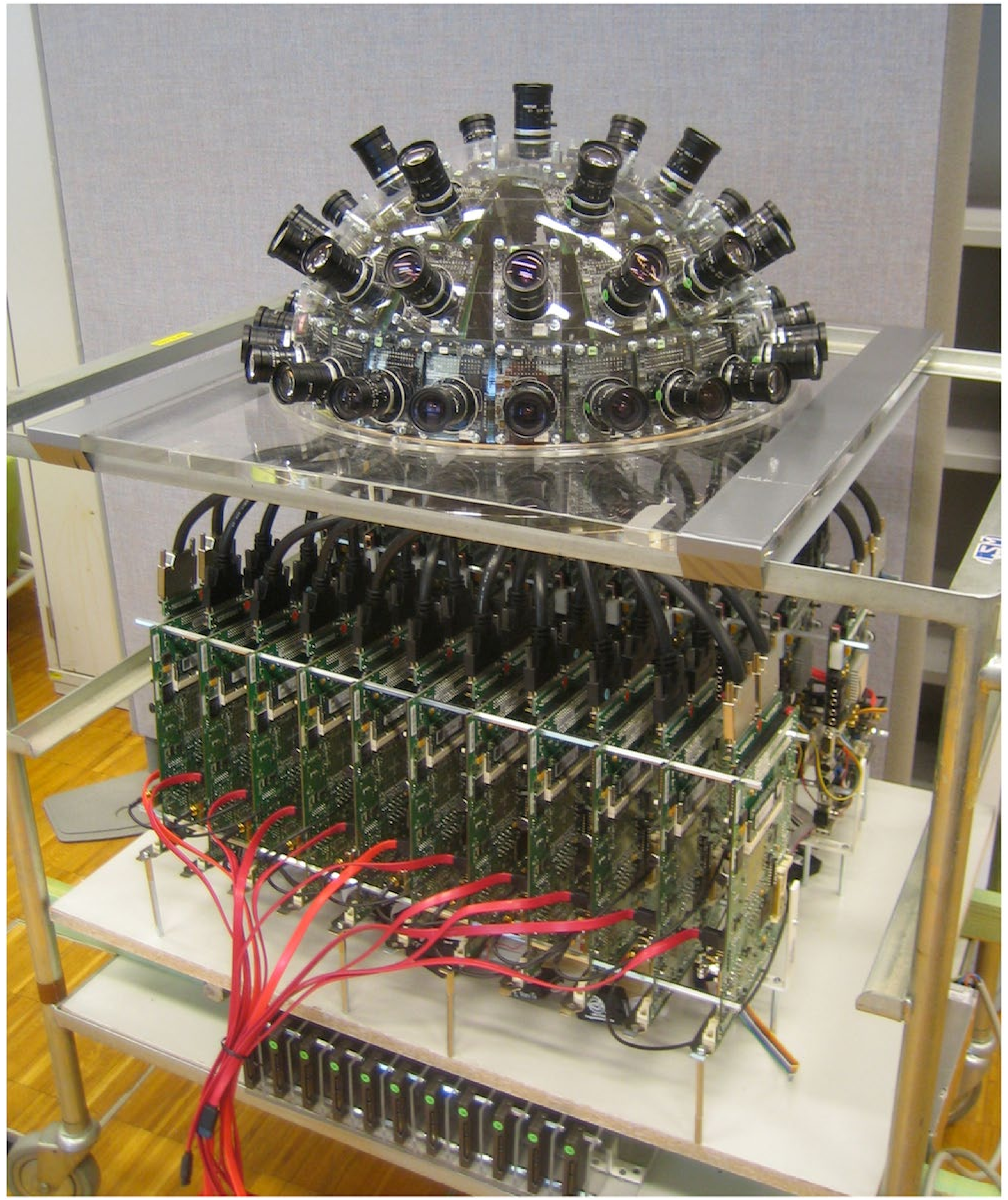
Acquisition - Even Lower Tech.

Acquisition - Stitched Panorama



source: photoanswers.co.uk

Acquisition - High Tech.



