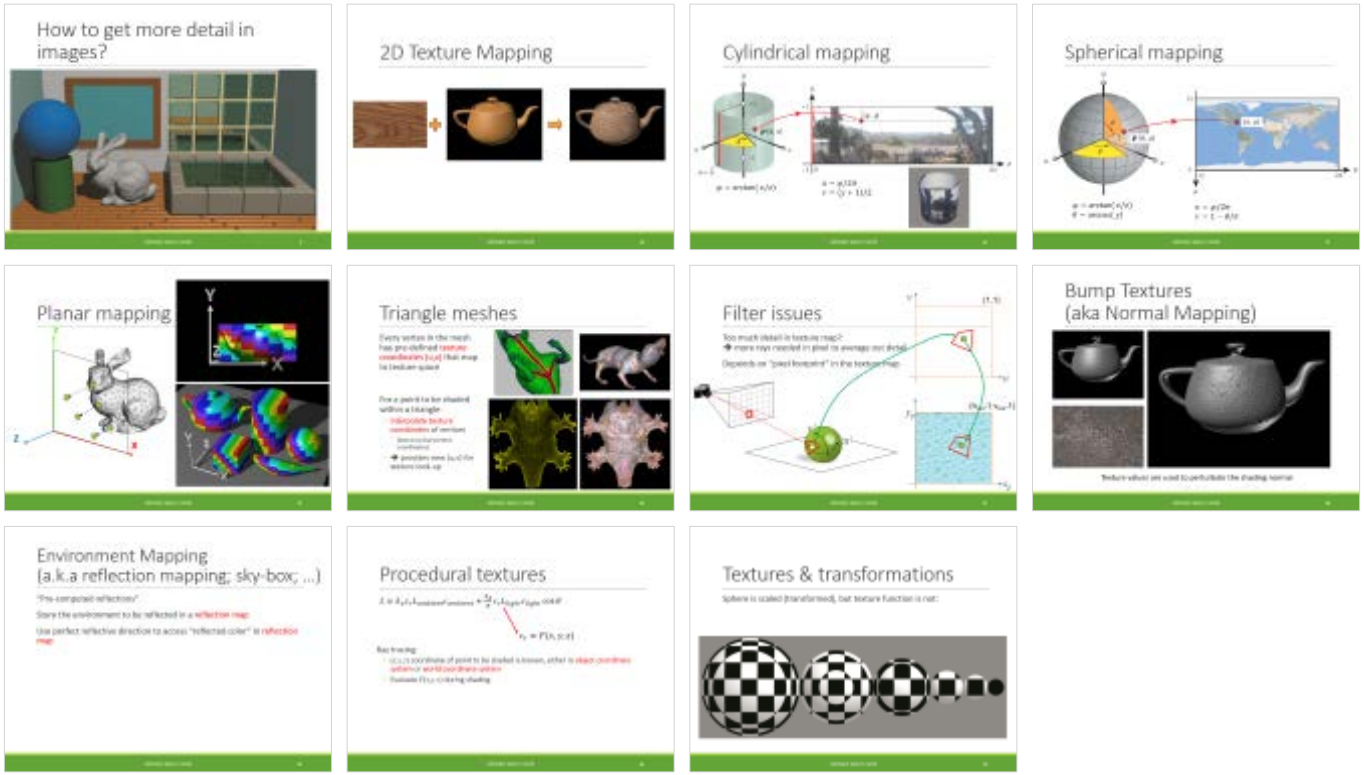
Texture Mapping

FUNDAMENTALS OF COMPUTER GRAPHICS

PHILIP DUTRÉ
DEPARTMENT OF COMPUTER SCIENCE

Overview Lecture



Relevant sections in book: Chapter 29, 30
(Illustrations from *Ray Tracing From The Ground Up*, *Physically-Based Rendering*, *Fundamentals of Computer Graphics*)

How to get more detail in images?

How to get more detail in images?

How to get more detail in images?

How to get more detail in images?

https://www.xooon.be/xn/salons/hoeksalons/bellagio-2-5-zits-arm-links-longchair-rechts/

How to get more detail in images?

https://sciencebehindpixar.org/pipeline/surfaces

How to get more detail in images?

Either: subdivide a geometric element in smaller elements

- Each small element has its own color and shading properties across the surface

Or: define a spatial-varying function over the surface of the geometric element

- Function describes variations in (shading) properties

Intuitively: detail in shading is defined at a higher resolution than geometric detail

Texture mapping

Modulate shading properties *within* the surface domain of a single geometric object

E.g.

$$L = k_a c_r L_{ambient} c_{ambient} + \frac{k_d}{\pi} c_r L_{light} c_{light} \cos\theta$$

$$c_r = F(x, y, z, \dots)$$

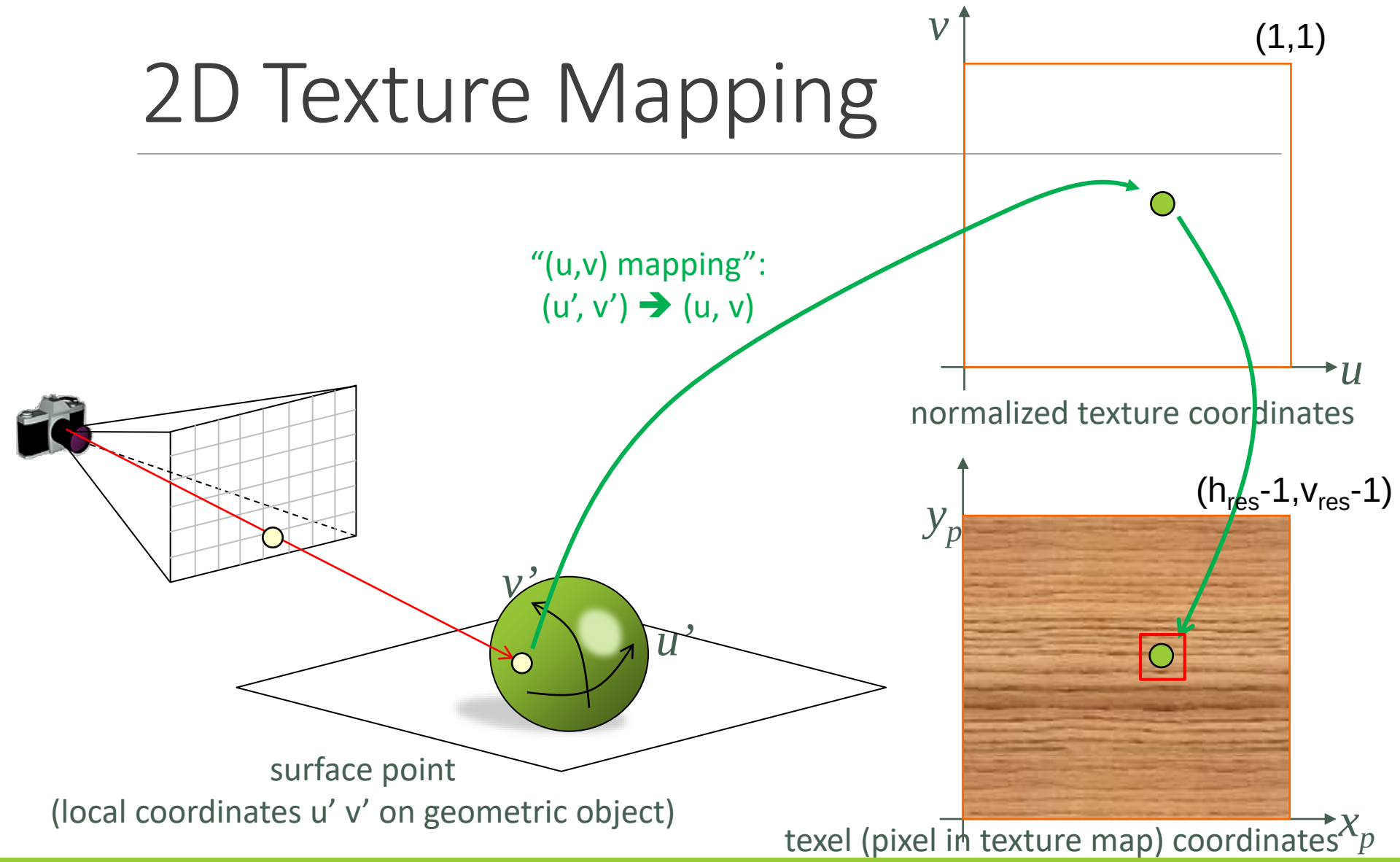
$F(x, y, z, \dots)$ = "texture map"

- … can be a function (procedural textures)
- … can be a look-up table or image (2D texture mapping)
- … can be a noise function (noise-based textures)
- …

2D Texture Mapping

2D Texture Mapping

2D Texture Mapping



2D Texture Mapping

Different (u,v) mappings define different projections of texture on geometric object

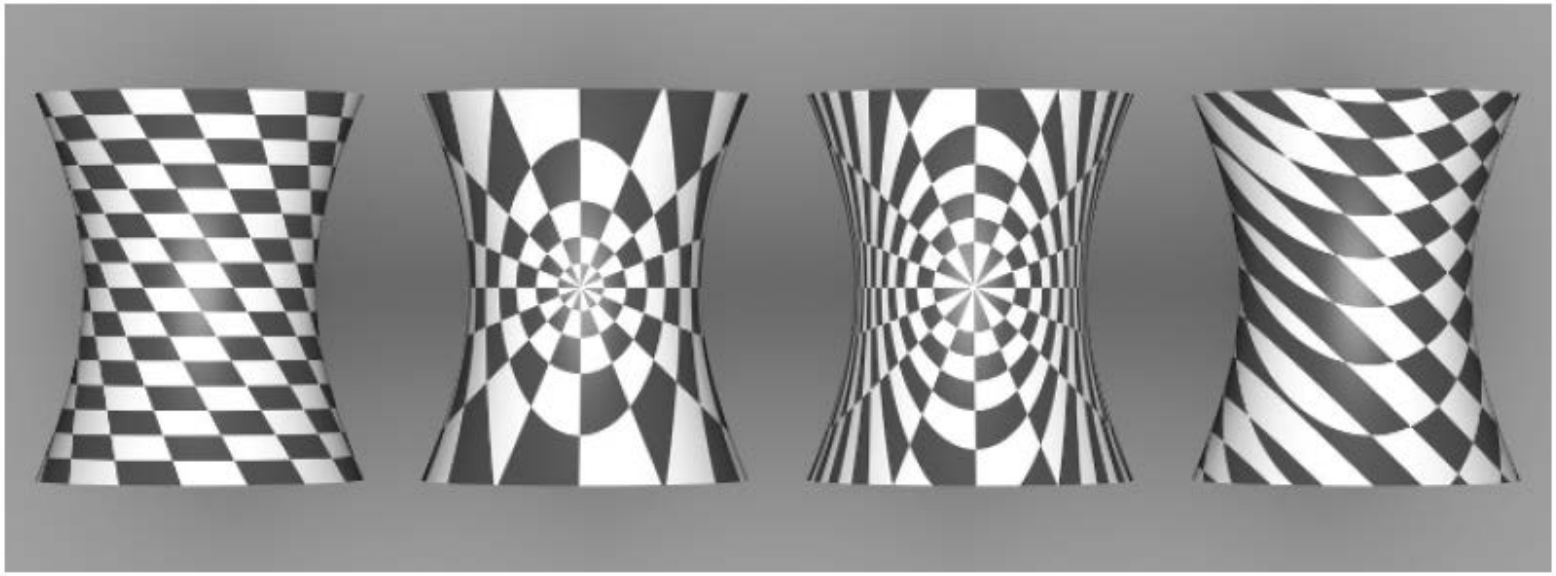
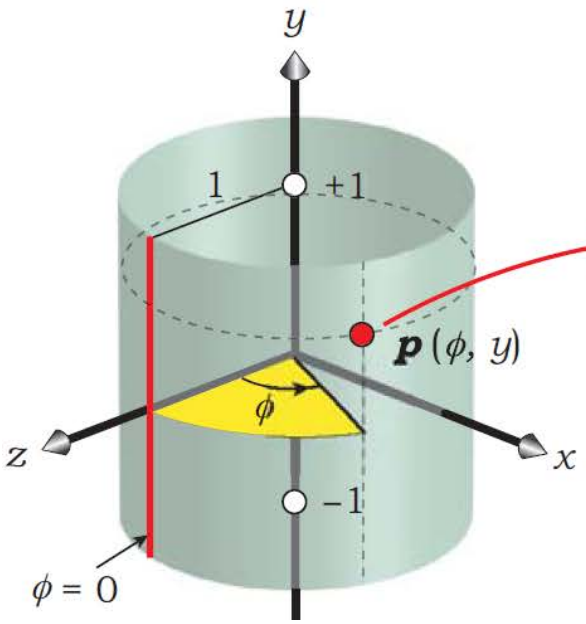
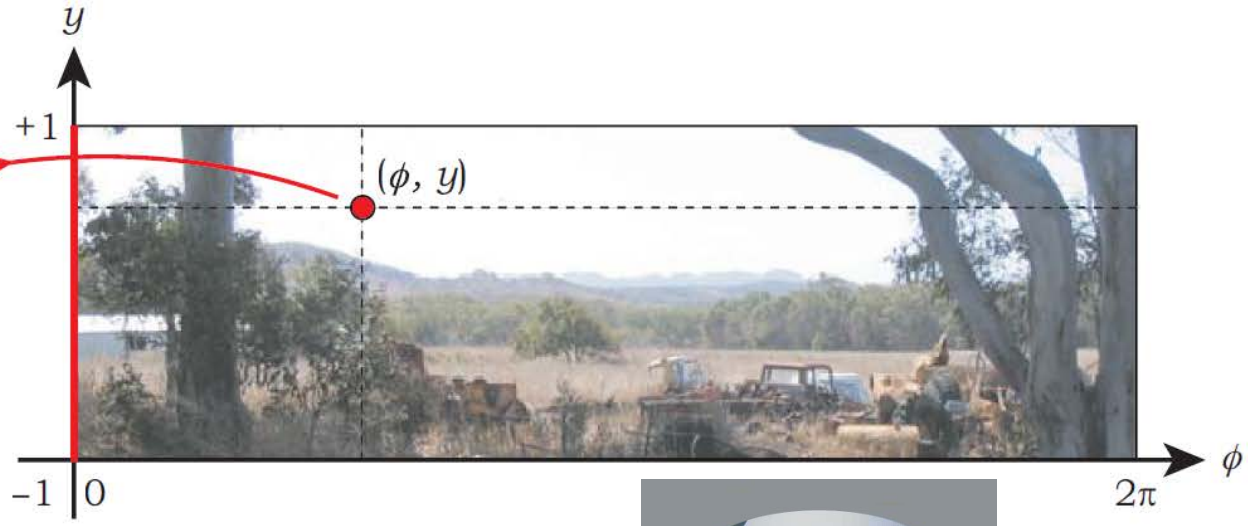


Figure 10.7: A checkerboard texture, applied to a hyperboloid with different texture coordinate generation techniques. From left to right, (u, v) mapping, spherical mapping, cylindrical mapping, and planar mapping.

