Computer Graphics

Texture, Environment and Other Maps

Based on slides by Dianna Xu, Bryn Mawr College

The Limits of Geometric Modeling

- Although graphics cards can render over 10 million polygons per second, that number is insufficient for many phenomena
  - Clouds
  - Grass
  - Terrain
  - Skin

Modeling an Orange

- Start with an orange-colored sphere
  - Too simple
- Replace sphere with a more complex shape
  - Does not capture surface characteristics (small dimples)
- Takes too many polygons to model all the dimples

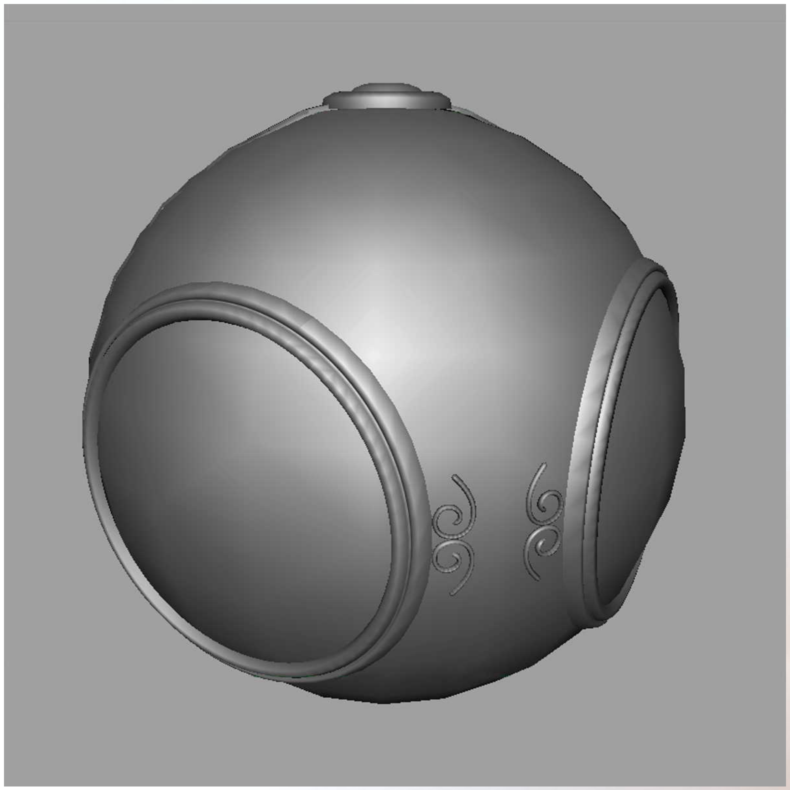
Modeling an Orange

- Take a picture of a real orange, scan it, and "paste" onto simple geometric model
  - This process is texture mapping
- Still might not be sufficient because resulting surface will be smooth
  - Need to change local shape
  - Bump mapping

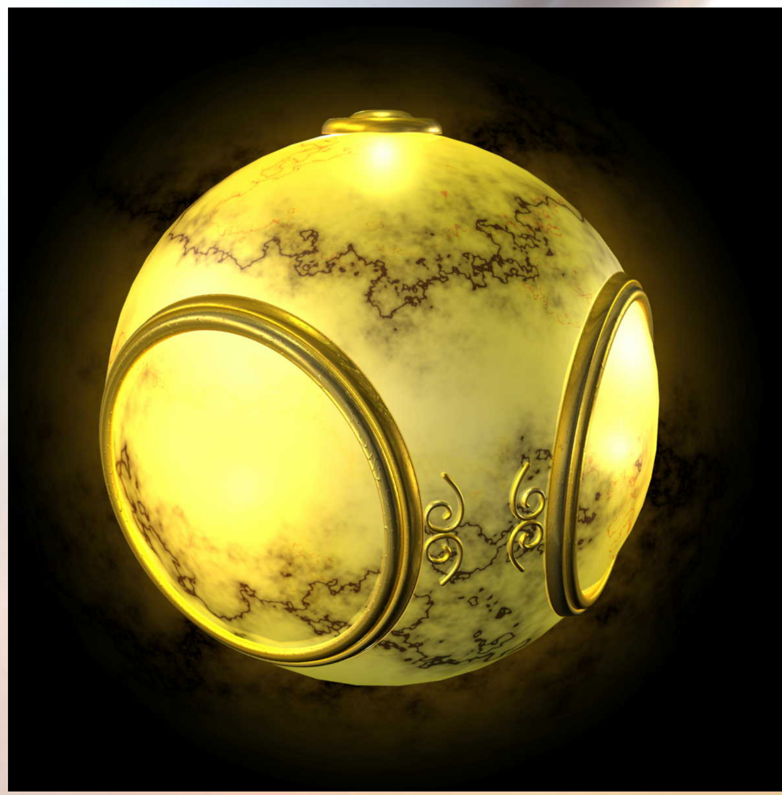
Three Types of Mapping

- Texture Mapping
  - Uses images to fill polygons
- Environmental (reflection mapping)
  - Uses a picture of the environment for texture maps
  - Allows simulation of highly specular surfaces
- Bump mapping
  - Emulates altering normal vectors during the rendering process