lsm.epfl.ch

immersivemedia.com

Acquisition

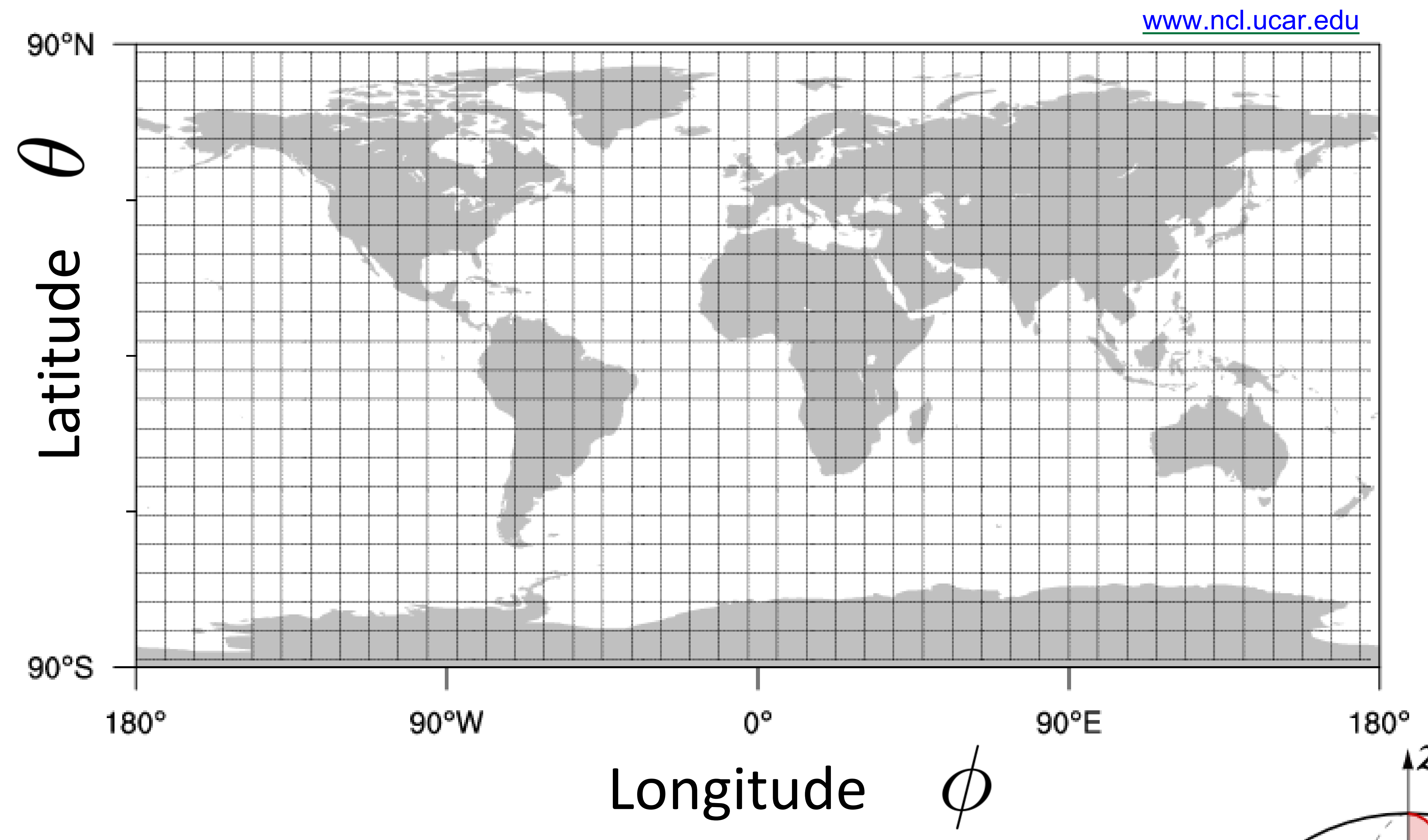
Storing environment maps

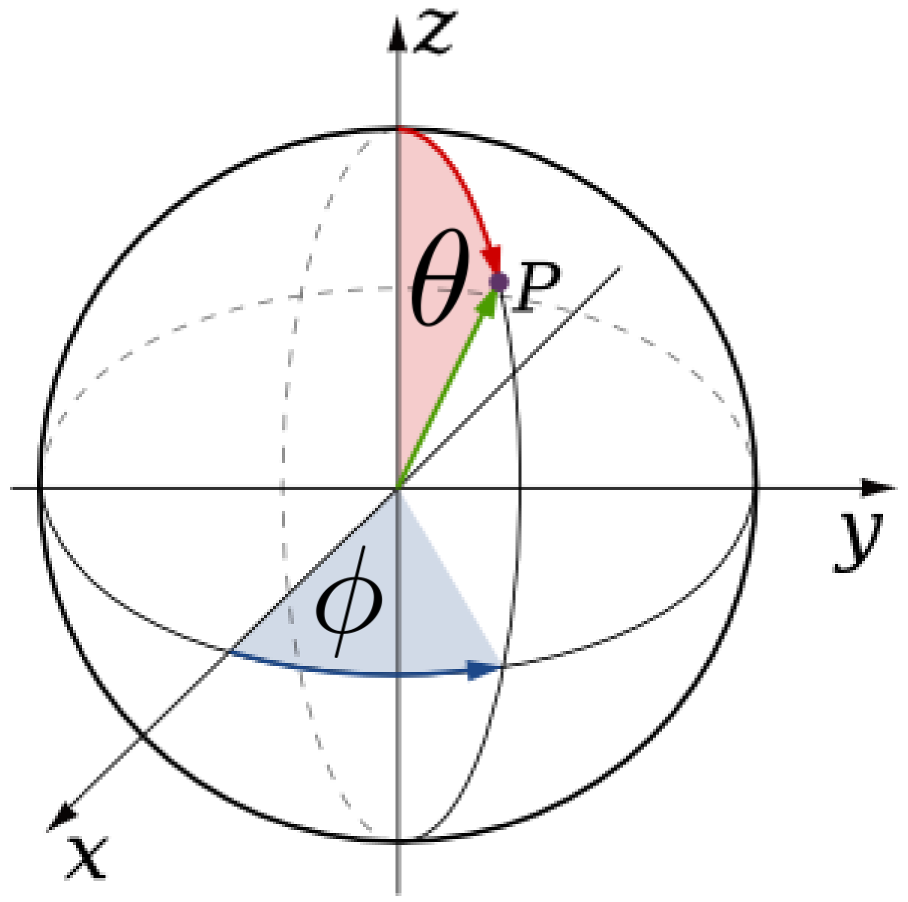
Various ways to parametrize environment maps

