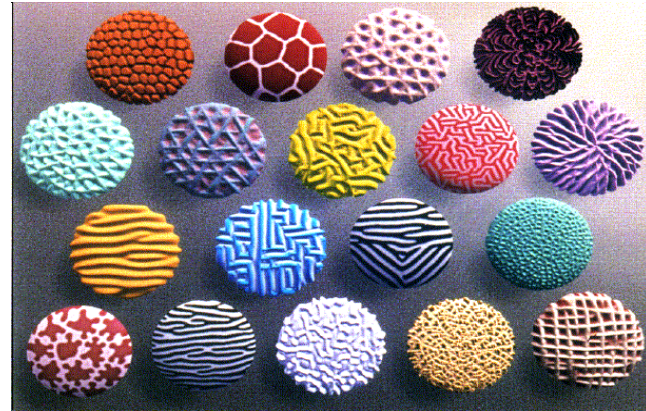


TEXTURE MAPPING



TEXTURE MAPPING

- real life objects have nonuniform colors, normals
- to generate realistic objects, reproduce coloring & normal variations = **texture**
- can often replace complex geometric details

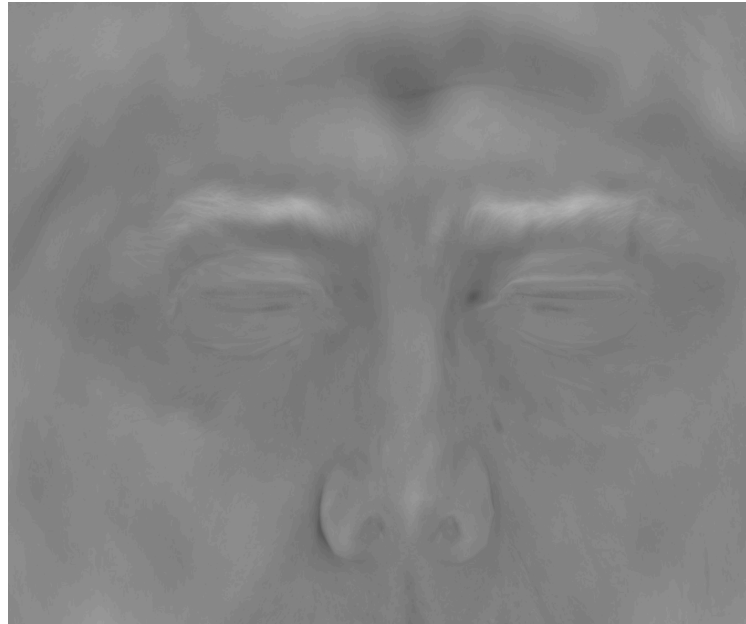


TEXTURE MAPPING

- hide geometric simplicity
 - images convey illusion of geometry
 - map a brick wall texture on a flat polygon
 - create bumpy effect on surface
- usually: 2D information associated with a 3D surface
 - point on 3D surface \leftrightarrow point in 2D texture
 - typically r,g,b colors
 - but can be any attributes that you would like to model over a surface

BUMP MAPS

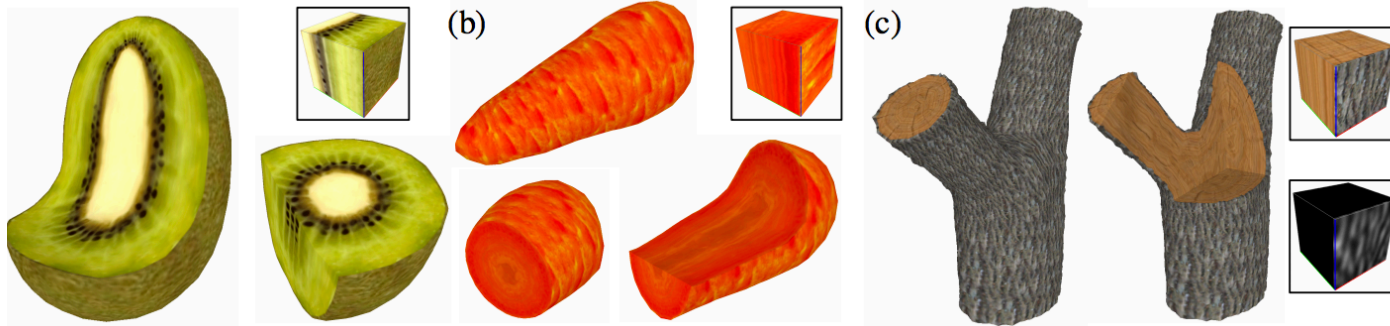
2D texture maps that are used to model the appearance of surface bumps, by adding small perturbations to the surface normals. The rendered geometry does not actually have bumps, i.e., it is smooth !!



threejs.org: materials/bumpmap

VOLUMETRIC TEXTURES

- model r,g,b for every point in a volume
- often computed using procedural function



[Lapped Solid Textures, SIGGRAPH 2008]