Cylindrical mapping

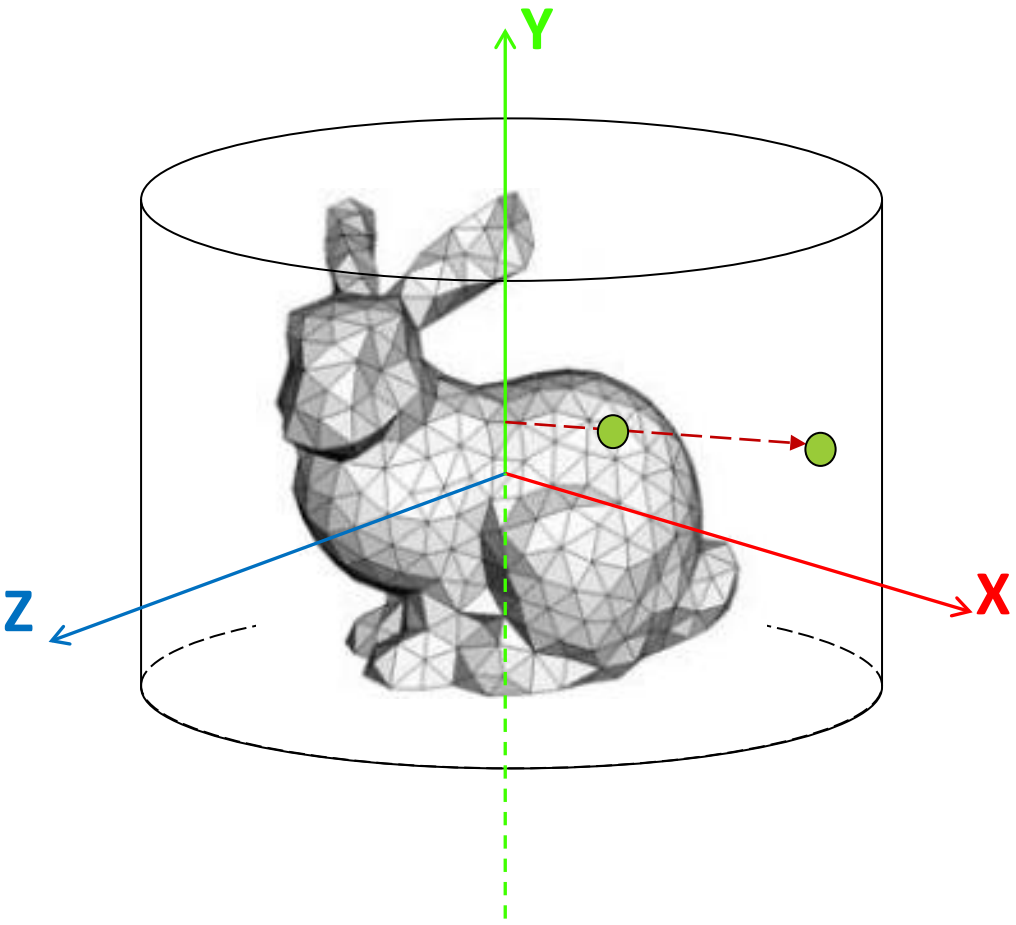


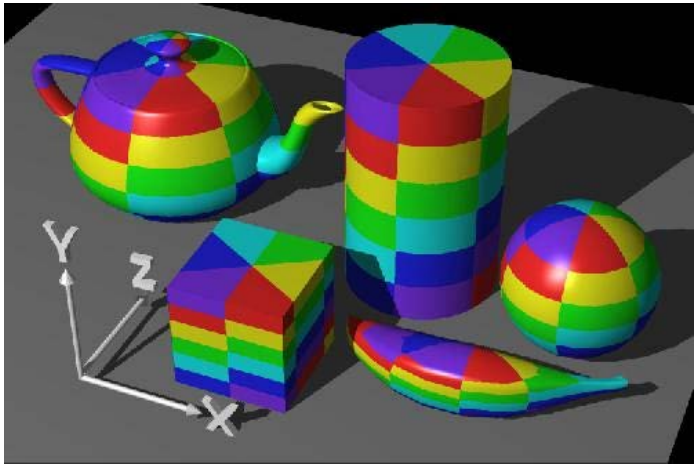
$\varphi = \arctan(x/z)$



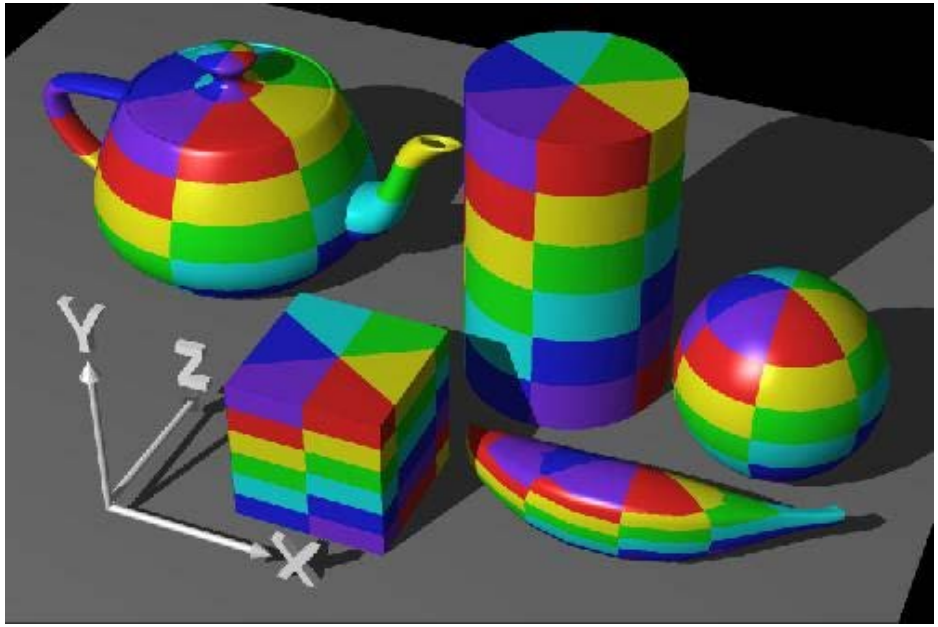
$u = \varphi/2\pi$
$v = (y+1)/2$

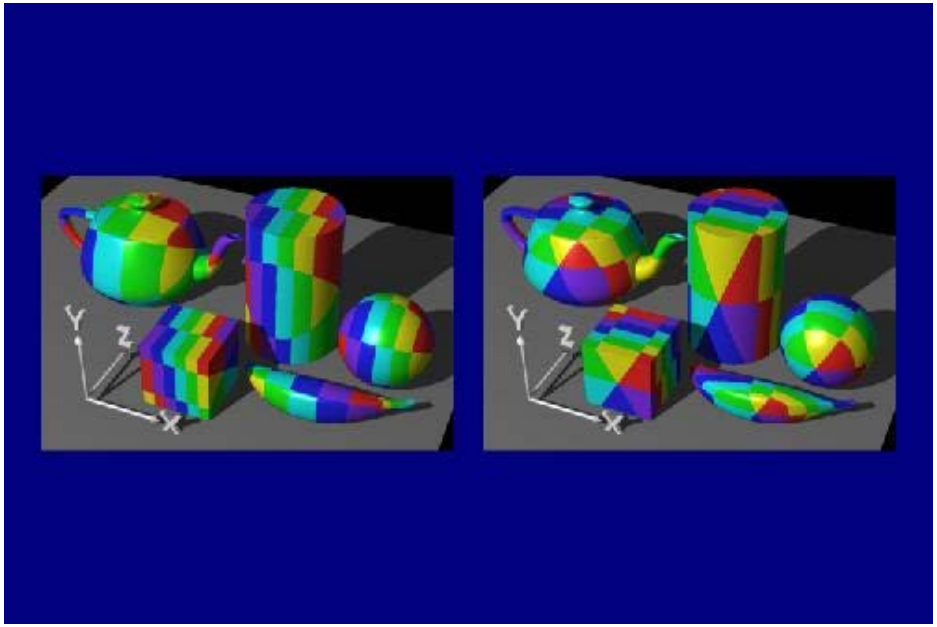
Cylindrical mapping



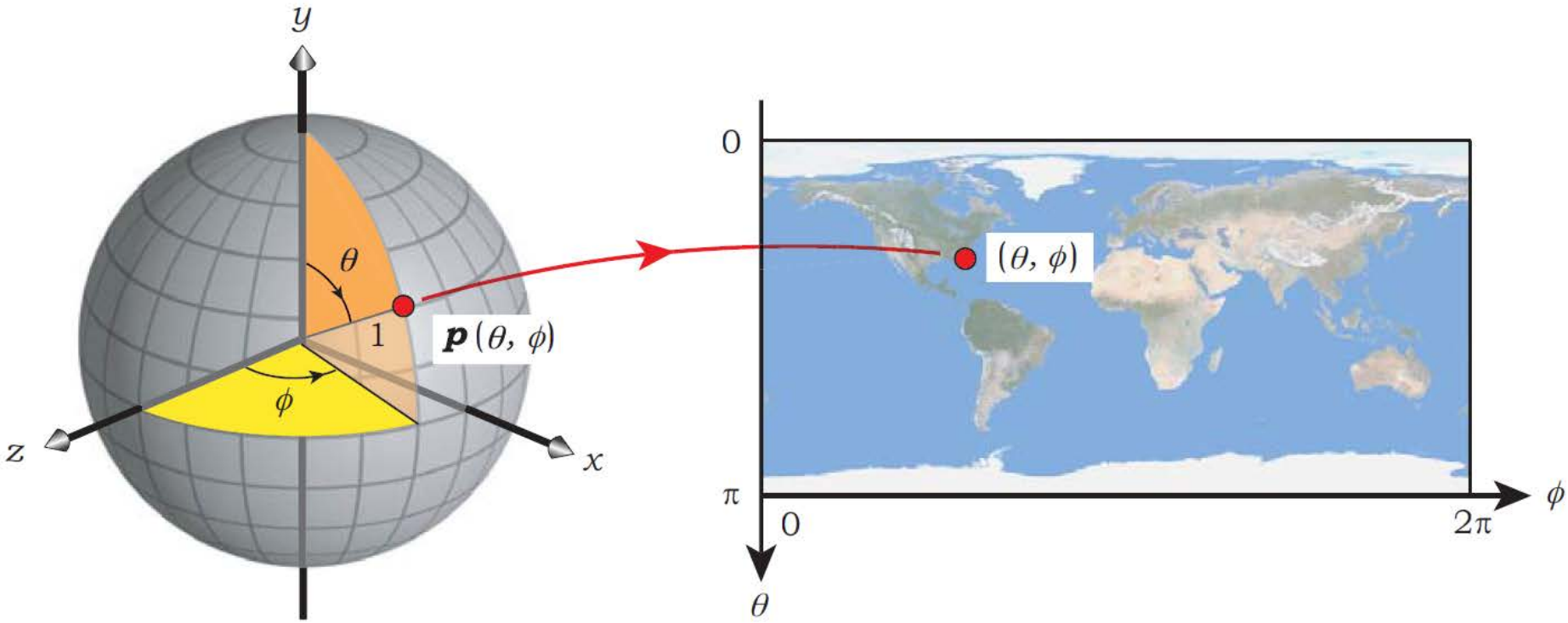


Cylindrical mapping





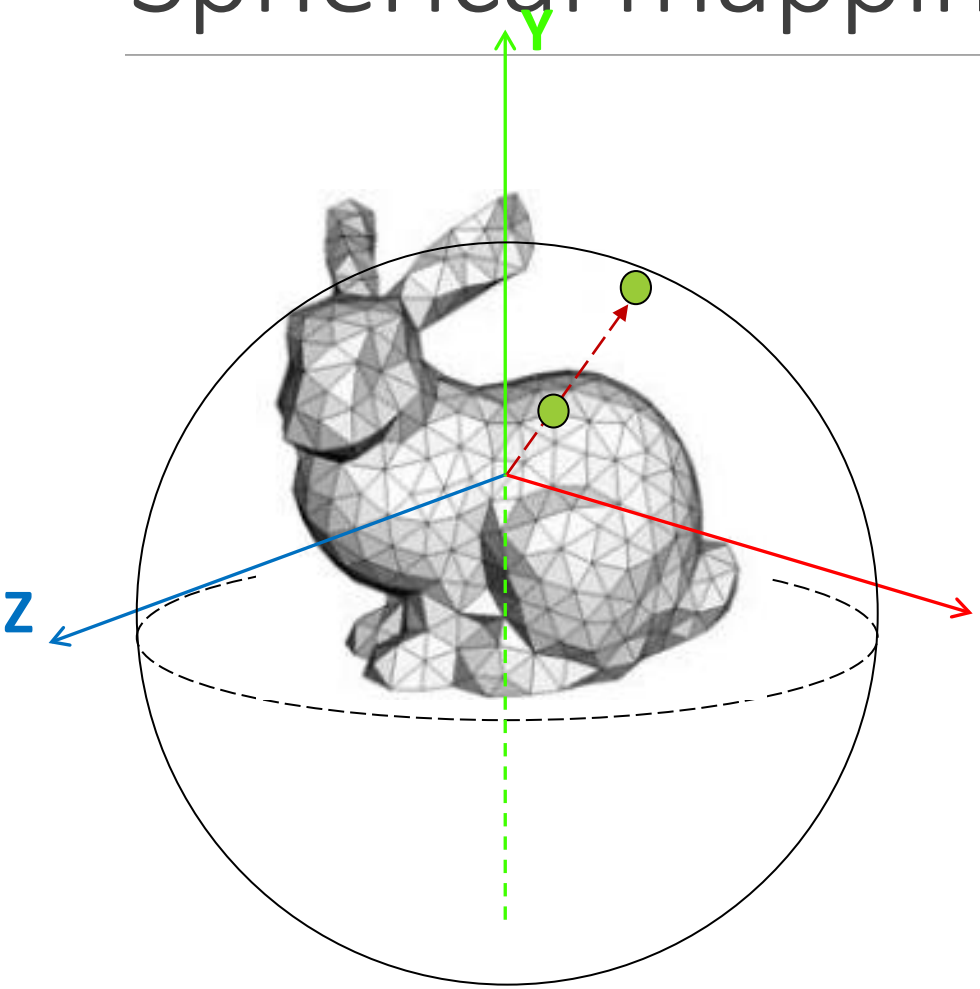
Spherical mapping

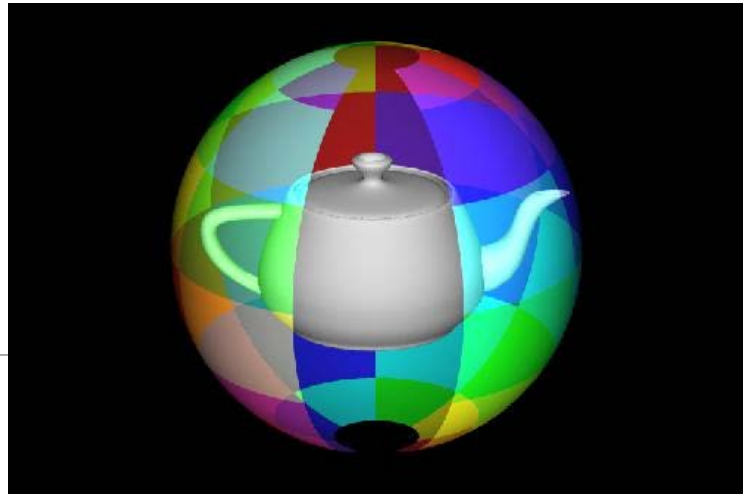


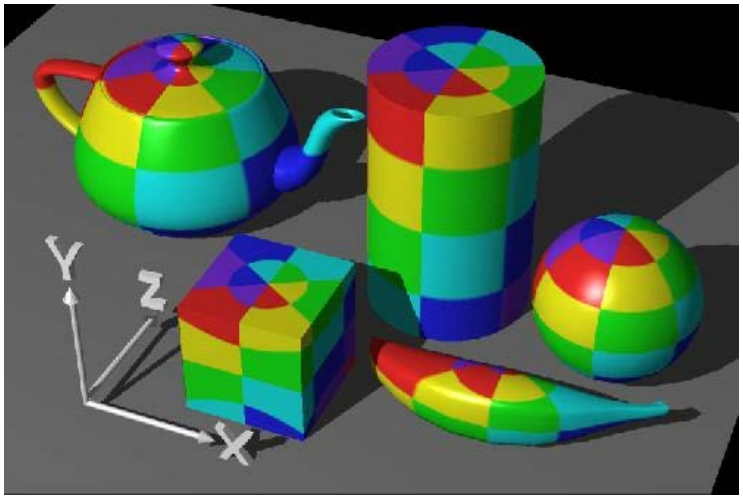
$\varphi = \arctan(x/z)$
$\theta = \arccos(y)$

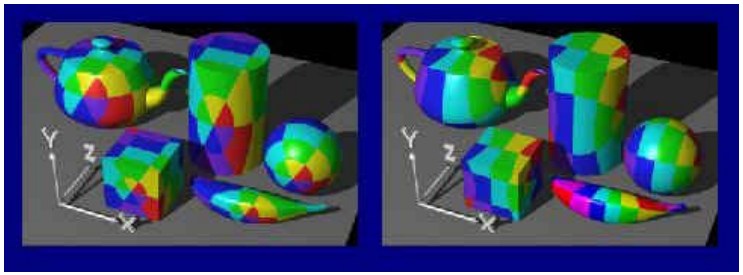
$u = \varphi/2\pi$
$v = 1 - \theta/\pi$

Spherical mapping

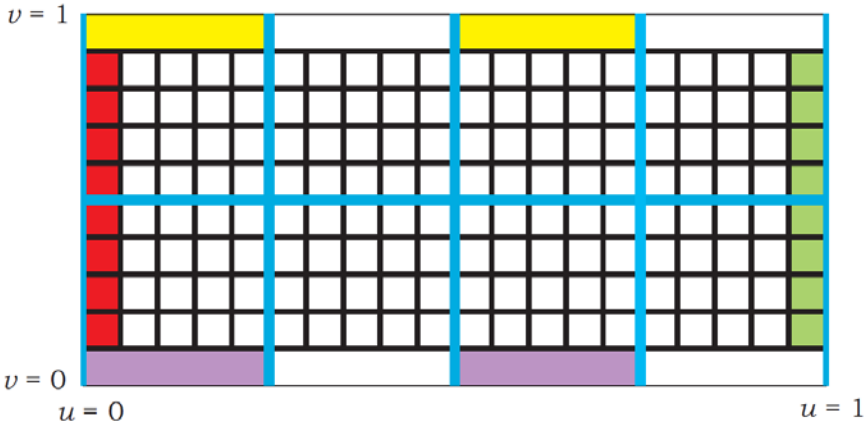


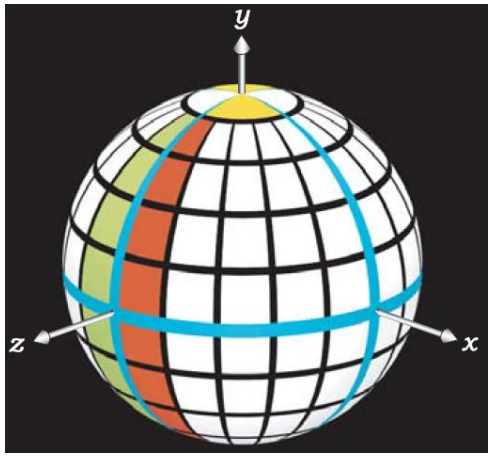


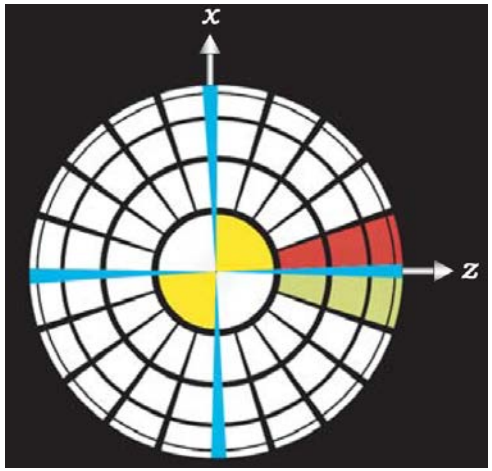




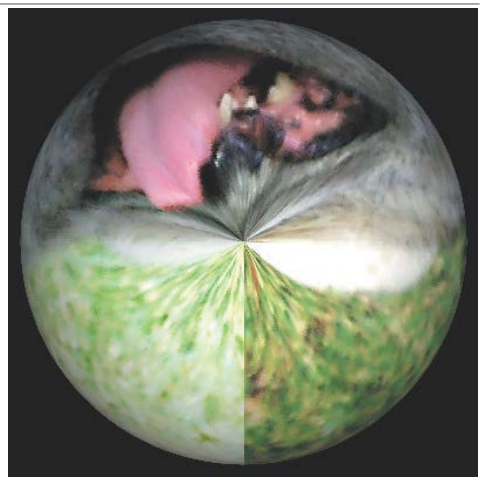
Spherical mapping: distortions





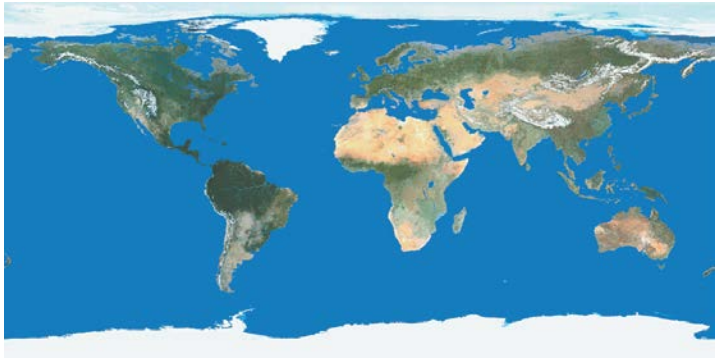


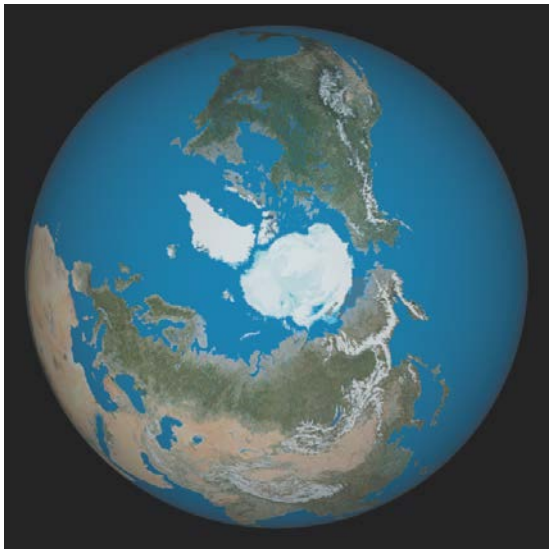
Spherical mapping: distortions



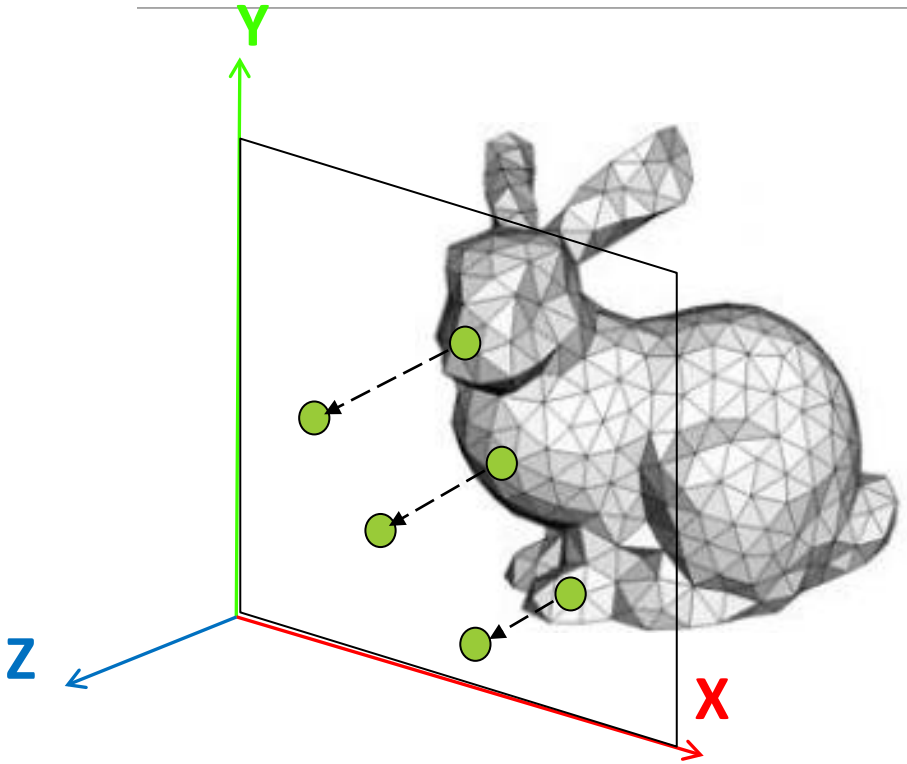
Spherical mapping: distortions

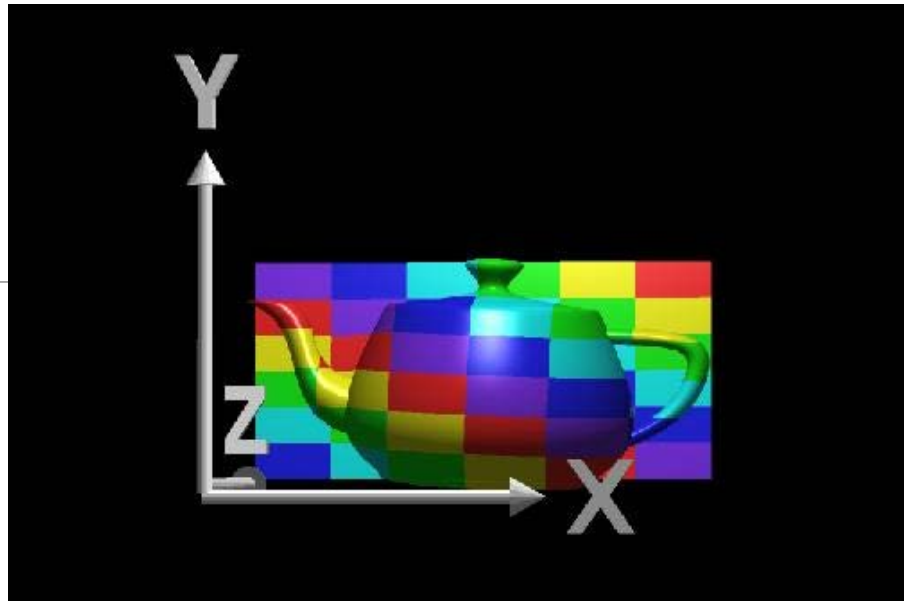
(pre-distort image)

