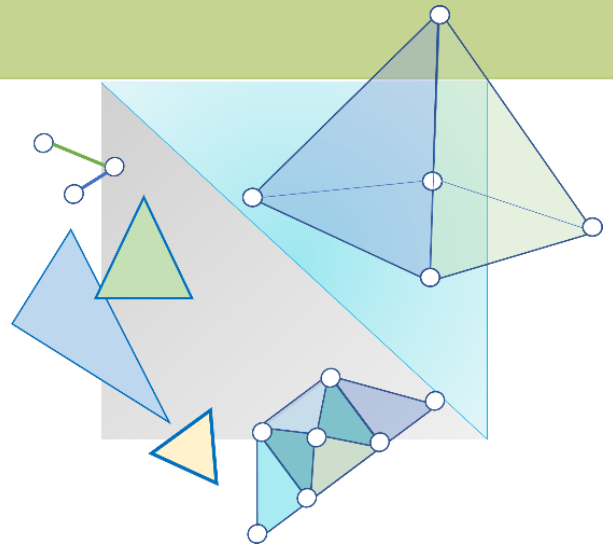
CITS3003 Graphics & Animation

Lecture 18:
Texture Mapping



Objectives

- Introduce Mapping Methods
  - Texture Mapping
  - Environment Mapping
  - Bump Mapping
- Consider basic strategies
  - Forward vs backward mapping
  - Two-part mapping

The Limits of Geometric Modeling

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

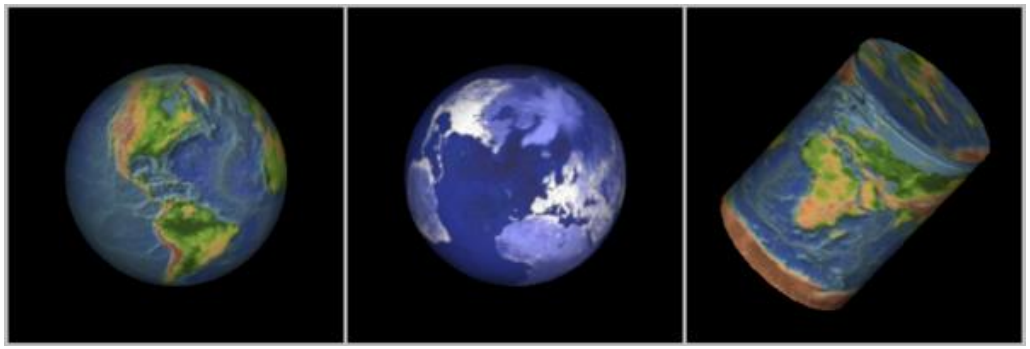
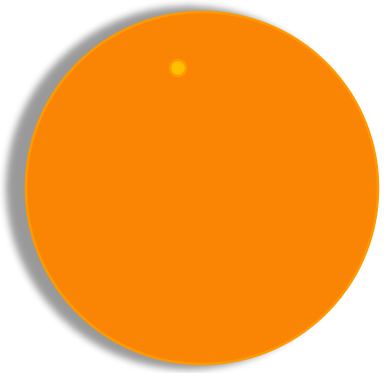


Image source

Modelling an Orange

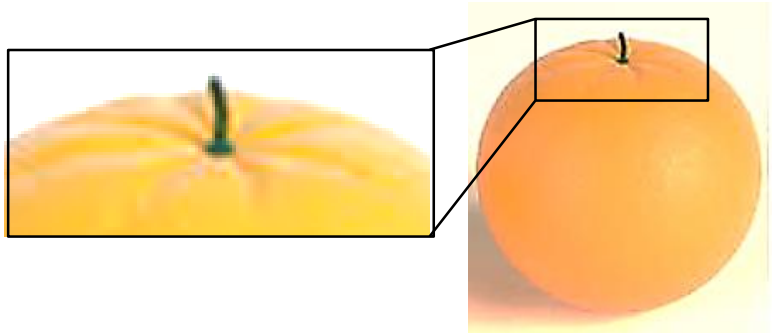
- Consider the problem of modelling an orange (the fruit)

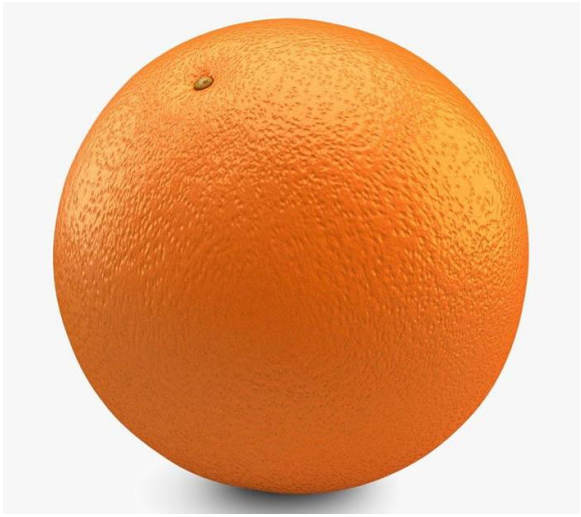
- We can model it as an orange-coloured sphere
  - Unfortunately, the rendering would be too regular to look like an orange.



Modelling an Orange

- Alternatively, we can replace the sphere with a more complex shape
- Unfortunately, the rendering still does not capture some fine surface characteristics (e.g., small dimples on the orange);
- Plus, increasing the number of polygons to capture the surface details would overwhelm the pipeline.