ENVIRONMENT MAP



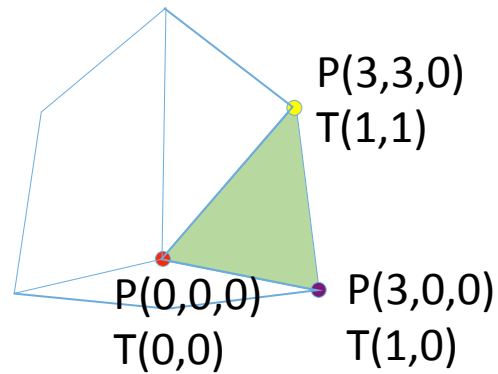
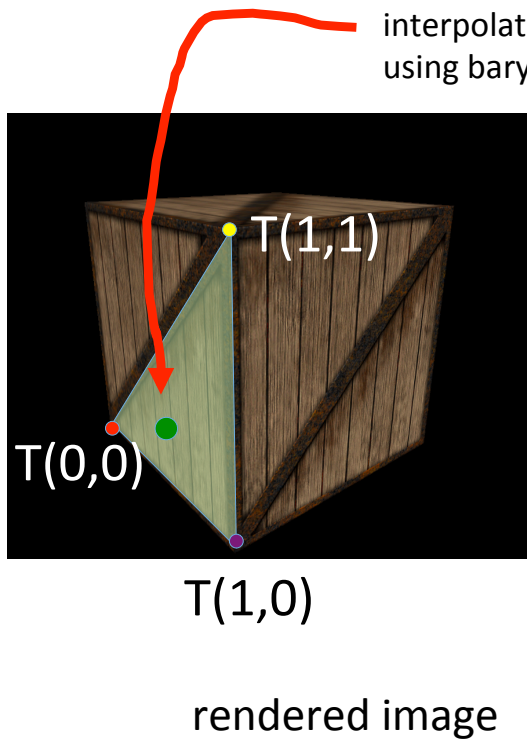
three.js - webgl environment mapping example
Equirectangular Map by University of Southern California
Spherical Map by Paul Debevec



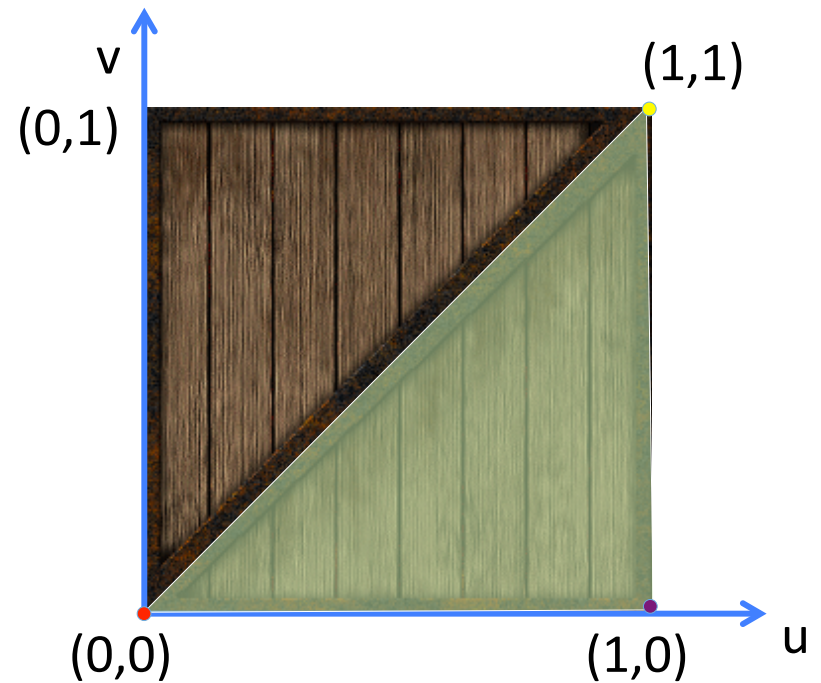
2 of 6 images for a cube map;
as a viewer, you are inside this cube!

There is an invisible corner seam in this image!