Texture Mapping



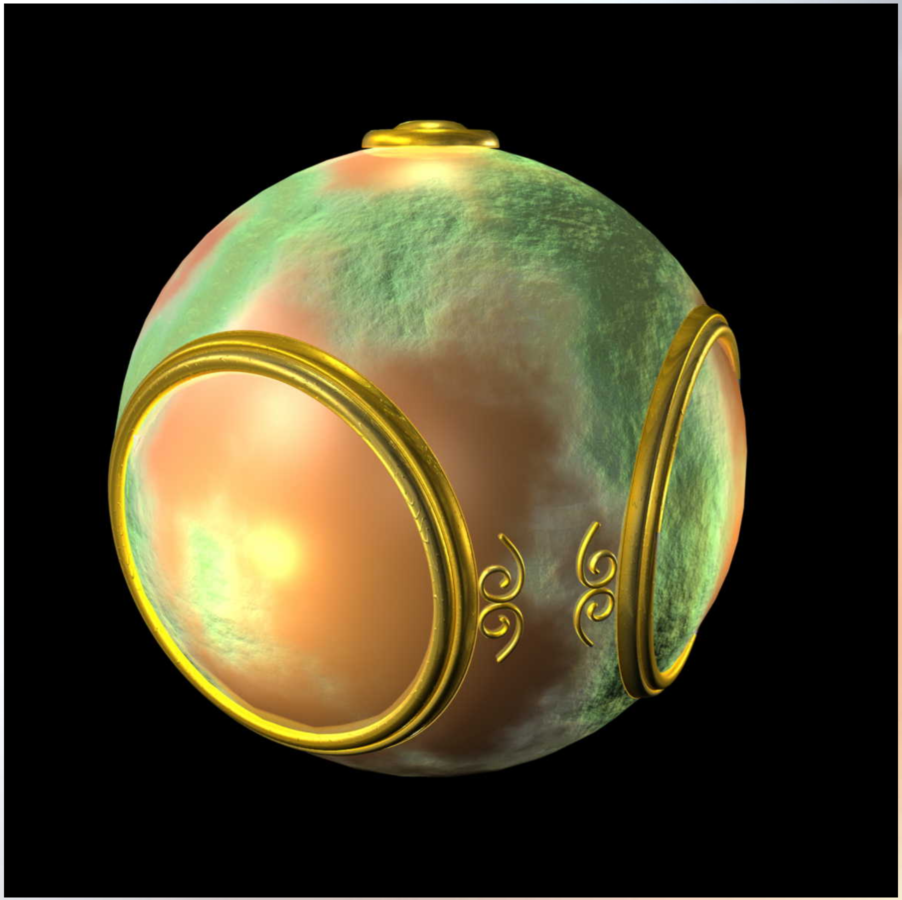
geometric model



texture mapped

Environment Mapping

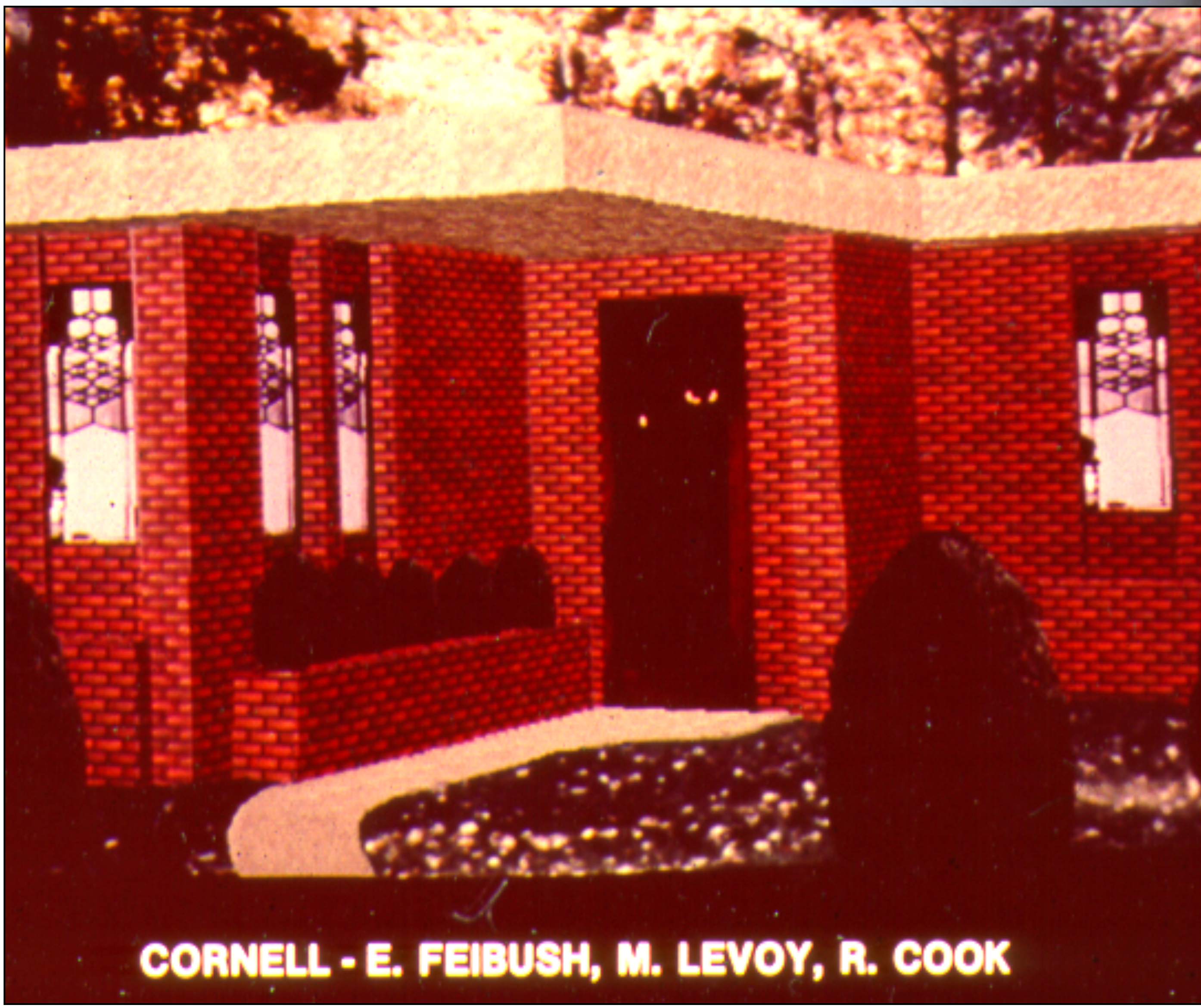
Bump Mapping



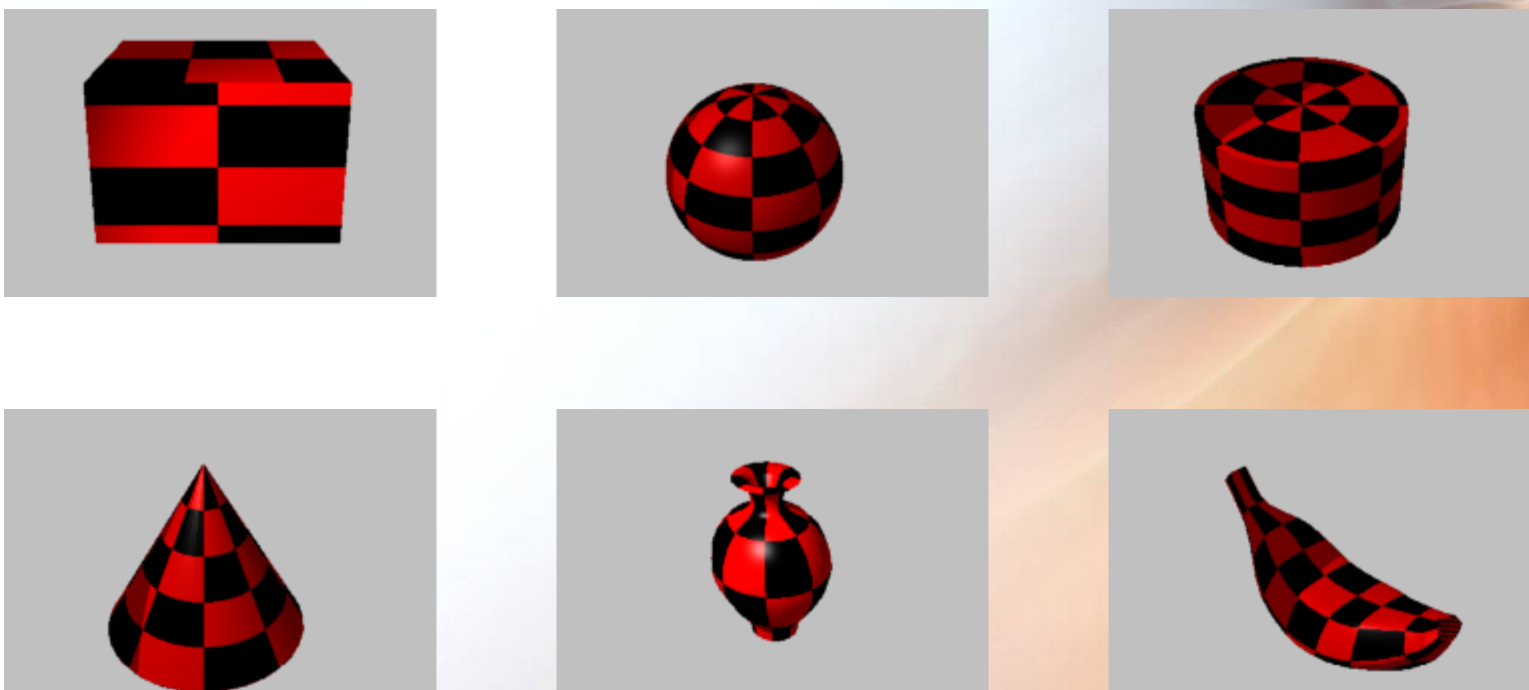
Mapping

- Mapping means taking a 2D (or 3D) function and applying it to any of the attributes of an object or object surface.
- Maps can be explicit arrays of values (such as in 2D images) or procedurally-defined functions F(u,v).
- Maps can modify colors, transmittance, reflective properties, shape, etc.