Related to cartography

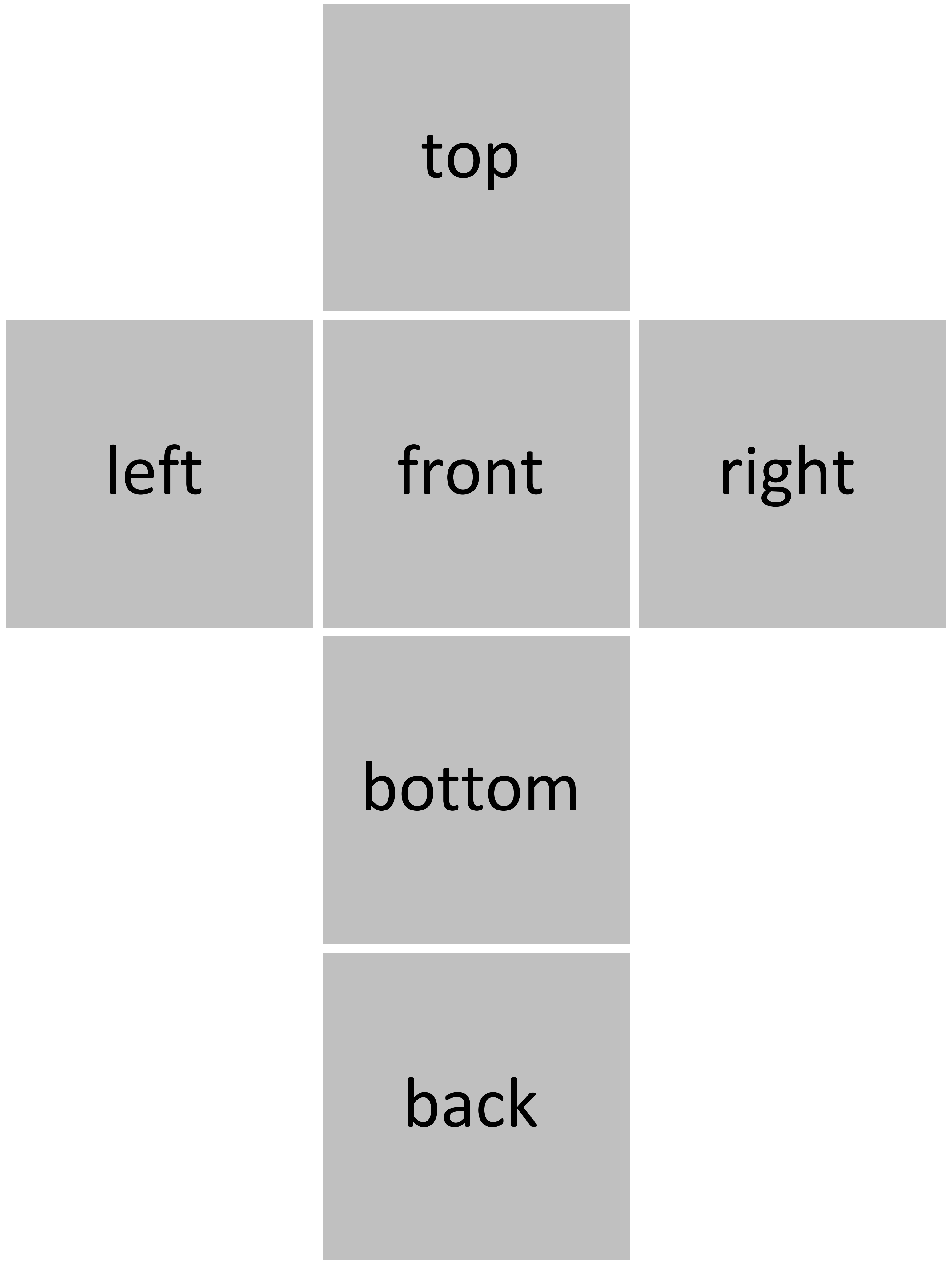
- Projecting the earth (sphere) onto a plane