BASIC TEXTURE MAP

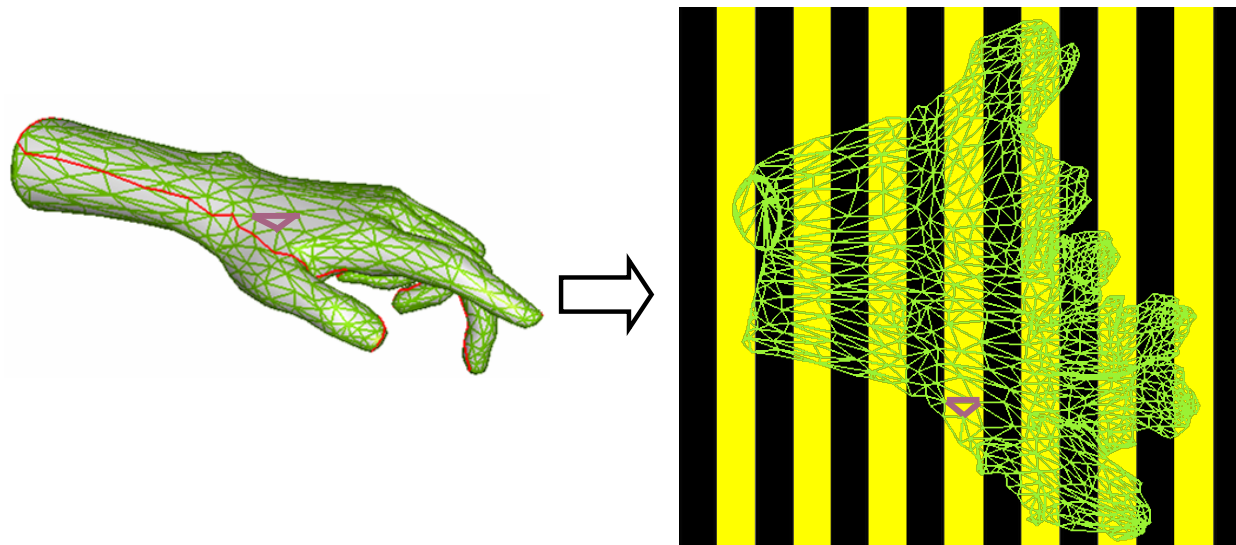
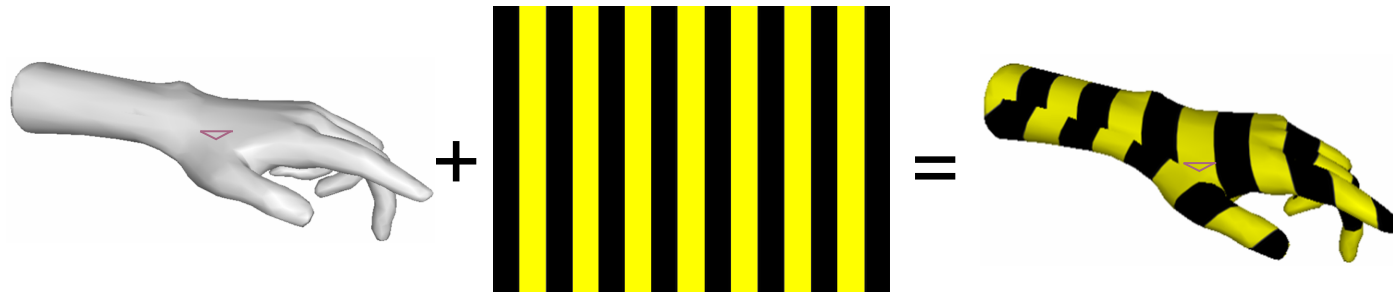


3D model:
 u,v texture coordinates are assigned to vertices by artist or program.



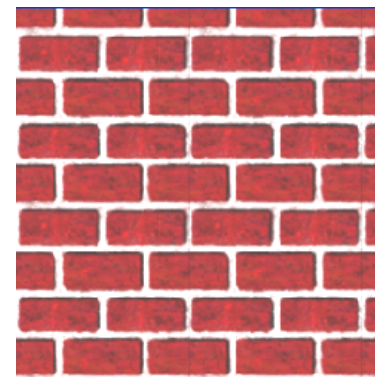
2D texture map: Image
Pixels here are called "texels"

TEXTURE MAPPING EXAMPLE



TEXTURE LOOKUP: TILING AND CLAMPING

- What if s or t is outside $[0 \dots 1]$?
- Multiple choices, e.g.:
 - `tex1.wrapS = THREE.RepeatWrapping`
 - `tex1.wrapS = THREE.ClampToEdgeWrapping`
 - `tex1.wrapS = THREE.MirroredRepeatWrapping`



TEXTURES: VERTEX SHADER & FRAGMENT SHADER

- javascript: texture is passed as a “uniform” to the fragment shader:
(slightly more complex than this due to async image load in js)

```
var myTexture = new THREE.TextureLoader().load( 'textures/crate.gif' );  
myTexture.wrapS = THREE.RepeatWrapping;  
var material = new THREE.MeshBasicMaterial( { map: myTexture } );
```

- vertex shader

```
attribute vec2 uv;  
varying vec2 uvCoords;  
uvCoords = uv;
```

- Fragment Shader:

```
uniform sampler2D myTexture;  
varying vec2 uvCoords;  
vec4 texColor = texture2D(myTexture, uvCoords);  
gl_FragColor = texColor;
```

