Note 8. Texture Mapping
(Reading: Angel: Section 9.2)

- Textures are patterns saved as rectangular array of data called texture-image data. Textures can be some regular patterns generated with a program or a scanned-in image. The individual elements in a texture-image data are often called texels.

  - 2D texture pattern is usually represented by $T(s, t)$ where $s, t$ are texture coordinates.

  - $T(s, t)$ is stored in texture memory as $n \times m$ array of texels.

- Texture mapping involves the display of a texture pattern on an area of geometric surface.

- Texture mapping makes images of 3D objects look more interesting and apparently more complex. It can mimic the surface detail of real object.

- Texture mapping has been used to provide a means by which images may themselves be used as display primitives.

- Overall process from texture space ($s, t$) to screen space ($x_S, y_S$) involves 2 concurrent, non-linear mappings or transformations.

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

![Diagram](image)

**Figure 10.3** Texture maps for a parametric surface.
• Indicated here is *forward texture mapping* (*mapping from texture space to screen space*)

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

**Properties of texture mapping**

- Mapping transformations may be *complex* functions.

- Use RGB values of $T(s,t)$ to either assign colors to the surface or modify the colors of the surface using $T(s,t)$.

- In forward texture mapping, a selected texture patch usually does not match up with pixel boundaries, thus introducing *aliasing*

*Aliasing* in texture mapping is due to the limiting resolution of both texture map and frame buffer. This problem is most visible when the texture is periodic.

![Figure 10.5 Aliasing in texture generation.](image)

- For a pixel in *screen space*, its *preimage* is an inverse pixel map in the *texture or (s, t)* space. It always involves fractional areas of texel coverage.

![Figure 10.4 Preimages of pixel.](image)
Intensity of the pixel in screen space is determined by averaging the texel values of the area covered by the preimage.

In inverse texture mapping, calculation of the area of the preimage in the texture space requires calculation of the inverse viewing-projection and inverse texture-map transformations.

**Texture Transformation (mapping):**

**Linear texture mapping**

To simplify calculations, the mapping from texture space to object (world) space is often specified with parametric linear functions:

\[
\begin{align*}
  u &= as + bt + c \\
  v &= ds + et + f
\end{align*}
\]  

(8-2)

\(a,b,c,d,e,f\) are constants determined by the geometry of the surface.

As long as \(ae \neq bd\), this mapping is invertible.

Alternately, other simpler linear functions may also be used:

\[
\begin{align*}
  u &= as + b \\
  v &= ct + d
\end{align*}
\]  

(8-3)

To recall

The object-to-image mapping is accomplished with the concatenation of the viewing and projection transformations.
1. Mapping from a square pattern to a rectangle

To map a rectangle with corners \((s_{\text{min}}, t_{\text{min}})\) and \((s_{\text{max}}, t_{\text{max}})\) in \((s, t)\) space to a rectangle with corners \((u_{\text{min}}, v_{\text{min}})\) and \((u_{\text{max}}, v_{\text{max}})\) in \((x_s, y_s)\) or \((u, v)\) space.

**Mapping function** \(M_{s,t} \rightarrow u,v\) can be obtained as:

\[
M_{s,t} \rightarrow u,v = T(u_{\text{min}}, v_{\text{min}}) \cdot S \left( \frac{u_{\text{max}} - u_{\text{min}}}{s_{\text{max}} - s_{\text{min}}}, \frac{v_{\text{max}} - v_{\text{min}}}{t_{\text{max}} - t_{\text{min}}} \right) \cdot T(-s_{\text{min}}, -t_{\text{min}})
\]

\[
\begin{align*}
t &= u_{\text{min}} + \frac{s - s_{\text{min}}}{s_{\text{max}} - s_{\text{min}}} (u_{\text{max}} - u_{\text{min}}) \\
v &= v_{\text{min}} + \frac{t - t_{\text{min}}}{t_{\text{max}} - t_{\text{min}}} (v_{\text{max}} - v_{\text{min}})
\end{align*}
\]  

(8-4)

2. Two-stage mapping – *mapping to 3D objects*

I. **S- mapping:** maps texture to a simple 3D intermediate surface, e.g. inside of a spherical surface.

\[
T(s,t) \rightarrow T'((x_0, y_0, z_0))
\]

II. **O- mapping:** intermediate mapped surface is mapped to the surface being rendered.

Three possible mapping strategies:

(i) Using normal from intermediate surface,

(ii) Using normal from object surface,
(iii) Using center of the object.

Figure 10.9 Second mapping. (a) Using the normal from the intermediate surface. (b) Using the normal from the object surface. (c) Using the center of the object.

a) Mapping from a square pattern to a cylindrical surface

Figure 10.7 Texture mapping with a cylinder.

\[ x_i = r \cos (2\pi u), \]
\[ y_i = r \sin (2\pi u), \quad 0 \leq u, v \leq 1 \]
\[ z_i = v \cdot h. \]
b) If a sphere of unit radius is used as the intermediate surface, a possible mapping is:

\[
\begin{align*}
  x_i &= \sin 2\pi u \sin 2\pi v, & 0 \leq u \leq 1 \\
  y_i &= \cos 2\pi v, & 0 \leq v \leq 1/2 \\
  z_i &= \cos 2\pi u \sin 2\pi v.
\end{align*}
\]

To digress at this stage to derive the texture mapping functions for mapping a square pattern to a quadrilateral patch formed from a spherical surface:

\[
\begin{align*}
  x_i &= \sin 2\pi u \sin 2\pi v, & 0 \leq u \leq 1/4 \\
  y_i &= \cos 2\pi v, & 1/8 \leq v \leq 1/4 \\
  z_i &= \cos 2\pi u \sin 2\pi v.
\end{align*}
\]
Using \textit{eqn (8-3)} the mapping functions are:

\begin{align*}
\text{Forward Mapping: } \quad (u, v) &= \left( \frac{s}{4}, \frac{2-t}{8} \right) \\
\text{Inverse Mapping: } \quad (s, t) &= \left( 4u, 2-8v \right)
\end{align*}

c) Mapping with box

Here we map texture map to the 6 square surfaces of a box using \textit{eqn (8-4)}.