Modelling an Orange (cont.)

- Another alternative is to take a picture of a real orange, and “paste” the image of the orange skin onto a sphere
  - This process is known as **texture mapping**
  - Computationally inexpensive way to add details

- However, this still might not be sufficient because the resulting surface will be smooth. To make the surface look a bit rough, we
  - need to change local shape, i.e., we need **bump mapping**

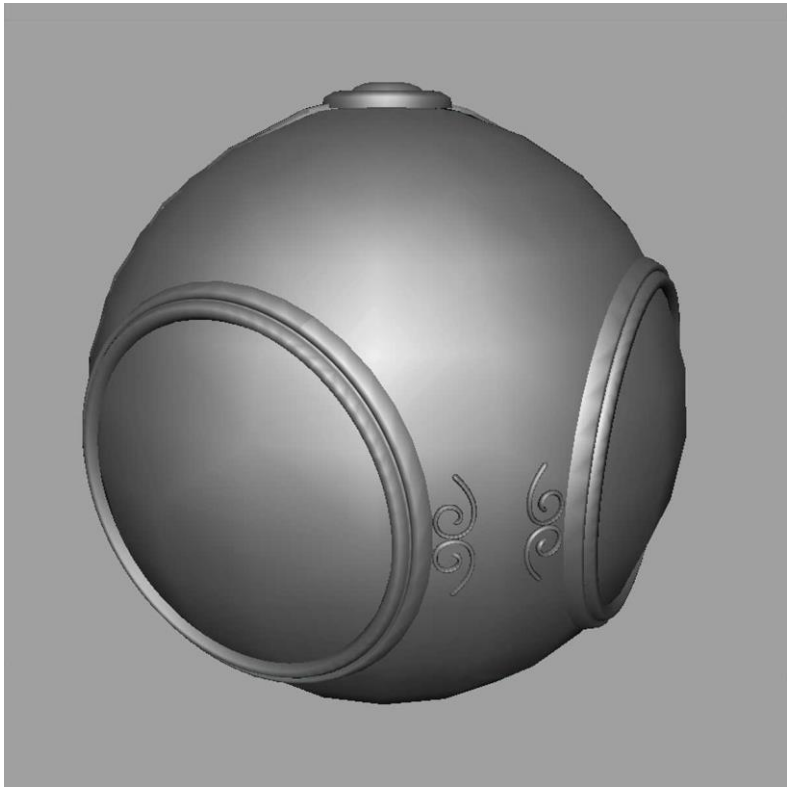
Three Types of Mapping Techniques

- **Texture Mapping**
  - Uses an image (or *texture map*) to influence the colour of a fragment; i.e., we paint patterns onto smooth surfaces.

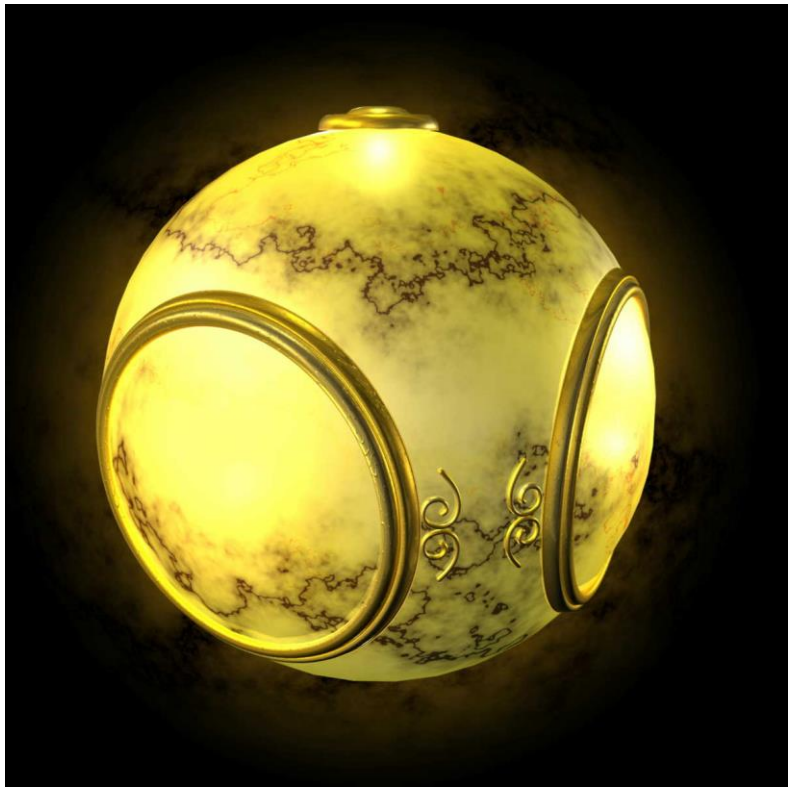
- **Environment Mapping (reflection mapping)**
  - Uses a picture of the environment as the texture map
  - This allows us to simulate highly specular surfaces

- **Bump mapping**
  - Adds small distortions to the surface normals before mapping the texture during the rendering process.
  - This allows us to simulate small variations in shape, such as the bumps on a real orange.