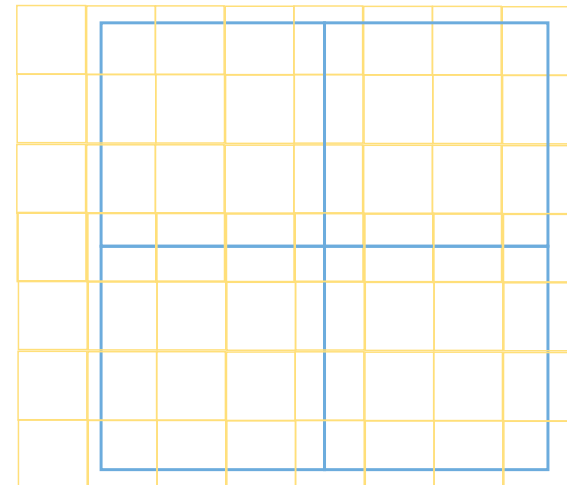
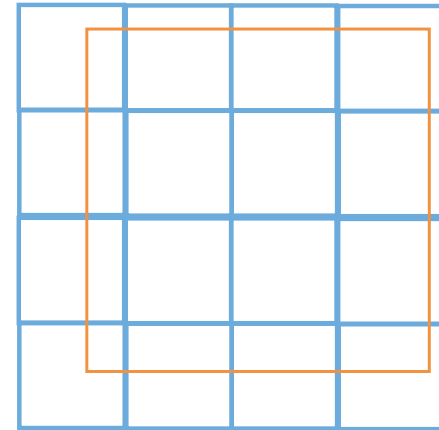
RECONSTRUCTION

- how to deal with:
 - **pixels** that are much larger than **texels**?
 - minification

```
THREE.NearestFilter  
THREE.NearestMipMapNearestFilter  
THREE.NearestMipMapLinearFilter  
THREE.LinearFilter  
THREE.LinearMipMapNearestFilter  
THREE.LinearMipMapLinearFilter
```

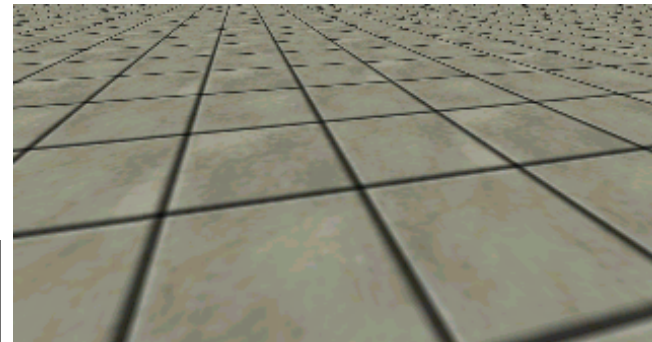
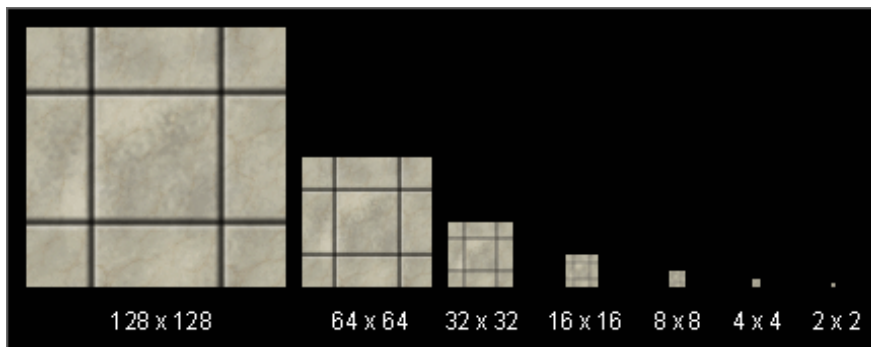
- **pixels** that are much smaller than **texels** ?
 - magnification

```
THREE.NearestFilter  
THREE.LinearFilter
```

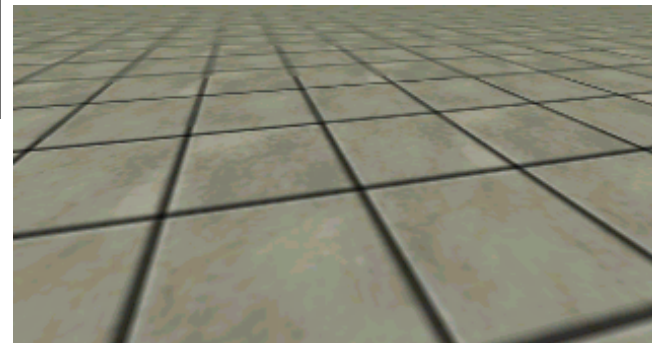


MIPMAPPING

use “image pyramid” to precompute averaged versions of the texture



Without MIP-mapping



With MIP-mapping

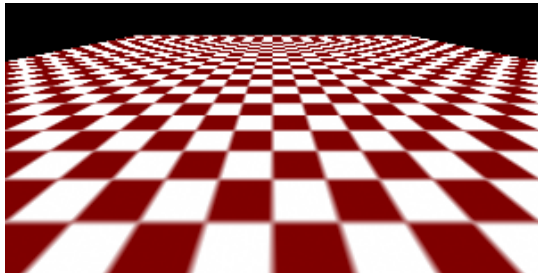
MIPMAPS

- **multum in parvo** -- many things in a small place
 - prespecify a series of prefiltered texture maps of decreasing resolutions
 - requires more texture storage
 - avoid shimmering and flashing as objects move