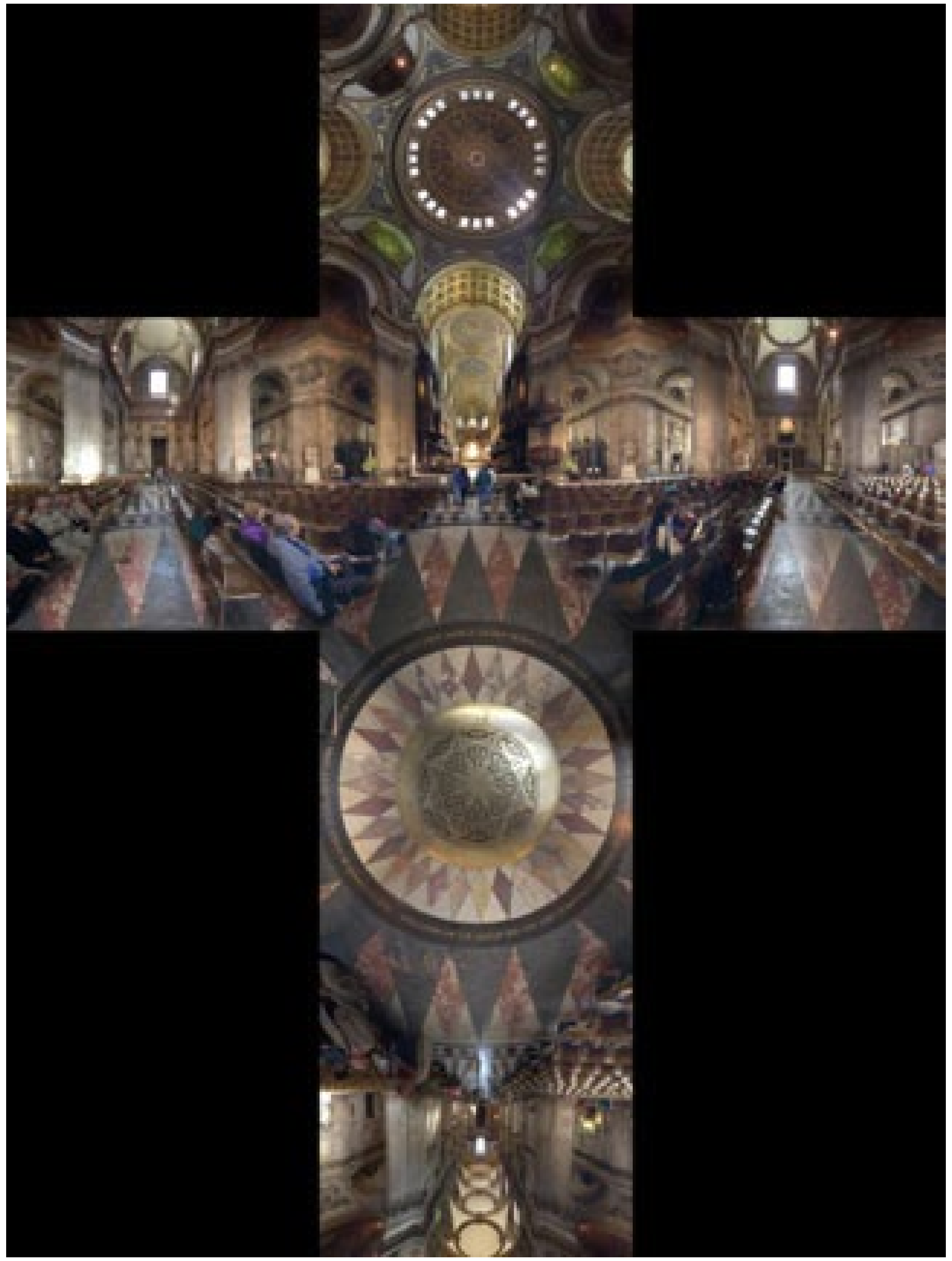
Latitude/Longitude Map





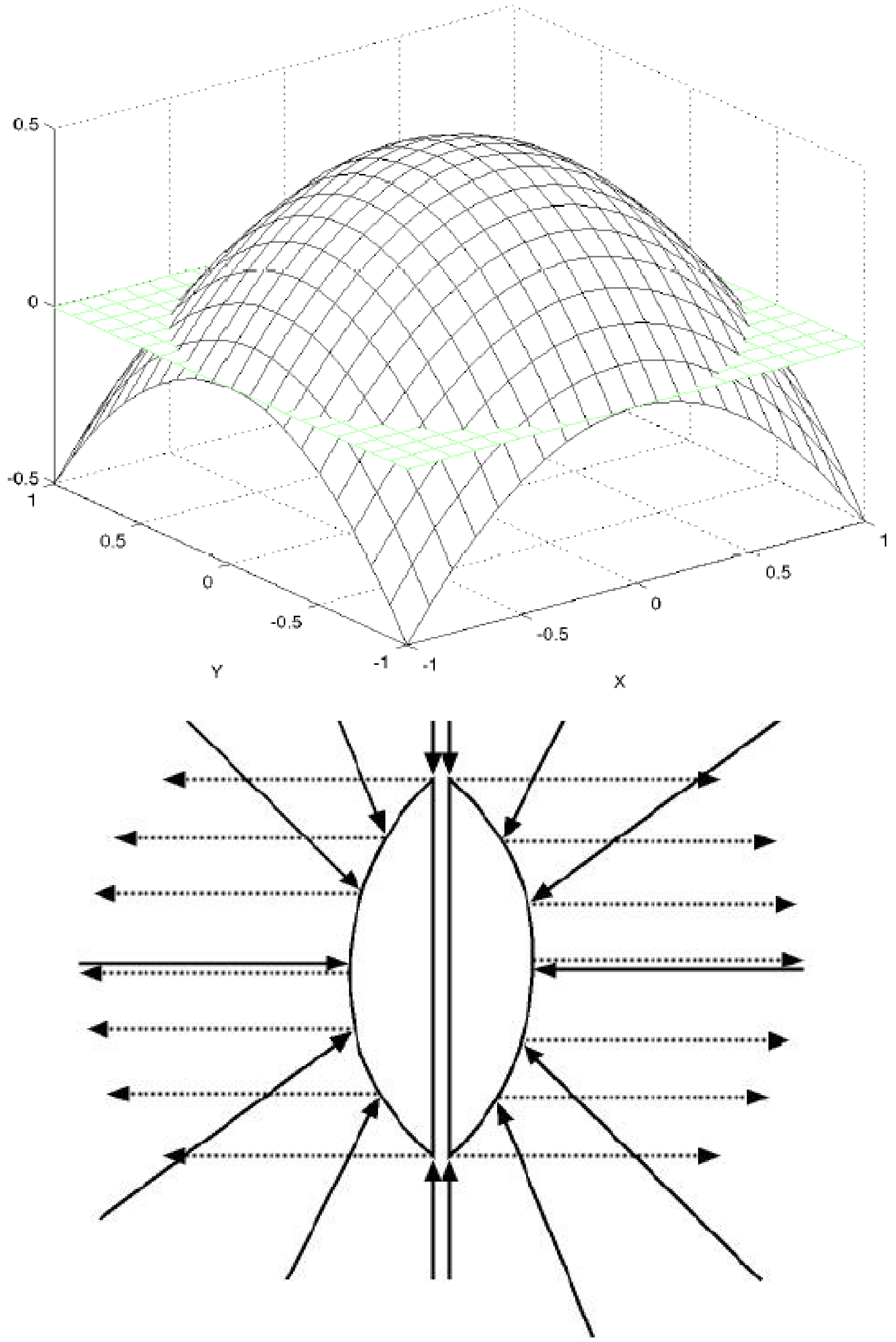
Cube Map (Skybox)