\textbf{Figure 10.8} Texture mapping with box.
Texture Mapping in Open GL

Steps in using 2D texture mapping in OpenGL:

1. Specify texture
2. Parameters for more mapping control
3. Enable texture mapping
4. Draw the scene, supplying both texture and geometric coordinates

1. Specify Texture

- Copy the texture-image data from the texture pattern \( T(s, t) \), e.g. from a .ppm file, to an \( n \times m \) image array. (may generate \( T(s, t) \) with a routine)

- Use \texttt{glTexImage2D()} to specify a 2D texture.

\begin{verbatim}
   glTexImage2D(GL_TEXTURE_2D, level, components, width, height, border, format, type, *image_buf);
\end{verbatim}

\texttt{e.g. Glubyte checkImage[ImageWidth][ImageHeight][3];
glTexImage2D(GL_TEXTURE_2D, 0, 3, 512, 512, 0, GL_RGB, GL_UNSIGNED_BYTE, &checkImage[0][0][0]);
}

- specify a 512x512 RGB image array, \texttt{checkImage}, to store the texture-image data.
- Components = 3 specifies number of color components for the texels.
- Level, border: fine control over detailed handling of texture.

2. Parameters for more mapping control

- Use \texttt{glTexParameter*()} to specify

\texttt{how the texture is to be wrapped}
\texttt{how the colors of pixels are to be filtered.}

The texture pattern defined in \texttt{glTexImage2D} normally covers a unit square in \((s, t)\) space. With larger range, \(e.g. \[0,3\]\) in both \(s, t\) direction in Fig. 9-6, a number of OpenGL parameters can be used to control the mapping in more detail.

\textbf{(a) Repeating / clamping mode}

\texttt{e.g.} a polygon can be repeatedly tiled with the texture map in repeating mode to generate Fig. 9-6:
glTexParameterf(GL_TEXTURE_2D, GL_TEXTURE_S_WRAP, GL_REPEAT);
gTexParameterf(GL_TEXTURE_2D, GL_TEXTURE_T_WRAP, GL_REPEAT);

if GL_CLAMP is used instead of GL_REPEAT, picture of Fig. 9-7 will be produced.

(b) Color filtering control

- Texture maps are square or rectangular, but after being mapped to a polygon and transformed into the screen coordinates, individual texels rarely correspond to individual pixels of the final image.

- A single pixel on the screen can correspond to a tiny portion of a texel (magnification), a large collection of texels (minification).

- Some form of filtering operations is needed to specify how the neighboring texels should be averaged to assign an appropriate color to the pixel.

Texture magnification
Texture minification

```plaintext
Texture Polygon
```

`glTexParameterf()`: the 2\textsuperscript{nd} parameter also allows for choice of color filtering operations:

Possible values for the 2\textsuperscript{nd} parameters:

- `GL_TEXTURE_S_WRAP` or `GL_TEXTURE_T_WRAP`, `GL_TEXTURE_MAG_FILTER` or `GL_TEXTURE_MIN_FILTER`.

E.g.:

```plaintext
glTexParameterf(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_NEAREST);
glTexParameterf(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_NEAREST);
```

(For more details, See OpenGL Programming Guide, pg 283).

Finally use `glTexEnvf()` to set the drawing mode of the texture mapped polygon.

```plaintext
glTexEnv(GL_TEXTURE_ENV, GL_TEXTURE_ENV_MODE, GL_DECAL);
```

decal mode - `GL_DECAL` - use the texture color as the final color of the texture mapped polygon.

modulating mode - `GL_MODULATE` - each of the color on the texture mapped polygon is multiplied by a corresponding value in the texture, useful for combining the effects of shading with texturing.

Blending mode - `GL_BLEND` - each of the color on the texture mapped polygon is mixed with a corresponding value in the texture like doing $\alpha$-blending of colors.

3. Enabling texture mapping
Like many other OpenGL features, texturing must be explicitly enable, using:

```
glEnable / glDisable (GL_TEXTURE_2D)```

to turn on/off texturing.

(Steps 2&3 do the texture-mapping initialization)

4. Draw the scene - supplying both texture and geometric coordinates:

- **Assign texture coordinates:**
  To draw the texture mapped scene, both the object coordinates and the texture coordinates must be provided side-by-side using

```
glTexCoord2*( );
glVertex*( );```

Refer to the checkerboard program

(*Listing 9-1 of OpenGL Programming Guide*):

*Texture image generated by a routine:*

Two texture-mapped squares

(size: 2x2 in defined in world coordinates)

Facing viewer

Tilting back 45°
Demo program

Source: Nate Robins (Utah)

- Try running the program experimenting with different texture maps.
- For the checkerboard, e.g. set vertex1 to (-0.4, 0.4, 0.0) and choose different option of \textit{GL\_TEXTURE\_MIN\_FILTER} mode for color filtering control.