Efficiency of Texture Maps over Detailed Geometric Modeling

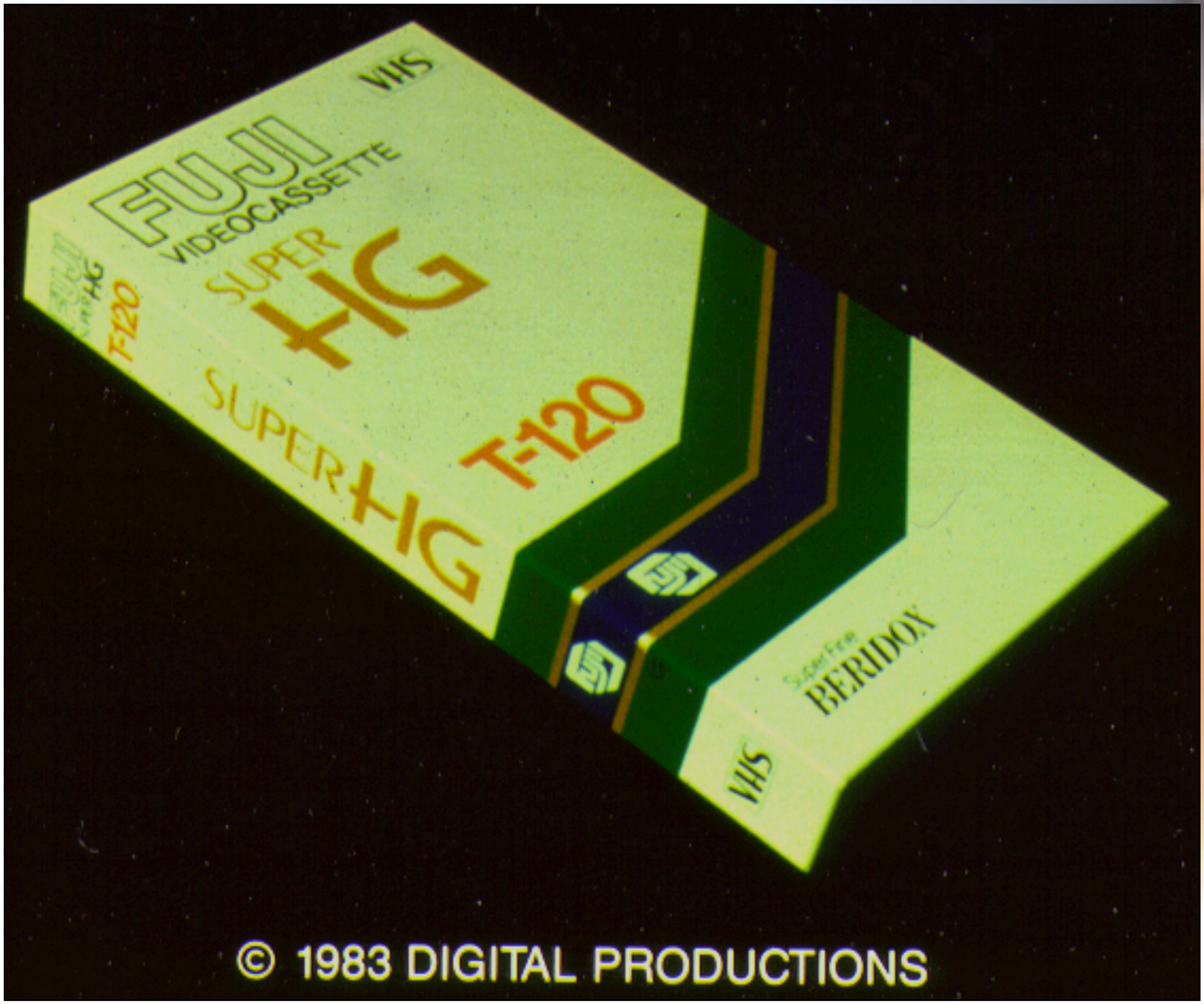


Texture Mapping Examples

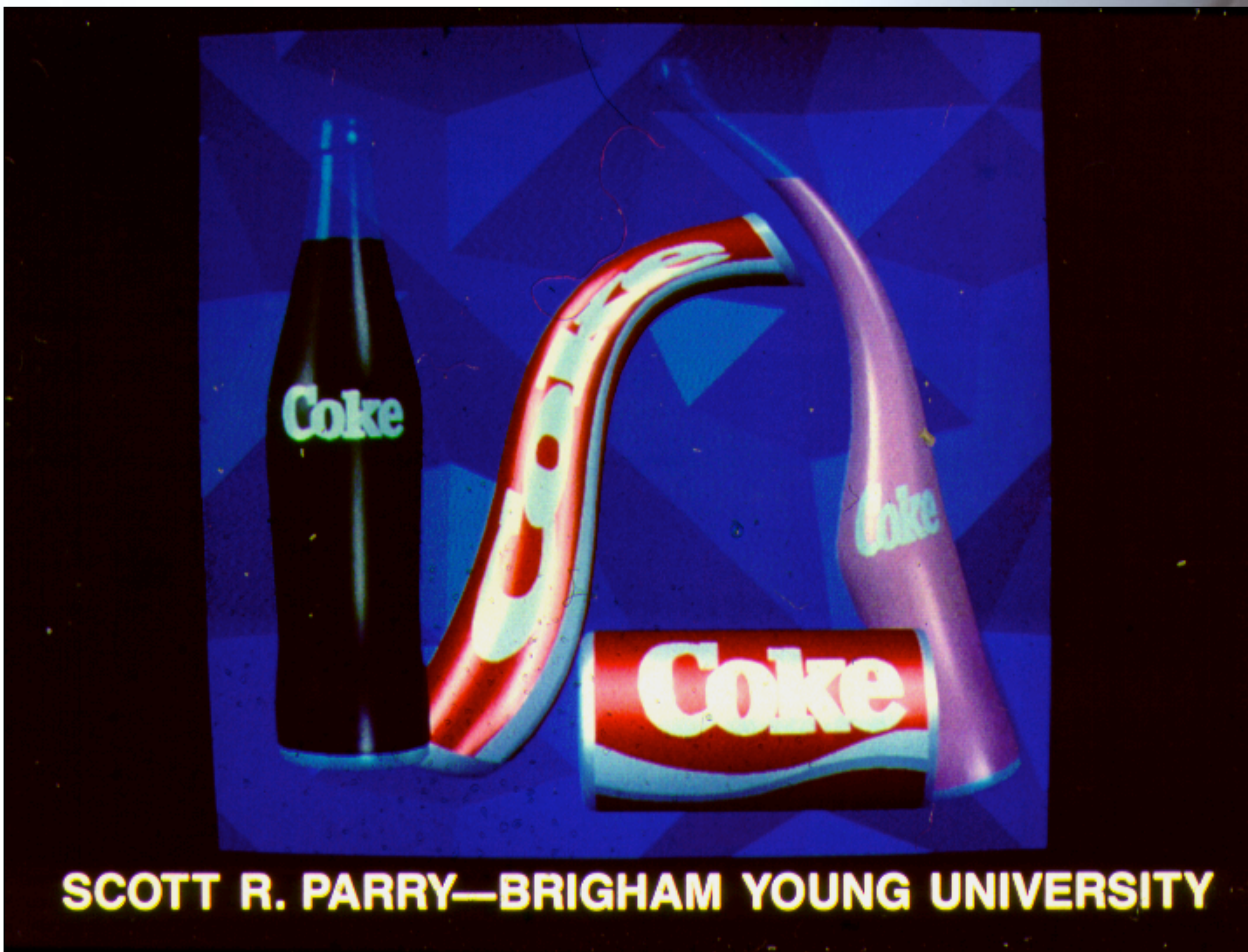


All based on same 2D checkerboard texture

Textures Save Excess Geometric Modeling

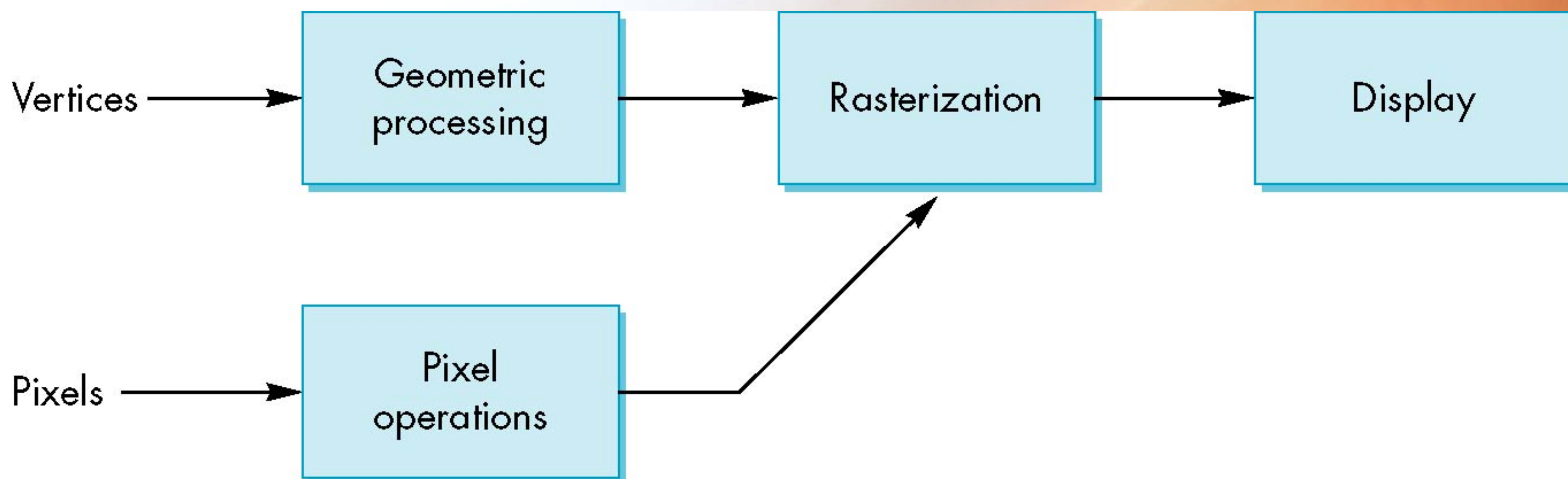


Texture attached to Geometry



Where does mapping take place?

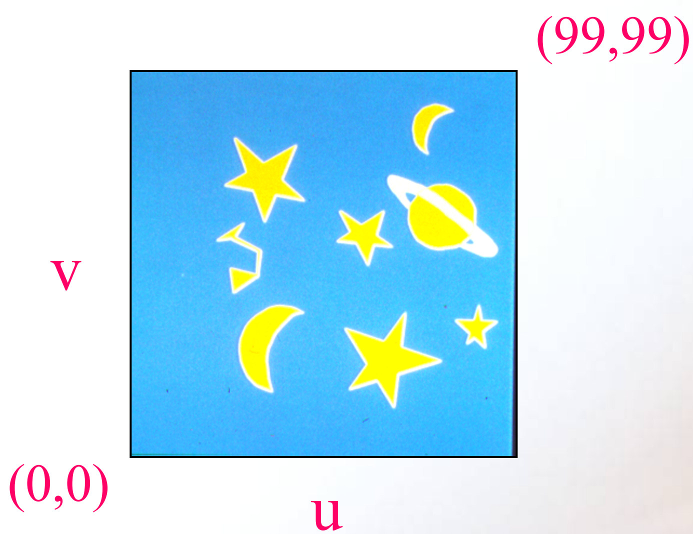
- Mapping techniques are implemented at the end of the rendering pipeline
  - Very efficient because few polygons make it past the clipper



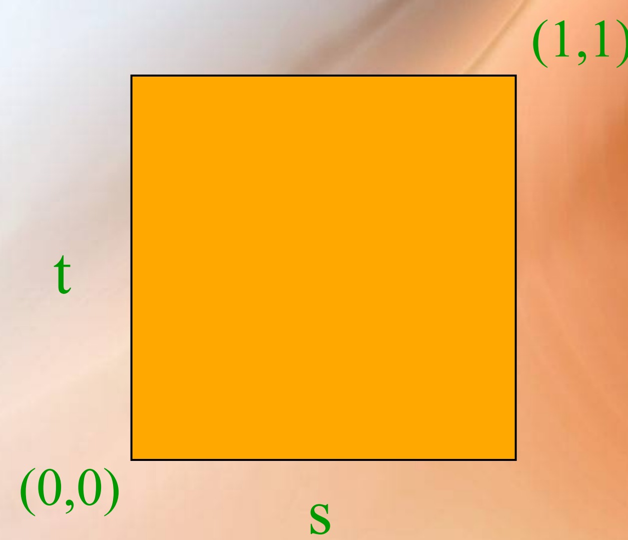
A Very Simple Example of Texture Mapping

- Texture is image (u,v) with 100×100 texels
- Polygon is unit square (s,t) with $0 \leq s \leq 1$ and $0 \leq t \leq 1$
- Texture mapping:
  - $s = u/99$
  - $t = v/99$
- Inverse texture mapping:
  - $u = \text{round}(99s)$
  - $v = \text{round}(99t)$

Simple Texture Mapping



Texture (u,v) coordinates

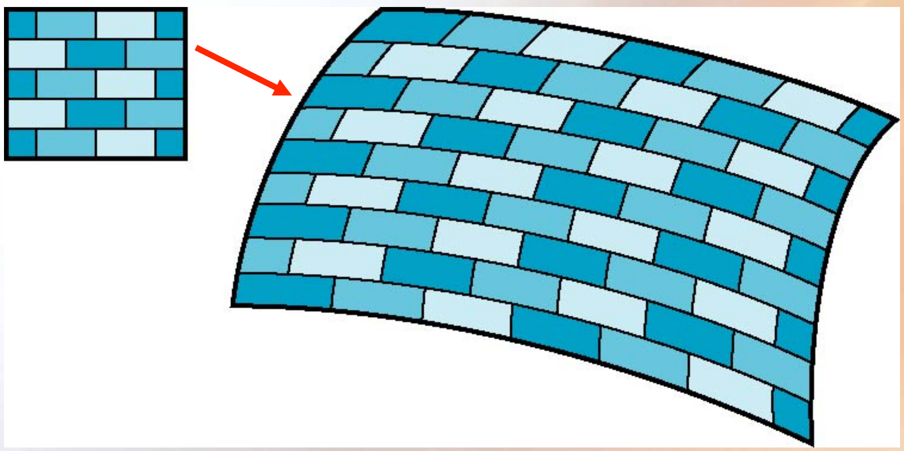


Polygon (s,t) coordinates

Is it Simple?

- **Although the idea is simple – map an image to a surface – there are 3 or 4 coordinate systems involved**