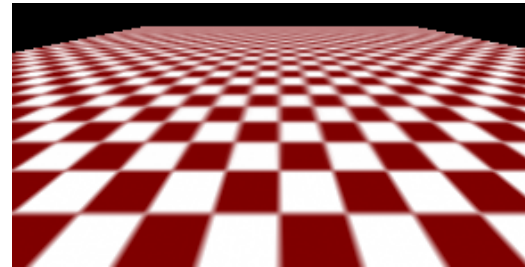
e.g.:

```
texture.magFilter = THREE.NearestFilter;  
texture.minFilter THREE.LinearMipMapLinearFilter;
```

without

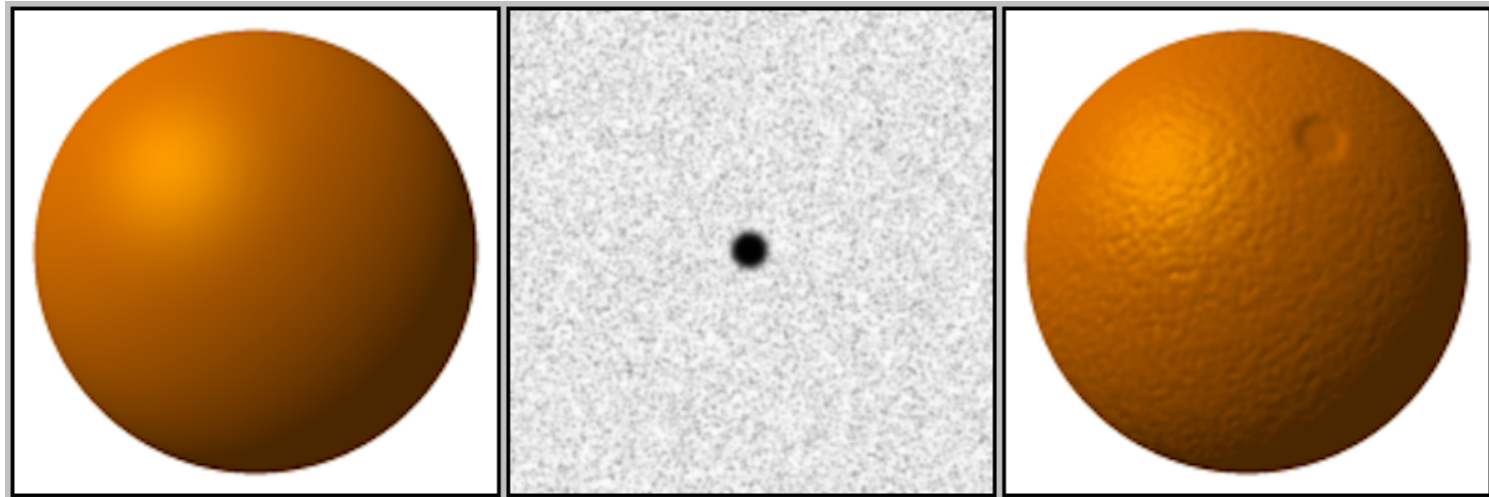


with

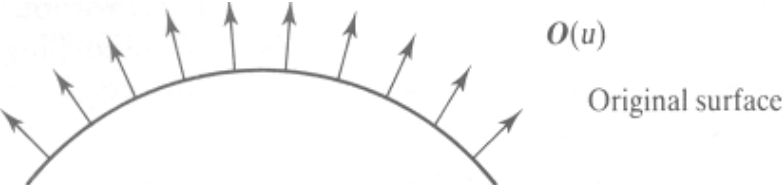


BUMP MAPPING: NORMALS AS TEXTURE

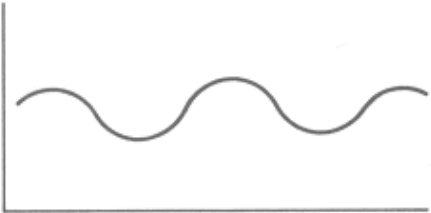
- object surface often not smooth – to recreate correctly need complex geometry model
- can control shape “effect” by locally perturbing surface normal
 - random perturbation
 - directional change over region