Texture Mapping



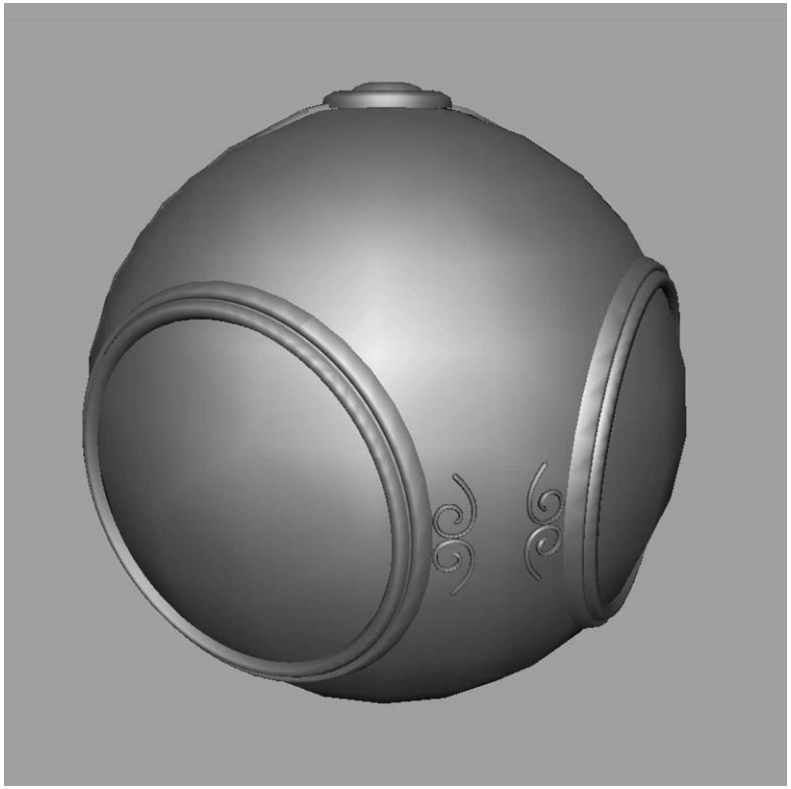
geometric model



texture mapping output

Paste image (marble) onto polygon

Environment Mapping

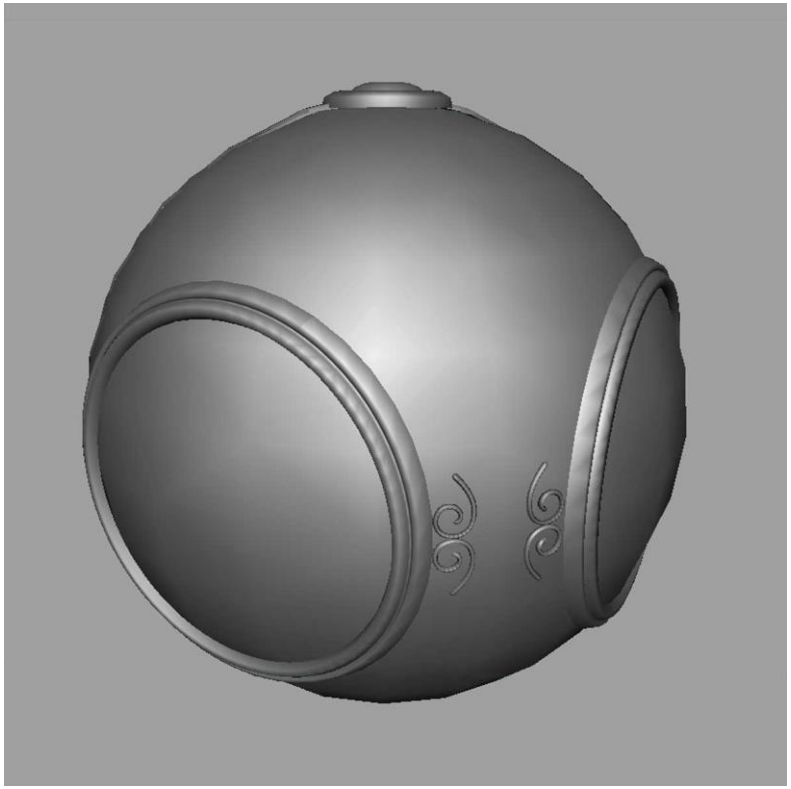


geometric model

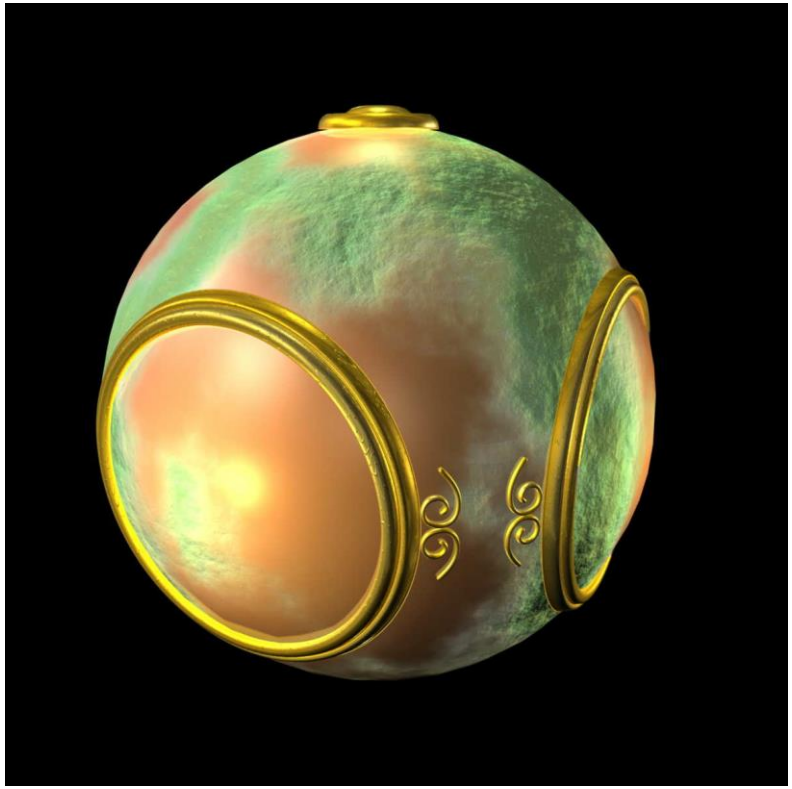
Environment mapping output

Picture of environment over object

Bump Mapping



geometric model

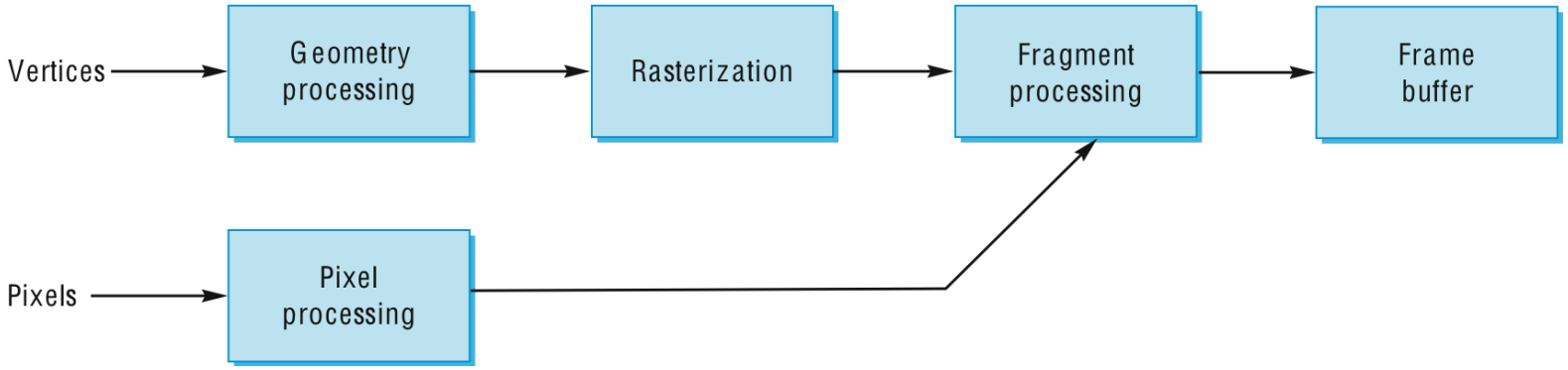


