



The University of New Mexico

Texture Mapping

Ed Angel

Professor of Computer Science,
Electrical and Computer
Engineering, and Media Arts
University of New Mexico



Objectives

- Introduce Mapping Methods
 - Texture Mapping
 - Environment Mapping
 - Bump Mapping
- Consider basic strategies
 - Forward vs backward mapping
 - Point sampling vs area averaging



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

- Consider the problem of modeling an orange (the fruit)
- 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 (2)

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



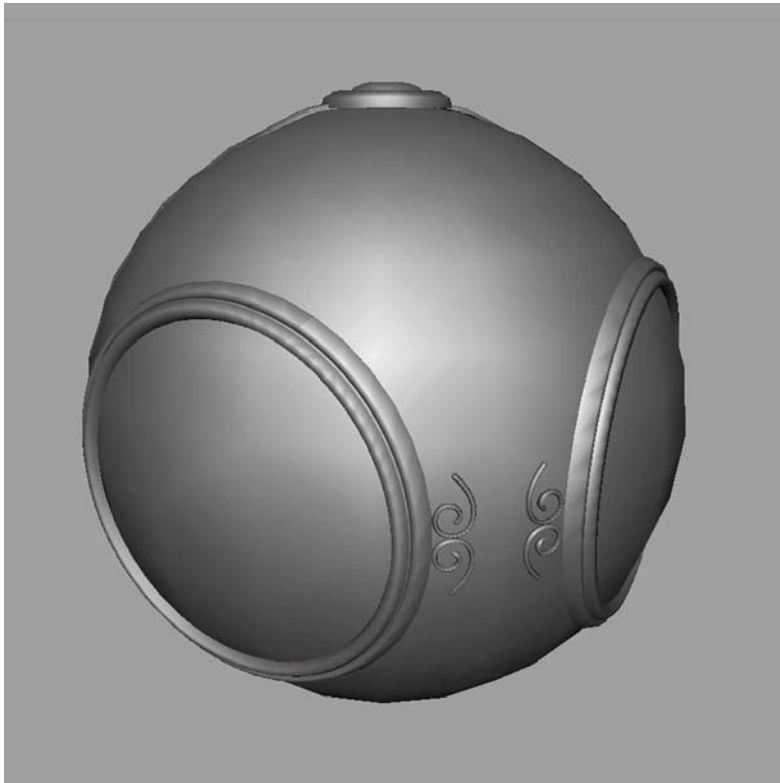
Three Types of Mapping

- Texture Mapping
 - Uses images to fill inside of polygons
- Environment (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