BUMP MAPPING

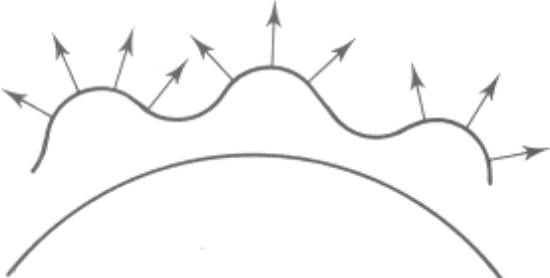


$O(u)$
Original surface



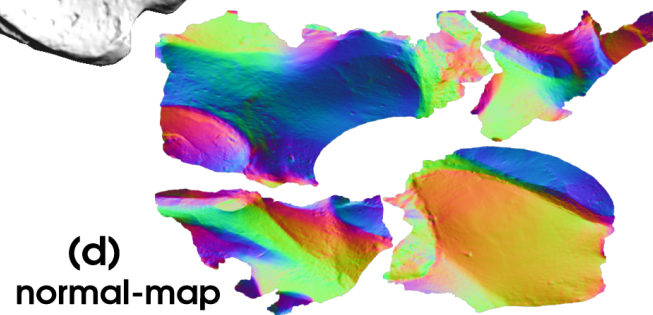
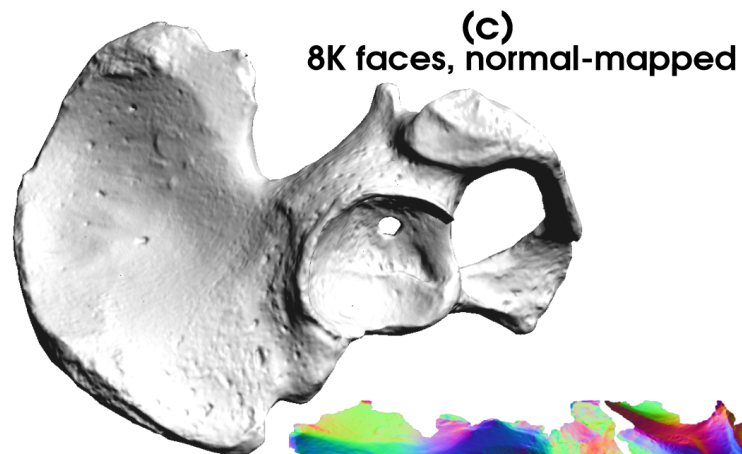
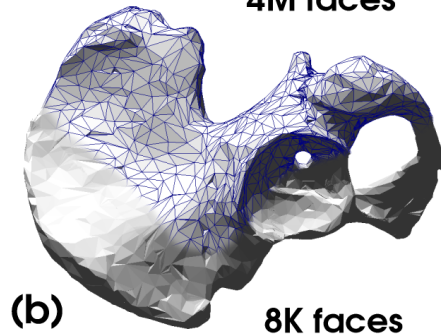
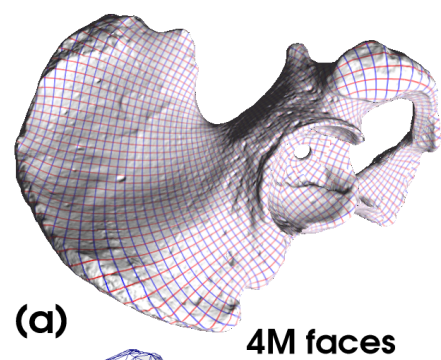
$B(u)$
A bump map