Bump mapping output

Simulate surface roughness

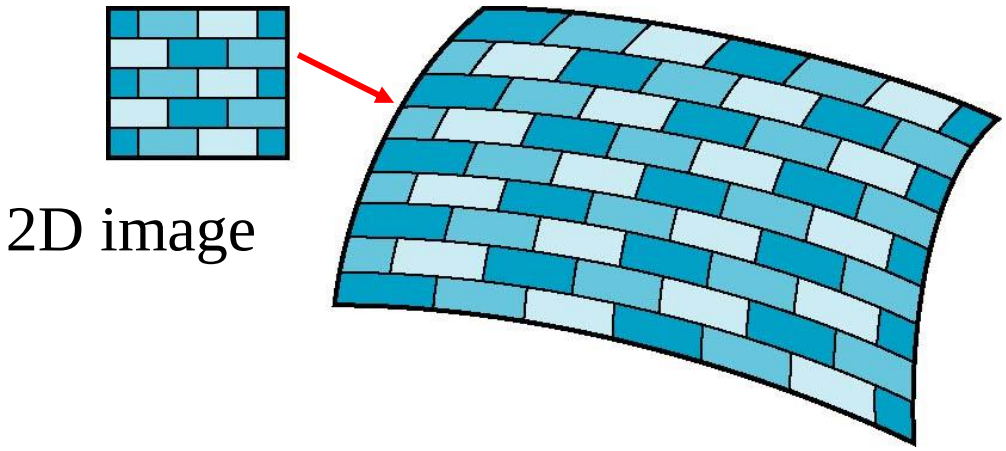
Where does texture mapping take place?

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



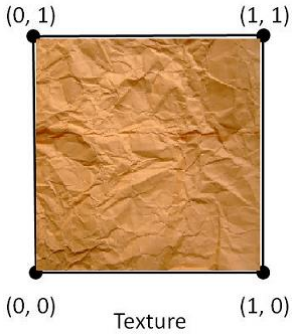
Is it simple?

Although the idea of mapping a texture (usually stored as an image in a file) onto a 3D surface is simple, there are 3 or 4 coordinate systems involved.