The University of New Mexico

Texture Mapping



geometric model



texture mapped



The University of New Mexico

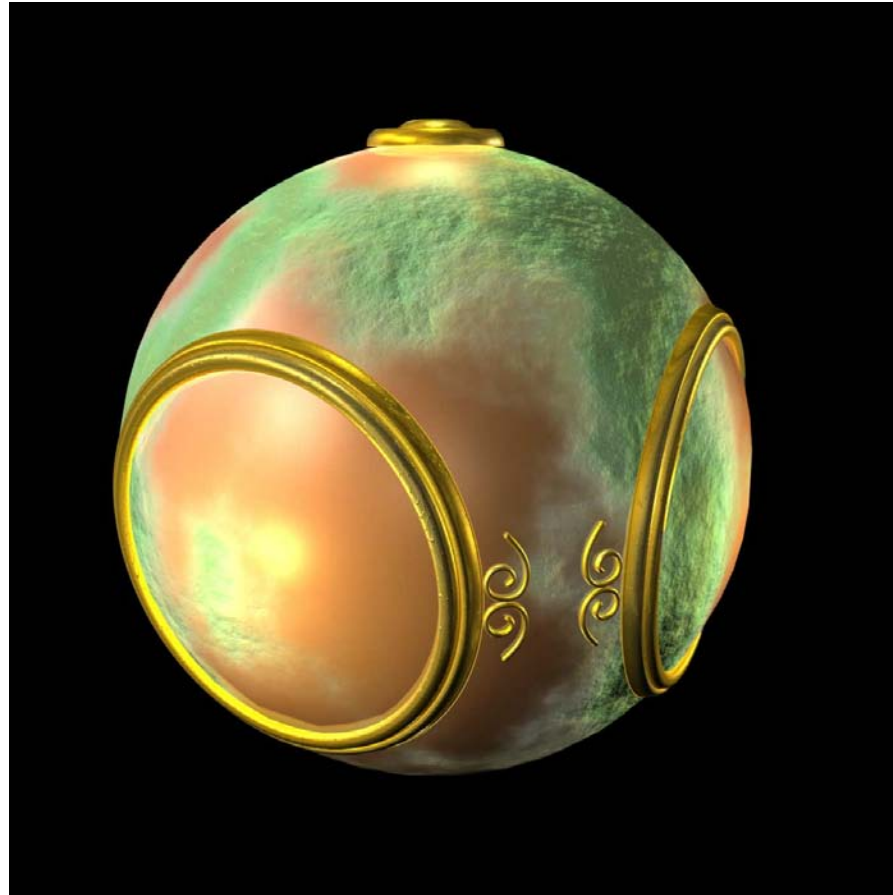
Environment Mapping





The University of New Mexico

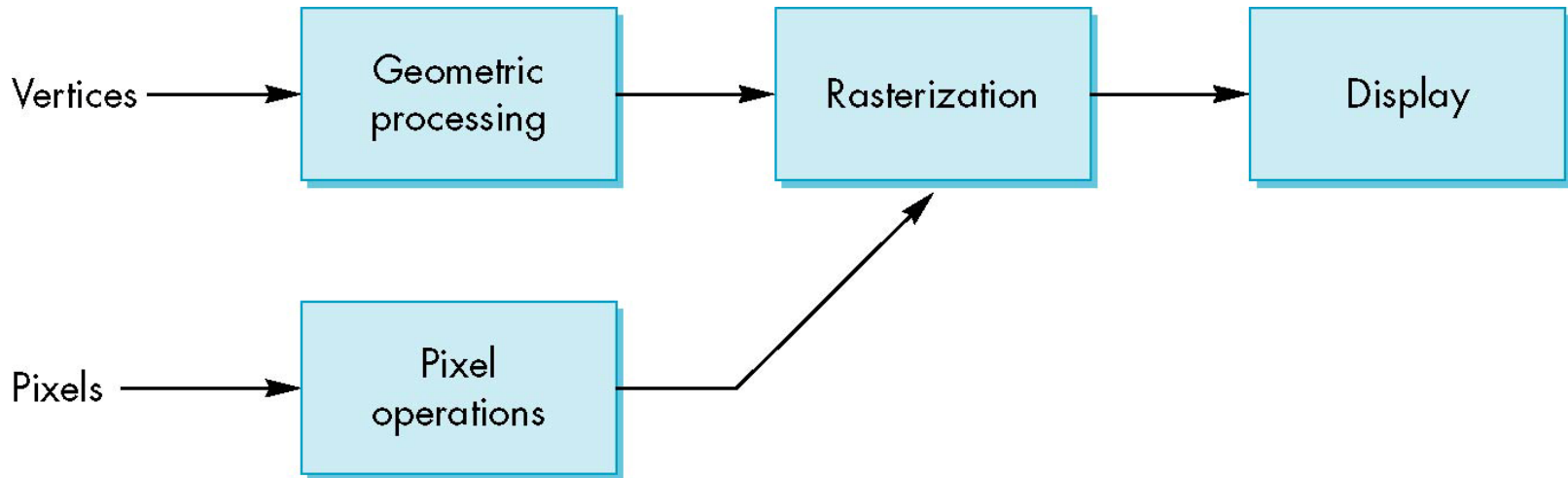
Bump Mapping





Where does mapping take place?