virtual surface
created with the bump map

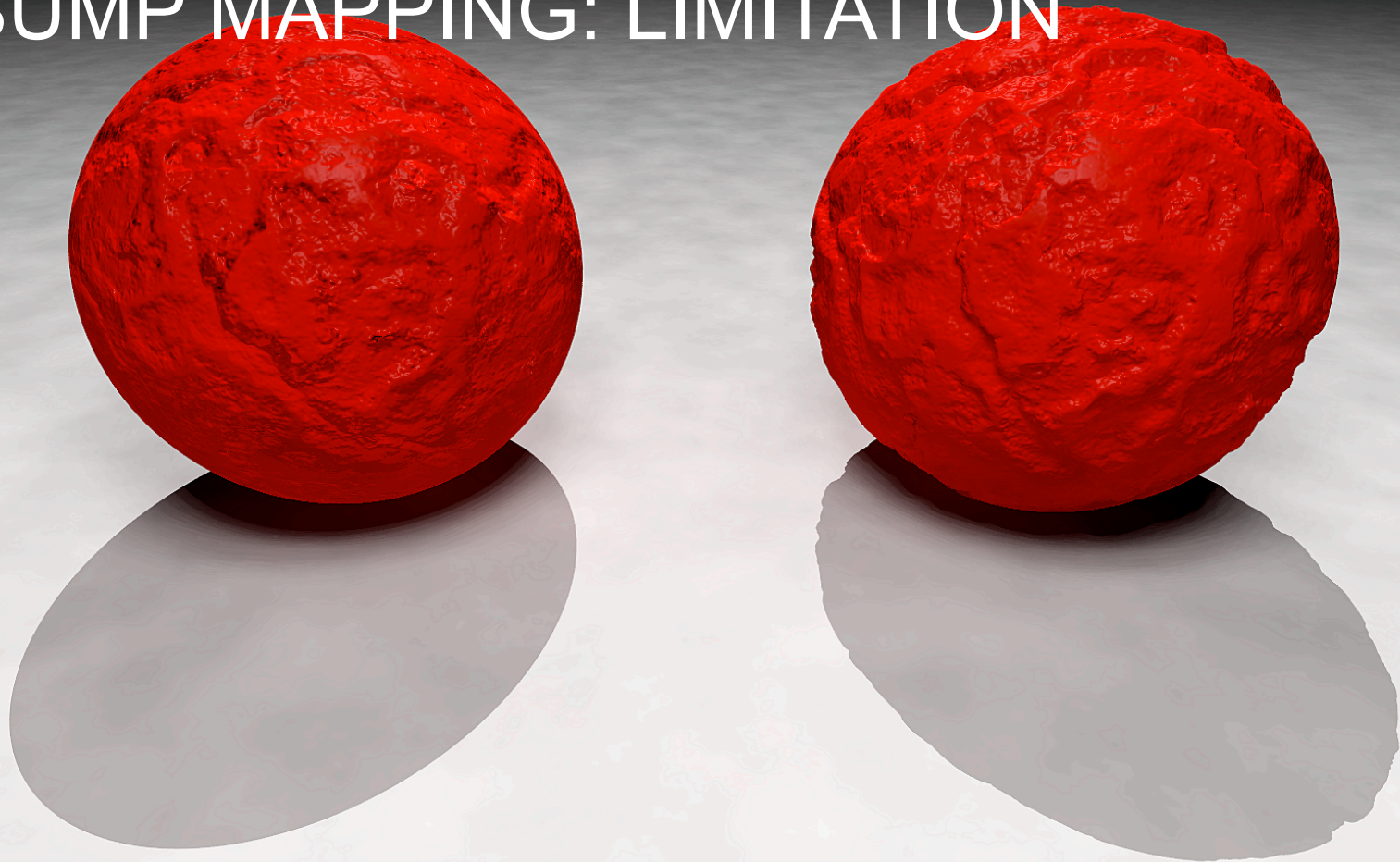


normals corresponding to this
virtual surface

Normal/Bump mapping

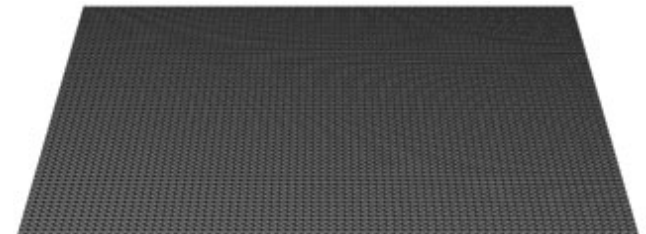


BUMP MAPPING: LIMITATION



DISPLACEMENT MAPPING

- bump mapping gets silhouettes wrong
 - shadows wrong too
- change surface geometry instead
 - need to subdivide surface
 - use tessellation shader