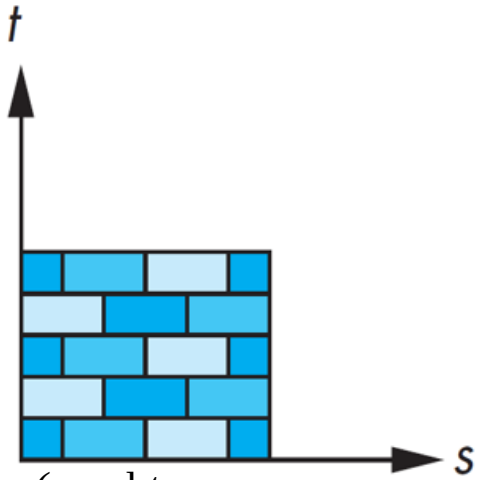


Texture Mapping

- **Texture Image**
  - In most applications, textures start out as two-dimensional images
    - Can scan texture images from the world (wood, skin, clouds) or paint yourself
  - A 2D image - represented by 2D array texture[height][width]
  - We call the elements of these arrays **texels**, or texture elements, rather than pixels to emphasize how they will be used.
  - Each element (or **texel**) has coordinate (s, t)



- **Texture Coordinates**
  - Used to identify points in the image to be mapped
  - **s** and **t** normalized to [0,1] range
  - **s** is used for the horizontal coordinate on the image and **t** is used for the vertical coordinate.
  - Note that texture coordinates are not based on pixels. No matter what size the image is, values of **s** and **t** between 0 and 1 cover the entire image.

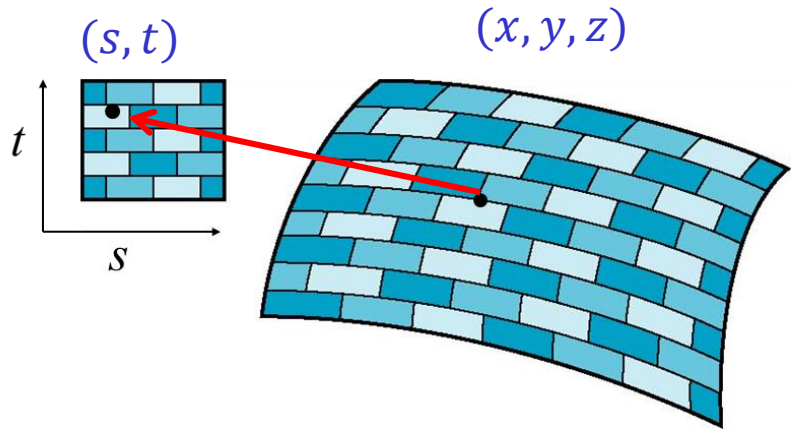


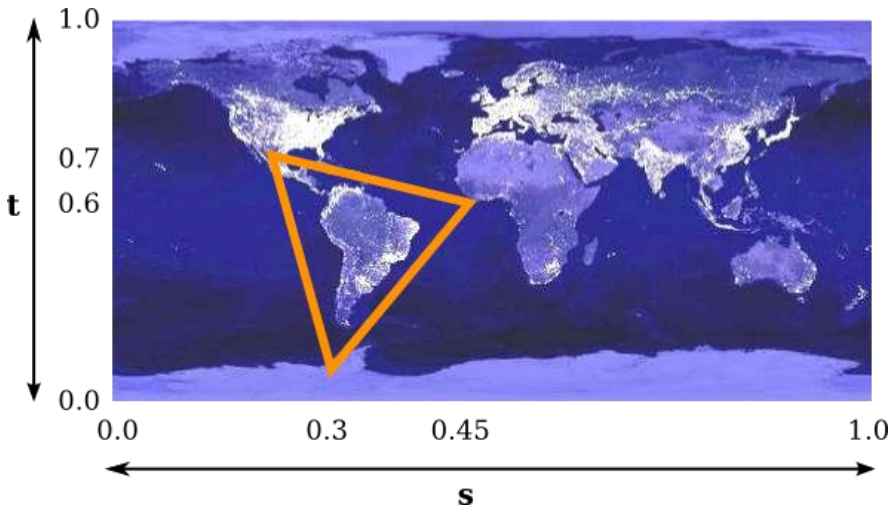
(*s* and *t* are both in the range 0..1)

Texture Mapping (cont.)

- To draw a textured primitive, we need a pair of texture coordinates (s,t) for each vertex.
- For the mapping from object coordinates (x, y, z) to texture coordinates (s, t), we need a mapping the form:
  - $s(x, y, z)$
  - $t(x, y, z)$

  i.e., given x, y, and z, what are the values for the parameters s and t to index into the texture map?





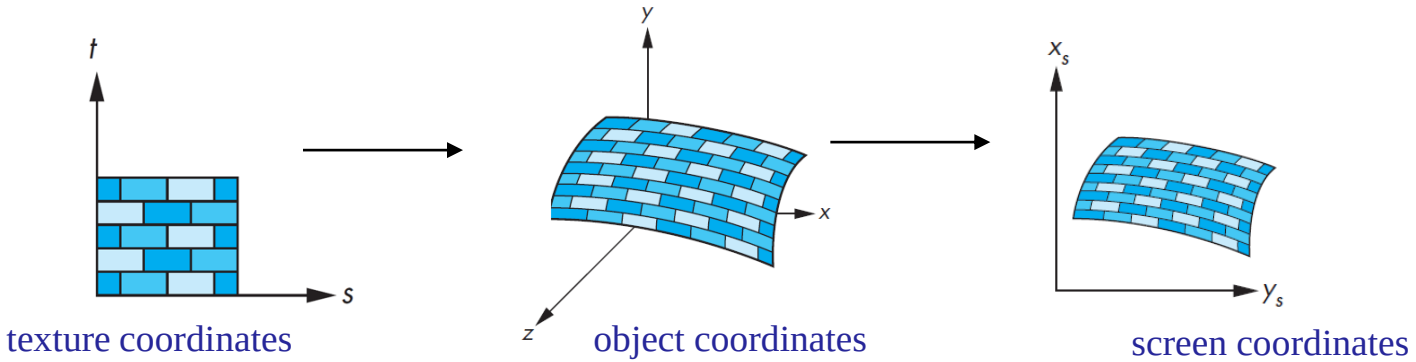
area in the image that is to be mapped onto the primitive is the triangle (outlined in thick orange)

Such mapping functions are difficult to find in general.