2D image

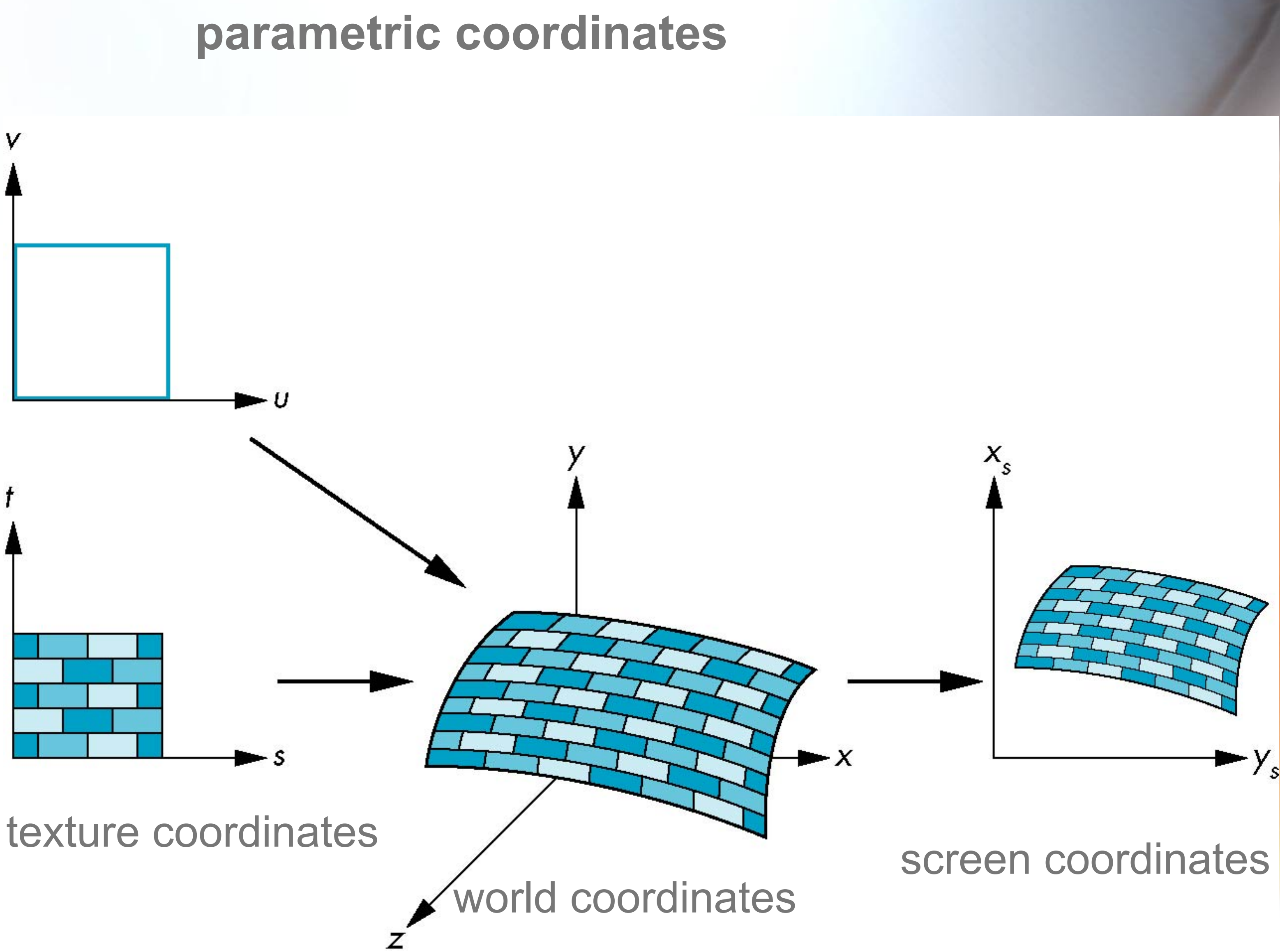


3D surface

Coordinate Systems

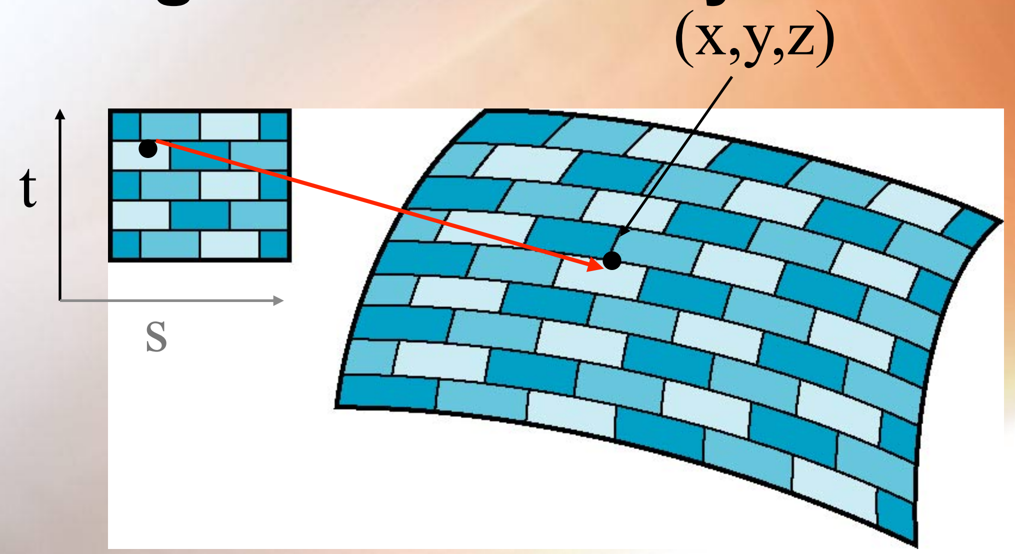
- Parametric coordinates
  - May be used to model curved surfaces
- Texture coordinates
  - Used to identify points in the image to be mapped
- World Coordinates
  - Conceptually, where the mapping takes place
- Screen Coordinates
  - Where the final image is really produced

Texture Mapping

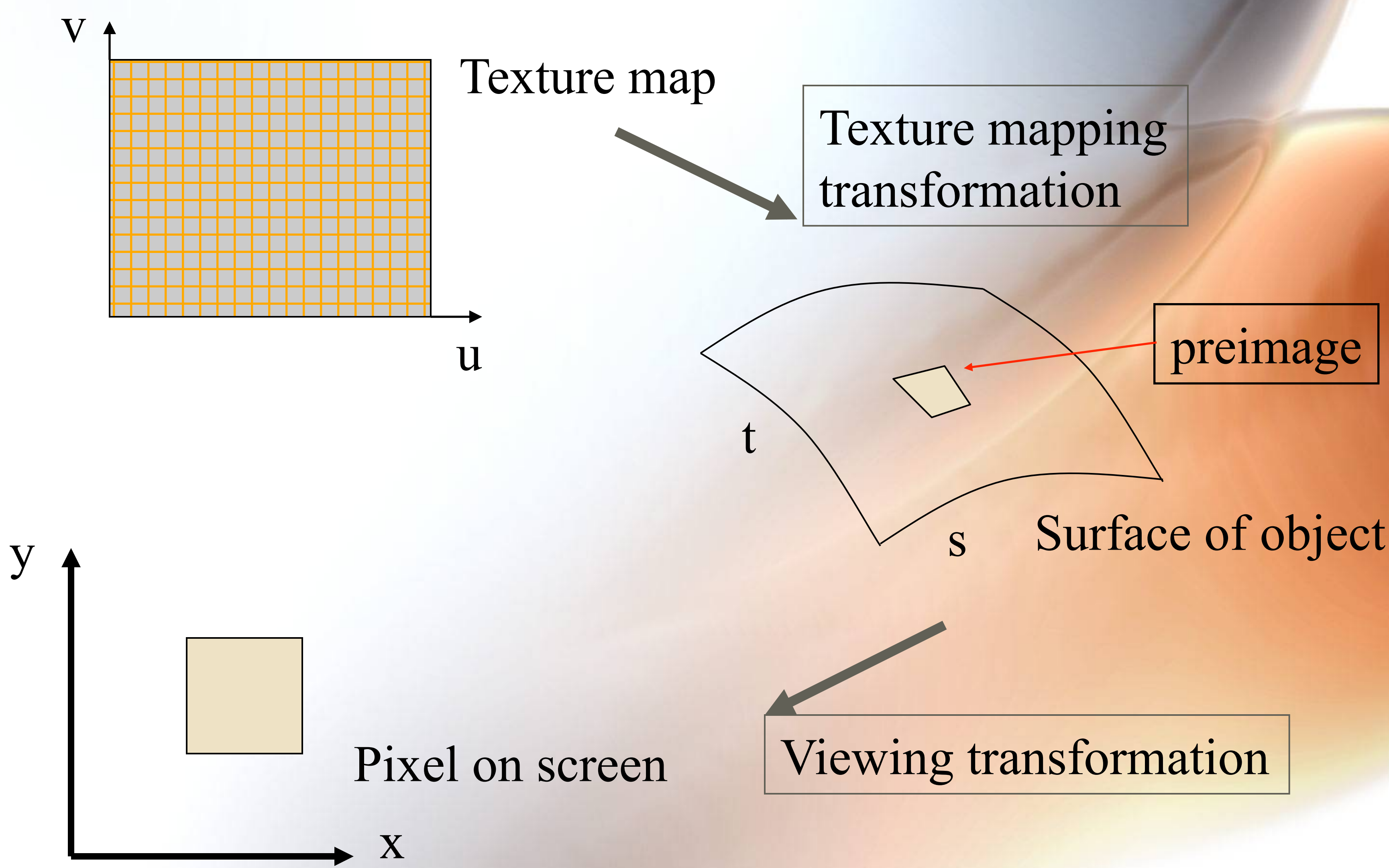


Mapping Functions

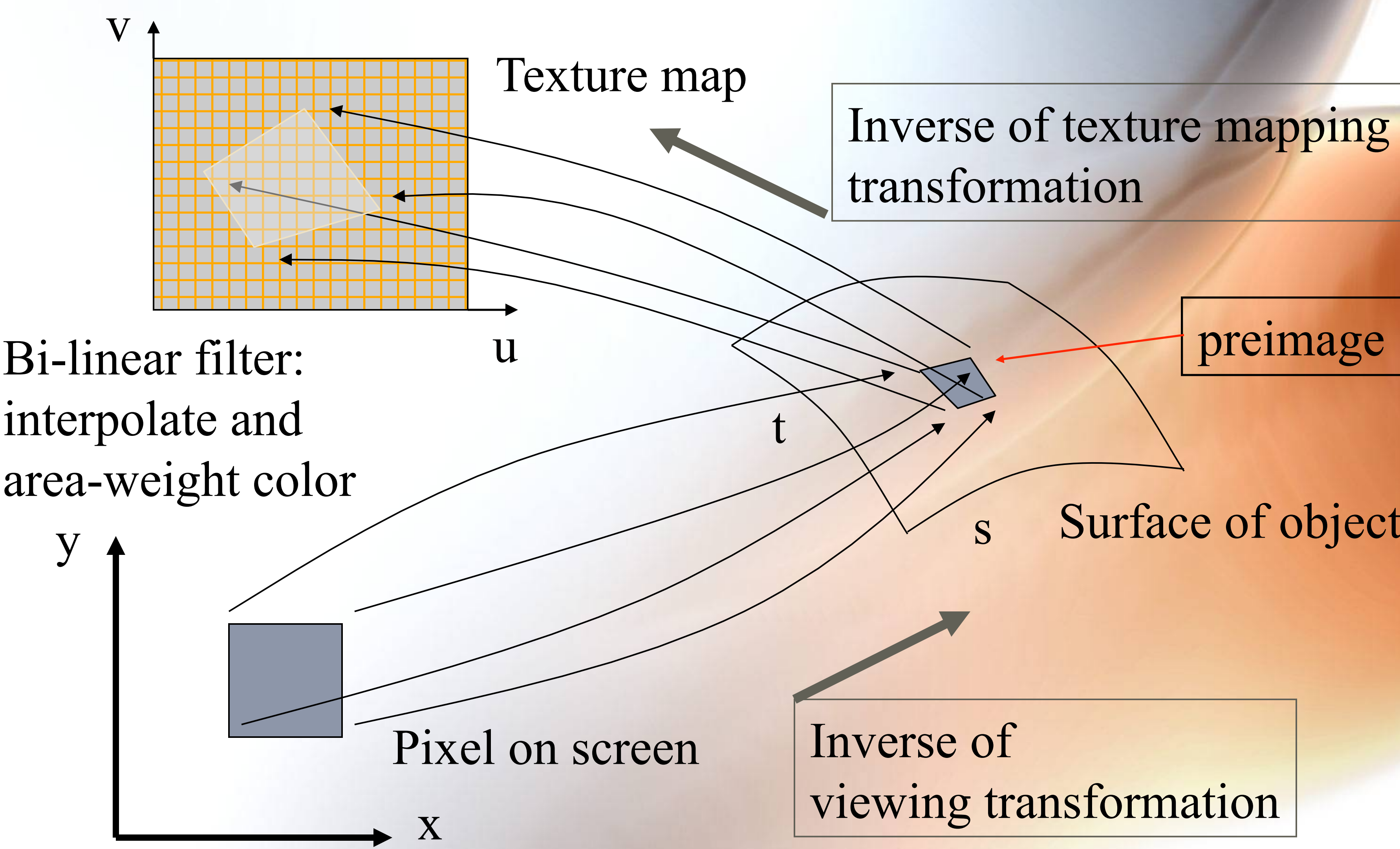
- Basic problem is how to find the maps
- Consider mapping from texture coordinates to a point a surface
- Appear to need three functions
  - x = Fx(s,t)
  - y = Fy(s,t)
  - z = Fz(s,t)
- But we really want to go the other way