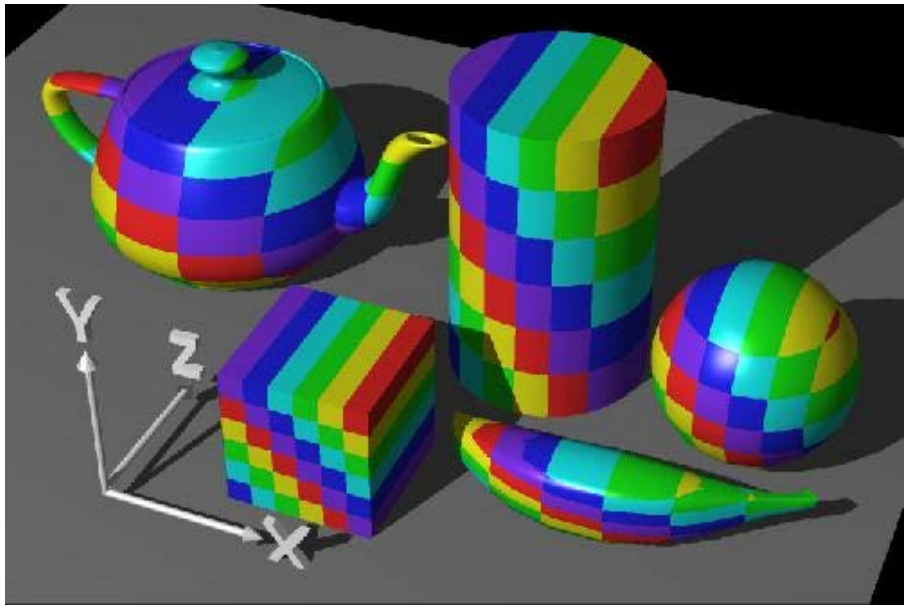




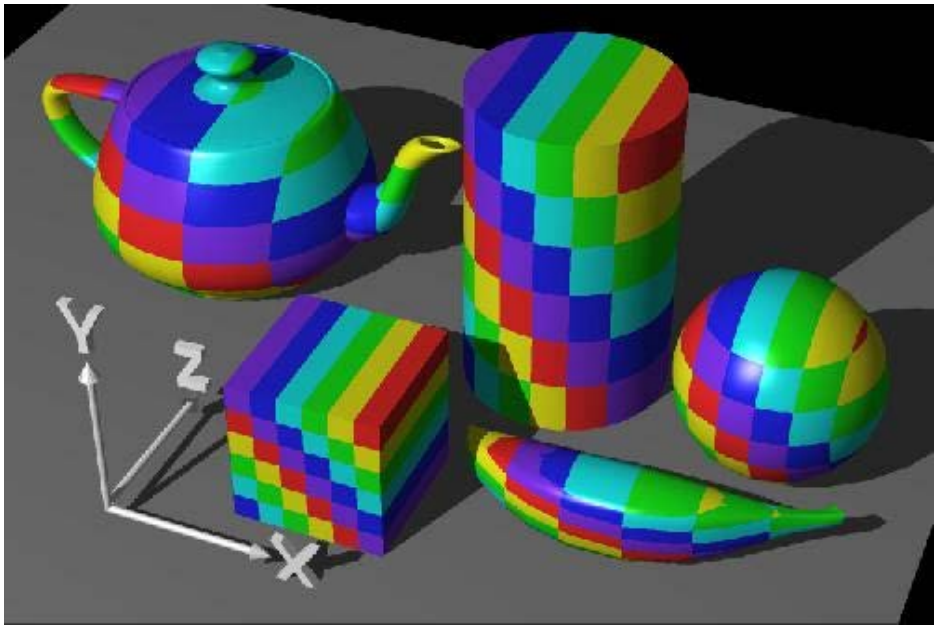
Planar mapping

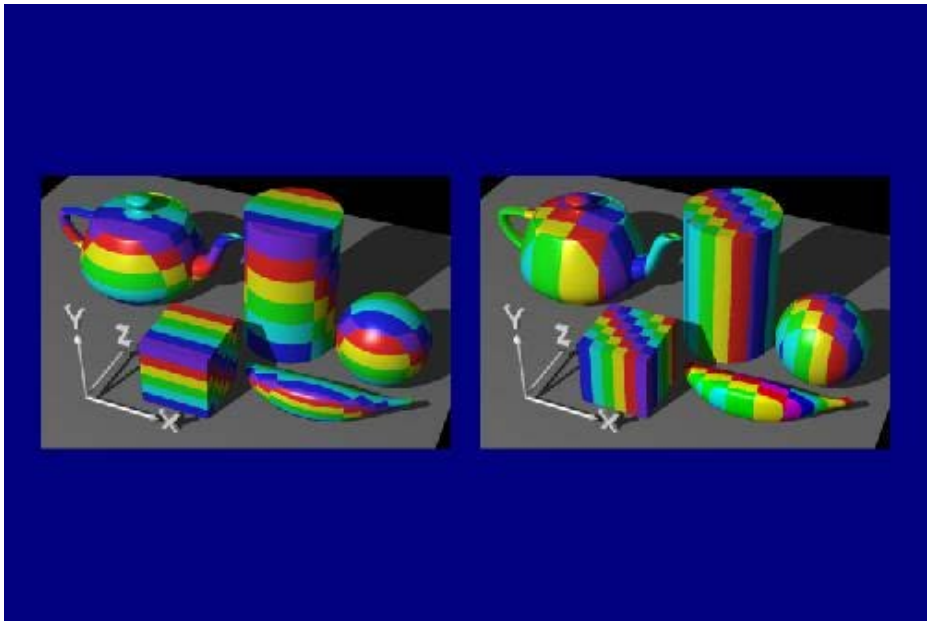






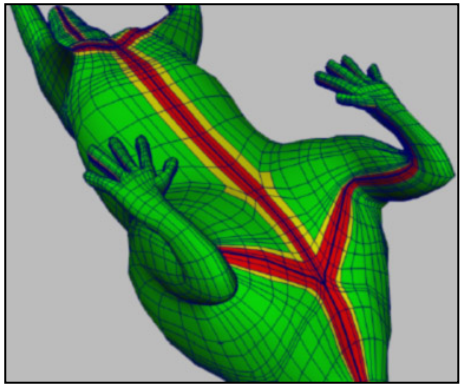
Planar mapping





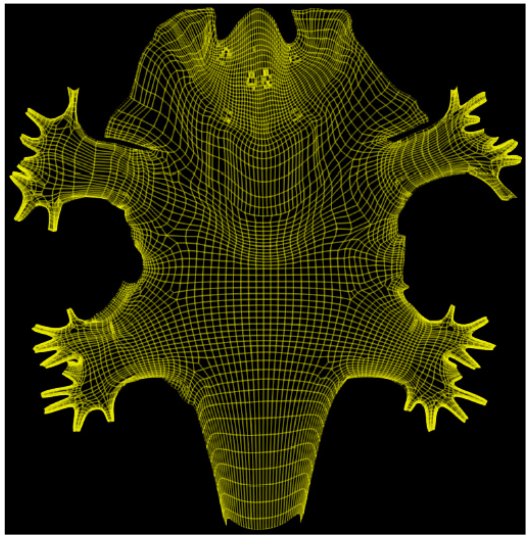
Triangle meshes

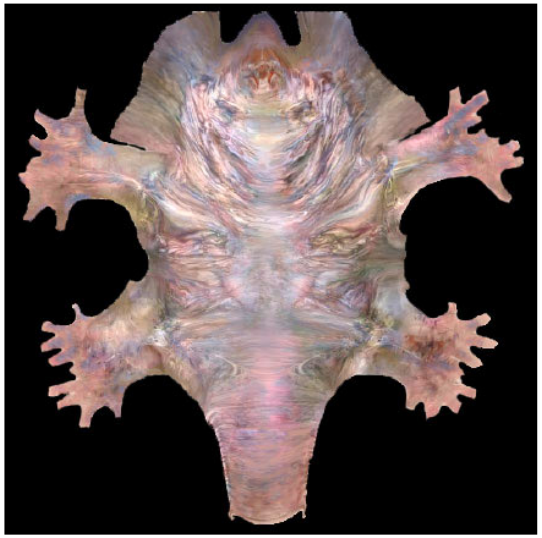
Every vertex in the mesh has pre-defined texture coordinates (u,v) that map to texture space



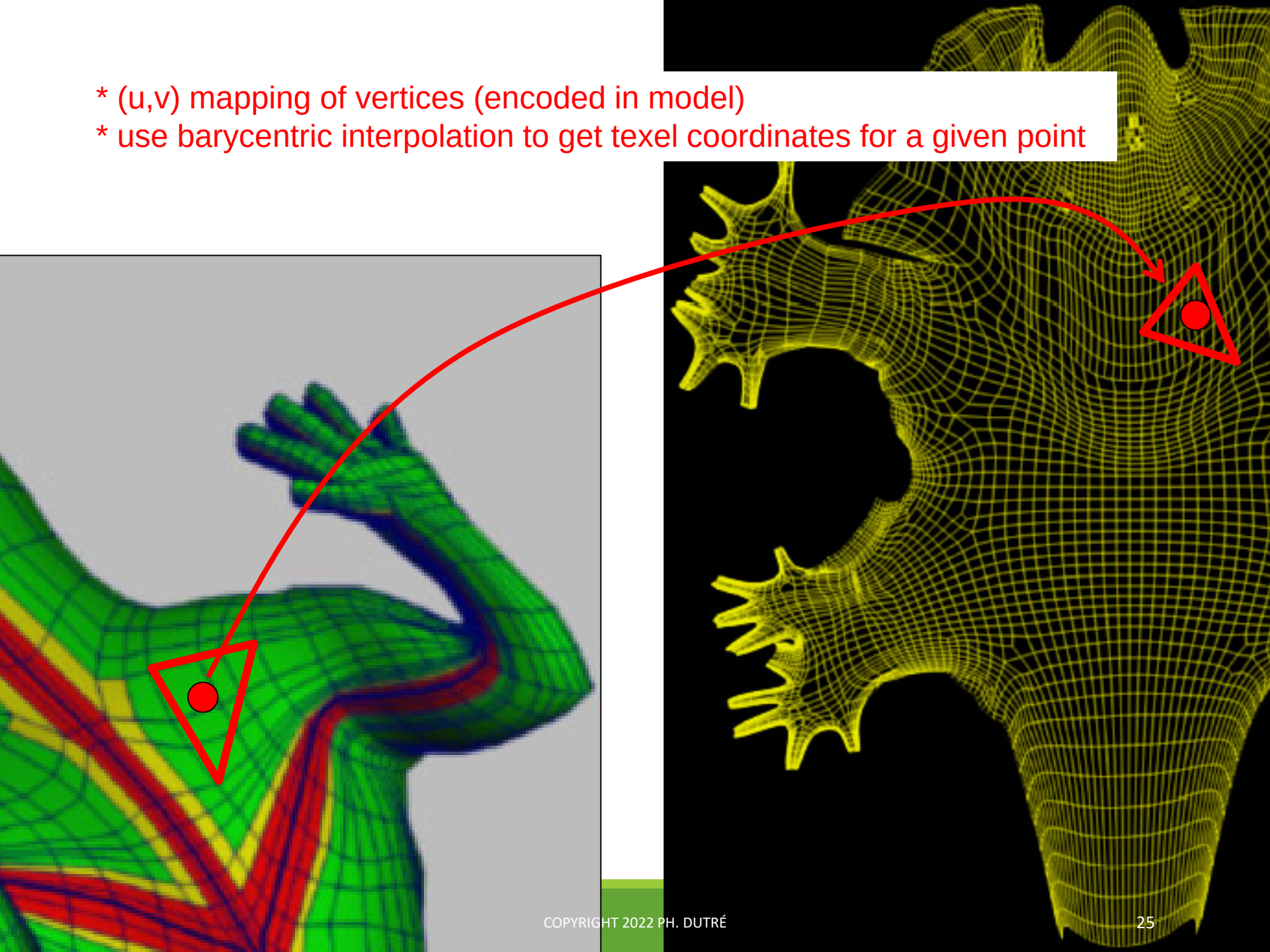
For a point to be shaded within a triangle:

- Interpolate texture coordinates of vertices
  - (based on barycentric coordinates)
- ➔ provides new (u,v) for texture look-up





* (u,v) mapping of vertices (encoded in model)
* use barycentric interpolation to get texel coordinates for a given point

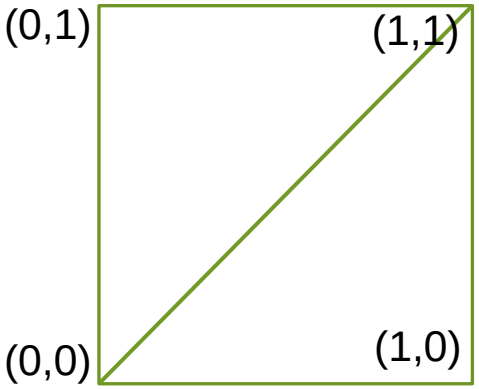


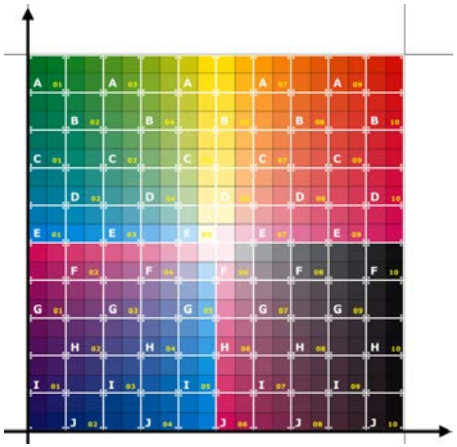
Triangle meshes

http://www.realtimerendering.com/udacity/?load=demo/unit8-texture-distortion.js

Triangle meshes

polygon mesh
(u,v) mapping



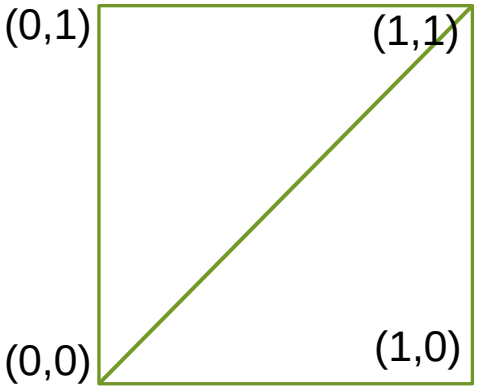


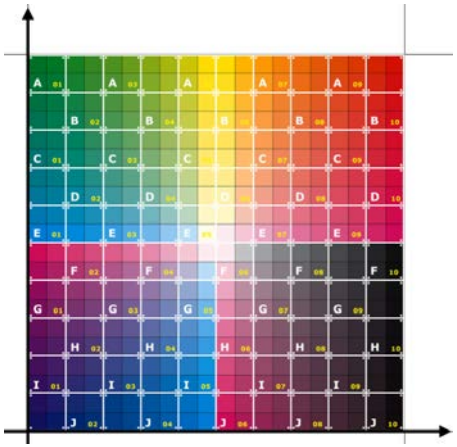
original texture

http://www.realtimerendering.com/udacity/?load=demo/unit8-texture-distortion.js

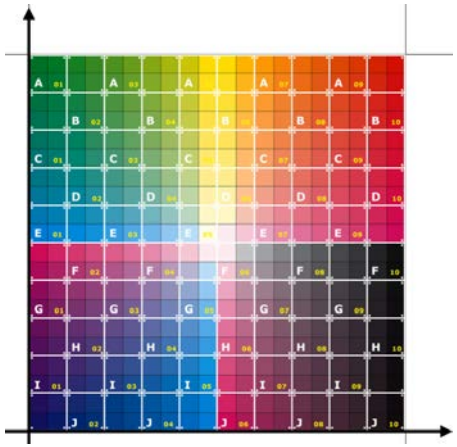
Triangle meshes

polygon mesh
(u,v) mapping





original texture

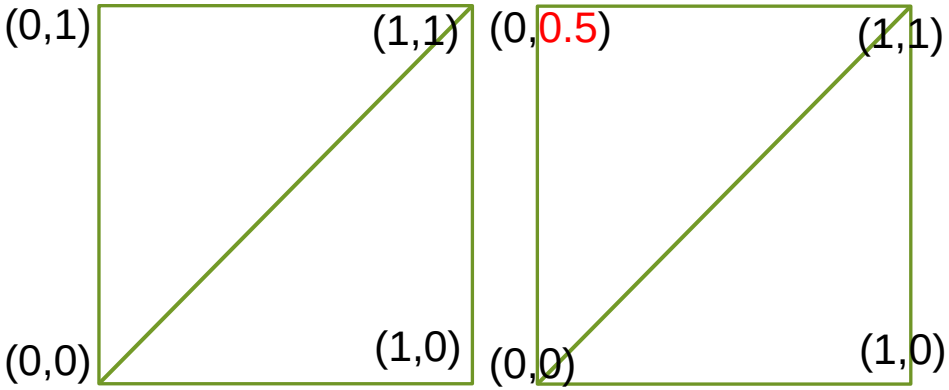


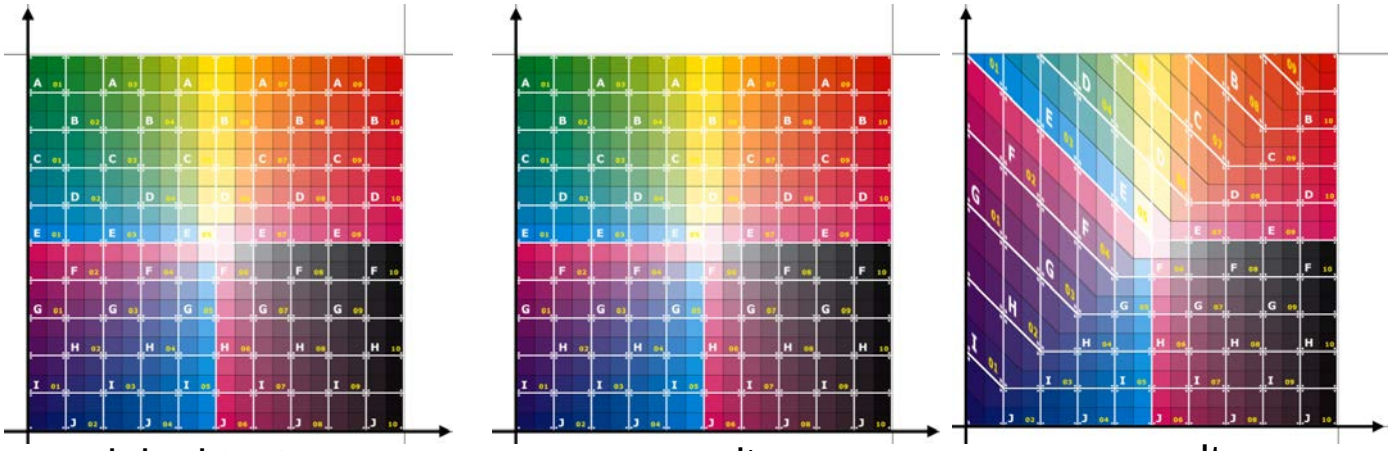
result

http://www.realtimerendering.com/udacity/?load=demo/unit8-texture-distortion.js

Triangle meshes

polygon mesh
(u,v) mapping





original texture

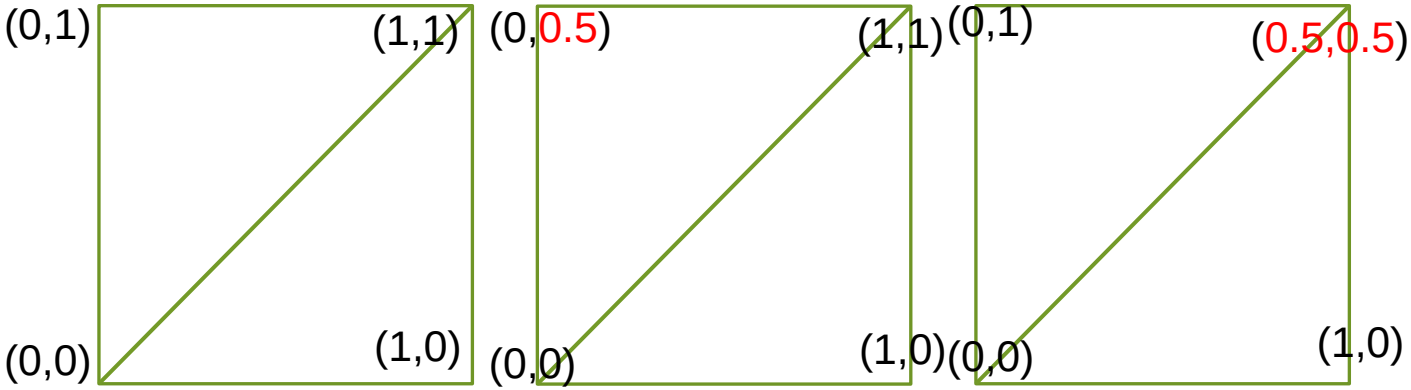
result

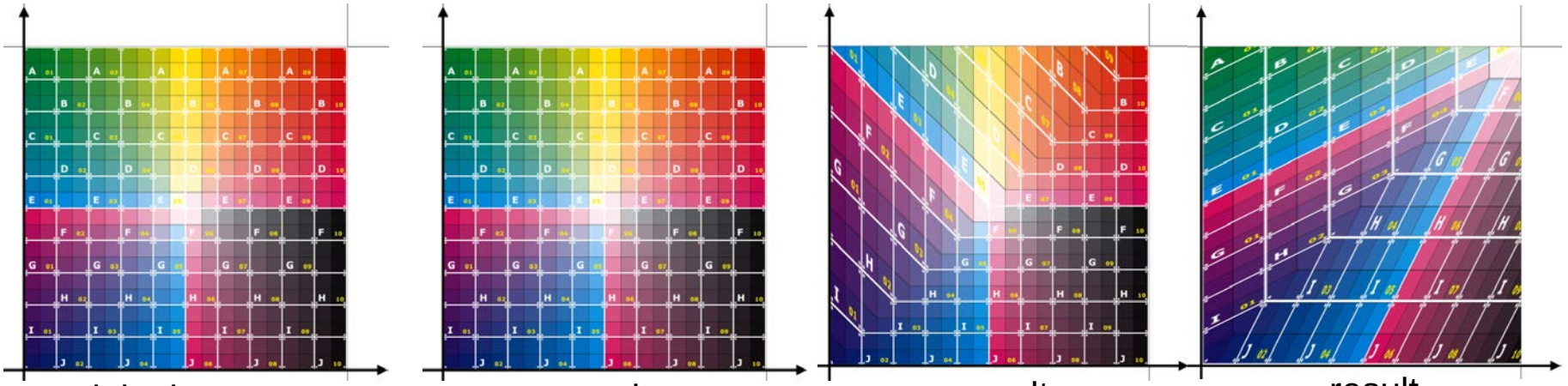
result

http://www.realtimerendering.com/udacity/?load=demo/unit8-texture-distortion.js

Triangle meshes

polygon mesh
(u,v) mapping

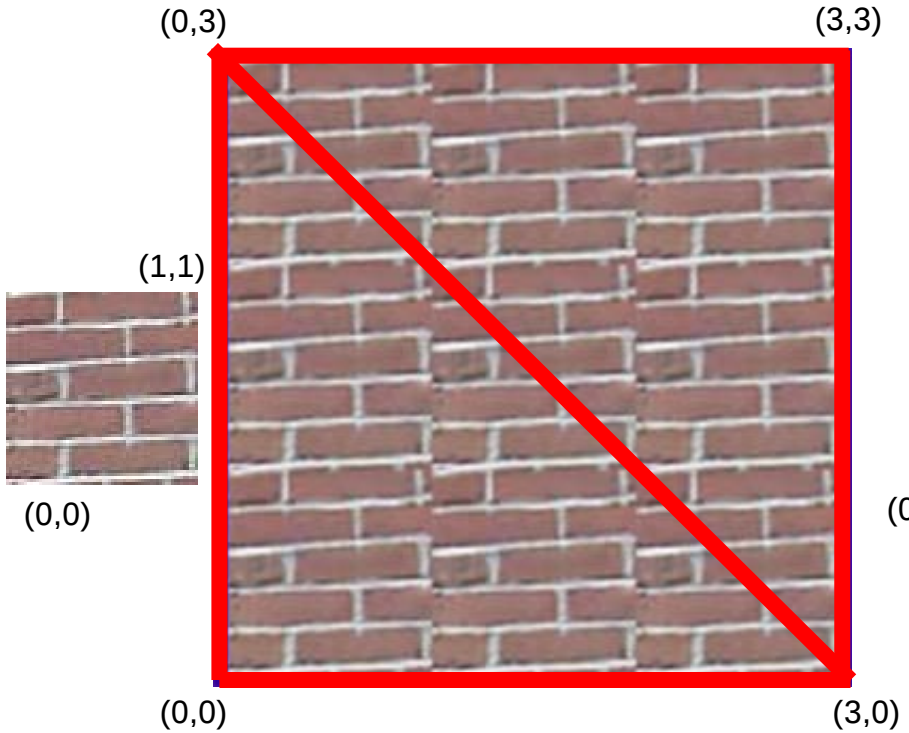




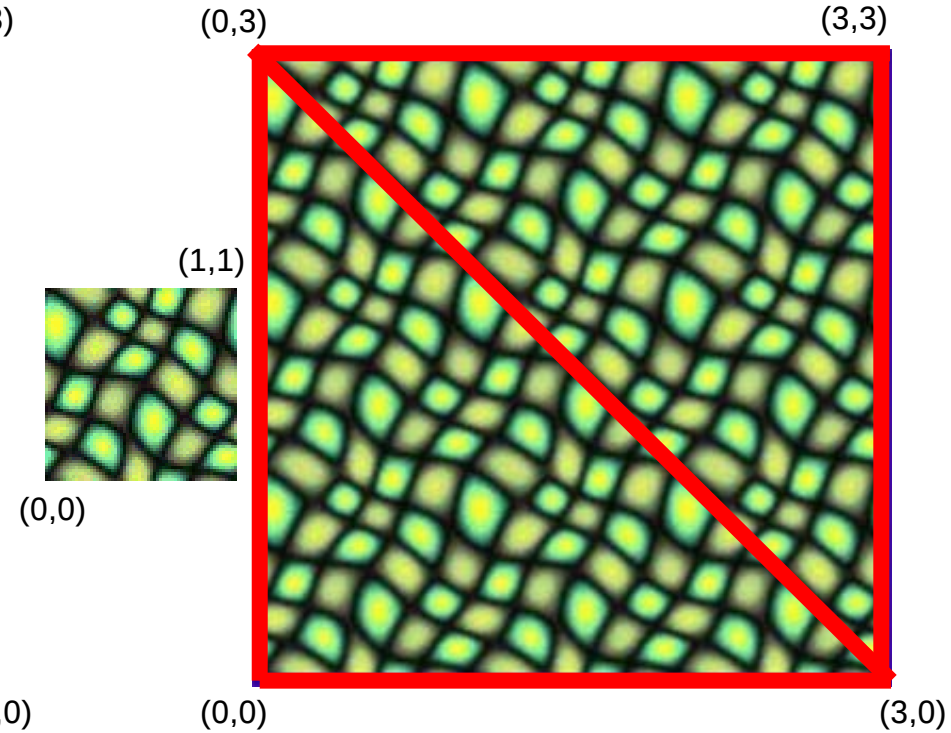
original texture | result | result | result

