Texture Mapping

Interactive Graphical Systems
HT2006

Lars Pettersson
lwp@it.uu.se

This lecture

- My research
- Texture Mapping
  - Usage of textures
  - Image sampling
  - Applying textures
  - Advanced texture mapping techniques
- Summary
My research

- View-dependent co-located visualization

Texture mapping in my research

- Textures based on geospatial data

From http://en.wikipedia.org/wiki/Texture_mapping
How are textures used?

- Transparency handling
- Camera back drop (= scene background texture)
- Shadow mapping
- Video- and picture animations
- Rendering applications in texture buffer.

Usage of Textures

- Camera Backdrop
Usage of Textures (cont.)

- Camera Backdrop

![Birds eye view on scene](image1.png)  ![Camera view on scene](image2.png)

Usage of textures

- Transparency handling
  - Alpha blending
    - VRT_SetTextureModulationMode( int texture_modulation )
      - VRT_TEXTURE_MODULATION_DECAL – replaces color of geometry
      - VRT_TEXTURE_MODULATION_MODULATE - multiplies color of geometry
  - Alpha masking
    - In VRT - use DECAL mode
      - set texture alpha value to zero for alpha masking
Usage of Textures (cont.)

- **Shadow mapping**
  - Render the scene from the lights point of view and save the depth values to a texture
    - The result is a depth map or “shadow map”
    - Every pixel in the map indicates its distance from the light source

- **Video- and picture animations**
  - Picture animations (see lecture 4)
  - Video animation
Usage of Textures (cont.)

- Rendering applications in texture buffer.
  - Project Looking Glass, Mac OS X, Microsoft Vista

Benefits of using textures

- to render 3D scenes (photo) realistic
- to provide detail without generating numerous geometric objects
- to allow for faster rendering of complex 3D scenes
- to accomplish special visual effects (e.g. animations, fog, plasma...)
- to fake lighting and shadow effects
What is a texture map?

- digital image matrix \((u,v)\) (most commonly 2D)
- single matrix elements = texels (texture elements) (contains most commonly color information)
- spatial resolution sampling
- amplitude resolution = texel depth \(\Rightarrow\) quantization

Spatial Sampling

600 x 460

300 x 230

150 x 115

75 x 58

38 x 29
Pixel (Texel) quantization

- 256 gray levels
- 64 gray levels
- 16 gray levels
- 4 gray levels
- 2 gray levels

Example of Color (Texel) formats

- 8 bit per texel gray scale
- 8 bit per texel color palette
- 16 bit per texel gray scale
- 24 bit per texel RGB
- 32 bit per texel RGBA (used in VRT)
The scene rendering process

Applying a texture?

- For each point in an object we must decide what texel in the texture to use.
- A common way to describe an object is by a number of polygons. Then we must decide for each point in each polygon what texel to use.
Texturing

What texel value should be used for the pixel at position \( w \) at scanline \( s \)?

Need to find a mapping.

In this case the texture coordinates for \( p_0, p_1, p_2 \) is \( a, b, c \).

\[
\begin{align*}
\alpha &= \frac{w_x - A_x}{B_x - A_x} \\
\beta &= \frac{w_y - A_y}{B_y - A_y}
\end{align*}
\]

By noting that \( A \) and \( B \) corresponds to base axis in the texture \( \alpha \) and \( \beta \) can be used to look up the right texel.
Texturing example

Given a texture of size 128x128 and a triangle defined by the vertices \( p = \begin{bmatrix} 5 \\ 0 \\ 9 \\ 0 \\ 128 \\ 0 \\ 128 \end{bmatrix} \) and corresponding texture coordinates \( t = \begin{bmatrix} 0 \\ 128 \\ 0 \\ 0 \\ 18 \\ 17 \end{bmatrix} \) Compute the texel to use for the point \( w_{\text{texel}} = \begin{bmatrix} 18 \\ 17 \end{bmatrix} \) inside the triangle:

\[
\begin{bmatrix}
A = p_2 - p_1 \\
B = p_3 - p_1
\end{bmatrix}
\]

\[
\begin{bmatrix}
\alpha = \frac{A_m - A_n}{A_m - A_n} \\
\beta = \frac{B_m - B_n}{B_m - B_n}
\end{bmatrix}
\]

\[
t_u = 127 \cdot 0.775 = 98.425 \\
t_v = 127 \cdot 0.725 = 92.075 \\
t = (u,v) = (98,92) \text{ rounded}
\]