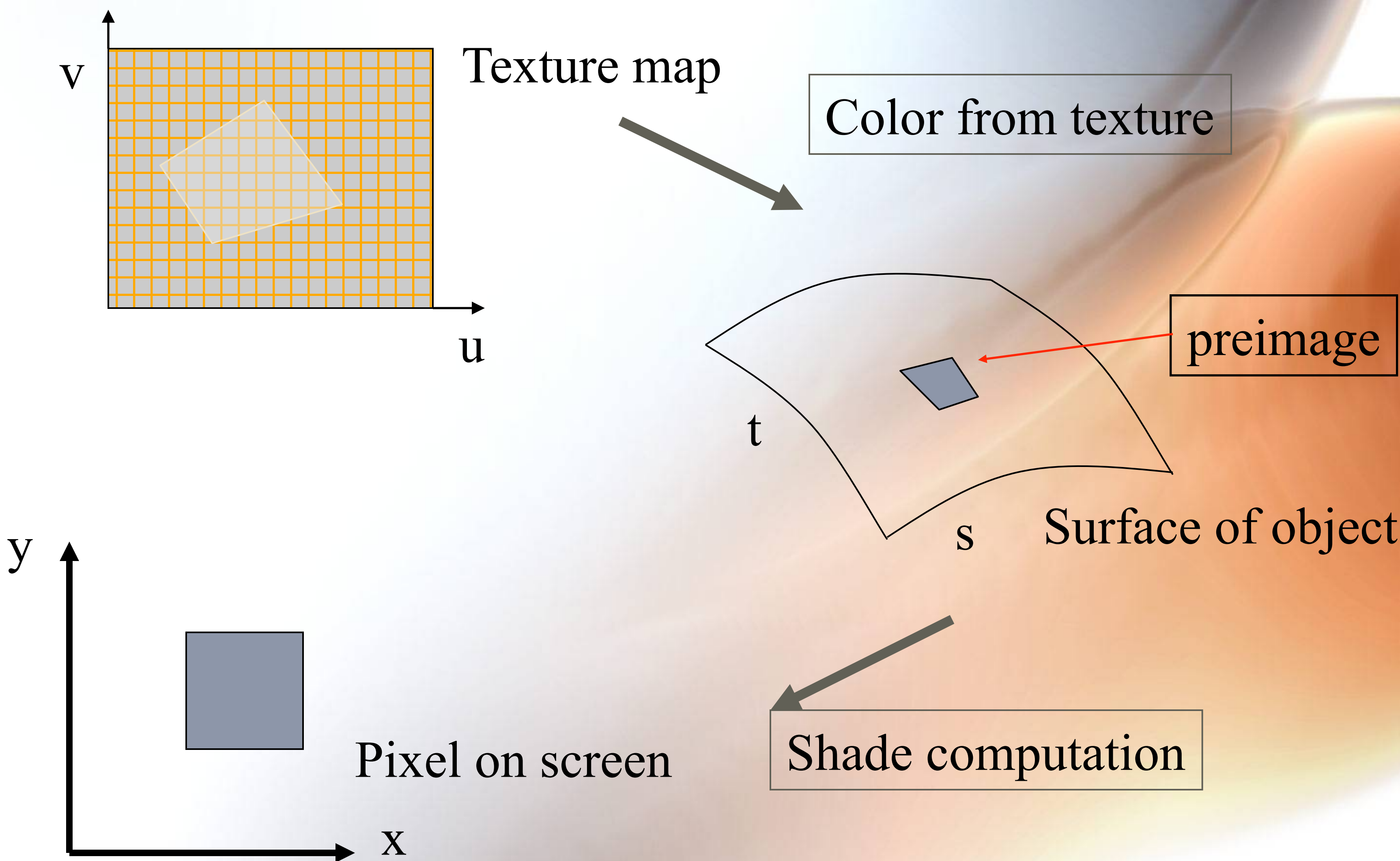
Texture Mapping Concept



Texture Mapping Computation



Mapping Texture Color Back to Screen



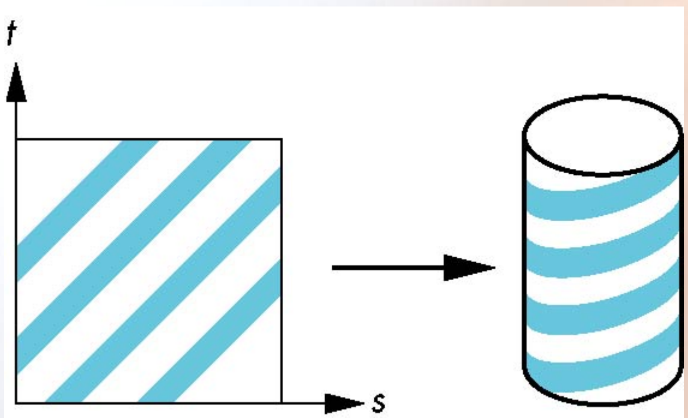
Backward Mapping

- We really want to go from model to texture
  - Given a point on an model, we want to know to which point in the texture it corresponds
- Need a map of the form
  - s = Fs(x,y,z)
  - t = Ft(x,y,z)
- Such functions may be difficult to find in general

Two-part mapping

$$p(u,v) = \begin{bmatrix} fx(u,v) \\ fy(u,v) \\ fz(u,v) \end{bmatrix}$$

- What we need is a parameterization.
- One solution is to first map the texture to a simple intermediate surface, which has a parameterization (u, v).
- Map to cylinder



Cylindrical Mapping

parametric cylinder

$$x = r\cos(2\pi u)$$
$$y = r\sin(2\pi u)$$
$$z = v / h$$

maps rectangle in (**`u,v`**) parameter space to cylinder of radius **`r`** and height **`h`** in world coordinates (**`x, y, z`**)

$$s = u$$
$$t = v$$

maps from texture space to parameter space

Environment Maps

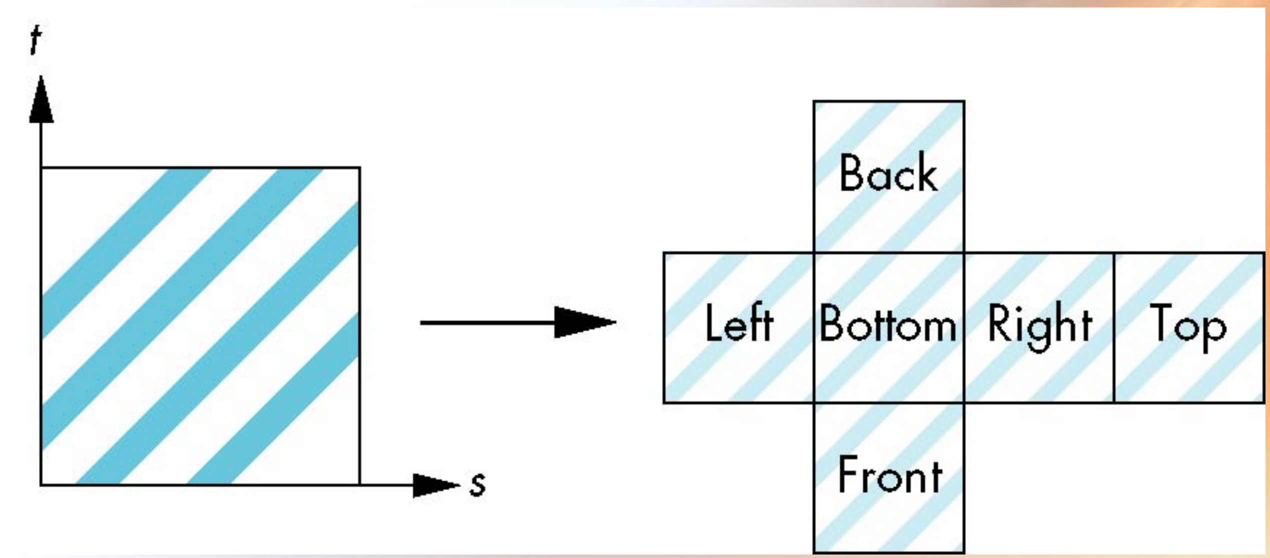
- Used as a cheap alternative to ray tracing shiny objects
- Object must be small w.r.p to the environment
- New map is required whenever the viewpoint changes

Spherical Maps

- Take a picture of the environment with a very wide-angle lens.
- Project the environment picture (map) onto a sphere centered at the center of projection.
- Shrink-wrap the sphere onto the object

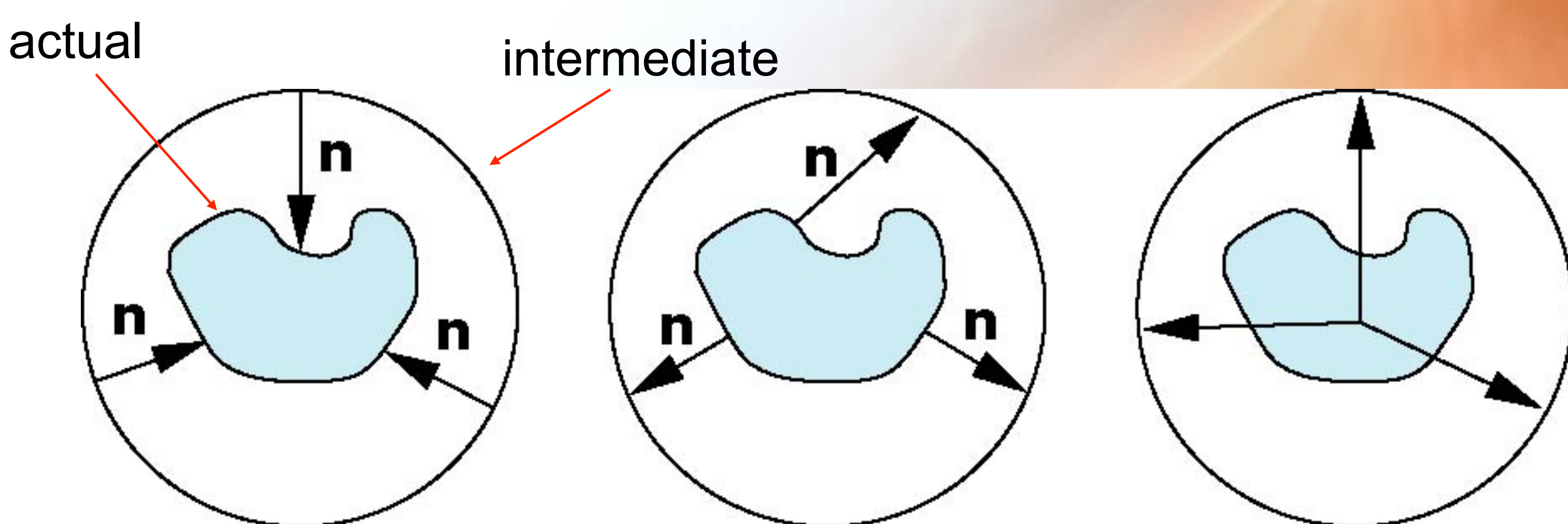
Box Mapping

- **Easy to use with simple orthographic projection**
- **Also used in environment maps**