ORIGINAL MESH



DISPLACEMENT MAP



MESH WITH DISPLACEMENT

https://en.wikipedia.org/wiki/Displacement_mapping#/media/

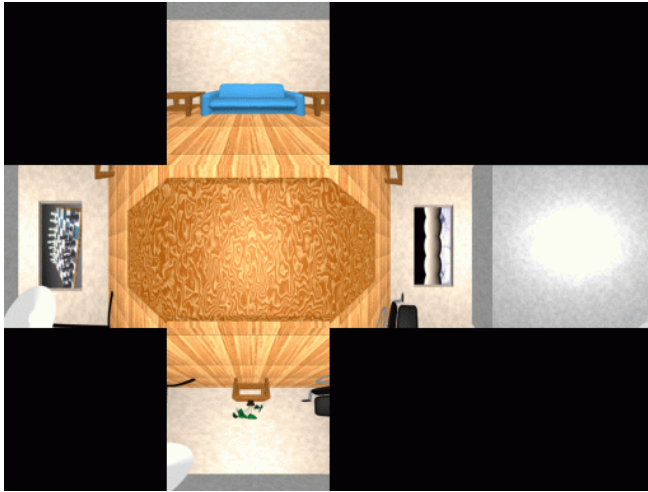
ENVIRONMENT MAPPING

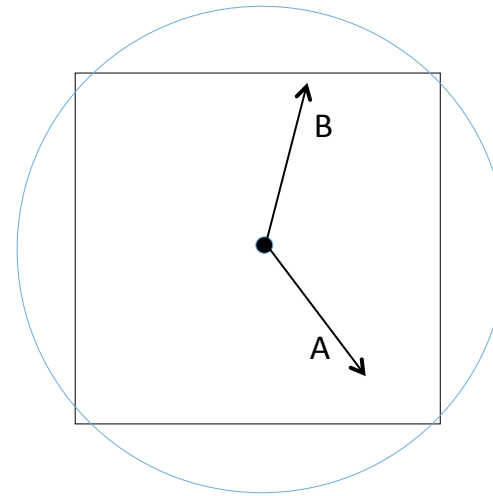
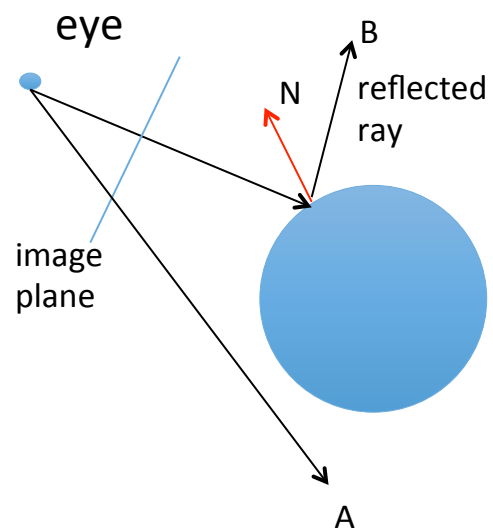
- generate image of surrounding or reflection
- sphere map or cube map



CUBE MAP

- 6 planar textures, sides of cube
 - point camera in 6 different directions, facing out from origin



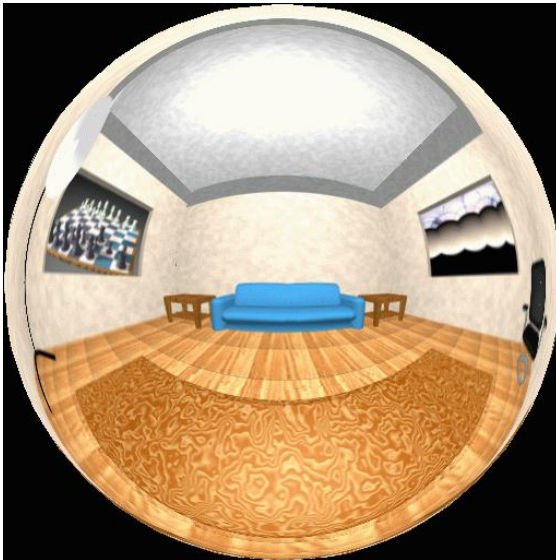


note: viewpoint is always at the center!

- Cube map: direction of vector selects the face of the cube to be indexed
 - co-ordinate with largest magnitude
 - e.g., the vector $(-0.2, 0.5, -0.84)$ selects the $-Z$ face
 - remaining two coordinates select the pixel from the face.

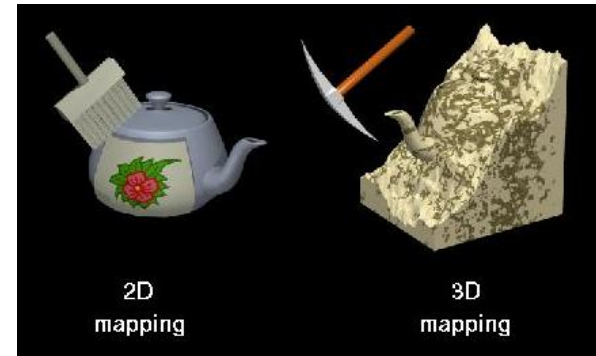
SPHERE MAP

- texture is distorted fish-eye view
 - point camera at mirrored sphere
 - spherical texture mapping creates texture coordinates that correctly index into this texture map



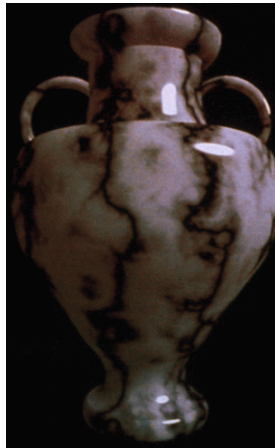
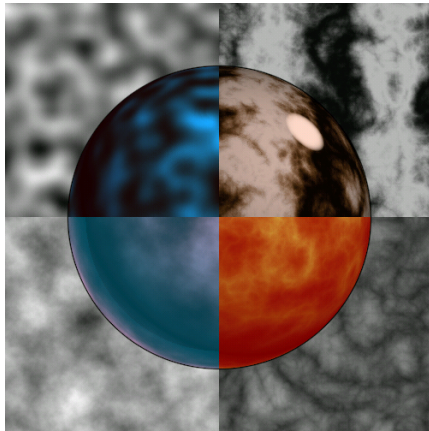
VOLUMETRIC TEXTURE

- define texture pattern over 3D domain - 3D space containing the object
- texture function can be digitized or **procedural**
- for each point on object compute texture from point location in space
- e.g., ShaderToy
- computation often cheaper than memory access



PROCEDURAL TEXTURES: PERLIN NOISE

- several good explanations
 - <http://www.noisemachine.com/talk1>
 - http://freespace.virgin.net/hugo.elias/models/m_perlin.htm
 - <http://www.robo-murito.net/code/perlin-noise-math-faq.html>



<http://mrl.nyu.edu/~perlin/planet/>