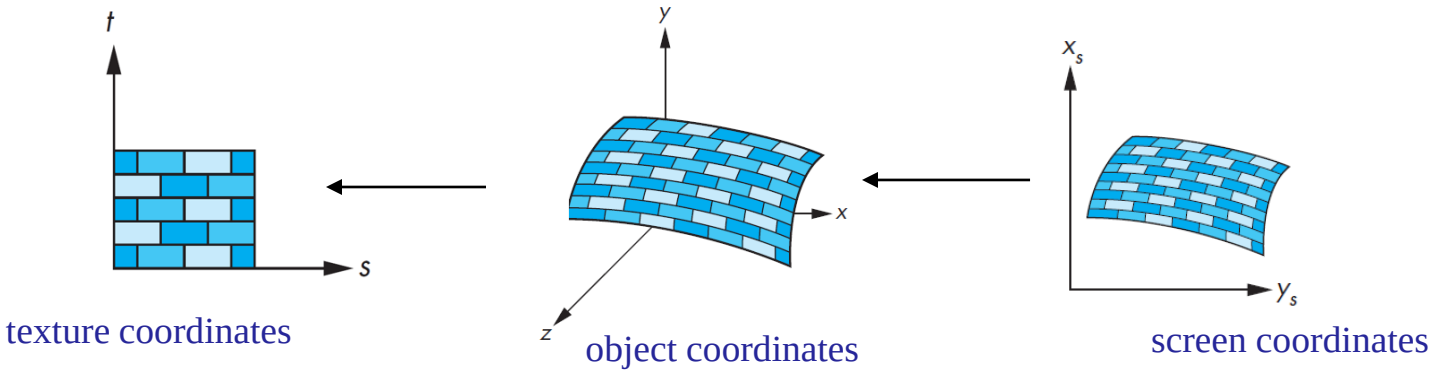
Image source

Texture Mapping (cont.)

Forward texture mapping (mapping from texture space to screen space)



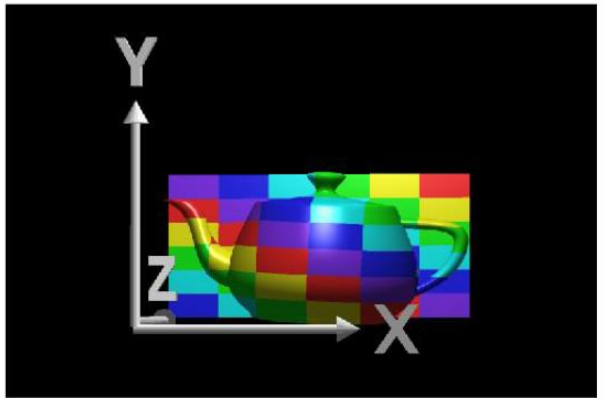
• very often inverse texture mapping/backward mapping (from screen space to texture space) is needed in rendering.



Texture Mapping

How do we map the texture onto an arbitrary (complex) object?

- Construct a mapping between the 3-D point to an intermediate surface?



courtesy of R. Wolfe

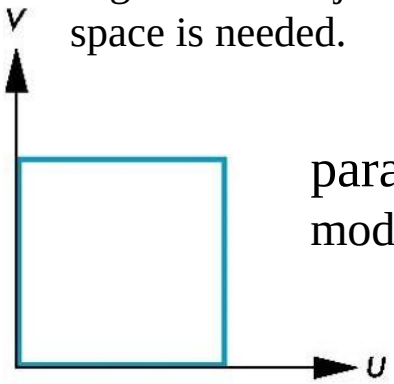
Two-part Mapping

- We mentioned earlier that backward mapping functions are difficult to find in general…

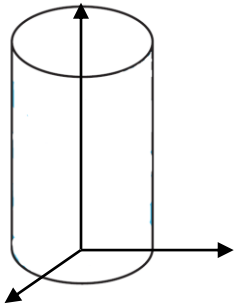
- One solution is to use a two-part mapping:
  - **First-part mapping:**
    - Look for an intermediate object that is closest to the 3D object that we want to map texture onto
    - Map the texture onto this intermediate object
  - **Second-part mapping:**
    - Map the texture from the intermediate object to the actual object.
    - There are 3 different ways (see later) to implement this second-part mapping.

Texture Mapping for Parametric Surfaces

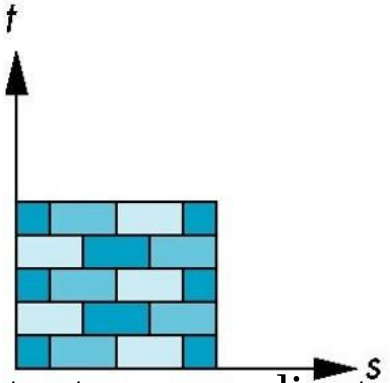
If geometric object is defined parametrically, an additional mapping function involving (u, v) space is needed.