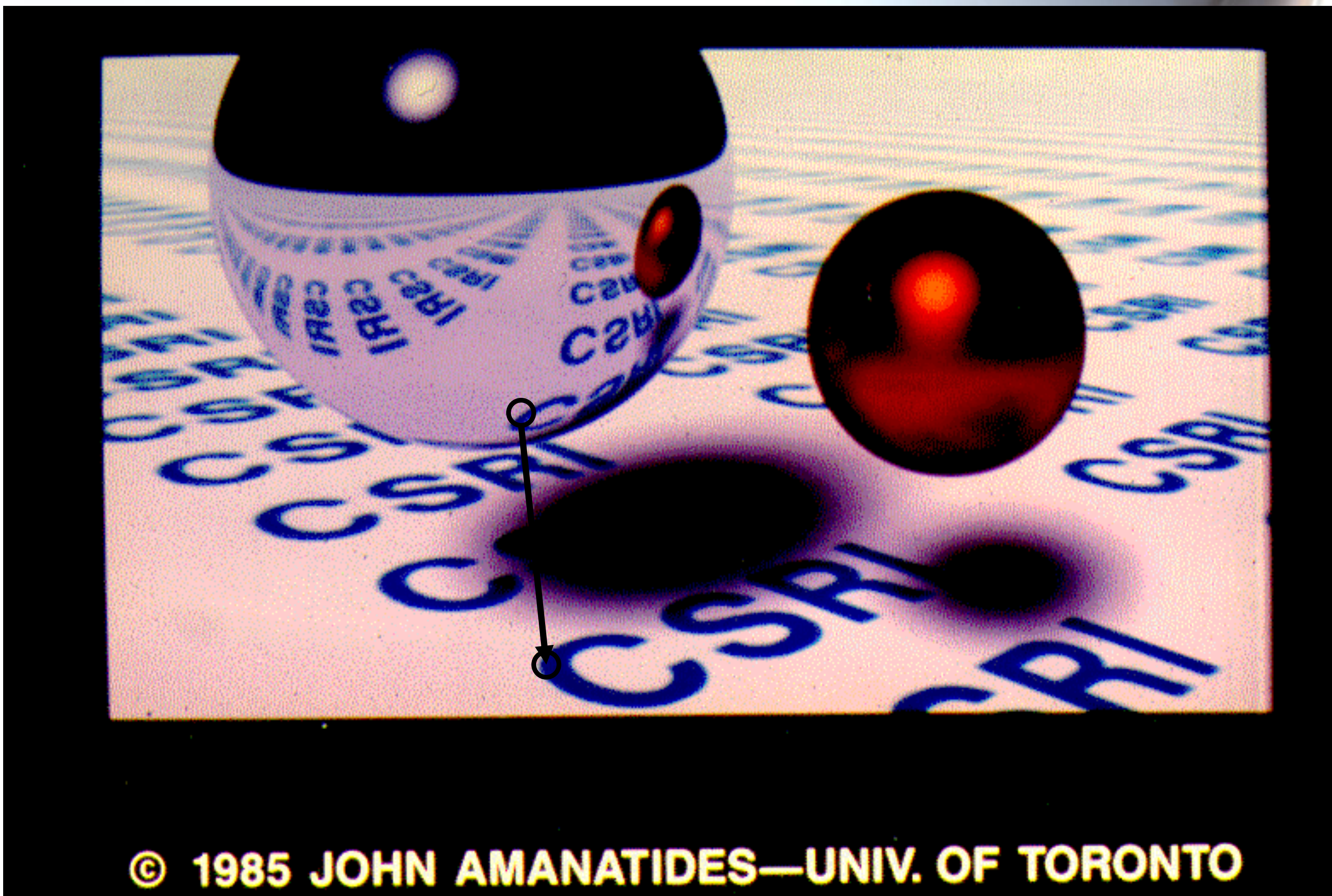


Second Mapping

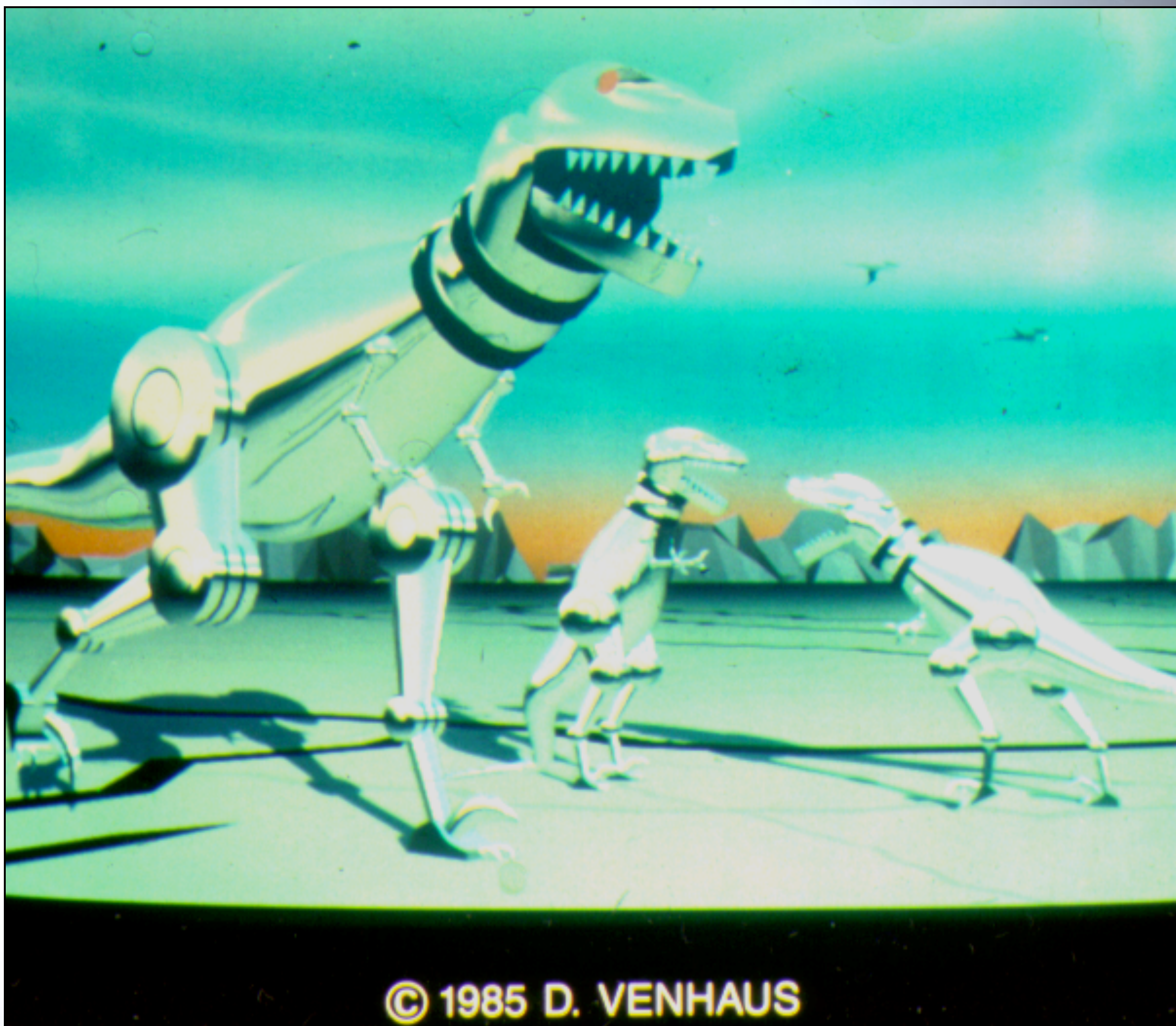
- Map from intermediate object to actual object
  - Normals from intermediate to actual
  - Normals from actual to intermediate
  - Vectors from center of intermediate



Mirror Reflection Example

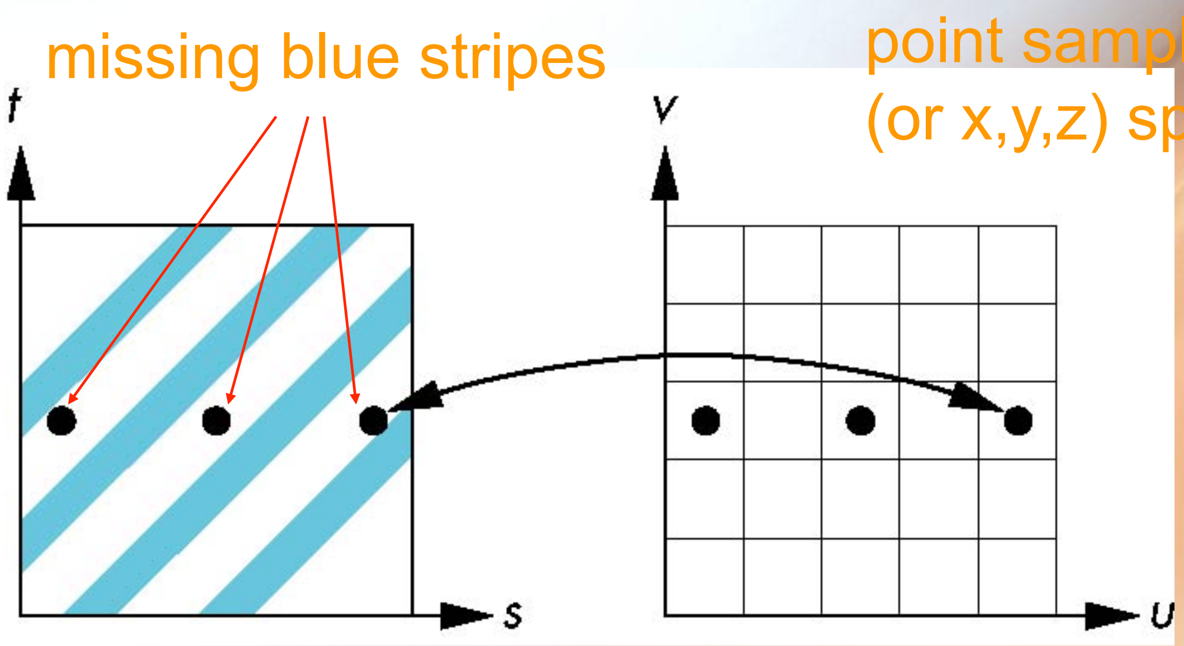


Chromosaurus



Aliasing

- **Point sampling of the texture can lead to aliasing errors**



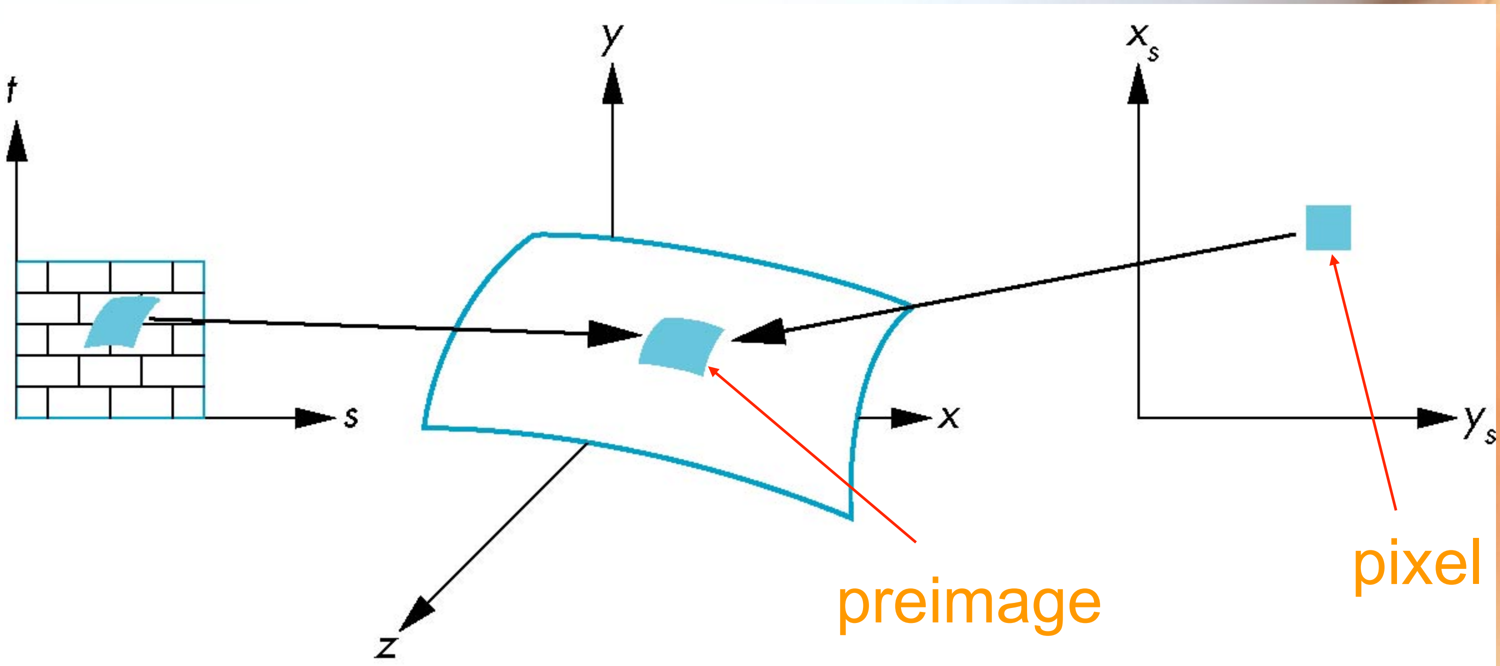
point samples in texture space

Aliasing

- Pixels have finite and discrete sizes.
- Any mapping is ultimately a discrete sampling of the texture, which can have the unfortunate tendency to miss the important parts.
- Most visible on periodic/repeated patterns.