http://www.realtimerendering.com/udacity/?load=demo/unit8-texture-distortion.js

Triangle meshes



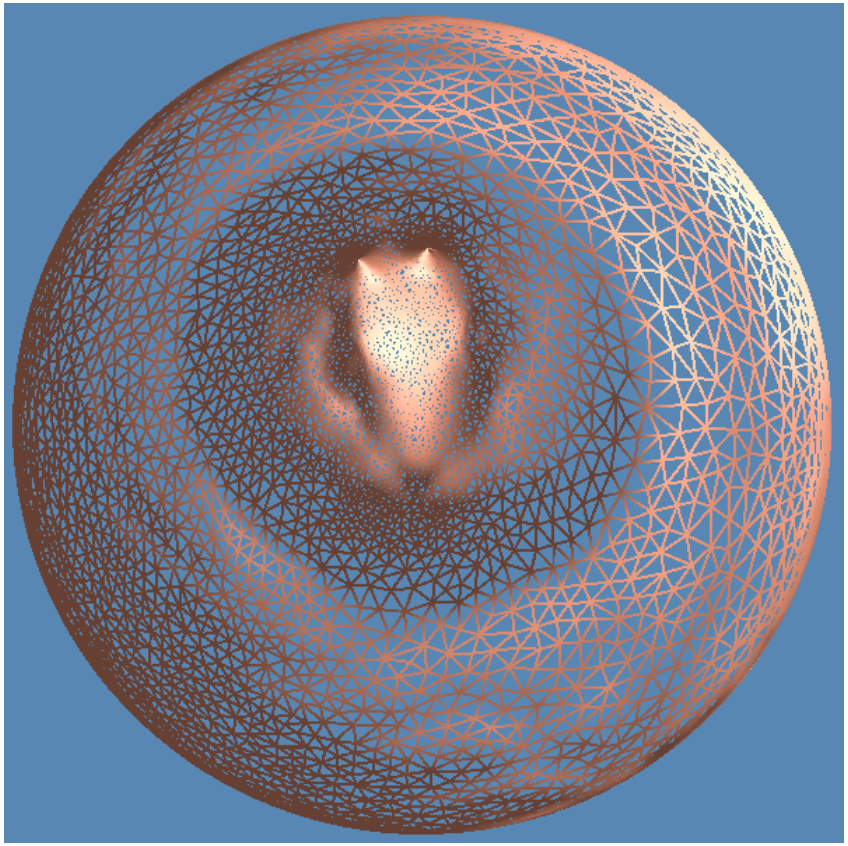
discontinuities in tiling

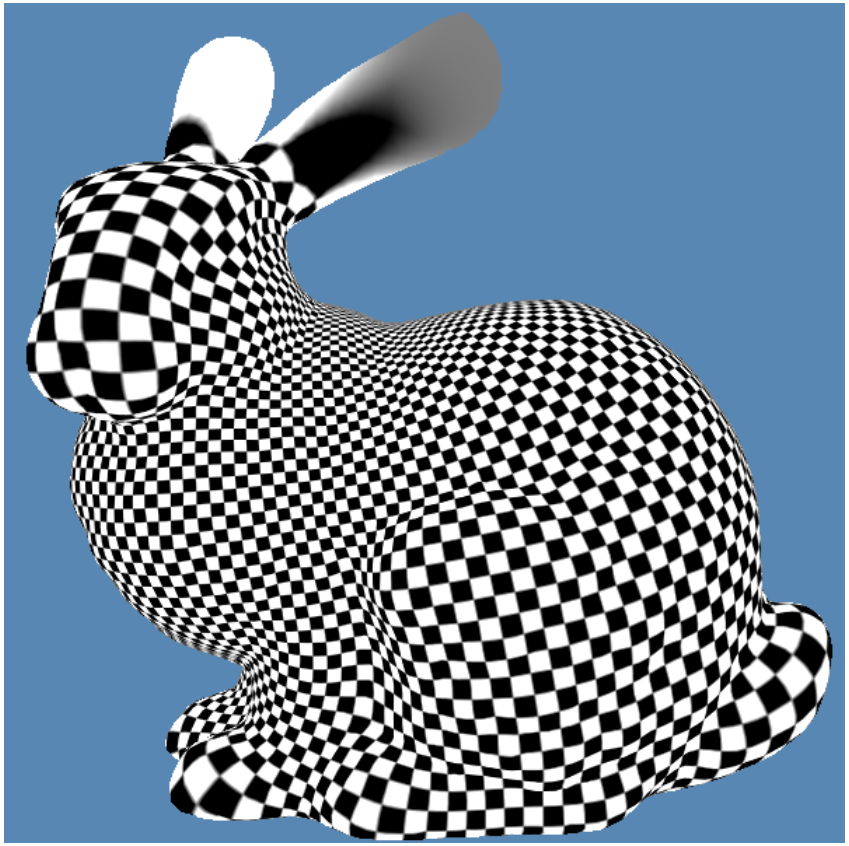


no discontinuities (repeating tile)

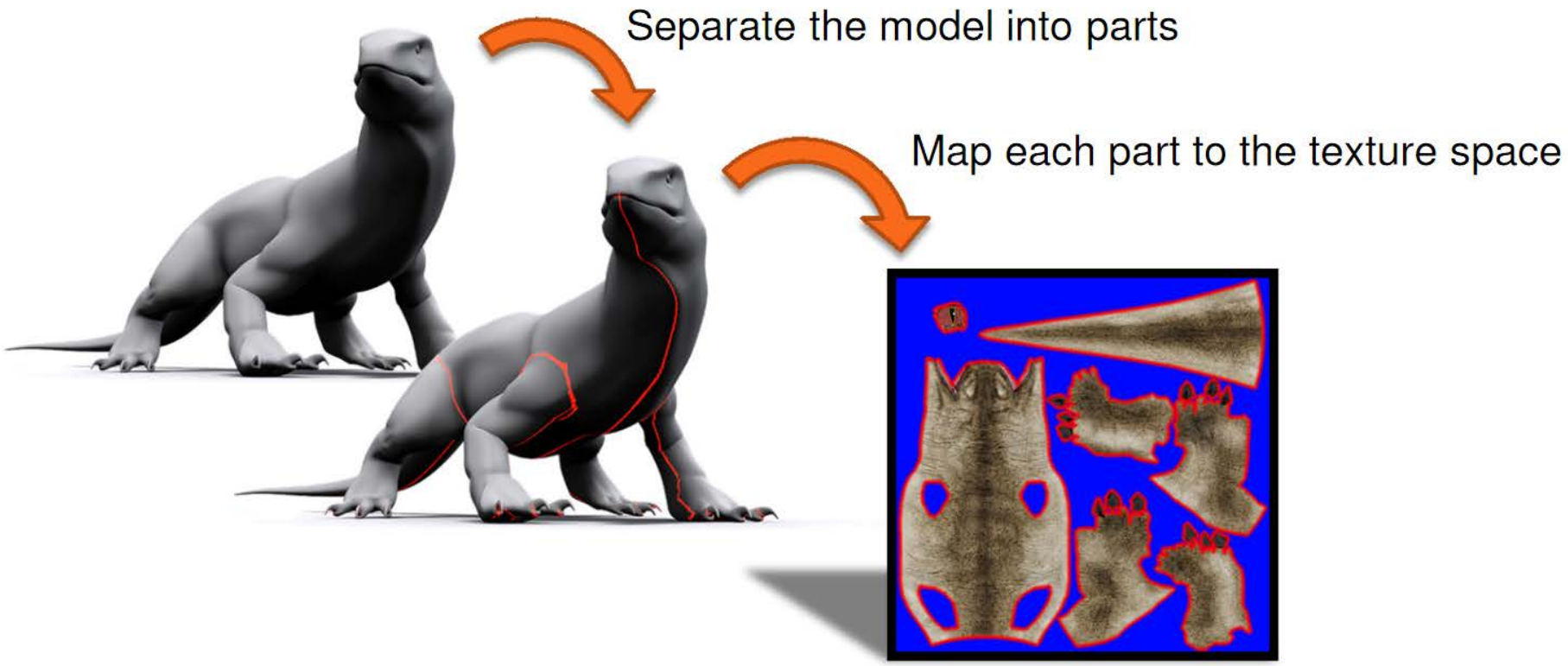
Triangle meshes

"Unfolding" is difficult -- distortions



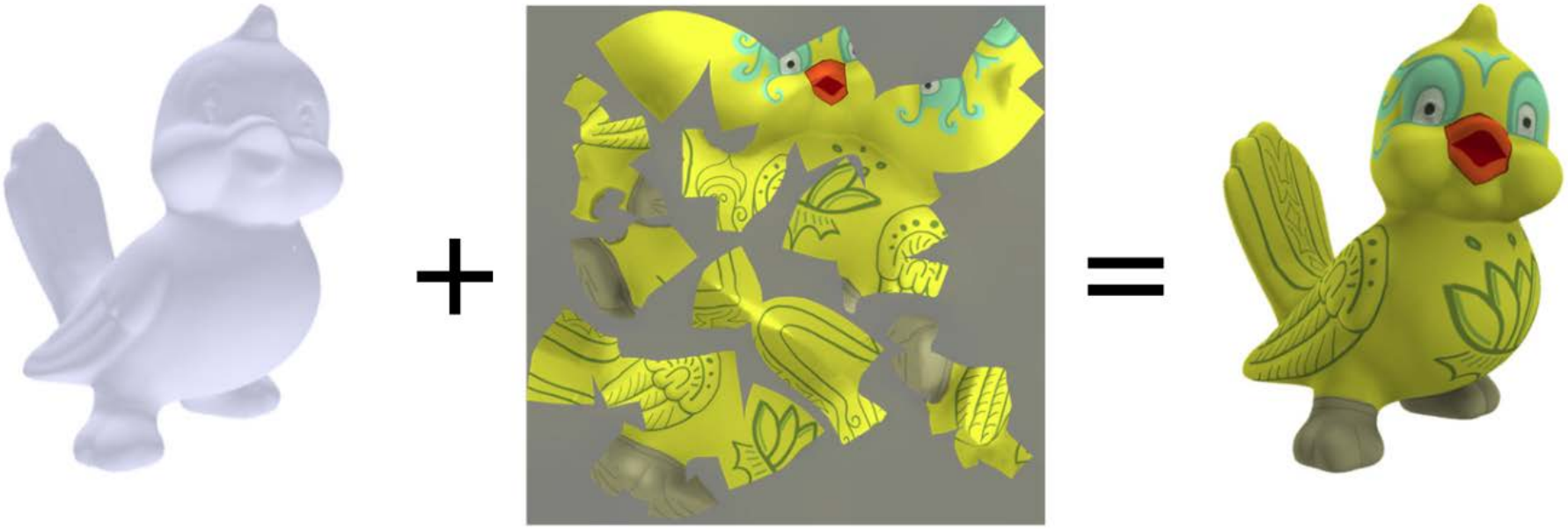


Triangle meshes – texture atlas



(Rethinking Texture Mapping – Tarini et al. – SIGGRAPH 2017 Course)

Triangle meshes – texture atlas

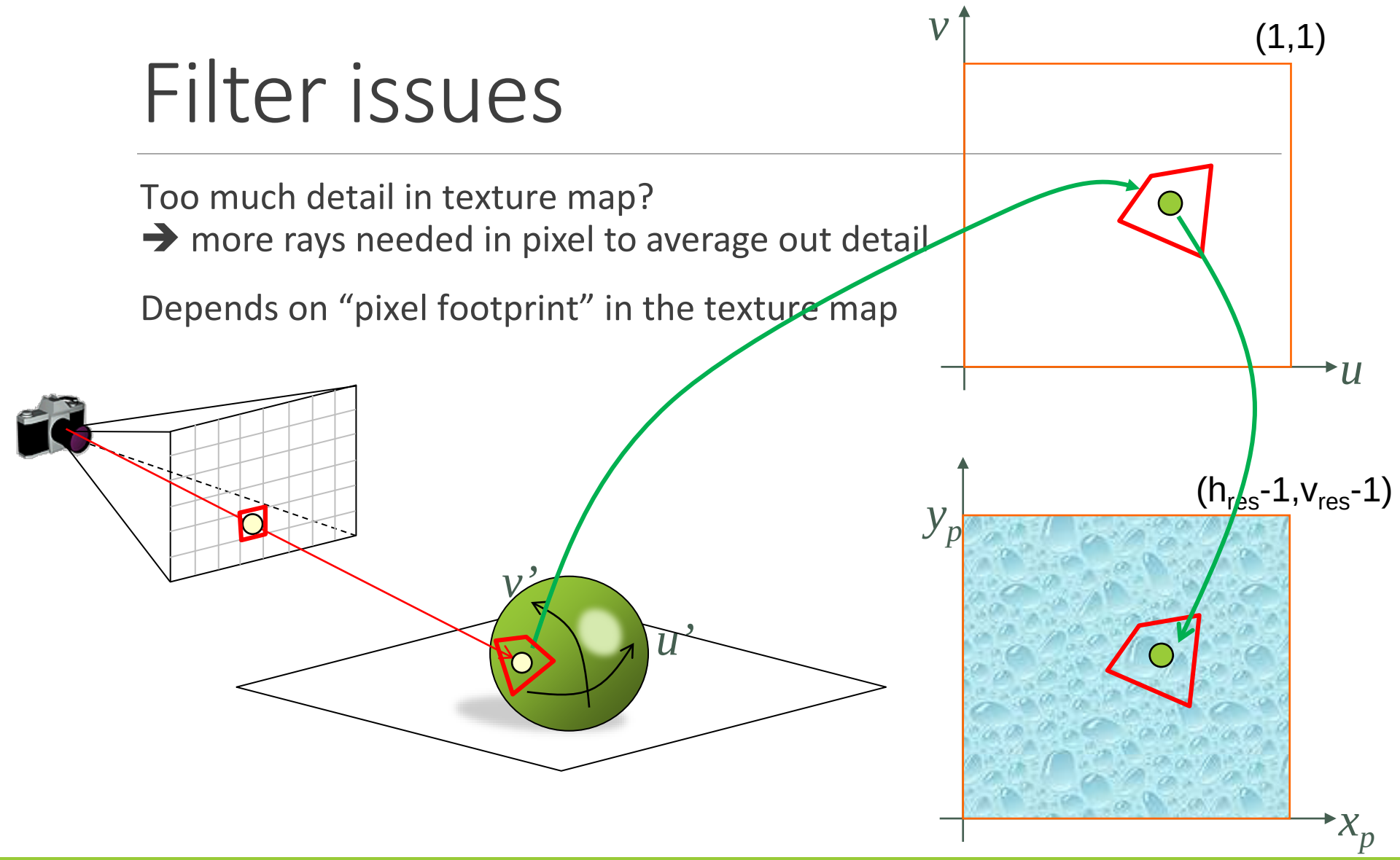


(Rethinking Texture Mapping – Tarini et al. – SIGGRAPH 2017 Course)

Filter issues

Too much detail in texture map?
➔ more rays needed in pixel to average out detail

Depends on “pixel footprint” in the texture map



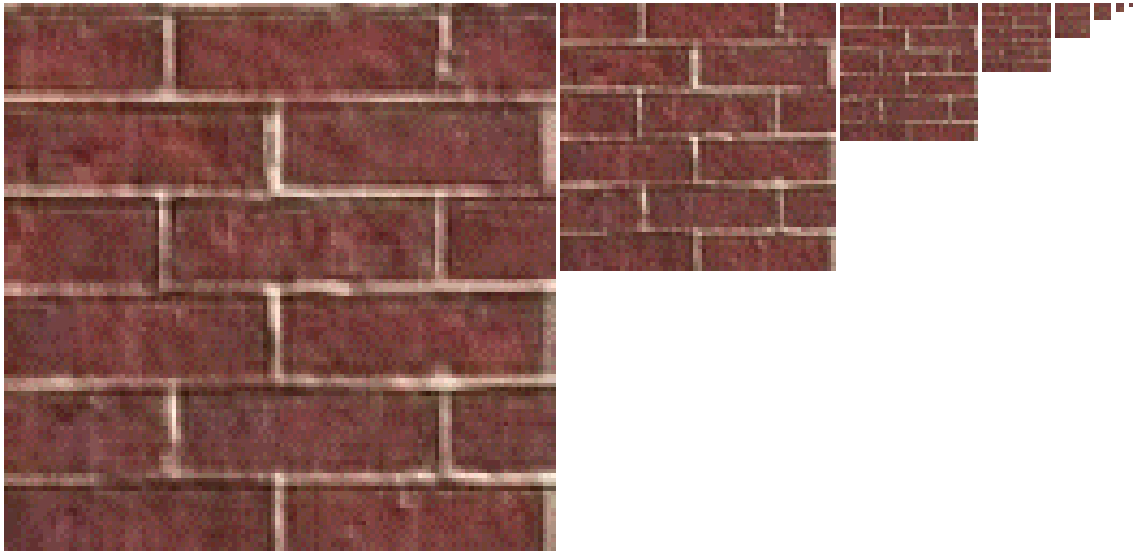
Pre-filter texture: MIP Mapping

"Pyramid" of images that are pre-filtered and re-sampled at 1/2, 1/4, 1/8, etc.,

During rendering:

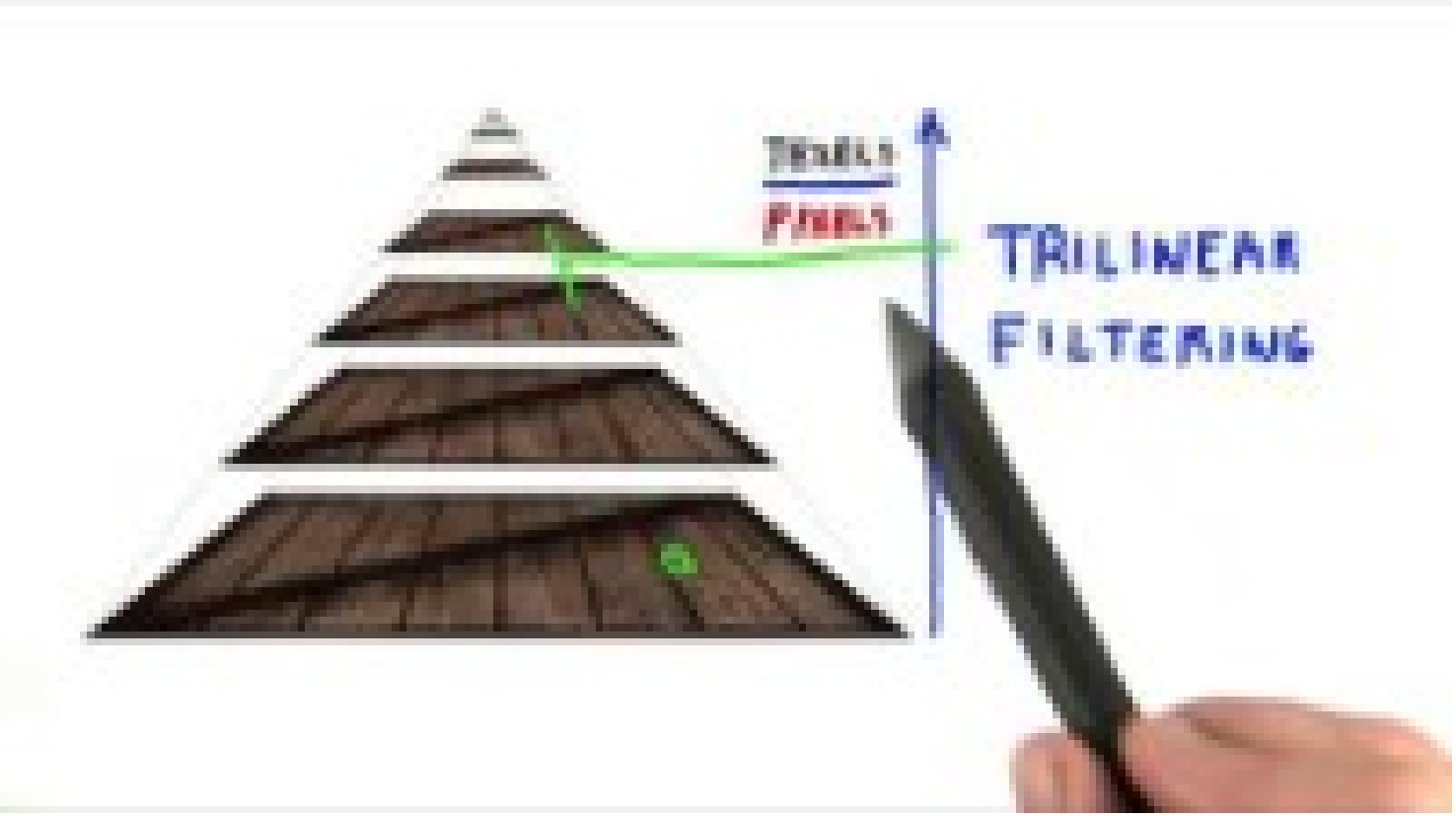
- pick the pre-filtered image that corresponds best in resolution:
  size of pixel footprint on object ~ size of texel in texture map
- Better: look up 2 closest levels (+ and -) and interpolate values fetched from each level