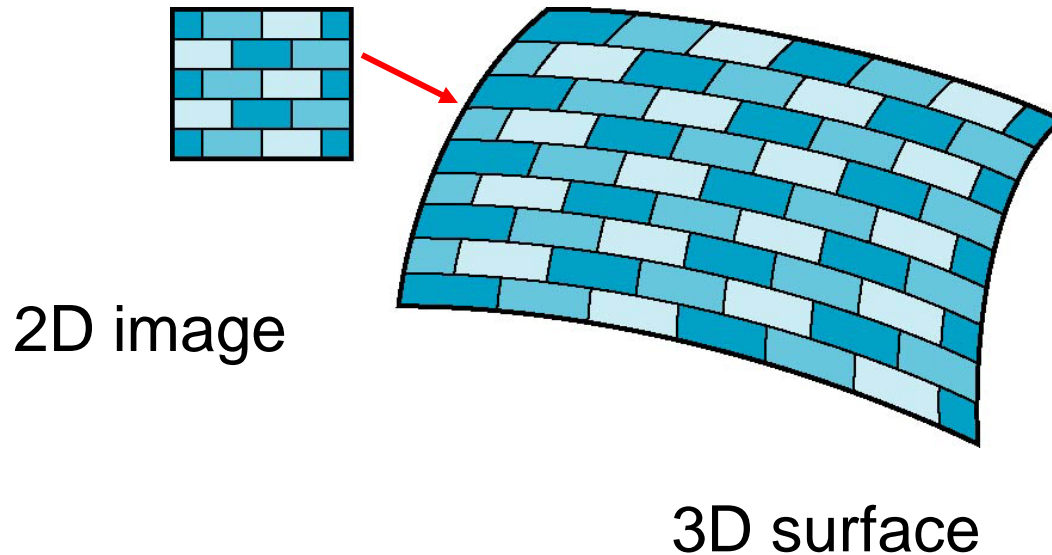
- Mapping techniques are implemented at the end of the rendering pipeline
 - Very efficient because few polygons make it past the clipper





Is it simple?