Area Averaging

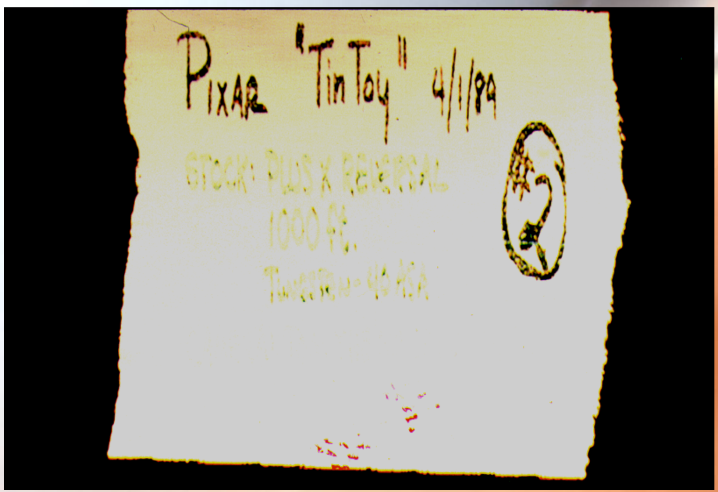
A better but slower option is to use *preimage area averaging*

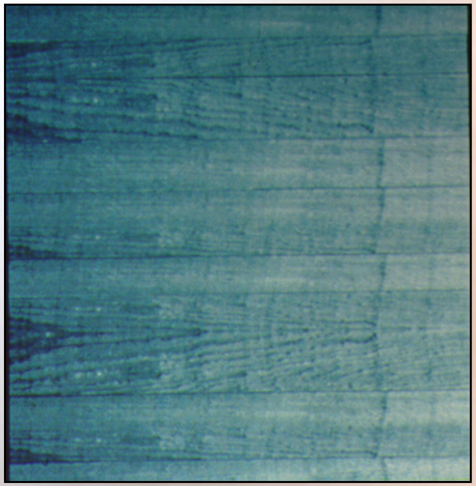


Note that *preimage* of pixel is curved

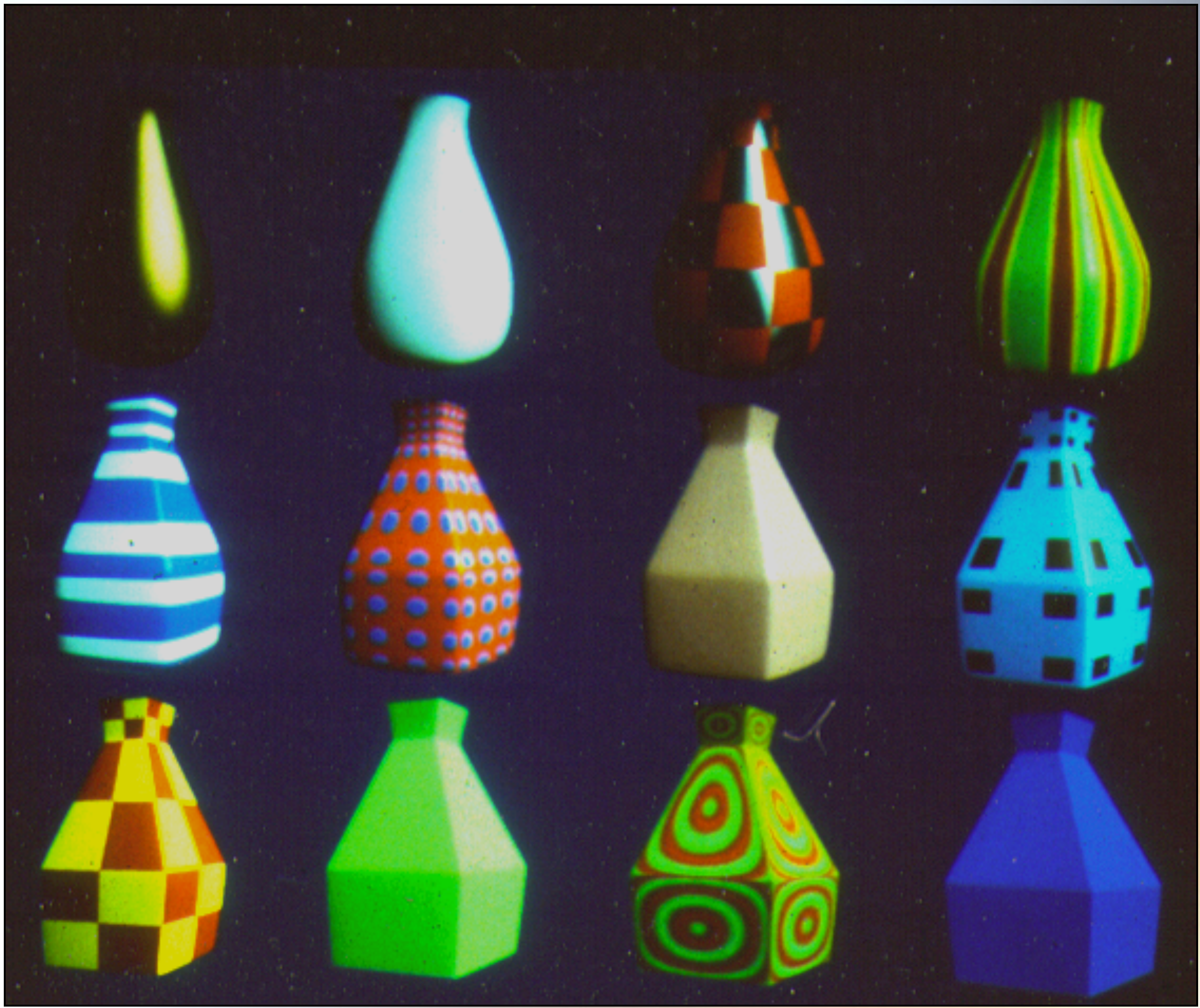
Textured Map Scene

Some of the Textures Used

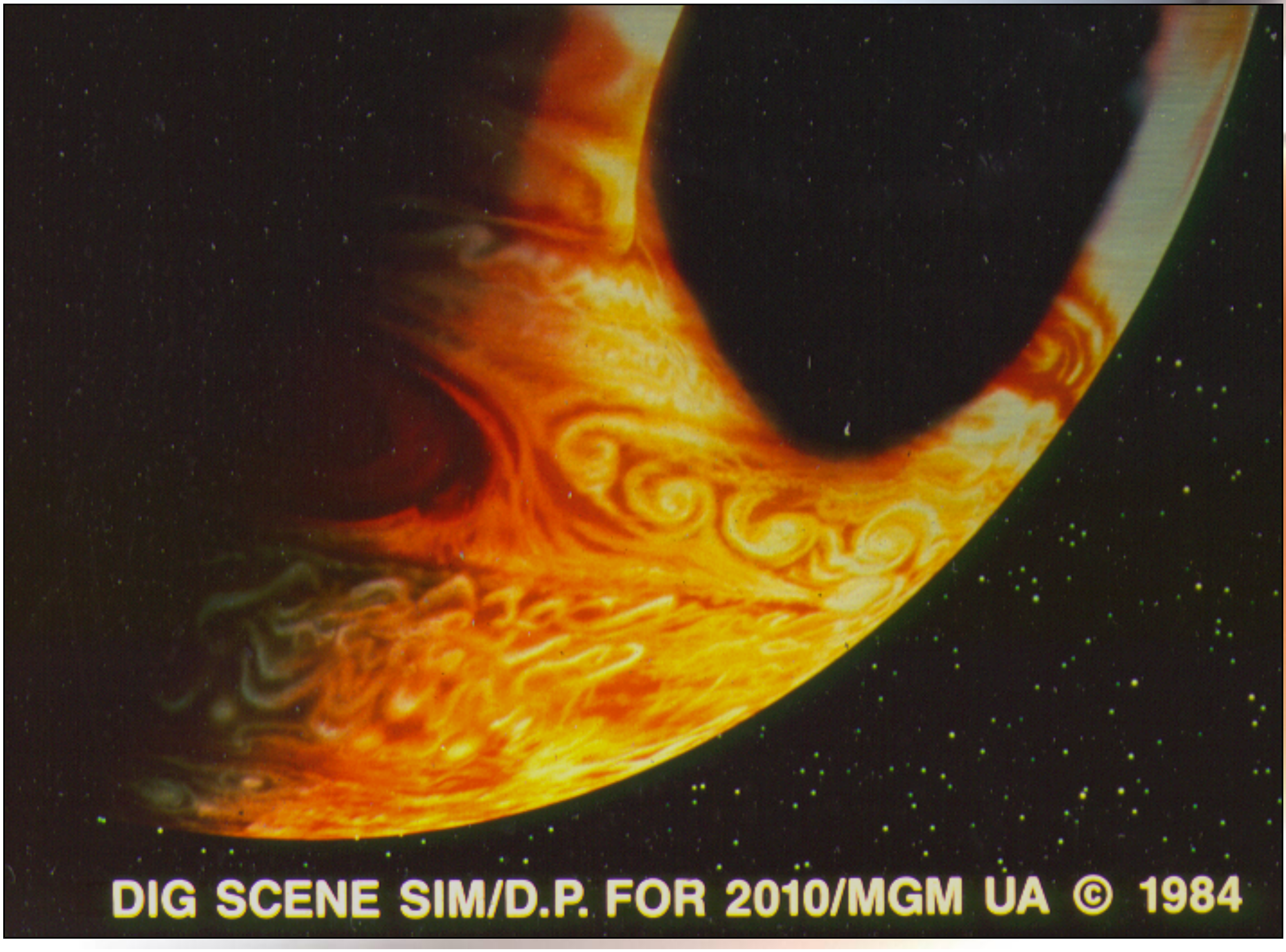




Mapping to Curved Surfaces



Using an Animated Texture Map



Texture Billboards (Real-Time Interaction)