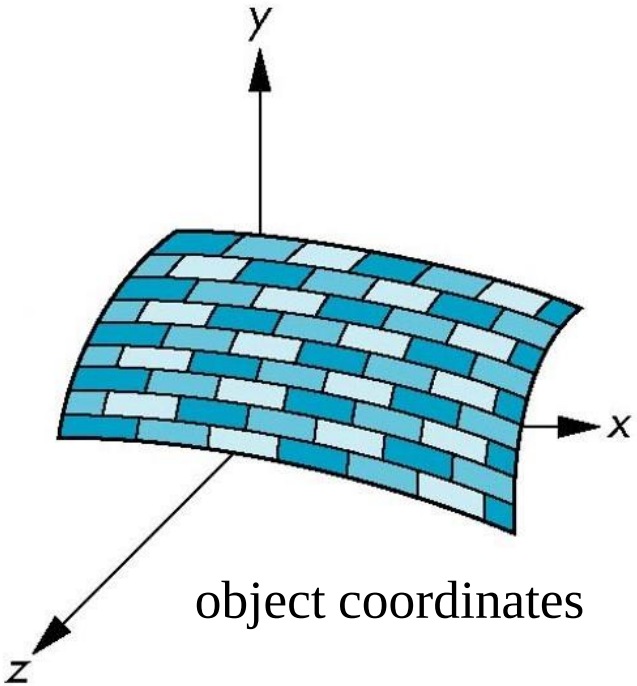
parametric coordinates: used to model curves and surfaces



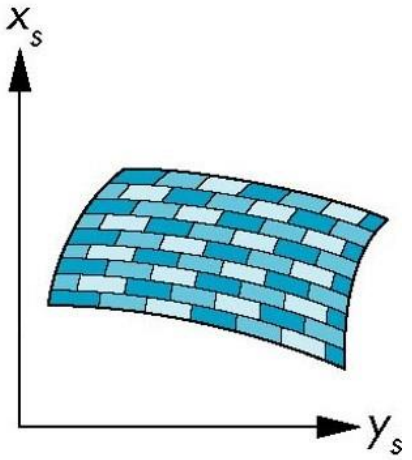
$$x = r\cos(2\pi u),$$
$$y = r\sin(2\pi u),$$
$$z = v/h,$$



texture coordinates (no unit, *s* and *t* are both in the range 0..1)



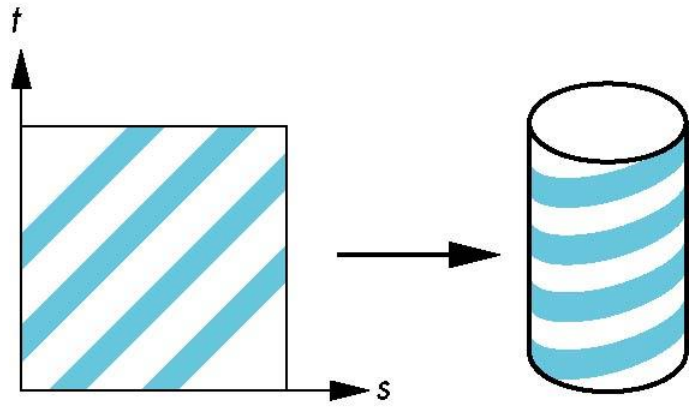
object coordinates



screen coordinates (in pixel unit)

Two-part Mapping

- Common intermediate objects:
  - cylinder
  - sphere
  - box

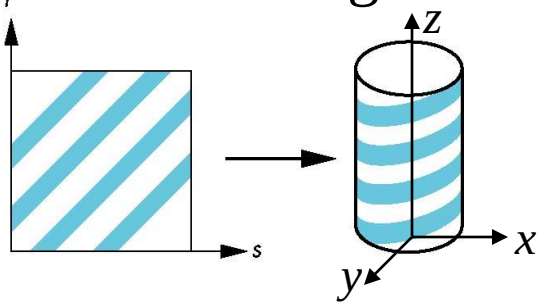


- Example: Suppose the intermediate surface is a cylinder. The first-part mapping is then to map the texture onto the cylinder.

First-part mapping: Intermediate object – cylinder

- Parametric equation of a cylinder with radius=r and height=h:
  - $x = r \cos 2\pi u$
  - $y = r \sin 2\pi u$
  - $z = v * h$

The parameters are u and v



- Given a 3D point (x, y, z) on the cylinder, do the following:
  - Find out the parameter values u and v
  - Since u and v vary from 0 to 1, we can simply let

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

  - retrieve the colour at coordinate (s, t) from the texture map.

Note that x and y take values in the range $-r \cdots r$, z takes values from 0 to h.