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





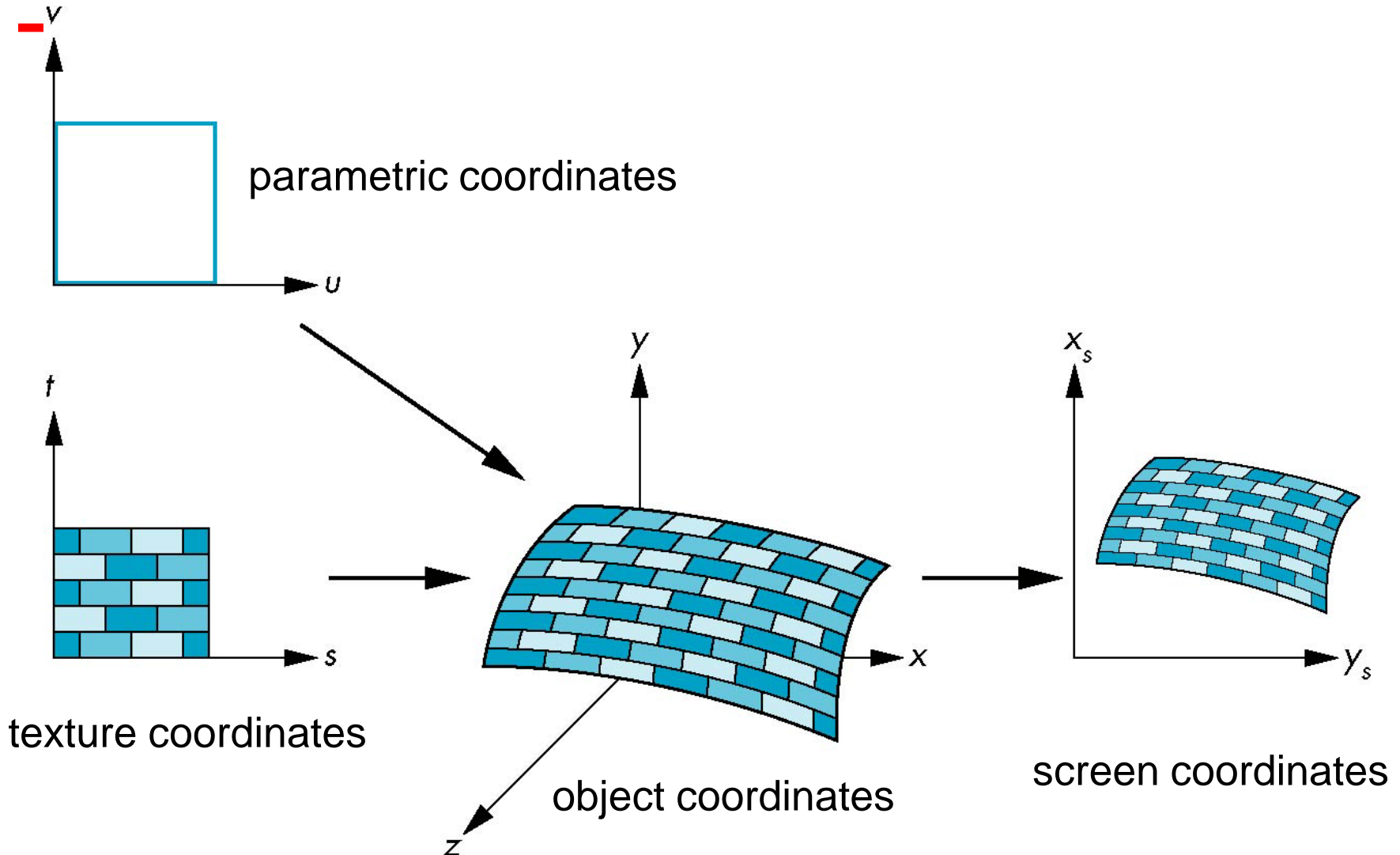
Coordinate Systems

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



The University of New Mexico

Texture Mapping





Mapping Functions

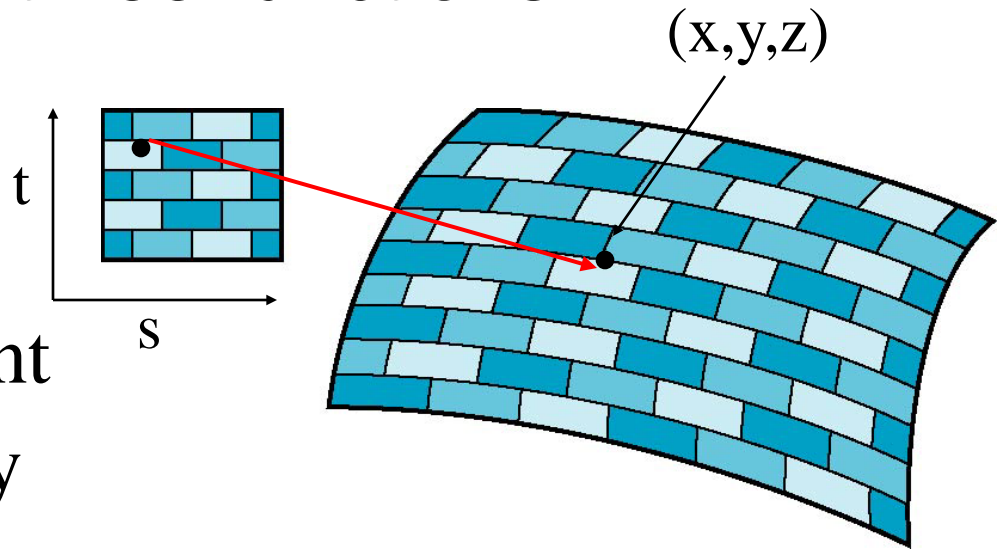
- Basic problem is how to find the maps
- Consider mapping from texture coordinates to a point a surface
- Appear to need three functions

$$x = x(s,t)$$

$$y = y(s,t)$$

$$z = z(s,t)$$

- But we really want to go the other way





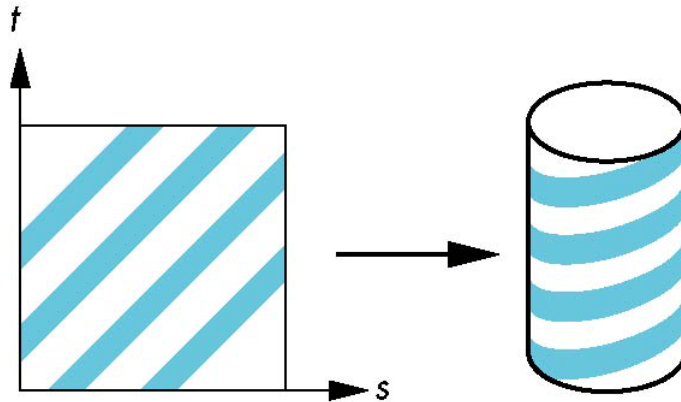
Backward Mapping

- We really want to go backwards
 - Given a pixel, we want to know to which point on an object it corresponds
 - Given a point on an object, we want to know to which point in the texture it corresponds
- Need a map of the form
$$s = s(x,y,z)$$
$$t = t(x,y,z)$$
- Such functions are difficult to find in general