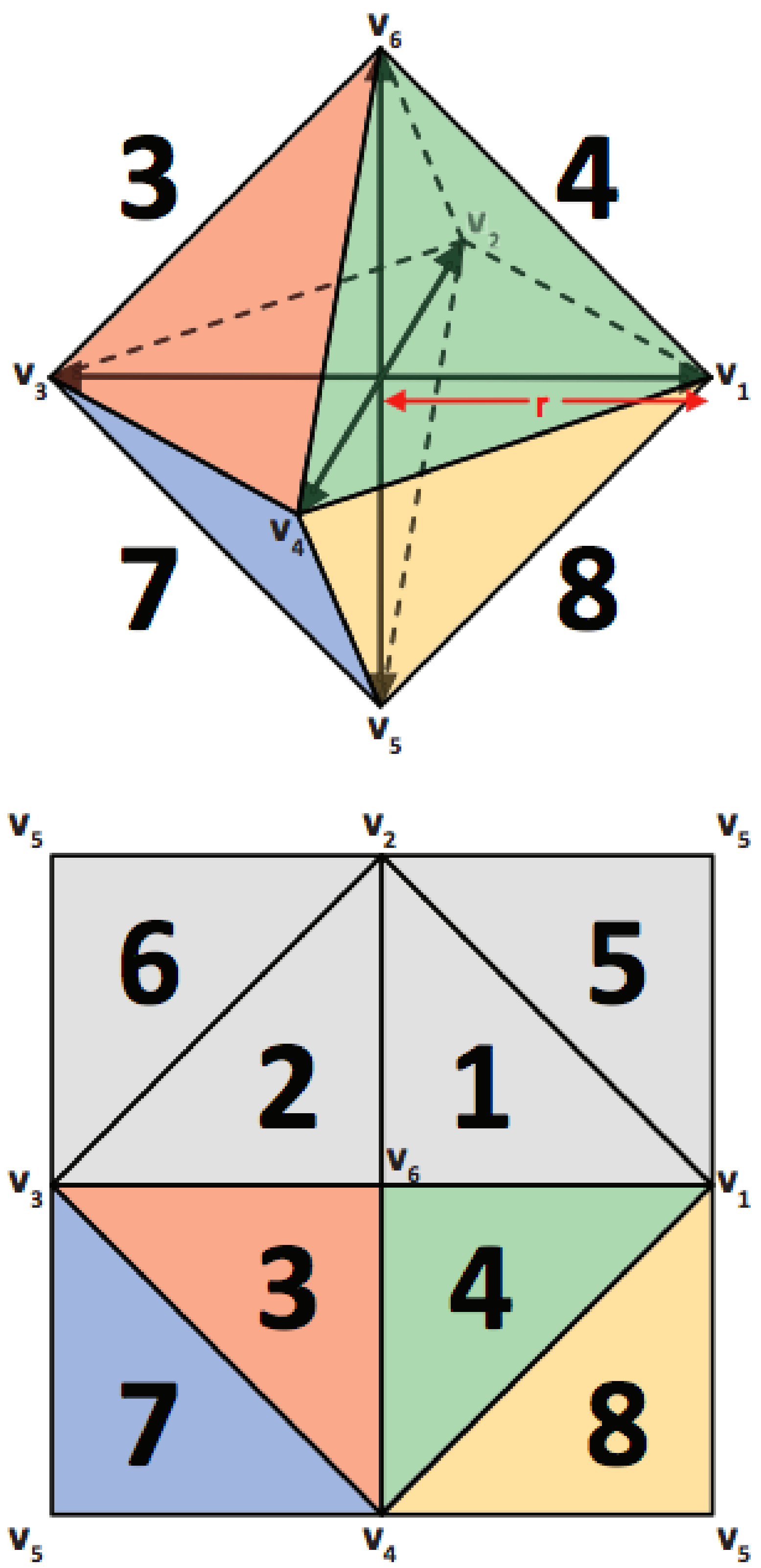
Other Parameterizations

Dual-paraboloid



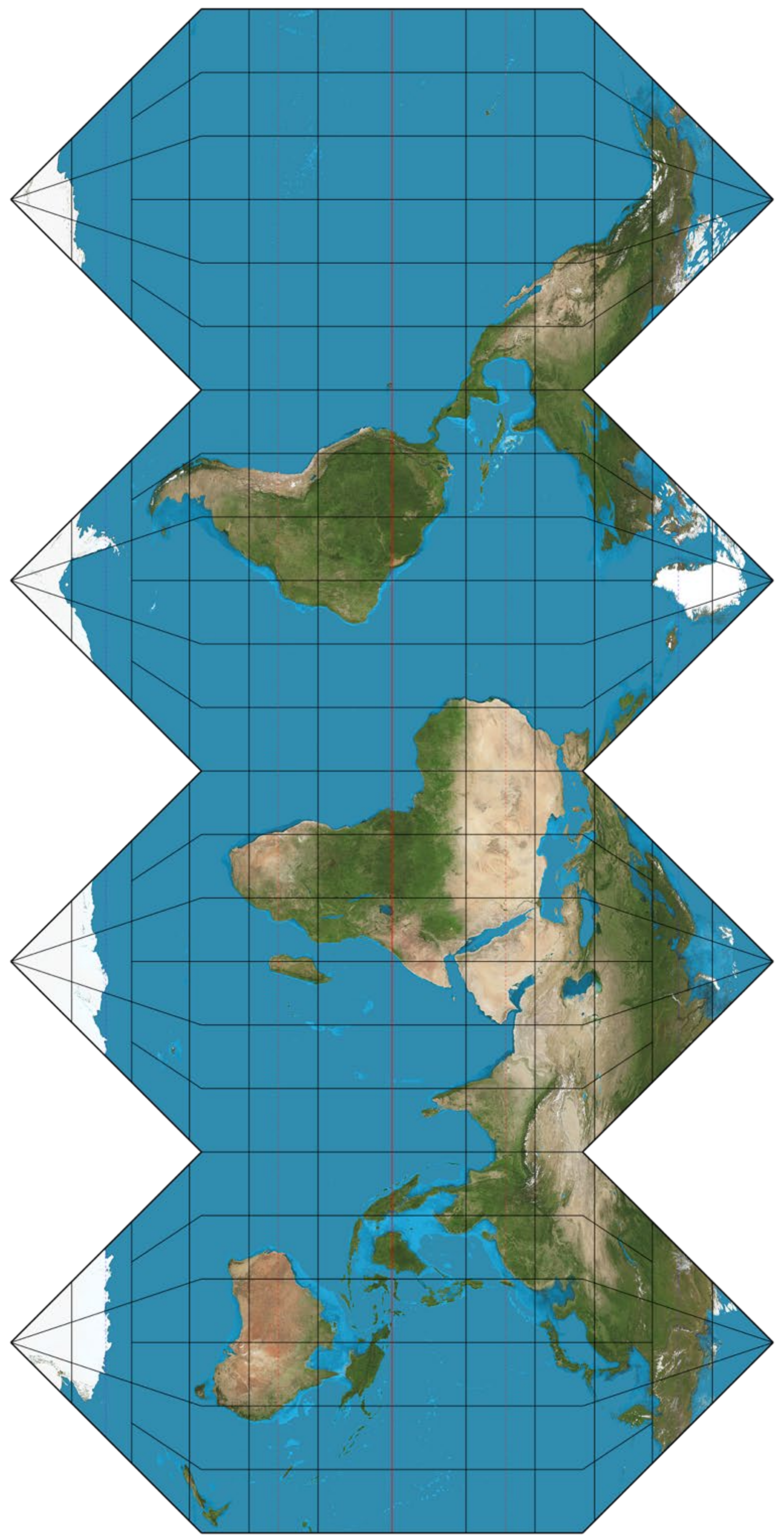
Courtesy of Brabec

Octahedron map