First-part mapping: Intermediate object – sphere

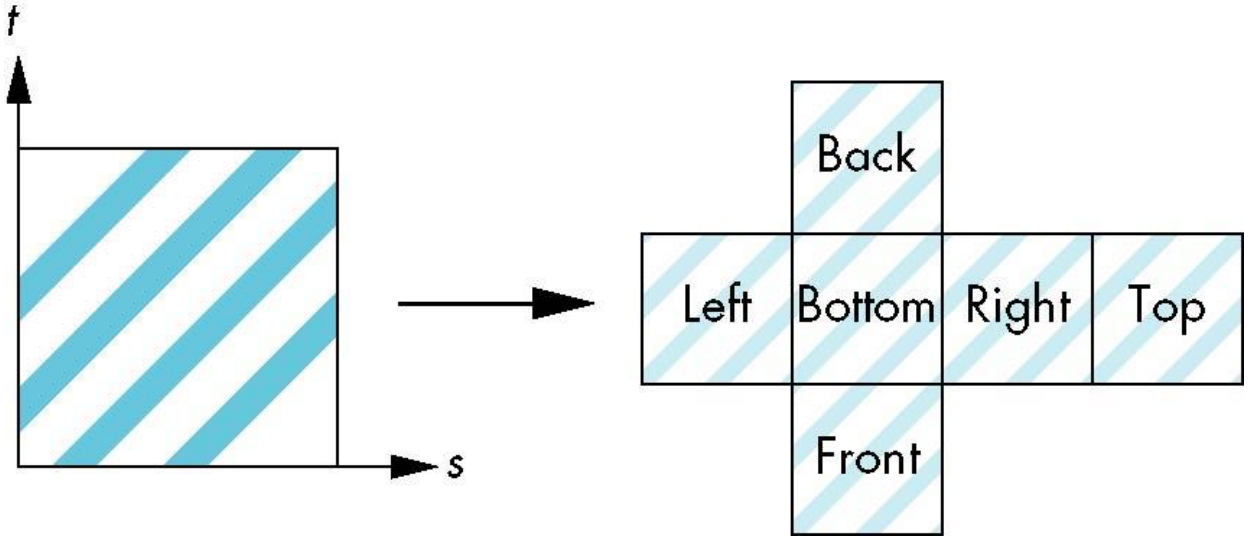
- We can also use a sphere as the intermediate object. The parametric equation of a sphere or radius r is given by:
  - $x = r \cos 2\pi u$
  - $y = r \sin 2\pi u \cos 2\pi v$
  - $z = r \sin 2\pi u \sin 2\pi v$

  in a similar manner to the cylinder:
  - Give (x, y, z) compute u and v
  - Let $s = u$ and $t = v$
- Spheres are used in environment mapping.

The parameters are u and v

First-part mapping: Intermediate object – box

- Boxes are also used as intermediate objects in environment mapping
- Easy to use with simple orthographic projection



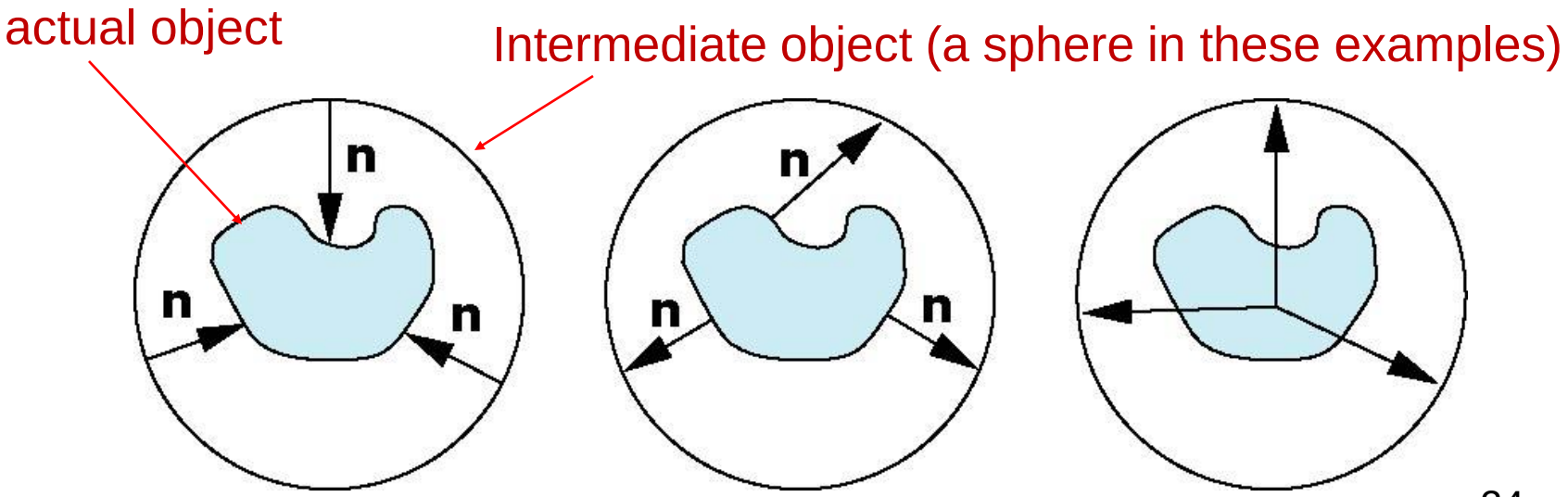
Example: First-part mapping: Intermediate object – cylinder

Convert rectangular coordinates (x, y, z) to cylindrical (r, u, v), use only (u, v) to index texture image

courtesy of
R. Wolfe

Second-part Mapping

- The second-part mapping is to map the texture from the intermediate object to the actual object.
- There are 3 ways to map texture from an intermediate object to the actual object:
  1. Normals from intermediate to actual
  2. Normals from actual to intermediate
  3. Vectors from centre of intermediate



Further Reading

"Interactive Computer Graphics – A Top-Down Approach with Shader-Based OpenGL" by Edward Angel and Dave Shreiner, 6th Ed, 2012

- Sec. 7.4 *Mapping Methods*
- Sec. 7.5 *Texture Mapping*, including
  - Sec. 7.5.1 *Two-Dimensional Texture Mapping*

References

- https://web.cse.ohiostate.edu/~shen.94/581/Site/Slides_files/texture.pdf
- https://www.cs.cmu.edu/~djames/15462/Fall03/notes/09-texture.pdf