MIP: multum in parvo (many in a small place)



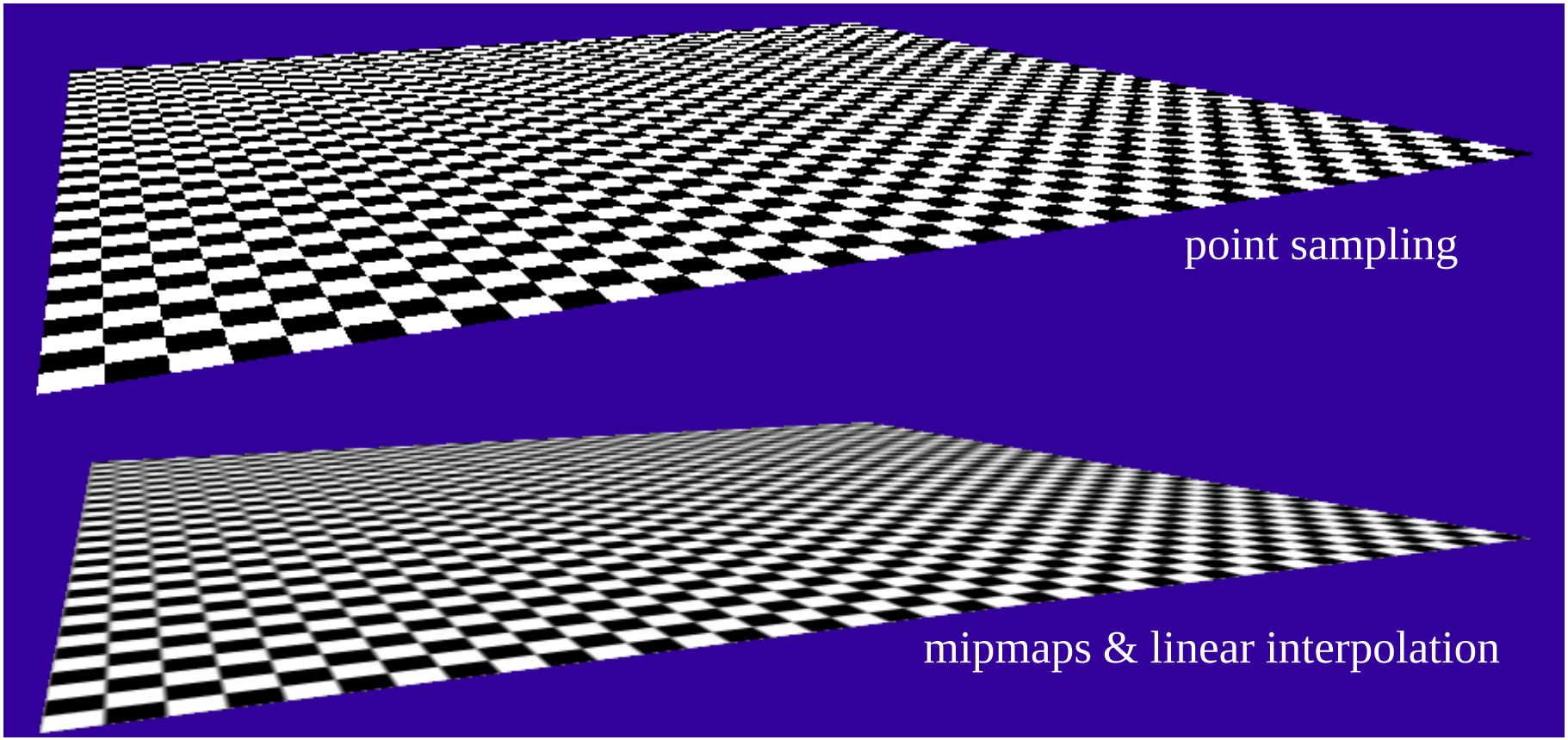
(FREDO Durand)

Pre-filter texture: MIP Mapping



https://youtu.be/8OtOFN17jxM

MIP Mapping Example

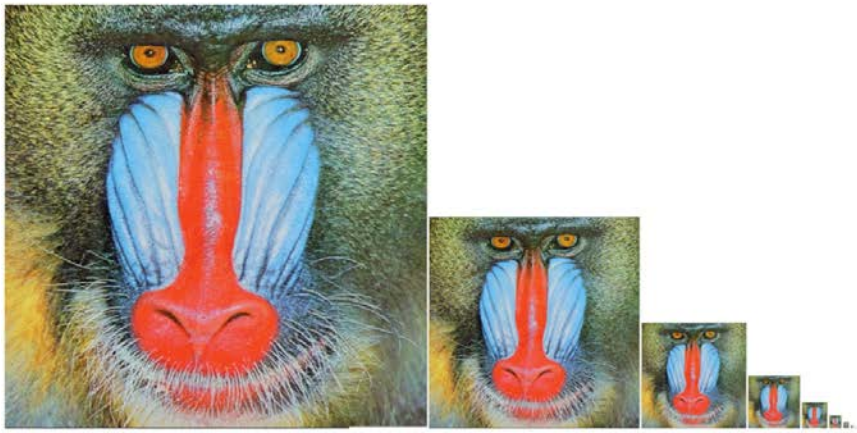


(Fredo Durand)

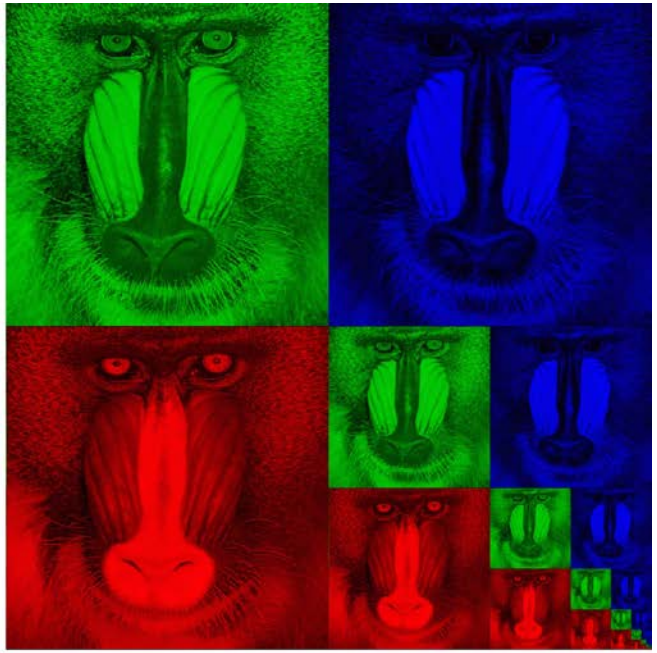
MIP Mapping: storage

Can be stored compactly

~ 1/3 extra storage space



10-level mip map



Memory format of a mip map

(Fredo Durand)

Bump Textures (aka Normal Mapping)

Texture values are used to perturbate the shading normal

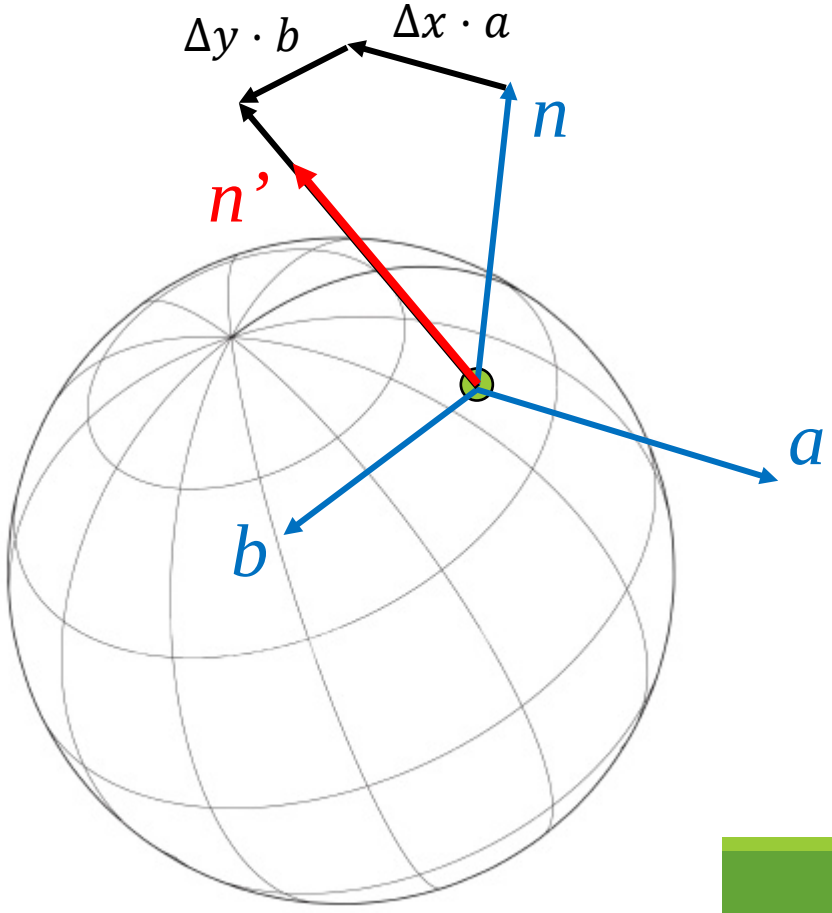
Bump Textures (aka Normal Mapping)

Read out values Δx and Δy from map, or evaluate 'bump-function' Δx = f(u,v), Δy = g(u,v)

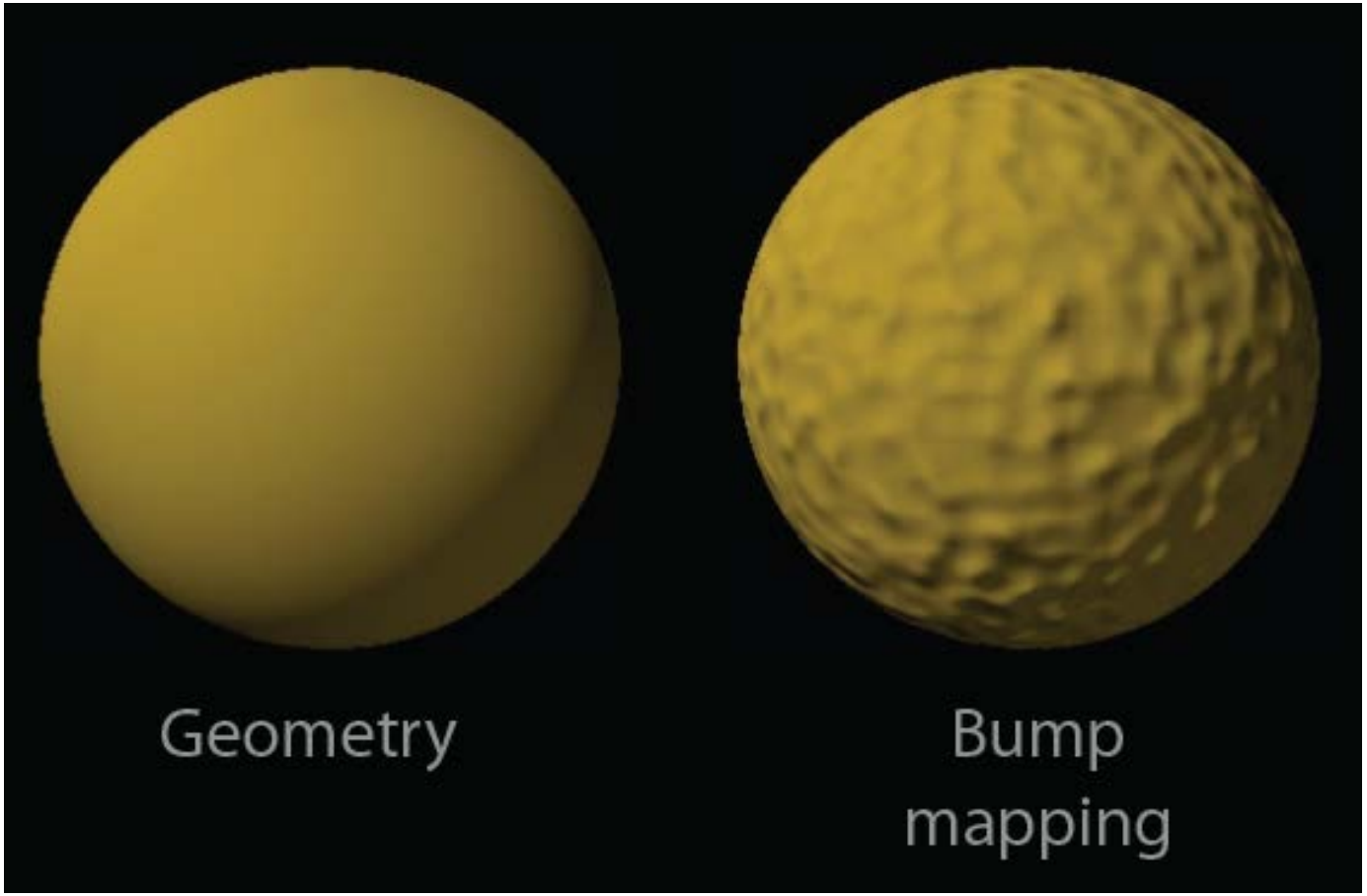
Compute vectors a, b perpendicular to n (local reference frame)

Compute $n' = \frac{n+\Delta x \cdot a+\Delta y \cdot b}{\|n+\Delta x \cdot a+\Delta y \cdot b\|}$

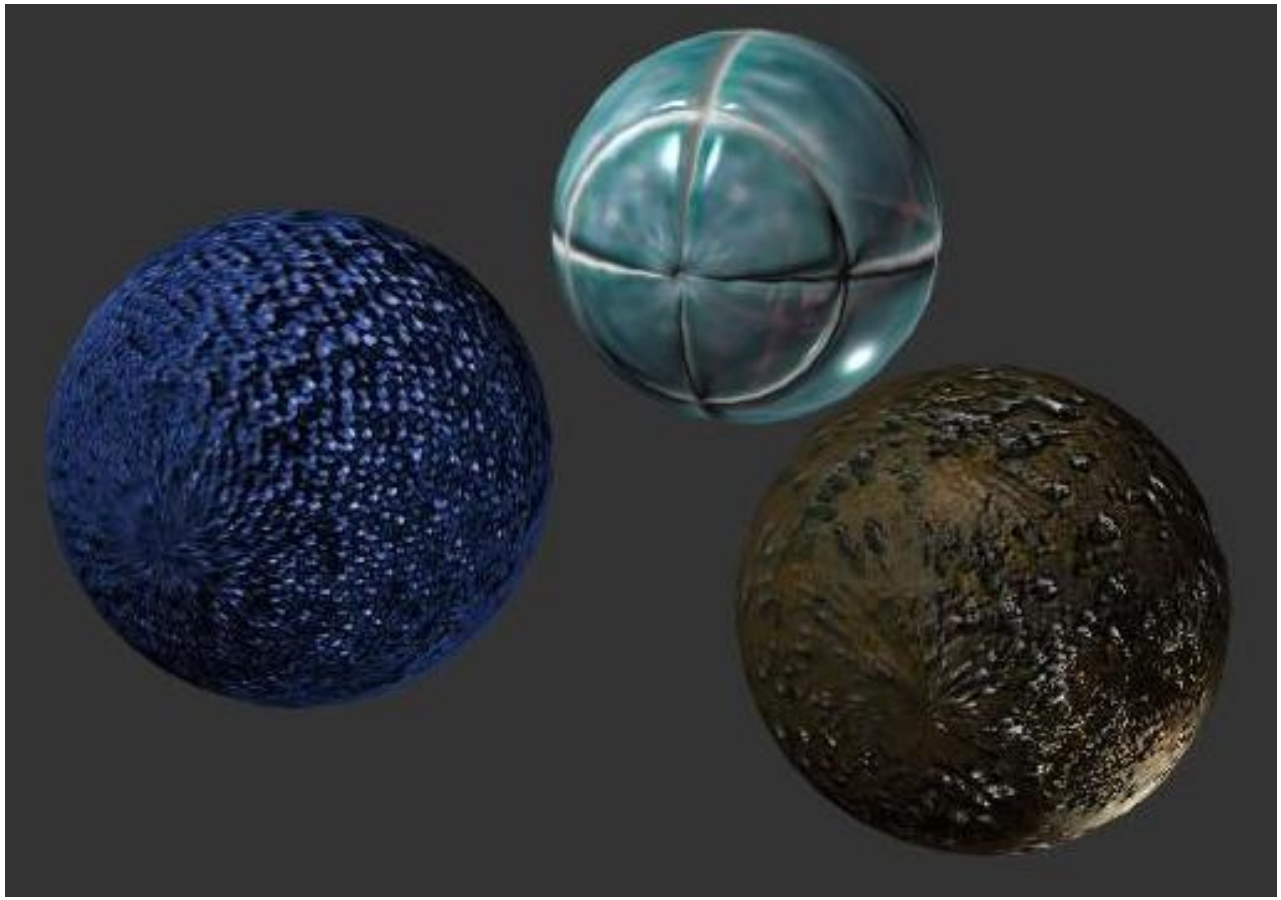
Shading normal n'
vs
Geometry normal n



Bump Textures (aka Normal Mapping)

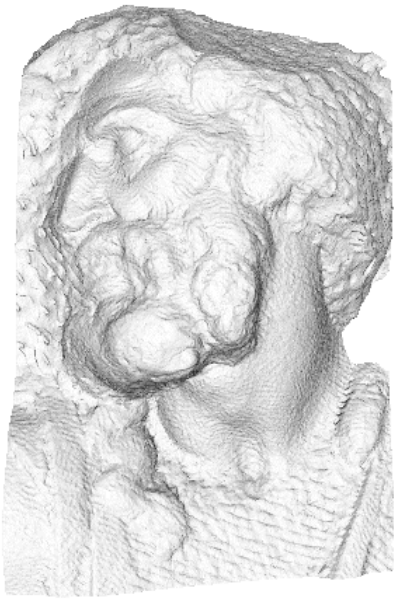


Bump Textures (aka Normal Mapping)

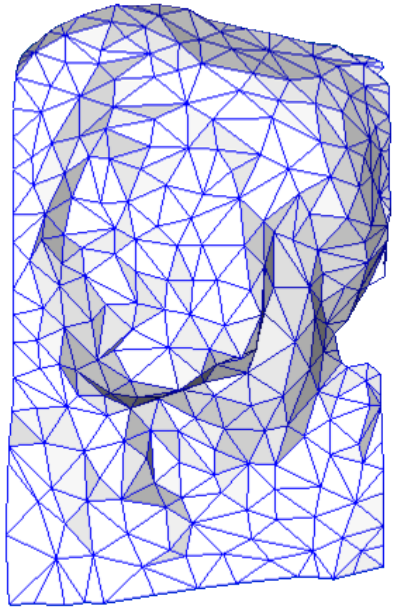


Bump Textures (aka Normal Mapping)

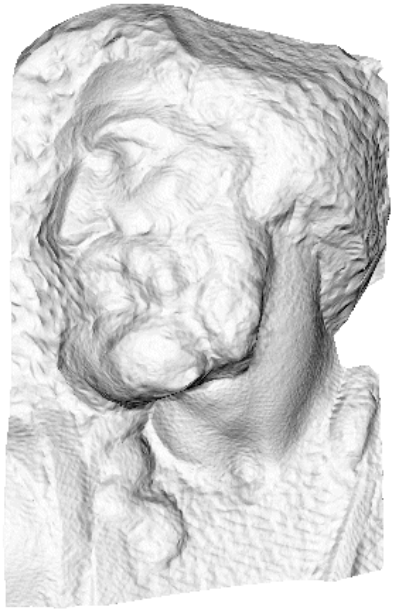
Mesh simplification + bump-map



original mesh
4M triangles



simplified mesh
500 triangles



simplified mesh
and normal mapping
500 triangles

[Paolo Cignoni]

Environment Mapping (a.k.a reflection mapping; sky-box; …)

"Pre-computed reflections"

