Texture Mapping
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 triangles per second, that number is insufficient for many phenomena
  - Clouds
  - Grass
  - Terrain
  - Skin
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);
  - Increasing the no. 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, scan it, and “paste” the image of the orange skin onto a sphere
  - This process is known as texture mapping

• 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
Environment Mapping

g geometric model

Environment mapping output
Bump Mapping

geometric model

Bump mapping output
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.
The Coordinate Systems involved in Texture Mapping

• Parametric coordinates
  - May be used to model curves and surfaces

• Texture coordinates
  - Used to identify points in the image to be mapped

• Object or World Coordinates
  - Conceptually, where the mapping takes place

• Window or Screen Coordinates
  - Where the final image is really produced
Texture Mapping

parametric coordinates (in pixel unit)

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

object coordinates

screen coordinates (in pixel unit)
Mapping Functions

• Consider mapping from texture coordinates to a point on a surface
• We need three parametric functions to describe the surface
  - \( x(s, t) \)
  - \( y(s, t) \)
  - \( z(s, t) \)
• Given the texture coord. \((s, t)\) of a pixel, find out which point on the surface it corresponds to.

But we really want to go the other way…
Backward Mapping

• However, we really want to go backwards, i.e., mapping from screen coordinates to texture coordinates.
Backward Mapping (cont.)

• Given a pixel on the screen, we want to find the corresponding 3D point \((x, y, z)\) on the object and then the \((s, t)\) coordinates in the texture map that the pixel corresponds to.

• For the mapping from object coordinates \((x, y, z)\) to texture coordinates \((s, t)\), we need a mapping of 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?

• Such mapping functions are difficult to find in general.
Need to Map Areas to Areas

• Because each pixel corresponds to a small rectangle on the display, when doing the backward mapping, we cannot just map *points to points* in the different coordinate systems; we need to map *areas to areas*.

• In the diagram below, the non-square area of the texture should contribute to the shading of the pixel.
Aliasing

• We need to map areas to areas as mapping points to points can also cause aliasing even for simple flat surfaces.

• What is aliasing?
  • In Computer Graphics, this refers to the process by which smooth curves or lines become jagged because of the resolution of the graphics device or insufficient data to represent the curves.

• Aliasing occurs in texture mapping too.
• Suppose that we have a periodic looking texture and we map points to points. Then even for flat surfaces, we can still encounter the aliasing problem. Especially when the camera zooms out of the scene, this results in each pixel covering a large region of the 3D surface…

![Diagram showing point samples in texture space and 3D flat surface](image)

Point samples in texture space. We miss blue stripes. Point samples in the \((u,v)\) of the 3D space.
Aliasing (cont.)

• Mapping *areas to areas* means that we assign the average shading value of the texture region to the pixel. This helps to reduce the aliasing effect; however, we would still miss the blue stripes when the resolution of both the frame buffer and the texture map drops.

![Diagram showing point samples in texture space and 3D space. Text reads: point samples in texture space. We miss blue stripes.](image)
Two-part Mapping

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

• One solution is to use a two-part mapping:
  - First-part mapping:
    o Look for an intermediate object that is closest to the 3D object that we want to map texture onto
    o Map the texture onto this intermediate object
  - Second-part mapping:
    o Map the texture on the intermediate object to the actual object.
    o There are 3 different ways (see later) to implement this second-part mapping.
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=$h$:
  - $x = r \cos 2\pi u$
  - $y = r \sin 2\pi u$
  - $z = v/h$

- 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 $s = u$ and $t = v$
  - 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:
  o Give $(x, y, z)$ compute $u$ and $v$
  o Let $s = u$ and $t = v$

Obviously there is distortion. We have to decide where to put the distortion.

• Spheres are used in environment mapping.
First-part mapping: Intermediate object – box

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


- Sec. 7.4 Mapping Methods
- Sec. 7.5 Texture Mapping, including
  - Sec. 7.5.1 Two-Dimensional Texture Mapping
  - For the aliasing problem, see Pages 370-371.