Two-part mapping

- One solution to the mapping problem is to first map the texture to a simple intermediate surface
- Example: 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 space to cylinder
of radius r and height h in object coordinates

$$s = u$$

$$t = v$$

maps from texture space



Spherical Map

We can use a parametric sphere

$$x = r \cos 2\pi u$$

$$y = r \sin 2\pi u \cos 2\pi v$$

$$z = r \sin 2\pi u \sin 2\pi v$$

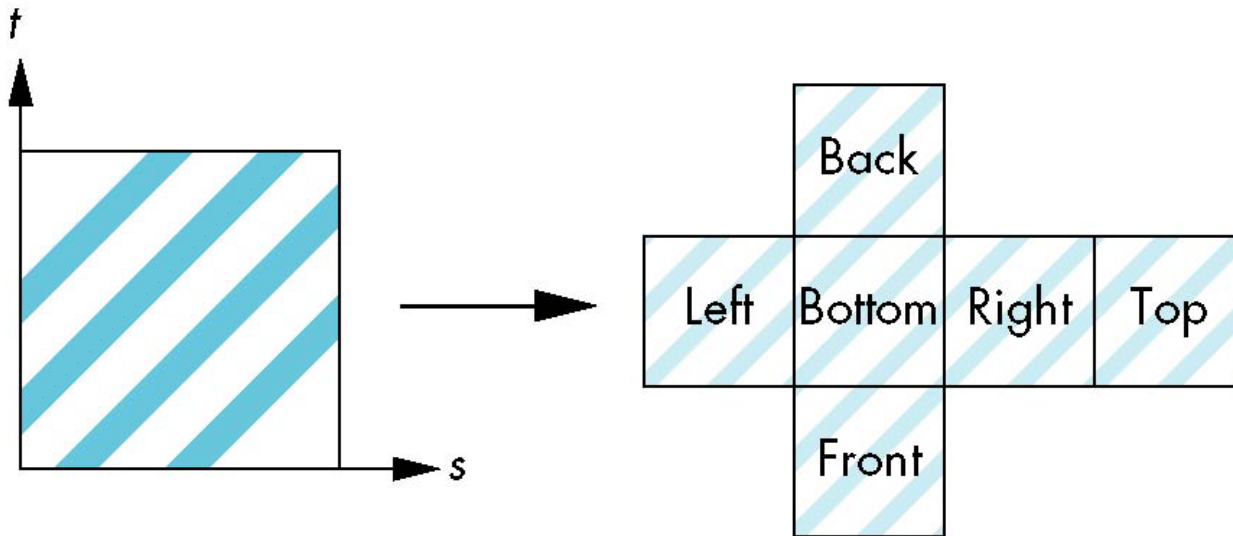
in a similar manner to the cylinder
but have to decide where to put
the distortion