Store the environment to be reflected in a reflection map

Use perfect reflective direction to access "reflected color" in reflection map

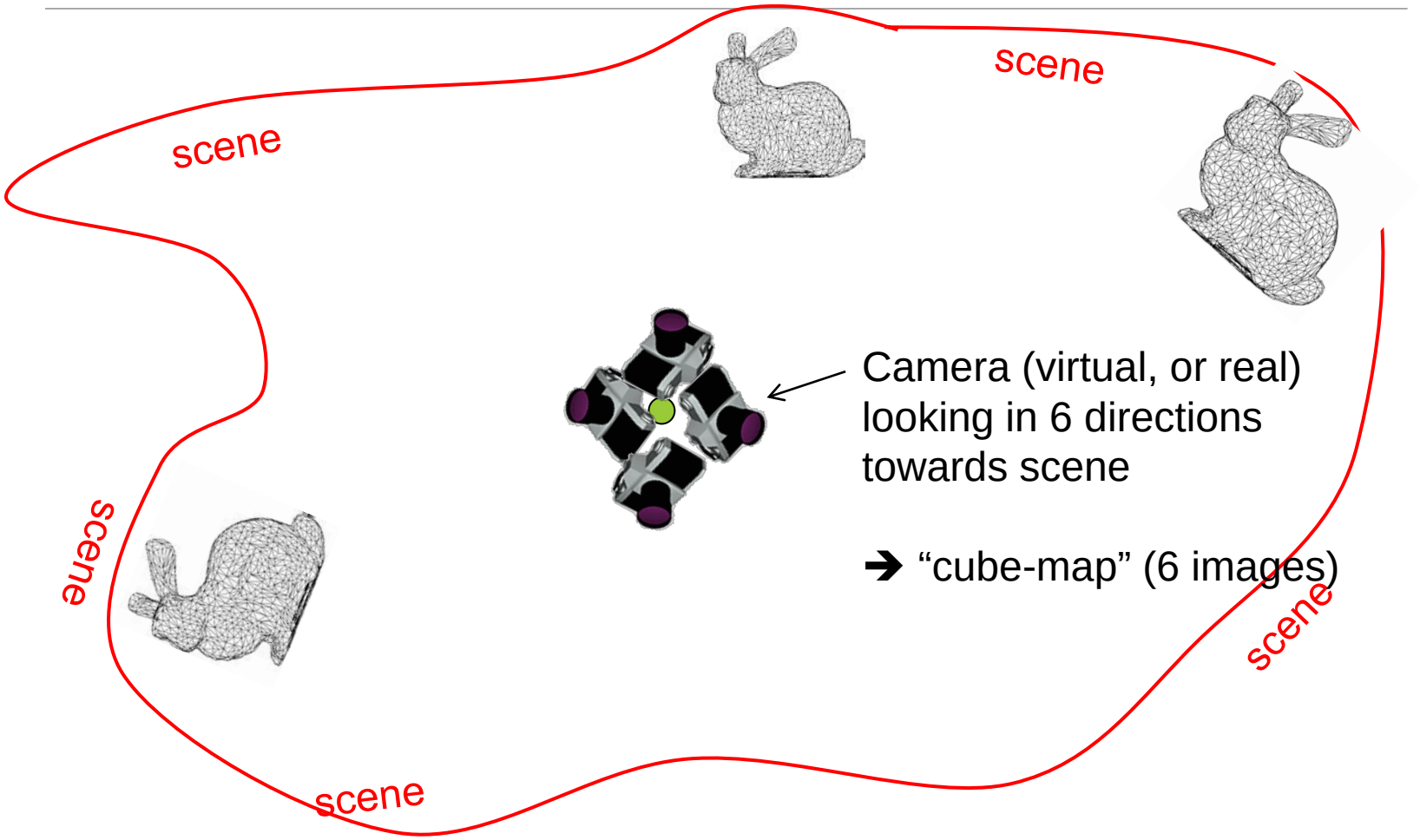
Environment map acquisition

360 degree camera

https://www.all3dfree.net/hdri-environment-for-free.html

Note: environments maps are often captured in high-dynamic range (HDR)

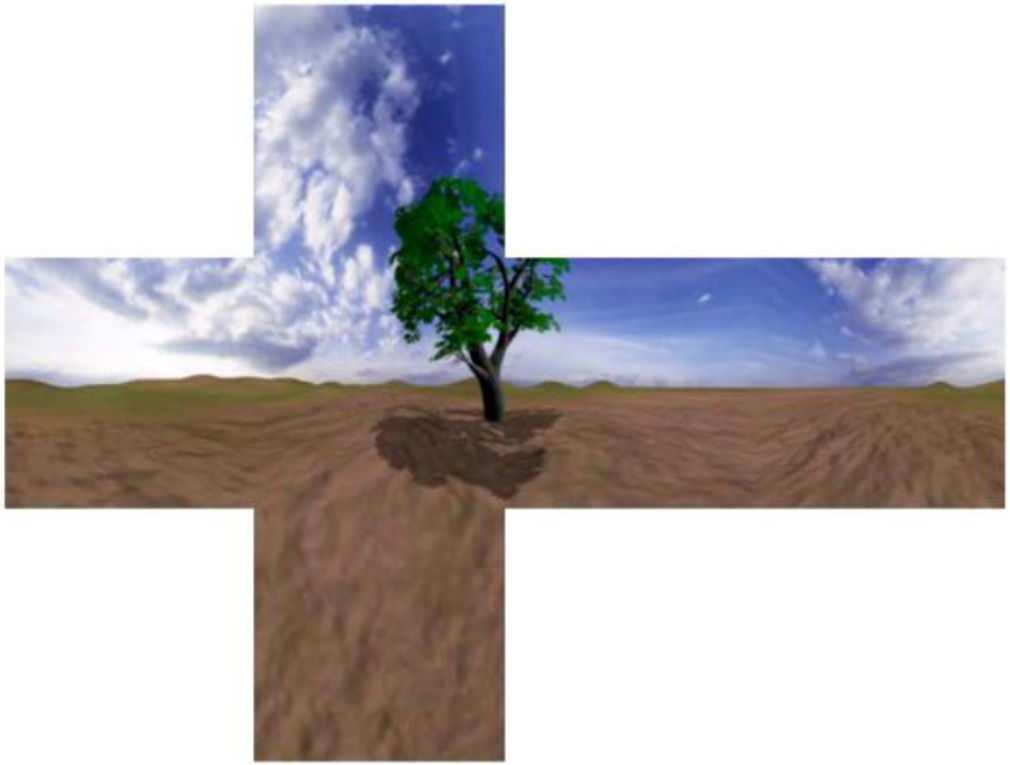
Environment map acquisition



Environment map acquisition

6 camera images of environment

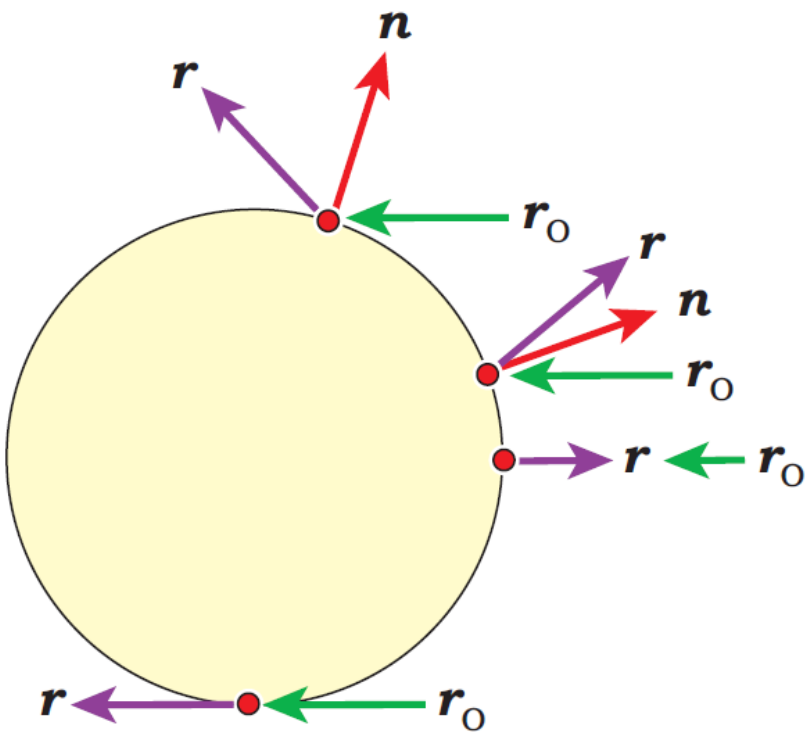
- (environment can be virtual or real)



Environment map acquisition Old School

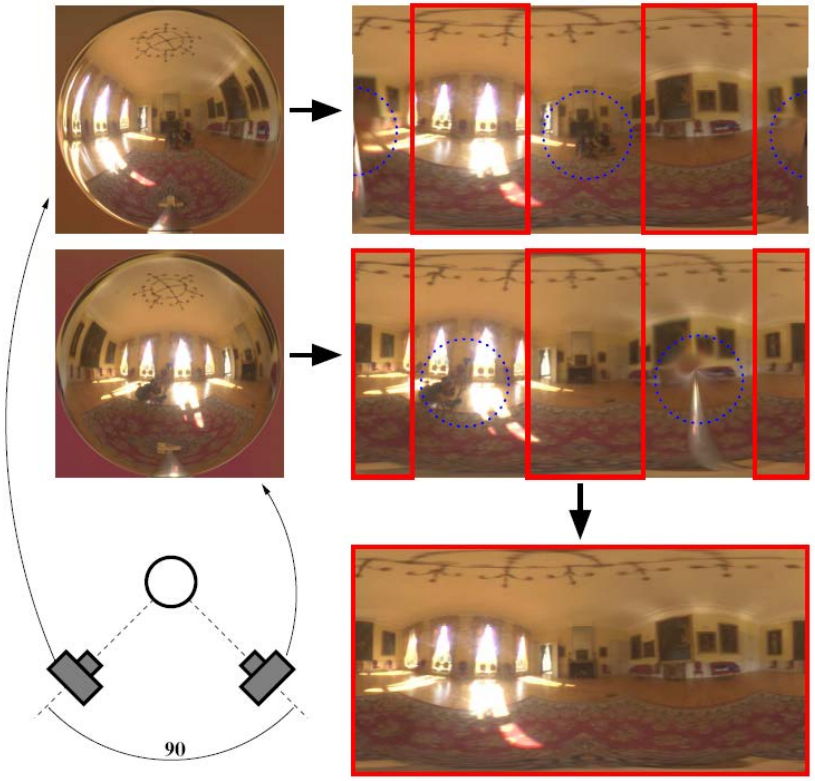
Take photograph of mirrored sphere

- Sphere reflects entire environment (except little area behind sphere)



Environment map acquisition Old School

Removing the camera from the image:



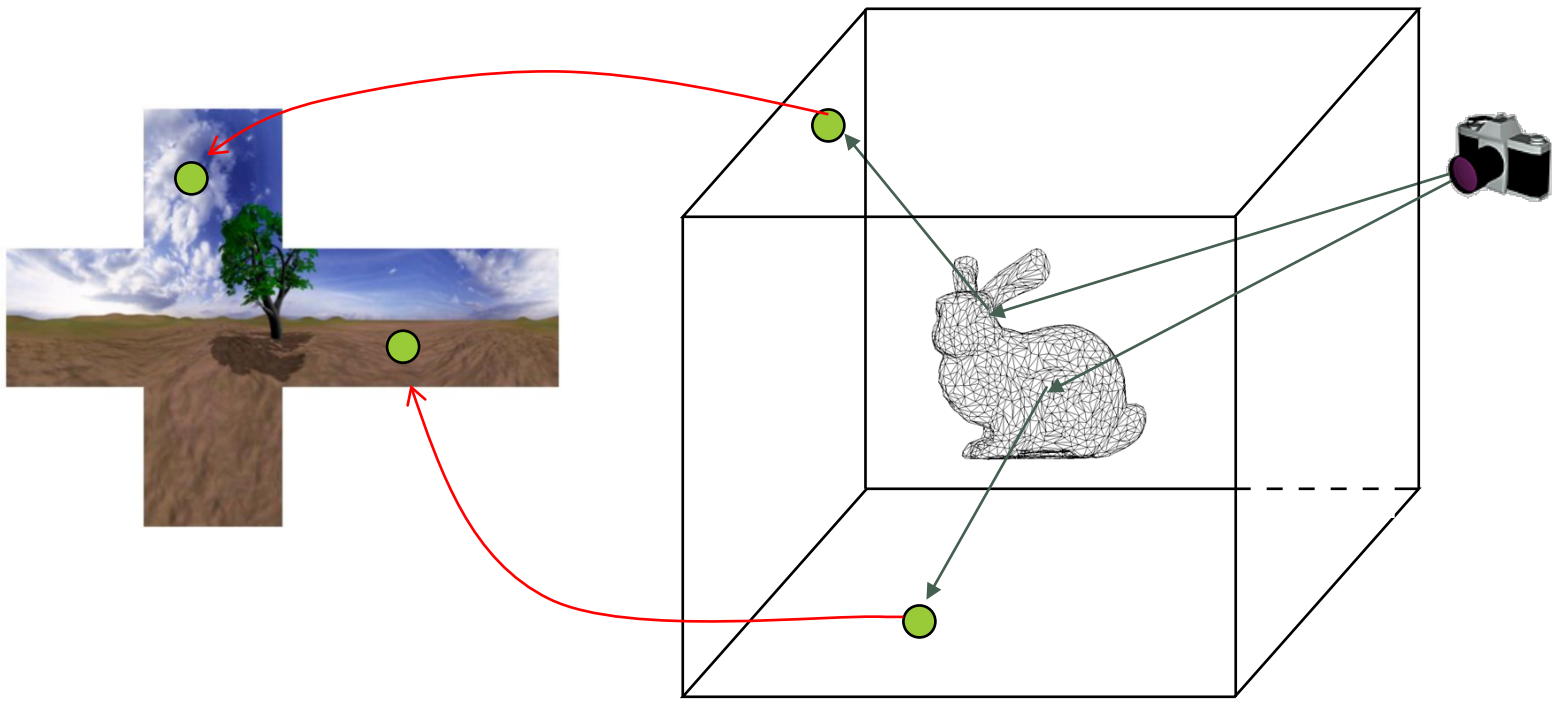
Environment map acquisition

http://graphics.cs.kuleuven.be/environmentmaps/index.html




How to use an environment map?

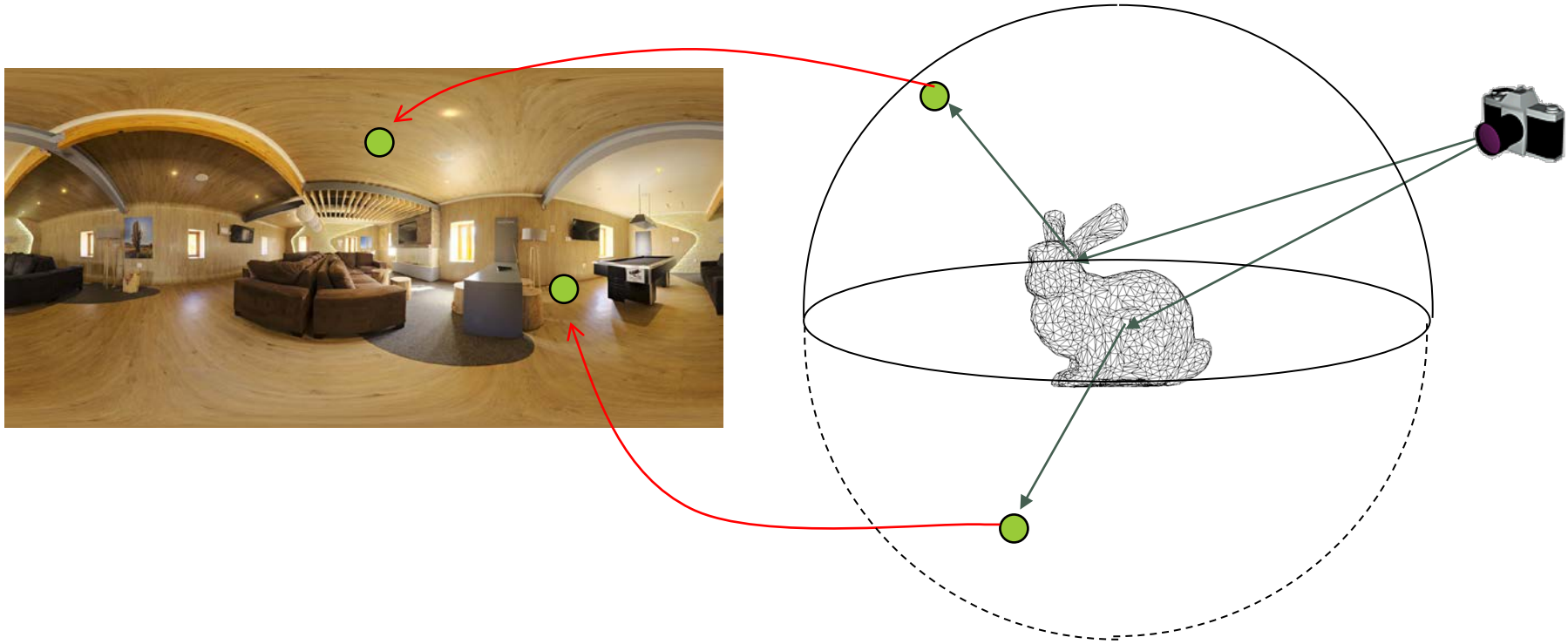
Pick up “reflections” from cube-map



(can be used for refraction rays as well)

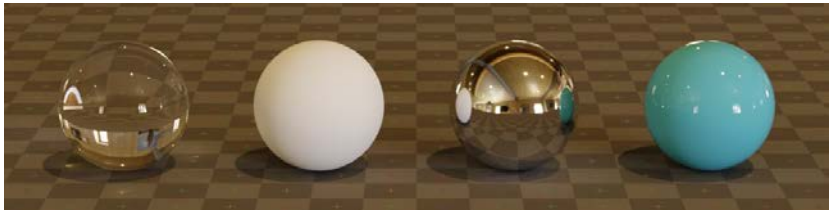
How to use an environment map?

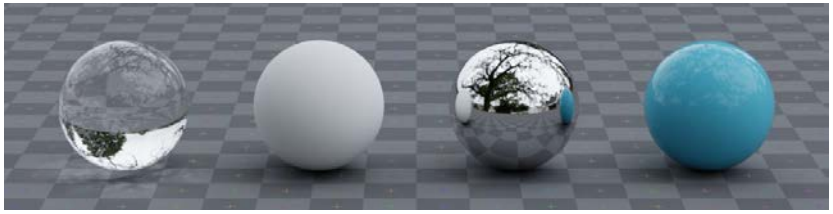
Pick up “reflections” from spherical map



How to use an environment map?

https://www.all3dfree.net/hdri-environment-for-free.html





Question: is this similar to using an environment light source?

Environment map vs Ray Tracing

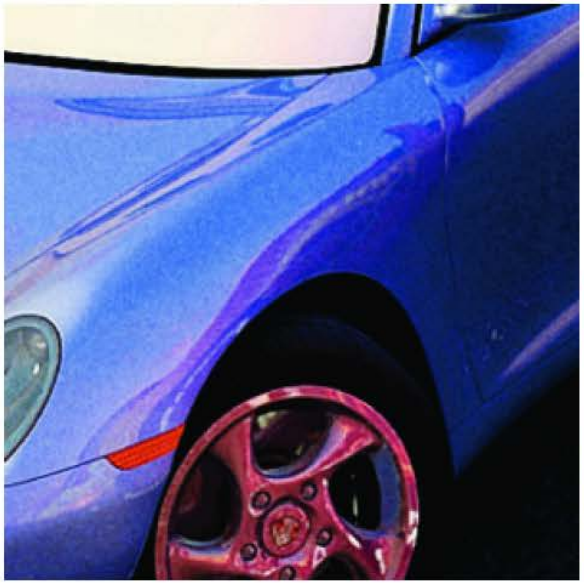
Environment map reflections
(reflects only the environment)

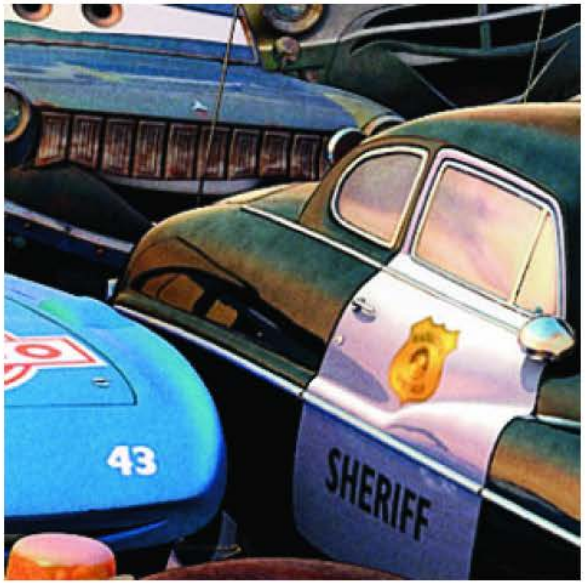
Ray-traced reflections
(reflects local object as well)

“Cars” used ray tracing for reflections

“Cars” used ray tracing for reflections

“Cars” used ray tracing for reflections





History of environment maps

History of environment maps

First demo of environment map – Interface – 1985

https://www.youtube.com/watch?v=J7U_ZQ0p8b4

History of environment maps

(First) use of environment map in film -- Terminator II - 1991

History of environment maps

Procedural textures

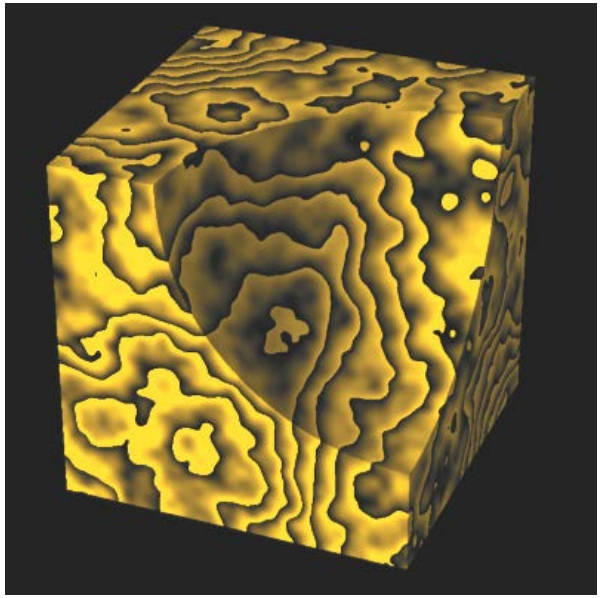
$$L = k_a c_r L_{ambient} c_{ambient} + \frac{k_d}{\pi} c_r L_{light} c_{light} \cos\theta$$