- **Use the camera position as a target for the normal vector of a polygon to be textured. Polygon thus always faces the camera.**
- **If one has different textures for different camera views, one can create the appearance of view-dependent 3D appearance on a billboard polygon (e.g., game character sprites).**

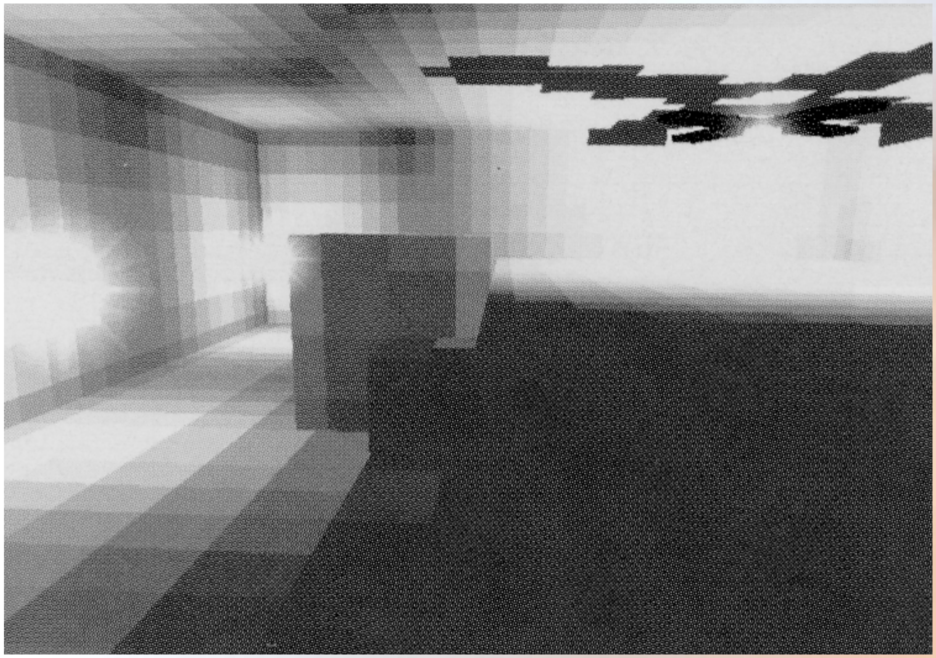
Example

Using a Map to Vary the Reflectance Function

Light Maps

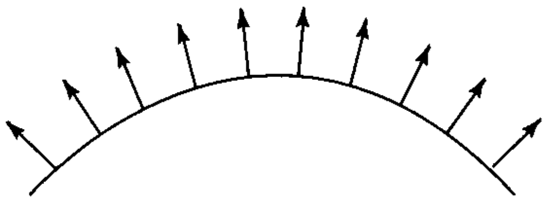
- An efficient technique for static objects and lighting.
- Pre-calculate light intensity and color across polygon surface.
- Linear filter (e.g. Gouraud) for pixel shade at run time.
- Add other dynamic lighting components at run time.

Light Maps

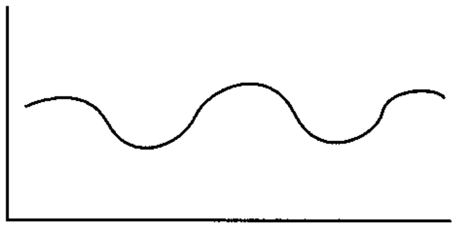


Policarpo and Watt

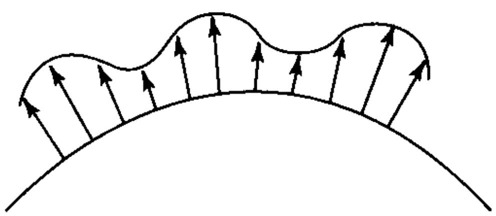
Bump Mapping (2D Analogy)



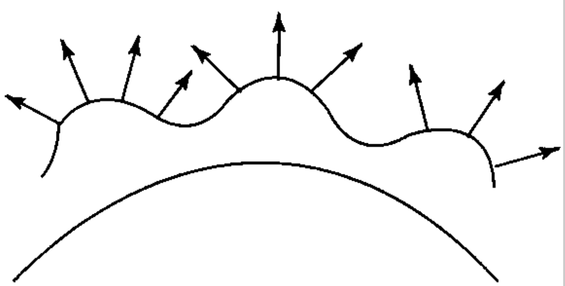
P(u)
Original surface



B(u)
A bump map – 2D height field



P'(u)
Lengthening or shortening P(u)
Using B(u)



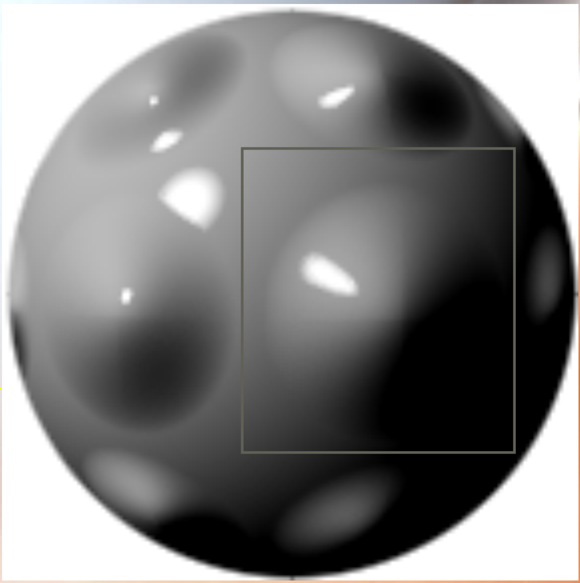
N'(u)
Perturb normals by partial derivatives of *B(u)*. Obtain the vectors to the "new" surface

Bump Mapping (2D Map to 3D Effect)

0	0	0	0	0	0	0	0
0	0	0	1	1	0	0	0
0	0	2	5	5	2	0	0
0	1	5	9	9	5	1	0
0	1	5	9	9	5	1	0
0	0	2	5	5	2	0	0
0	0	0	1	1	0	0	0
0	0	0	0	0	0	0	0

Bump map

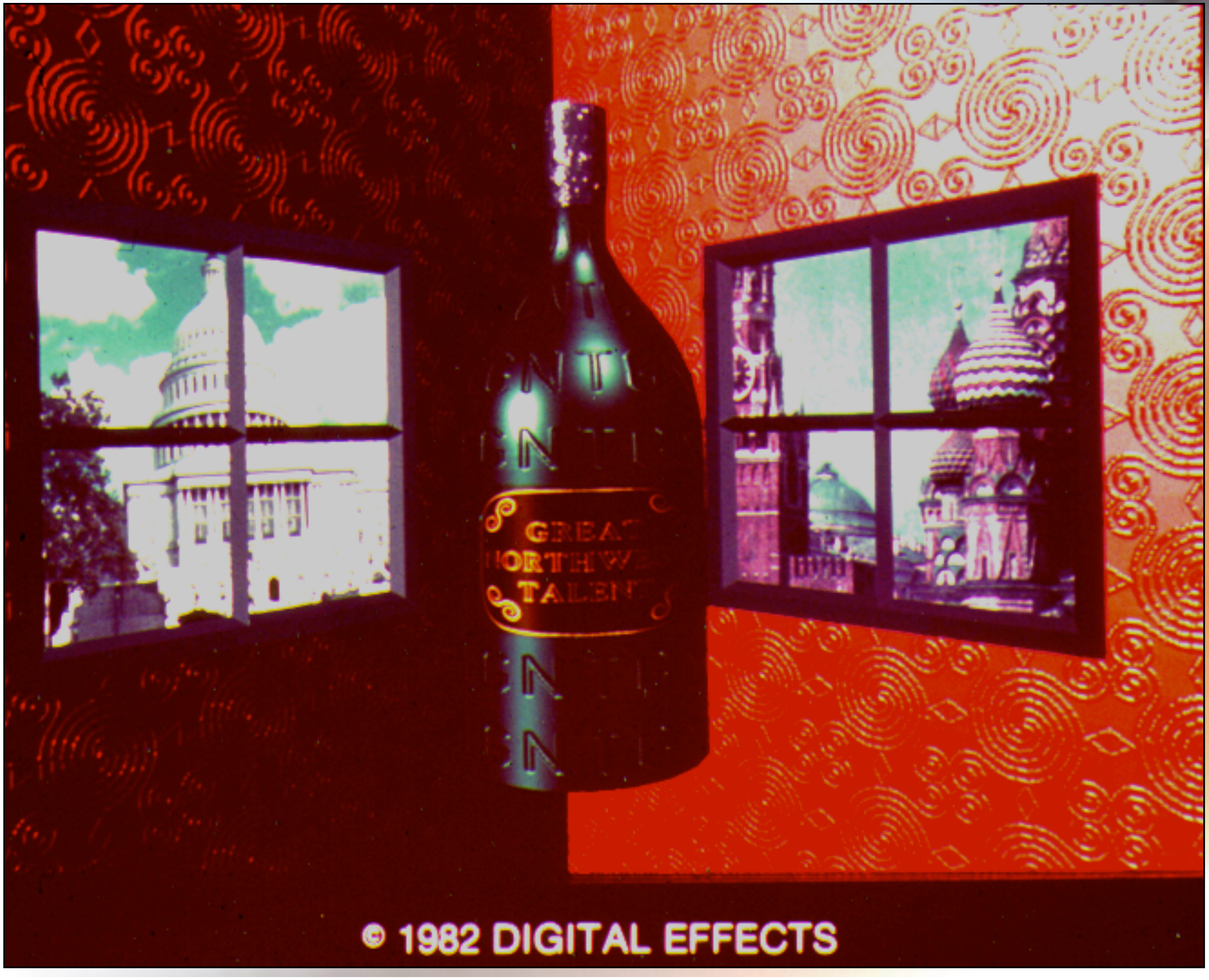
applied like texture map, but...



Demo

Bump map perturbs the local normal vector by the partial derivatives of the map values, giving the illusion of curvature.

Bump Mapping Adds Visual Complexity Cheaply

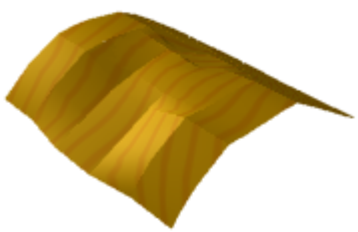


Certain Textures Models Work Well with Bump Maps

Displacement Mapping

- Start with parameterized object surface S (u,v).
- Displacement map: a 2D height field or function D(u,v).
- Apply corresponding height (displacement) S(u,v)+D(u,v).

Example



Original surface
S(u,v)

0	0	0	0	0	0	0	0
0	0	0	1	1	0	0	0
0	0	2	5	5	2	0	0
0	1	5	9	9	5	1	0
0	1	5	9	9	5	1	0
0	0	2	5	5	2	0	0
0	0	0	1	1	0	0	0
0	0	0	0	0	0	0	0

Bump map D(u,v)

New surface with
actual displacements
S(u,v)+D(u,v)

Scene with Displacement Map

Reflection Map

Scene with Displacement and Reflection Maps