Spheres are used in environment maps



Box Mapping

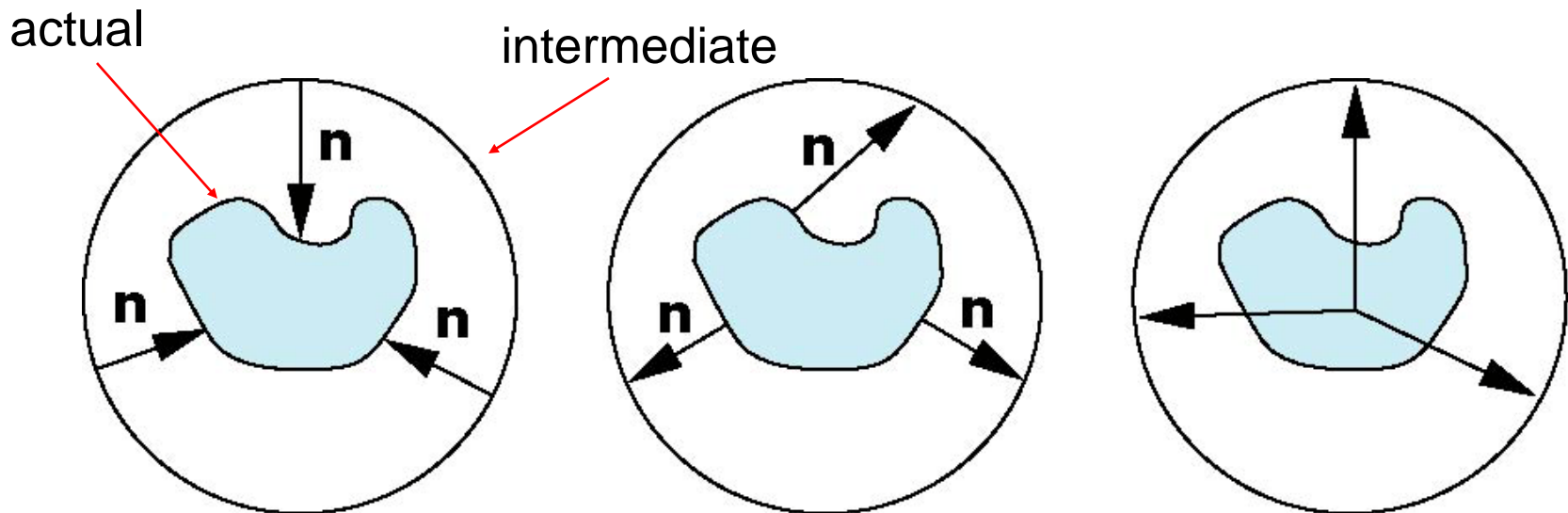
- Easy to use with simple orthographic projection
- Also used in environment maps





Second Mapping

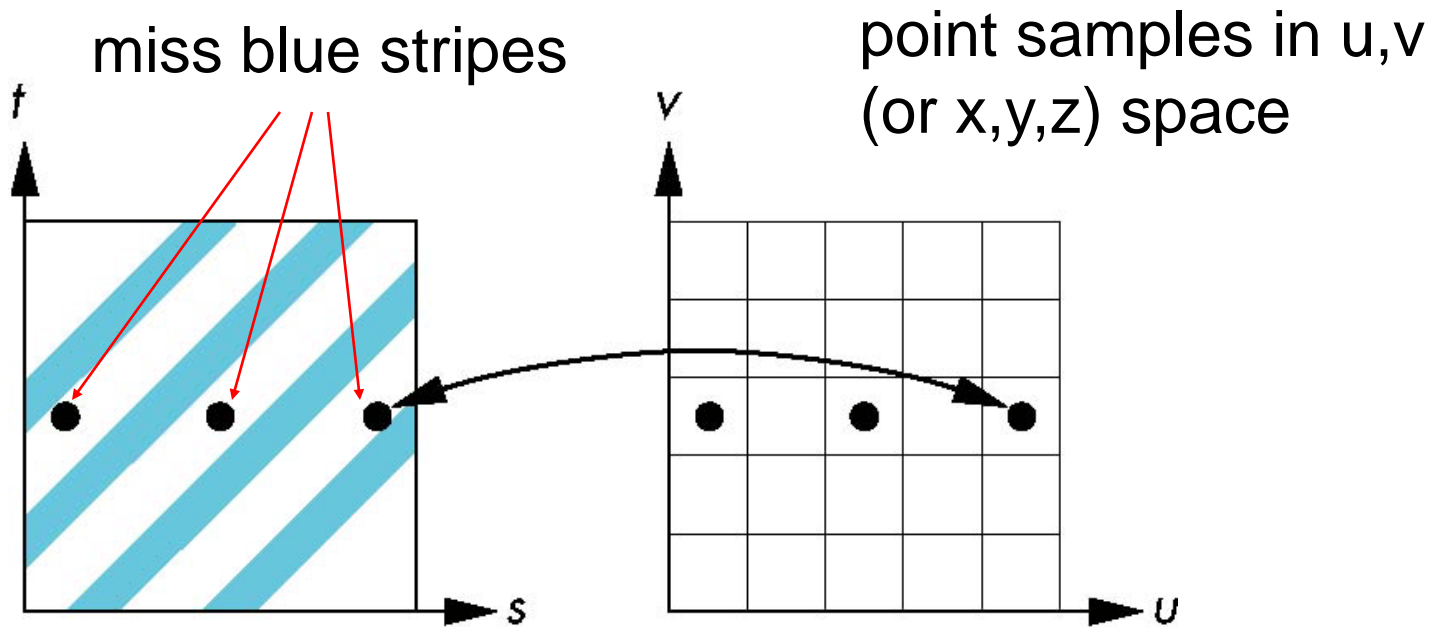
- Map from intermediate object to actual object
 - Normals from intermediate to actual
 - Normals from actual to intermediate
 - Vectors from center of intermediate