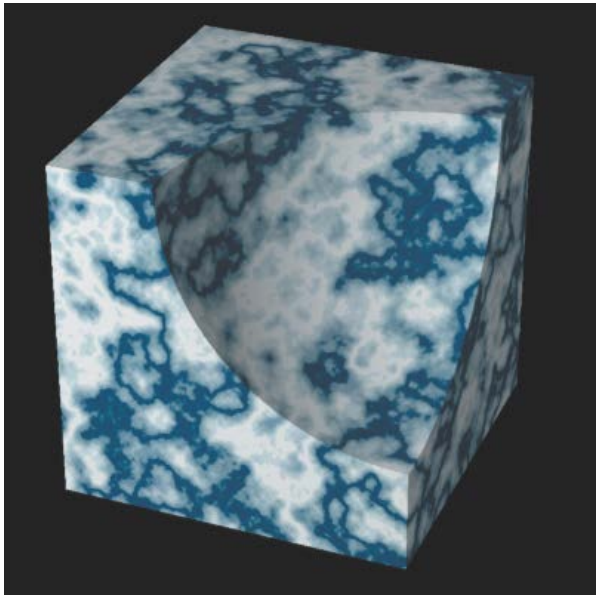
$$c_r = F(x, y, z)$$

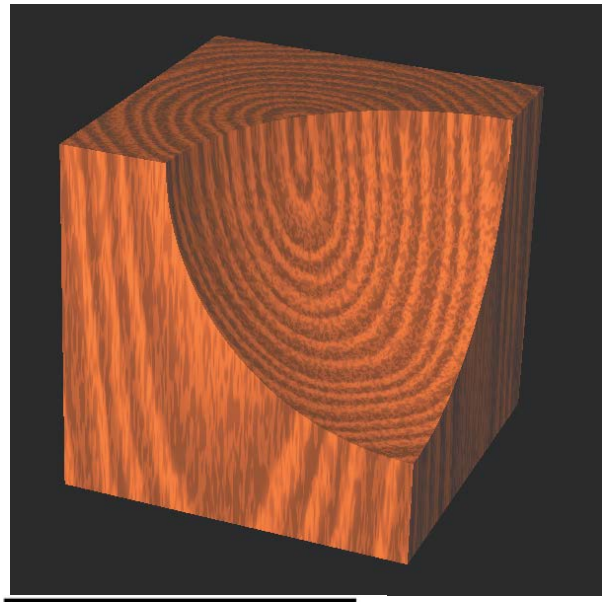
Ray tracing:

- (x,y,z) coordinate of point to be shaded is known, either in object coordinate system or world coordinate system
- Evaluate $F(x,y,z)$ during shading

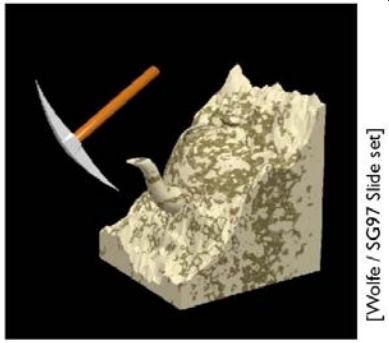
Procedural textures 3D



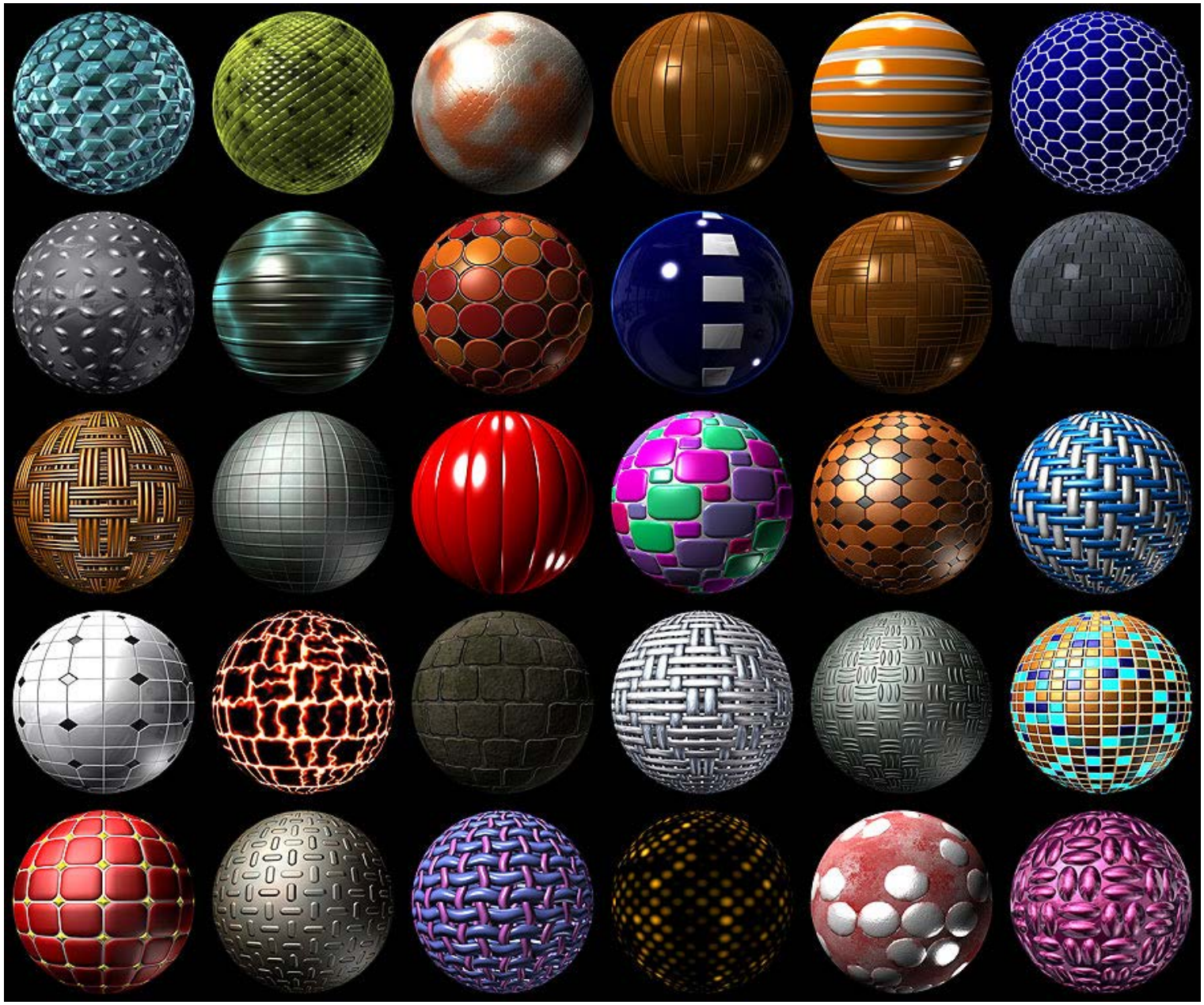




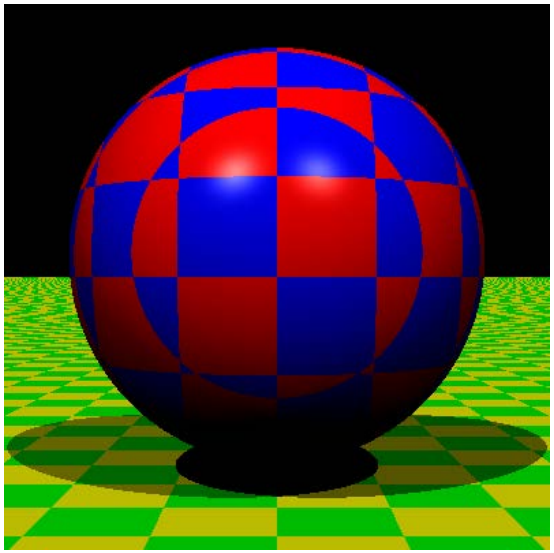
Visual Effect: 3D model "*carved*" out of solid material



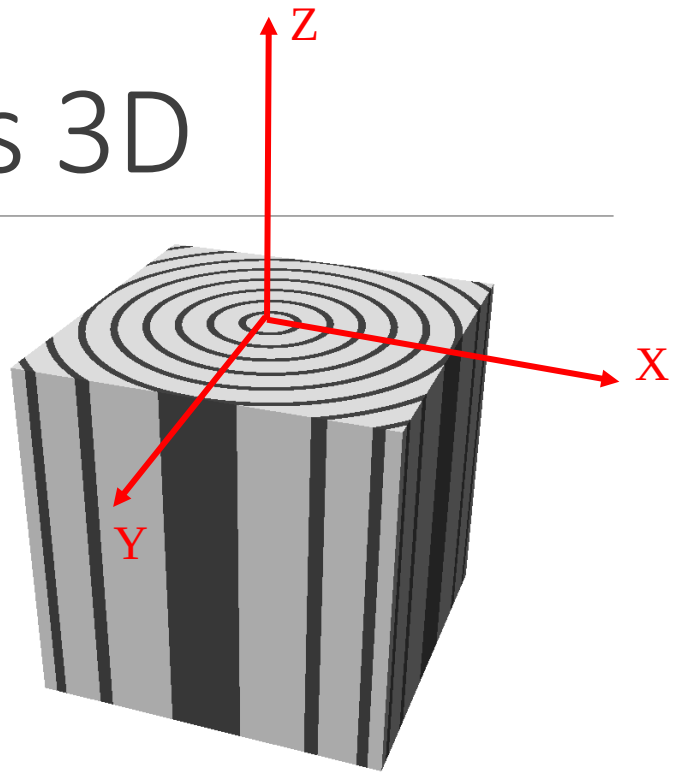
[Wolfe / SG97 Slide set]



Procedural textures 3D



```
F(x,y,z) {
a = int(x+y+z)
if a%2 == 1 c_r = blue
    else c_r = red
return c_r
}
```

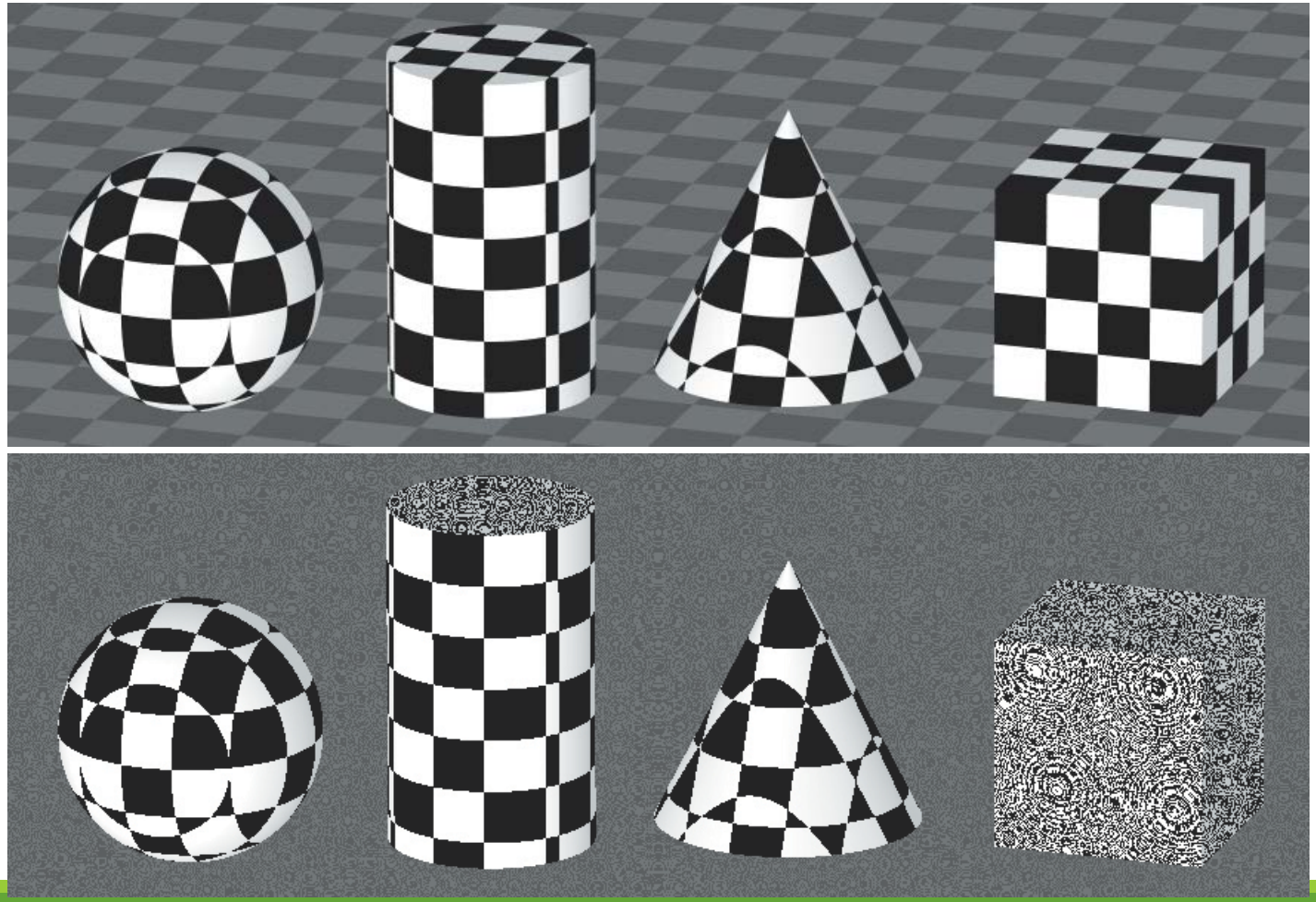



```
F(x,y,z) {
r = int(sqrt(x*x+y*y))
if (r%4 == 0) c_r = black
    else c_r = white
return c_r
}
```

Procedural textures 3D

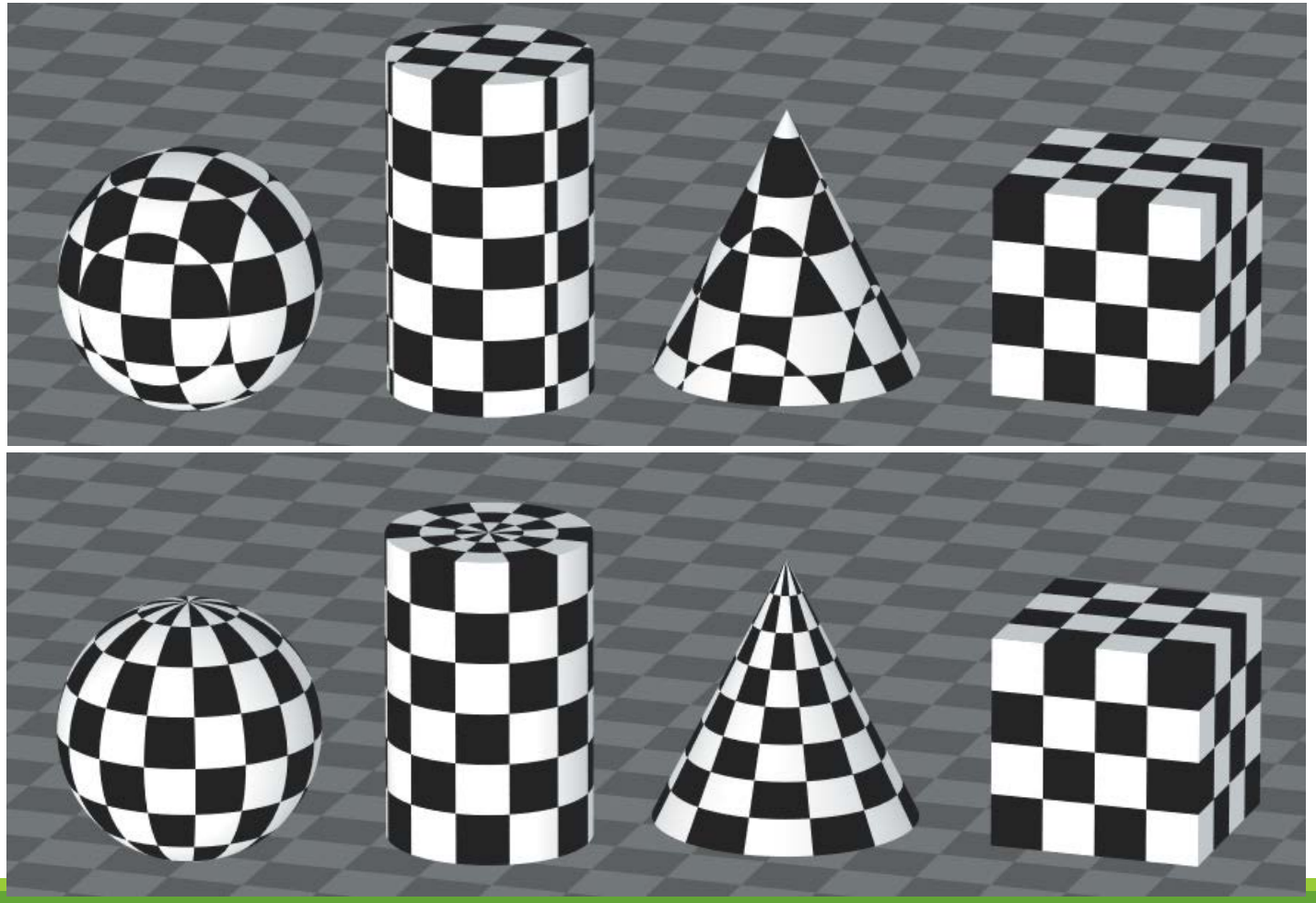
Figure 10.23: The Dragon Model, Textured with the `Checkerboard3DTexture` Procedural Texture. Notice how the model appears to be carved out of 3D checks, rather than having them pasted on its surface.

Procedural textures 3D



(1 ray
per pixel)

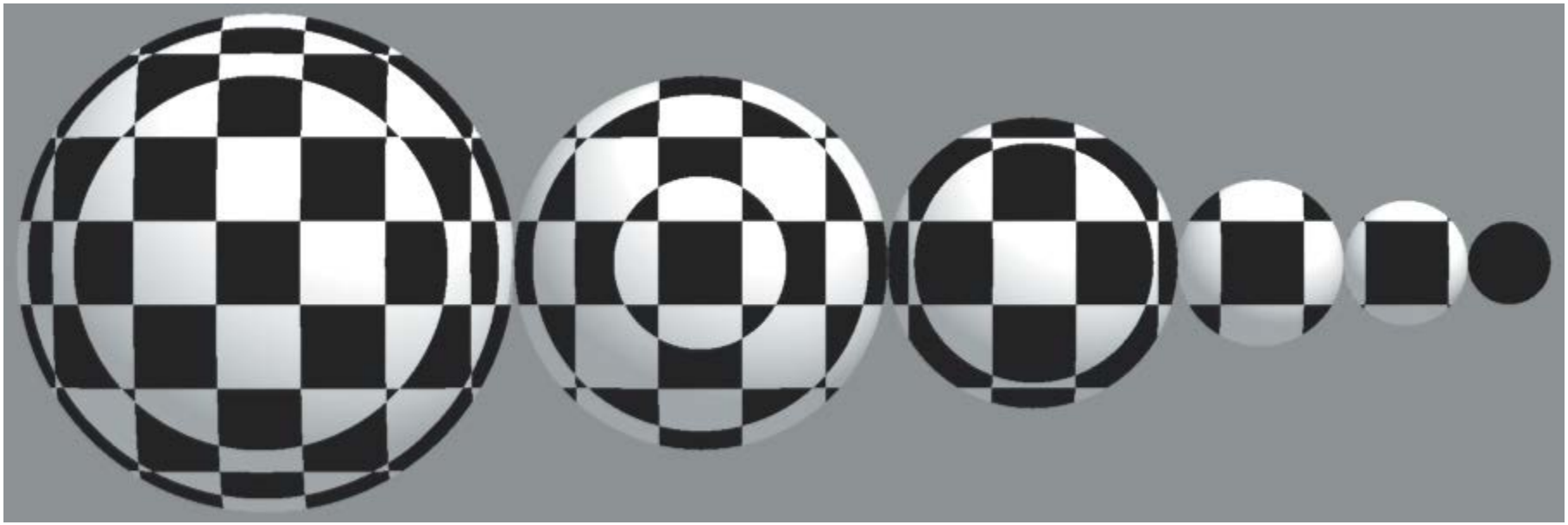
Procedural textures 2D



(cfr. uv-mapping)

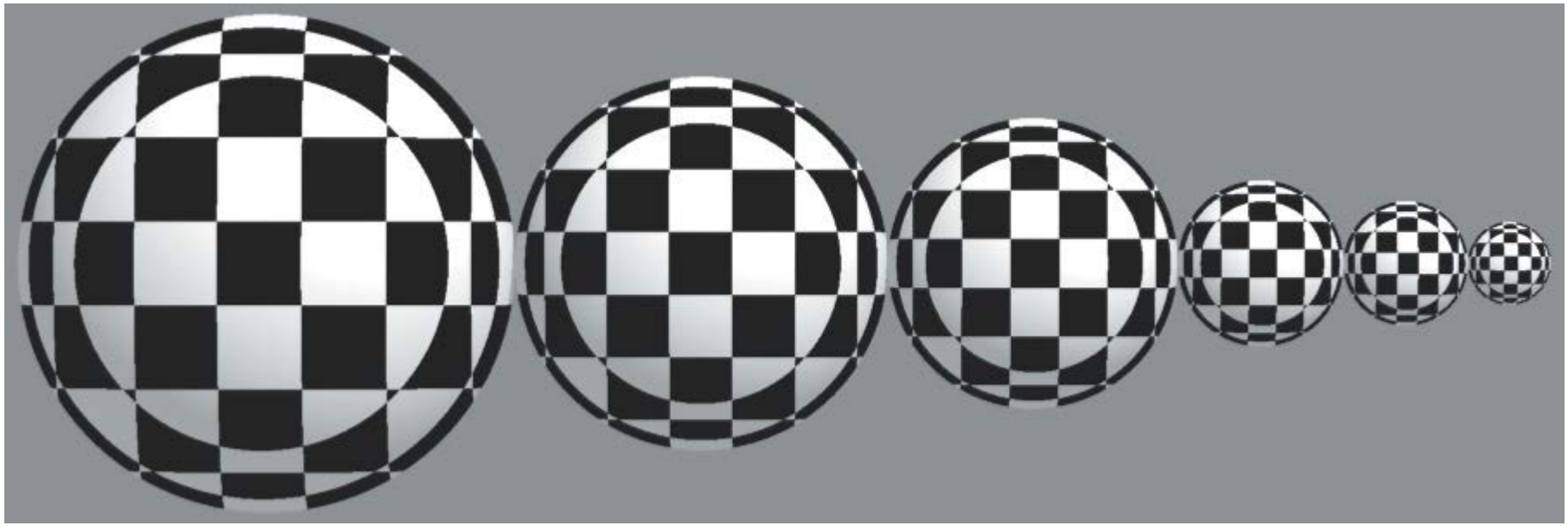
Textures & transformations

Sphere is scaled (transformed), but texture function is not:



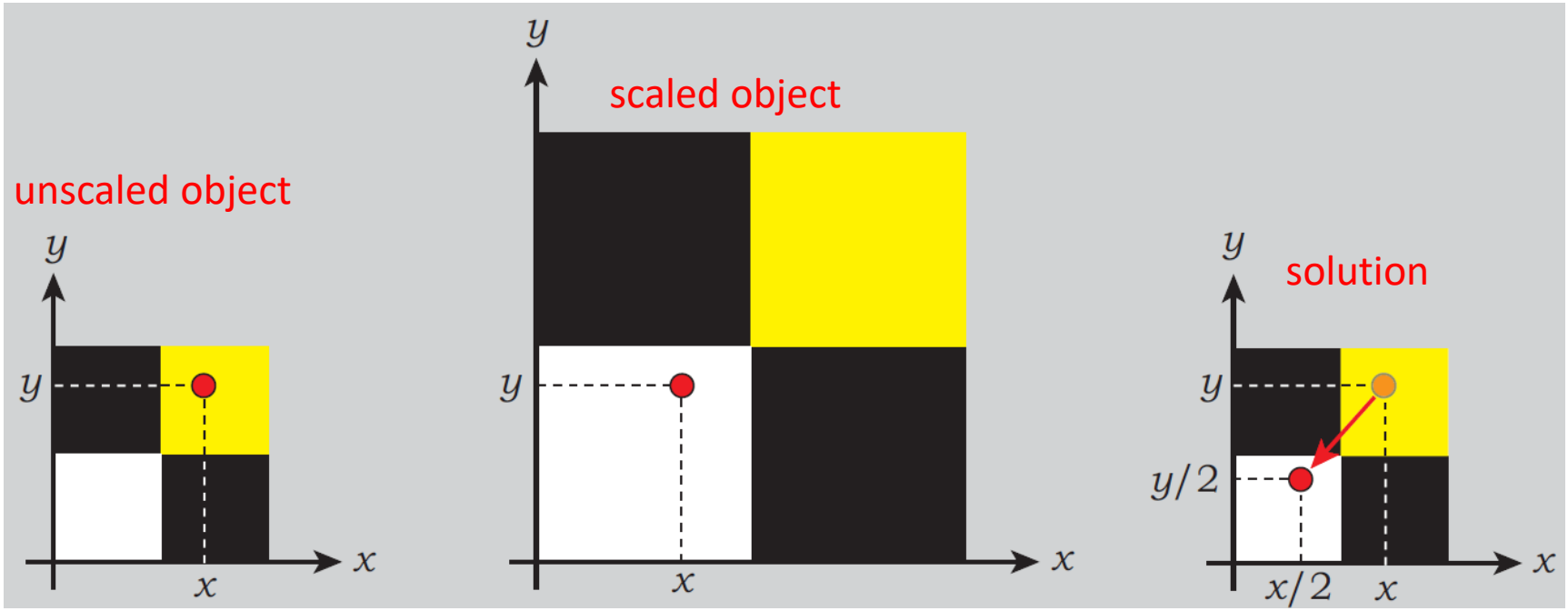
Textures & transformations

What we want:



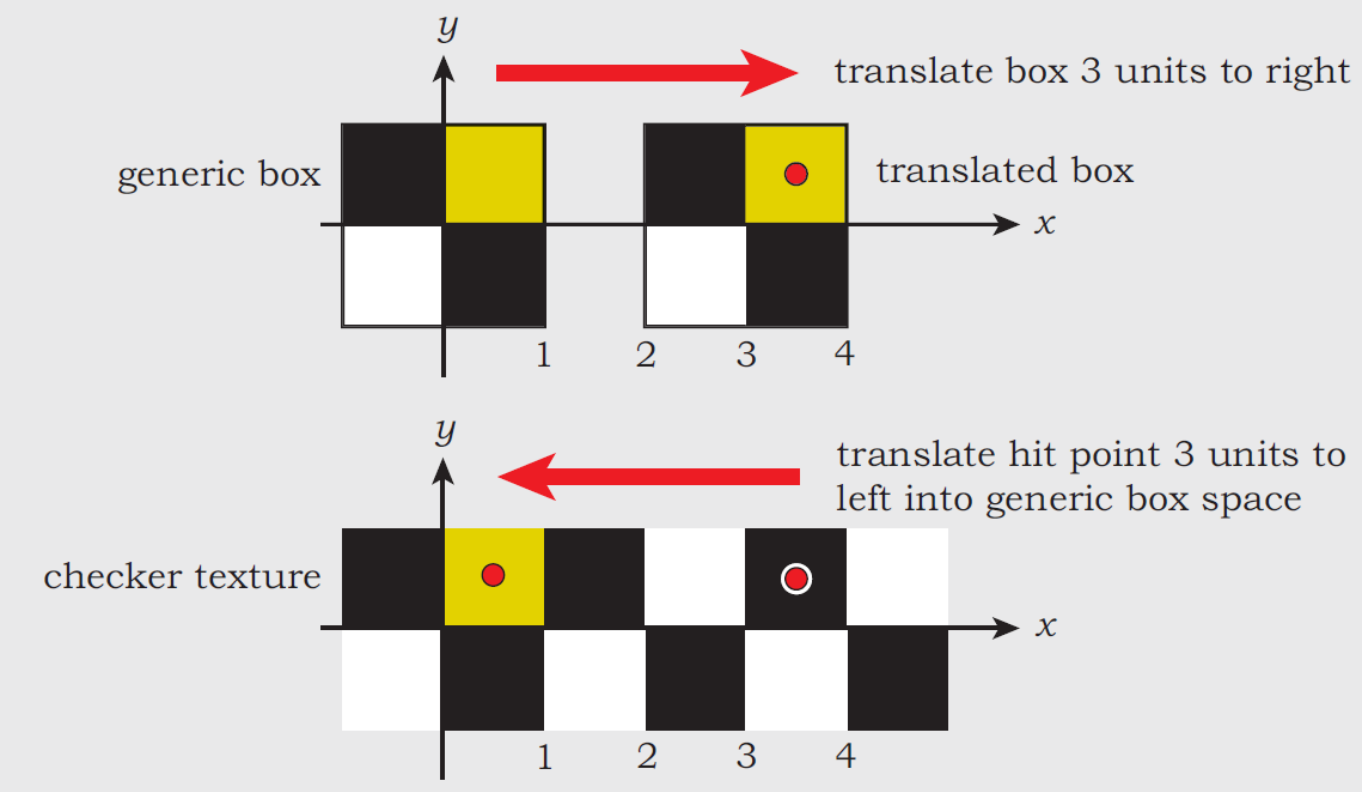
Textures & transformations

Inverse-scale the hit-point
(cfr. intersection of ray with transformed object)



Textures & transformations

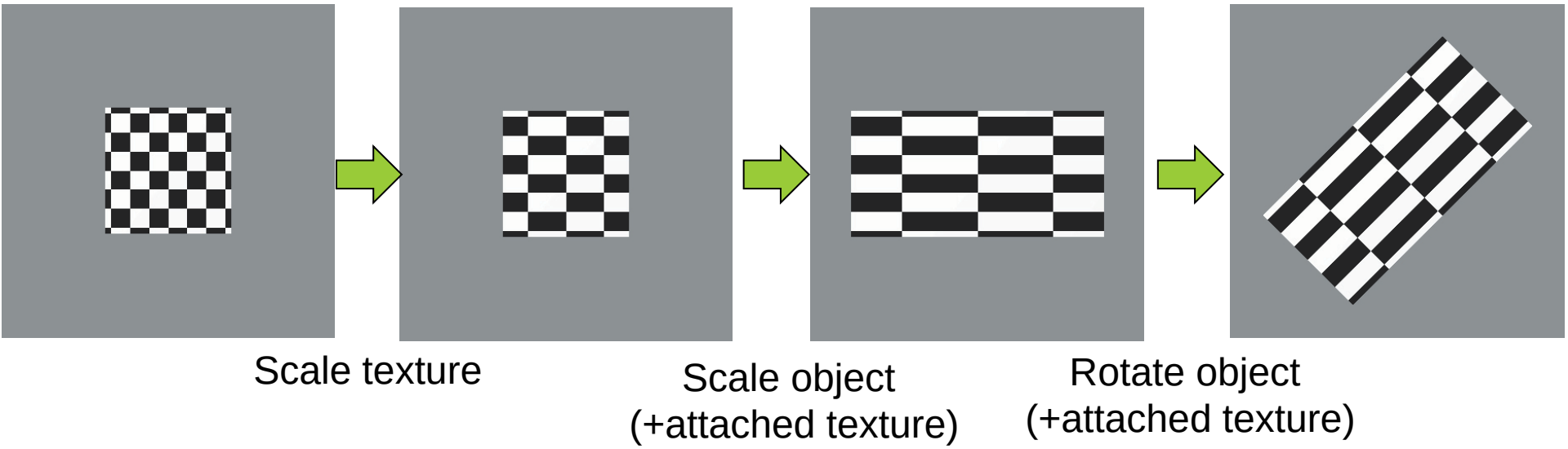
Translation example



Textures & transformations

Transforming object / Transforming texture

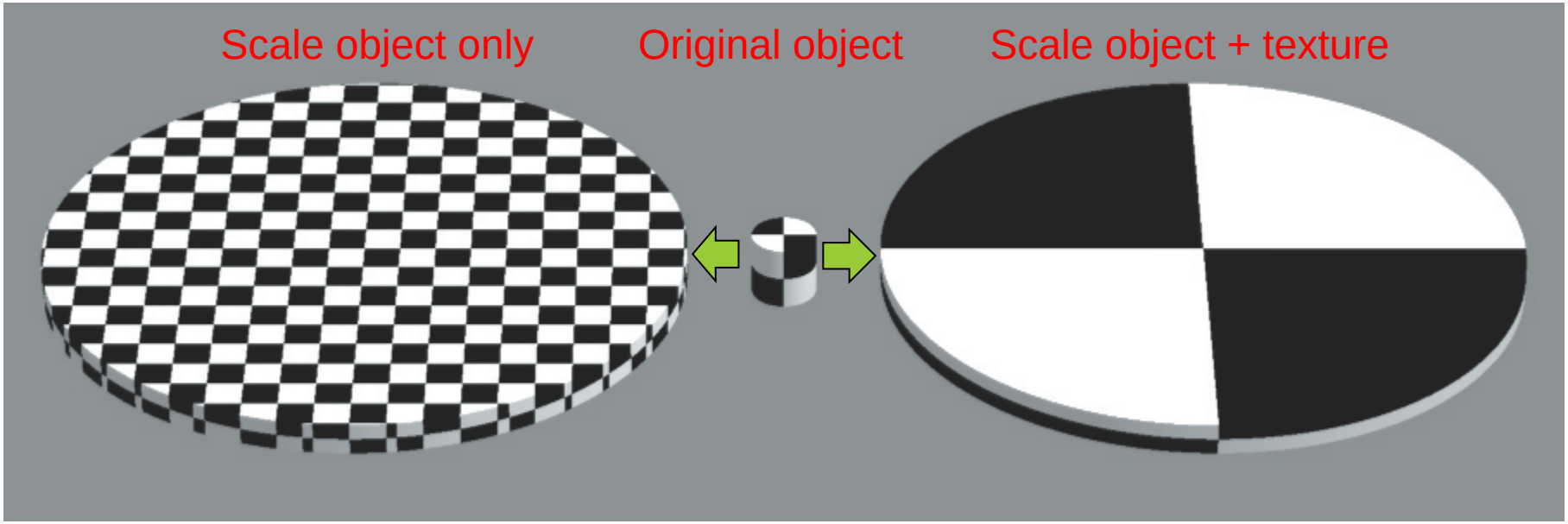
- Sometimes we want them independently from each other



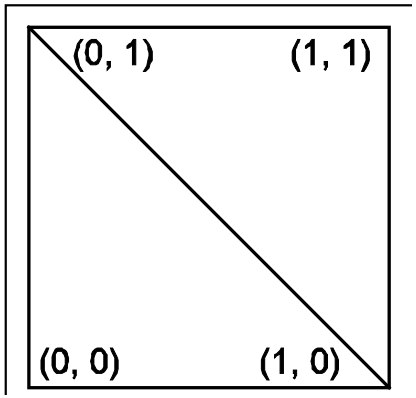
Textures & transformations

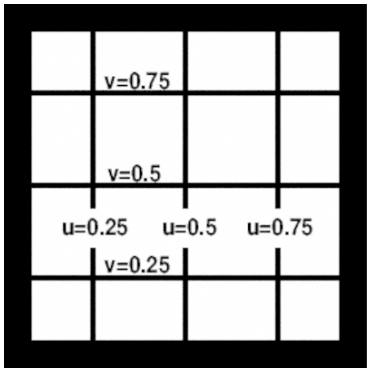
Transforming object / Transforming texture

- Sometimes we want them independently from each other



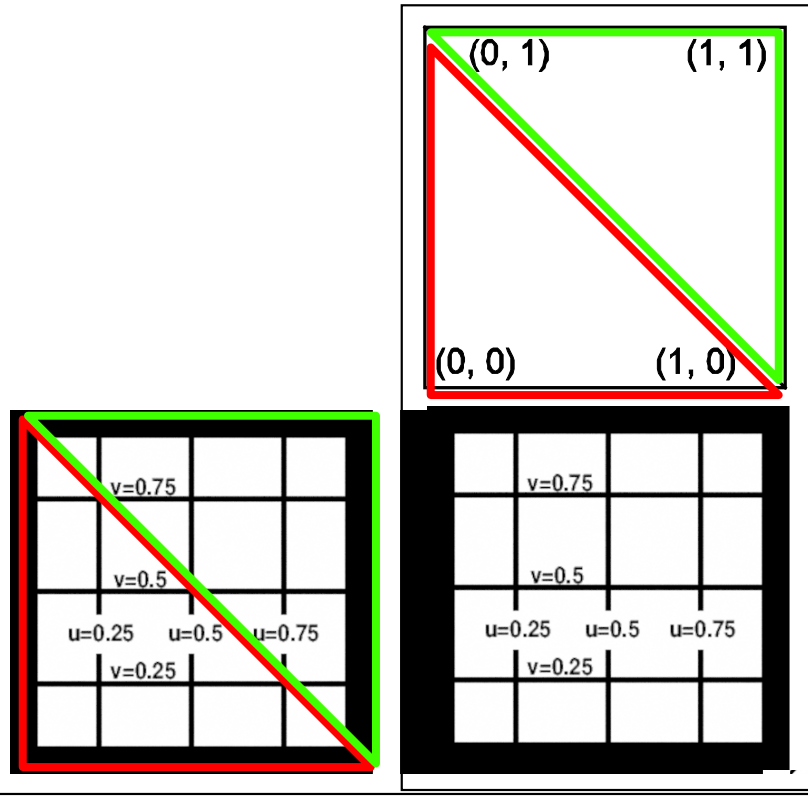
Triangle meshes





original texture

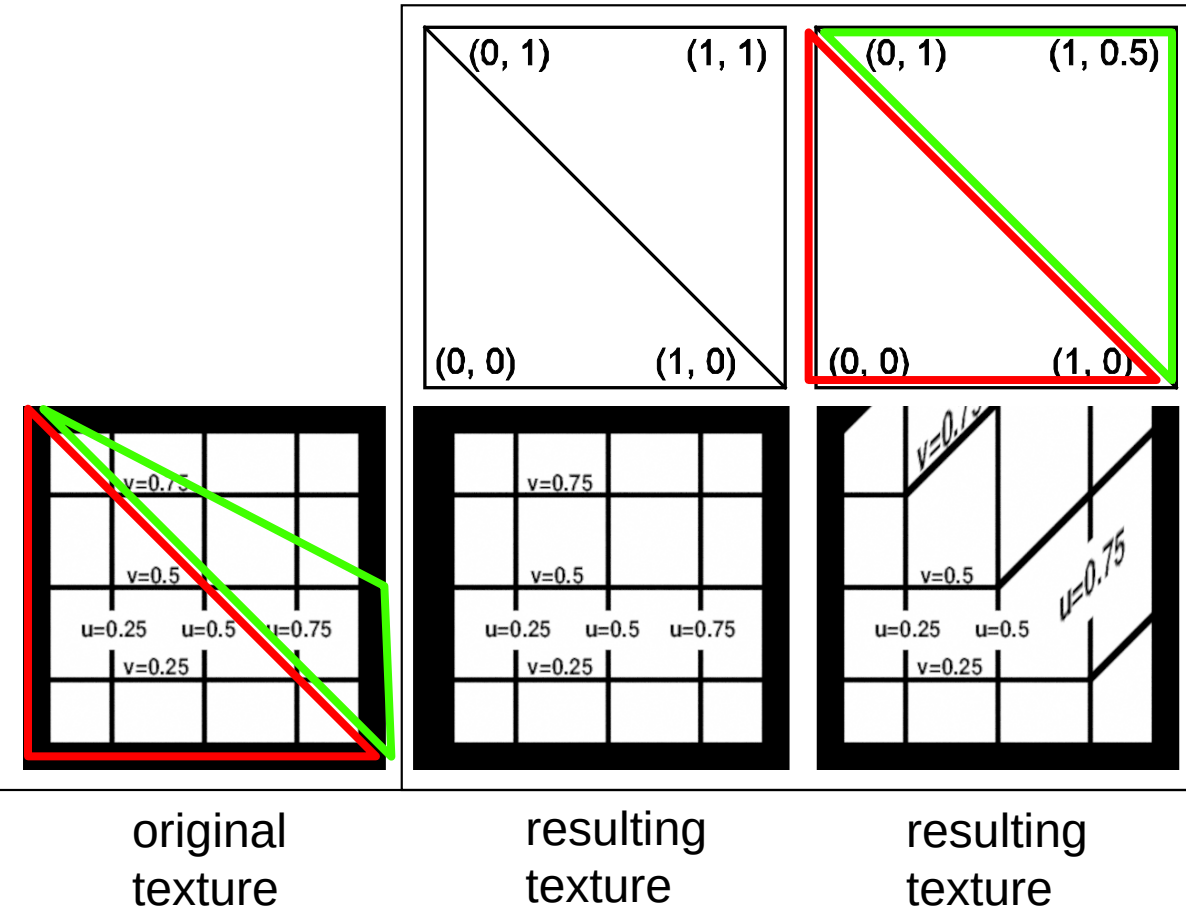
Triangle meshes



original texture

resulting texture

Triangle meshes

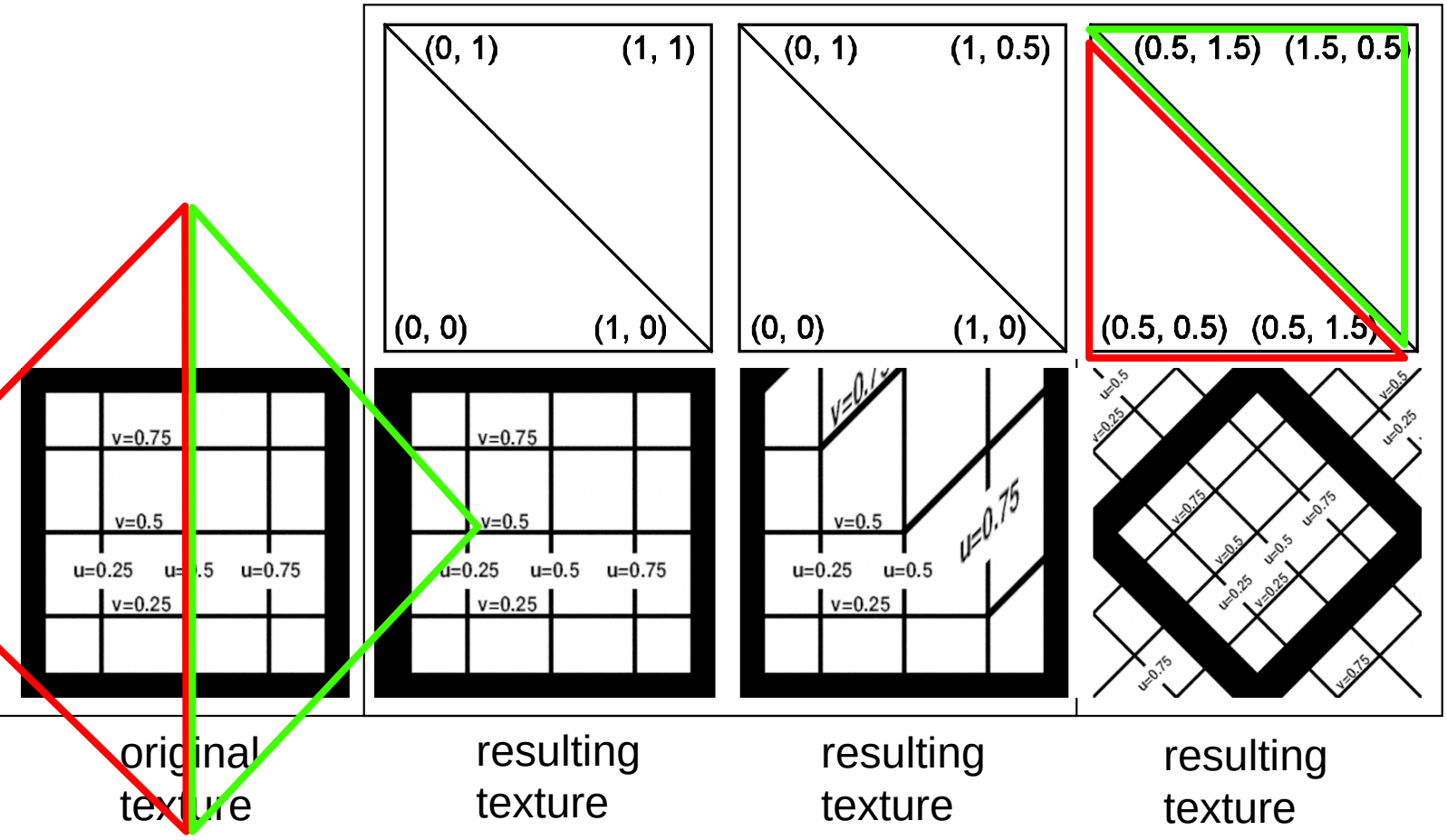


original texture

resulting texture

resulting texture

Triangle meshes



original texture

resulting texture

resulting texture

resulting texture