Courtesy of Engelhardt and Dachsbacher

HEALPix



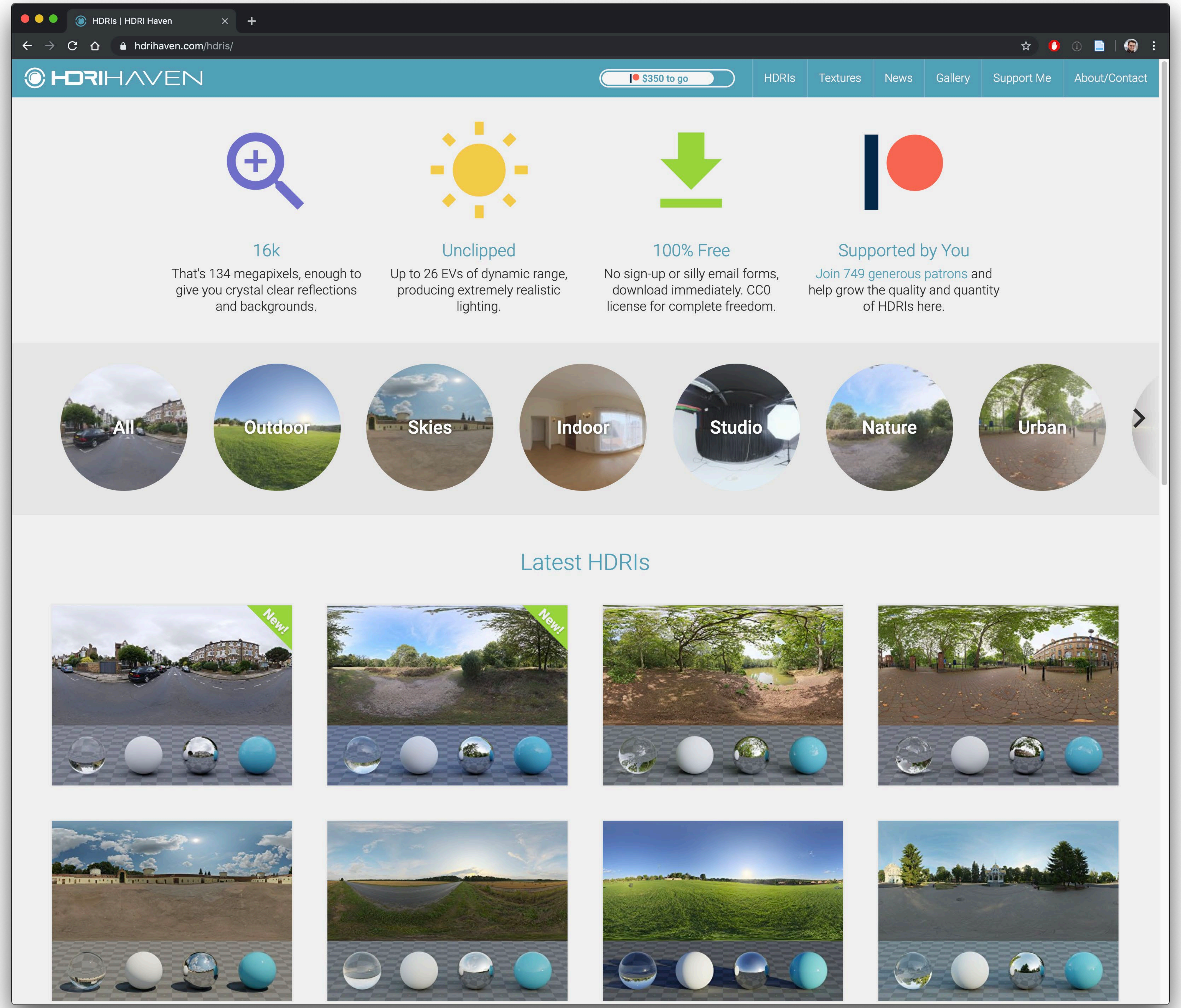
Courtesy of Ryazanov

Get them online

hdrihaven.com

The original:

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

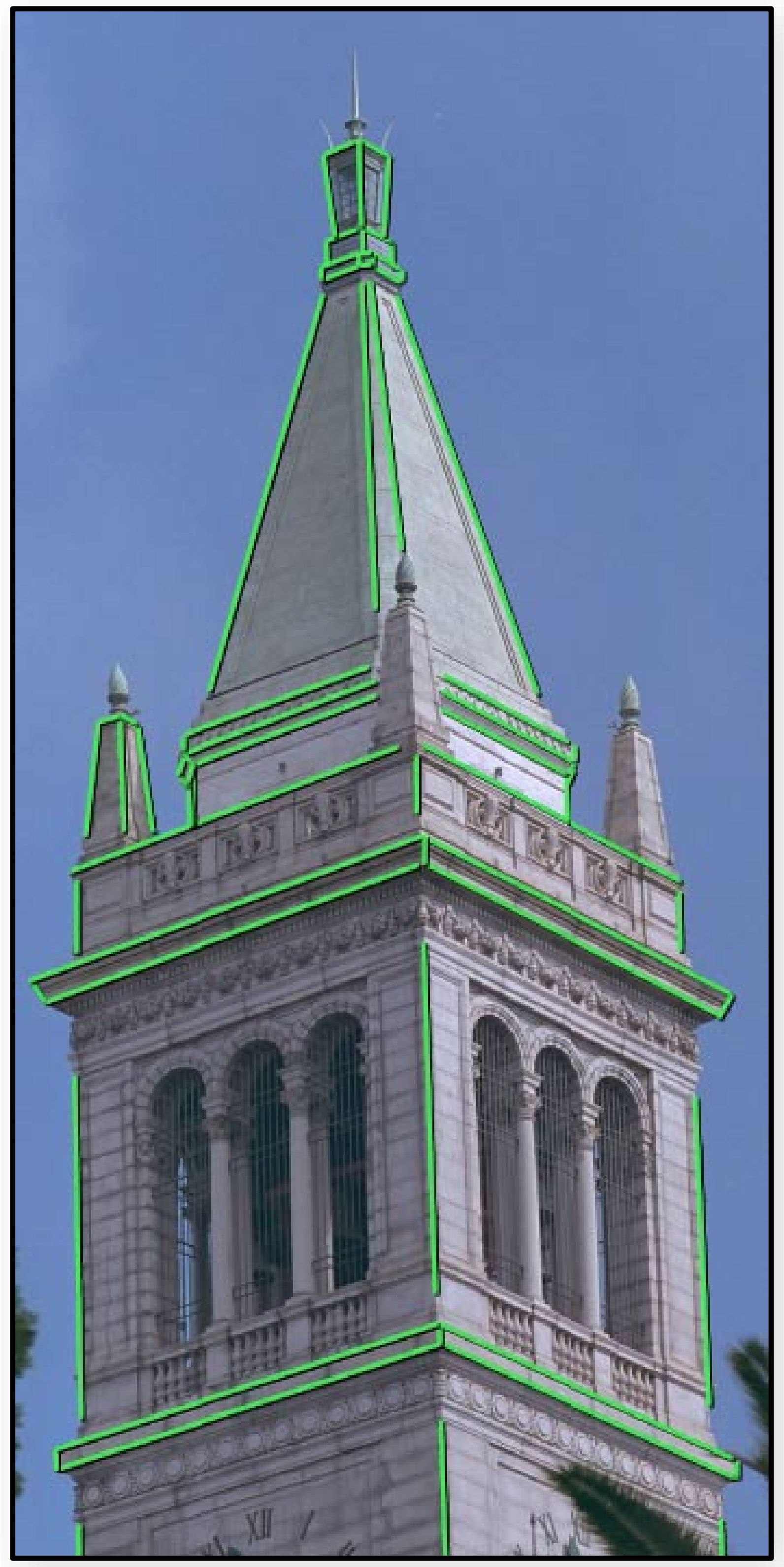
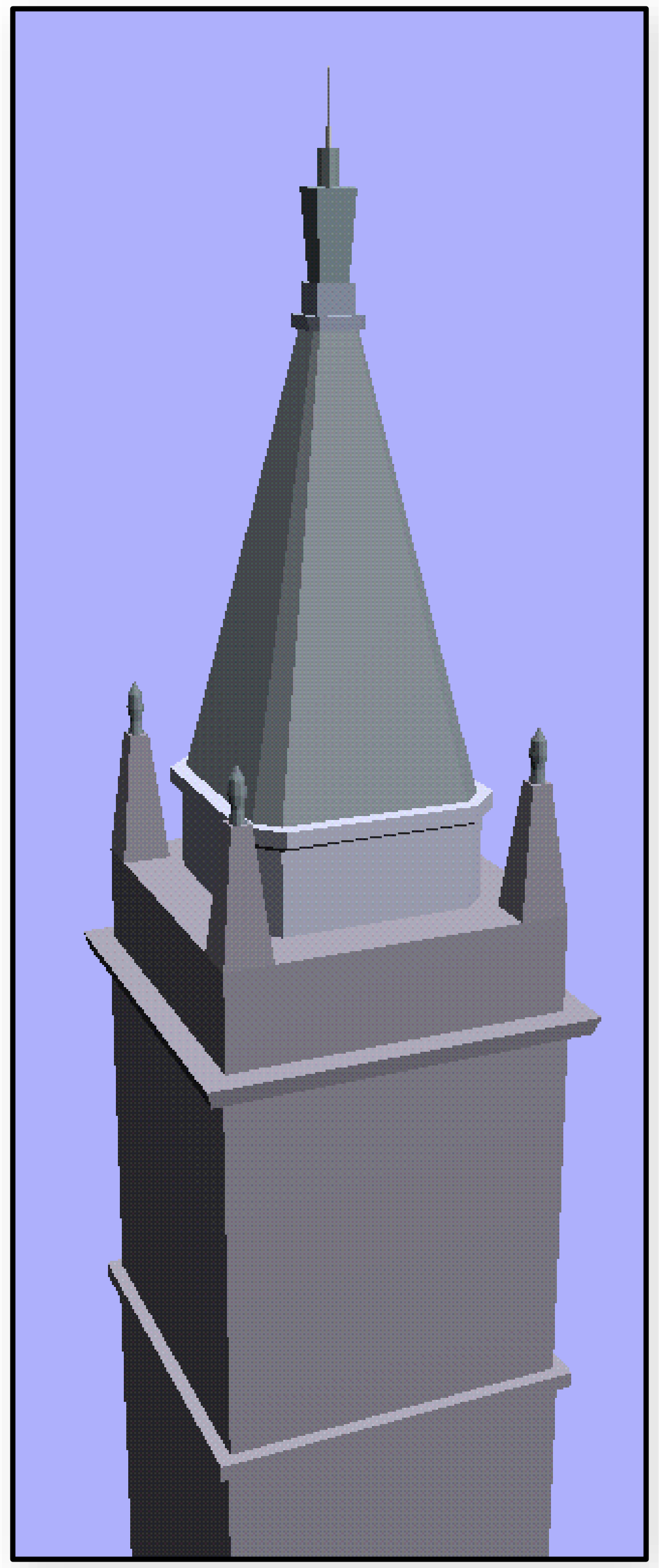
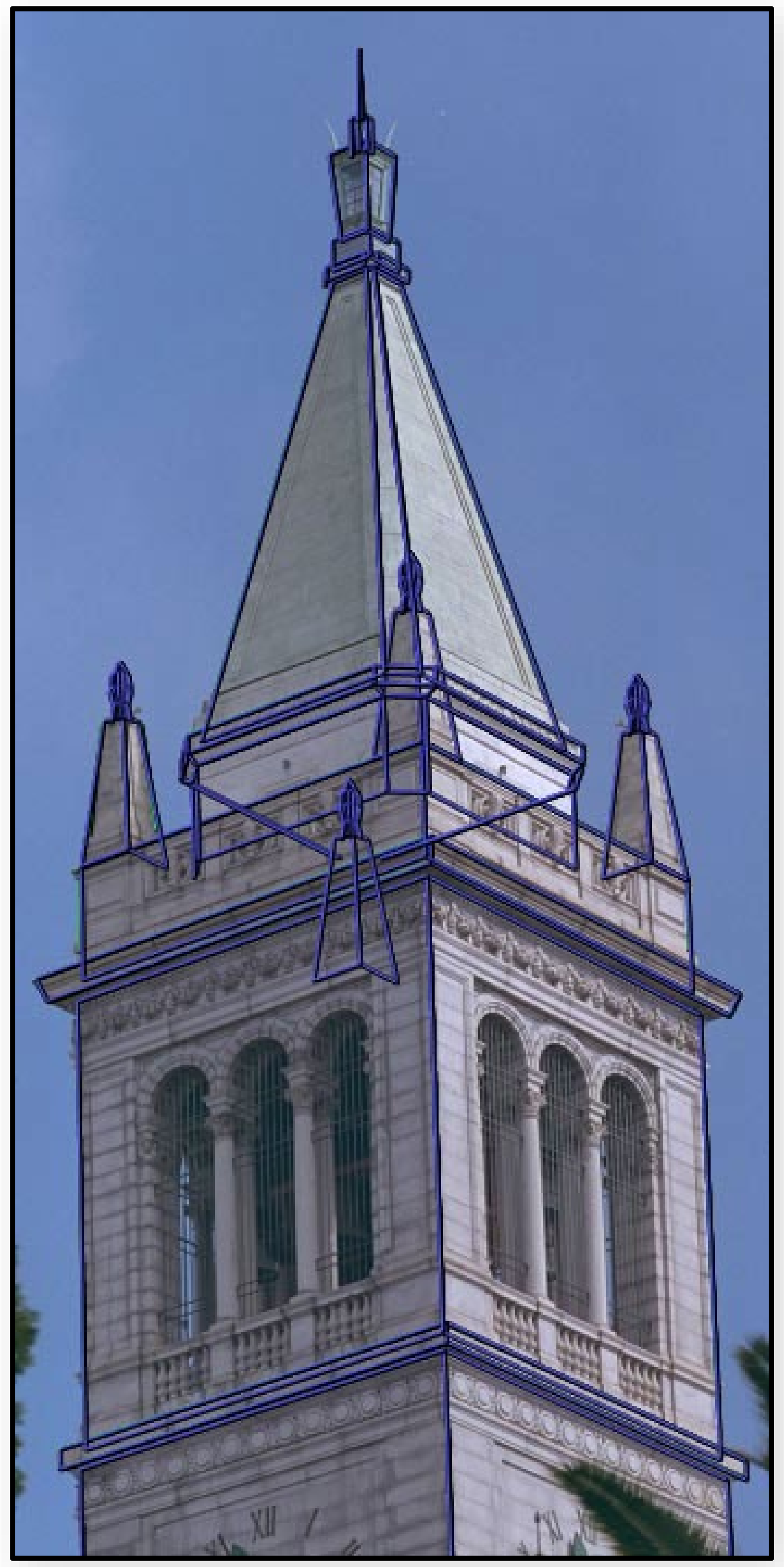
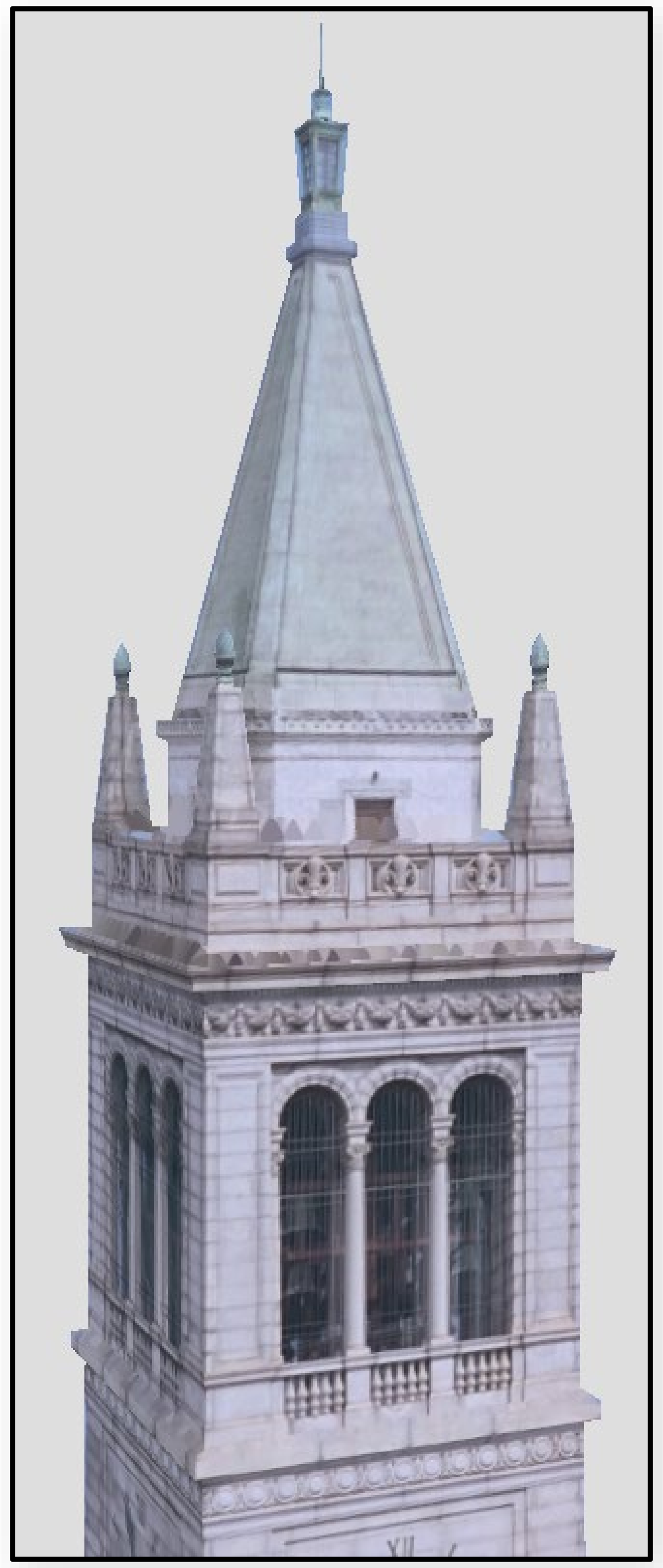


Exploit texturing!

Projective texture example

Modeling from photographs

Using input photos as textures

After a slide by Emily Whiting

[Debevec et al. 1996]

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/