Aliasing

- Point sampling of the texture can lead to aliasing errors

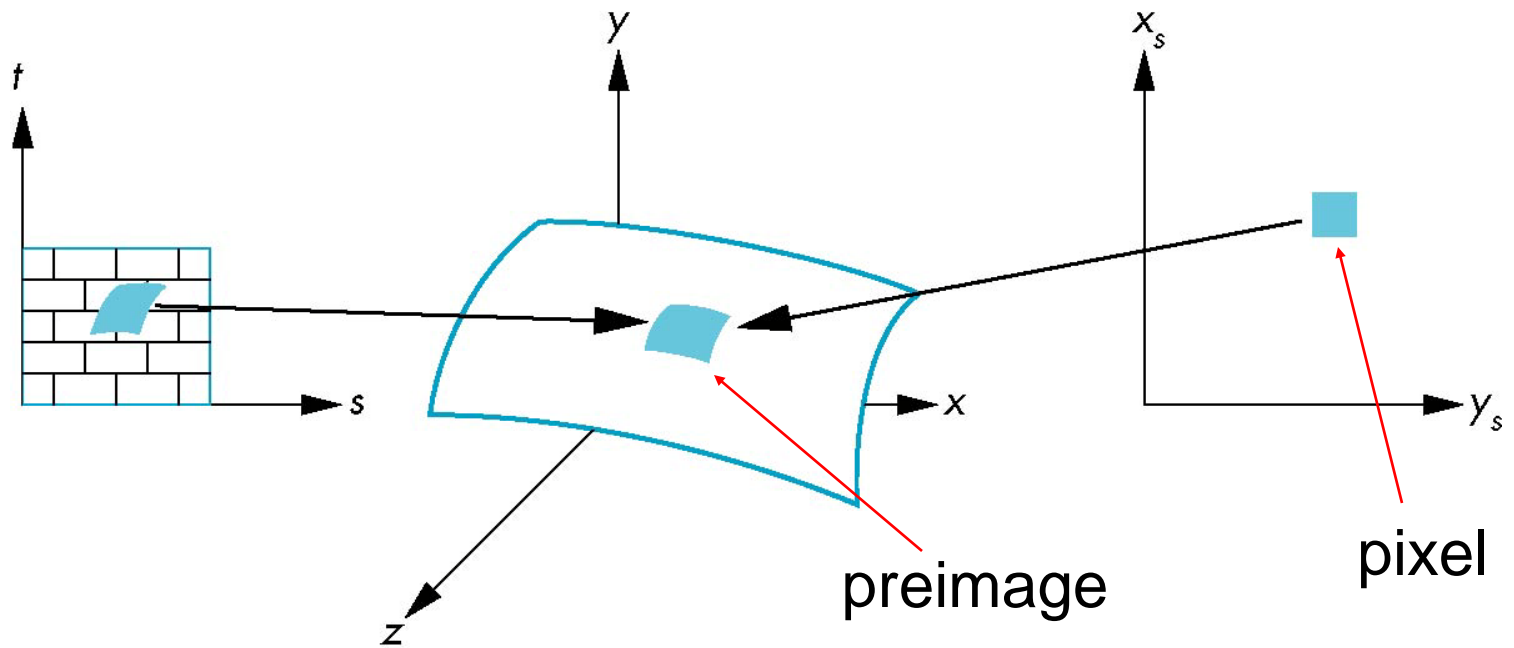


point samples in texture space



Area Averaging

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



Note that *preimage* of pixel is curved