Addressing texels in a texture map

width, height of texture map usually \( 2^n \) (e.g. \( 2^7 = 128 \))

\[
\Rightarrow \text{ texture size: } 128 \times 128
\]

\[
t_u = 127 \cdot 0.775 = 98.425 \\
t_v = 127 \cdot 0.725 = 92.075 \\
t = (u,v) = (98,92) \text{ rounded}
\]

address \( t_{uw} = t_u \times \text{width} + t_v \times \text{bpt} \)

\[
= (t_u \times \text{width} + t_v) \times \text{bpt}
\]

ex.: \( (92 \cdot 128 + 98) \cdot 1 = 11874 \)

if width \( 2^n \) \( 2^n \leftrightarrow \text{shl n} \)

\[
\begin{bmatrix}
0 \\ 128 \\ 0 \\ 128 \\ 0 \\ 128 \\ 0 \\ 128
\end{bmatrix}
\]

\[
\begin{bmatrix}
0 \\ 128 \\ 0 \\ 128 \\ 0 \\ 128 \\ 0 \\ 128
\end{bmatrix}
\]
Example: Draw on texture

\( u, v \) is continuously known from collision detection in the rendered scene.

For each \( t(u,v) \) change its RGBA component in memory to black

The "Texel Footprint"

The area in screen pixels covered by a texture element

It is very uncommon that one texture element covers exactly one pixel, due to different scale and rotational alignment.

\[ \text{-> How should the color of a pixel then be calculated?} \]
Practical problems:

- Texture map is bigger than the rendered polygon
  \[\Rightarrow\] sub-sampling

- Texture map is smaller than the rendered polygon
  \[\Rightarrow\] super-sampling

Texel neighborhood

- 2 - point neighborhood
- 4 - point neighborhood
- 6 - point neighborhood
- 8 - point neighborhood
Sub-sampling

1. point sampling / nearest-neighbor
⇒ might result in irregular rendered texture due to rounding errors

Sub-sampling (cont.)

2. averaging in a defined neighborhood
⇒ improves visual result but does not necessarily eliminate the problem
Super-sampling

1. point sampling / multiple nearest-neighbor

2. Bilinear interpolation
   includes weighted neighborhood
Mip maps

**Problem:** Artifacts due to image sub-sampling

**Solution:** Pre-processing of the texture map with local filters (e.g. smoothing)

Texture acquisition

- Scanning of photographs or drawings
- Digital drawings
- Digital camera images
- Rendered images
- Procedural Images
Example

The source code below creates a fireplace model composed of 3 polygons based on 8 vertices.

```c
geom = VRT_GeometryNew();
VRT_GeometryAddVertex (geom, 0.0,0.0,0.0, 0.0,0.0,0.0, ?, ?);
VRT_GeometryAddVertex (geom, 1.0,0.0,0.0, 0.0,0.0,0.0, ?, ?);
VRT_GeometryAddVertex (geom, 1.0,1.0,0.0, 0.0,0.0,0.0, ?, ?);
VRT_GeometryAddVertex (geom, 0.0,1.0,0.0, 0.0,0.0,0.0, ?, ?);
VRT_GeometryAddVertex (geom, 1.0,0.0,-0.25, 0.0,0.0,0.0, ?, ?);
VRT_GeometryAddVertex (geom, 1.0,1.0,-0.25, 0.0,0.0,0.0, ?, ?);
VRT_GeometryAddVertex (geom, 1.0,1.0,-0.25, 0.0,0.0,0.0, ?, ?);
VRT_GeometryAddVertex (geom, 0.0,1.0,-0.25, 0.0,0.0,0.0, ?, ?);
texture = LoadTexture ("c:\vrlab\fireplace.bmp");
```

```c
poly = VRT_GeometryNewQuad (geom, 3, 2, 1, 0);
VRT_PolygonSetTexture (poly, texture);
poly = VRT_GeometryNewQuad (geom, 7, 6, 2, 3);
VRT_PolygonSetTexture (poly, texture);
poly = VRT_GeometryNewQuad (geom, 2, 5, 4, 1);
VRT_PolygonSetTexture (poly, texture);
```

```c
node = VRT_NodeNew (root, "a fireplace");
VRT_NodeSetGeometry (node, geom);
```

Example (cont.)

You are given a texture map of size 256 x 256 texels (s. below). This texture is supposed to be mapped onto all three polygons. Give for all vertices v_n (n = 0..7) the normalized texture coordinates u and v such that the results of the mapping equals to the picture in the lower left corner of the picture shown below!
Inverse mapping based on ray tracing

Highly complex 3D shape (10,000 polygons)
Photographic picture of the real 3D object

Question: How can we map the photographic picture upon the 3D shape?

Simulate the photographic scenario:
- Suppose a given camera angle
- Align 2D picture and 3D shape such that 3D shape and picture “match” (rotation, translation, scale)
- Perform ray tracing from polygon vertices towards observer's eye
- Calculate u/v coordinates in textured polygon
- Assign hit coordinates (u/v) to polygon vertices

Result:
- After the inverse mapping procedure 3D shape appears photographically correct for a limited field of view/rotations

Page 18
View-dependent texture mapping

When many photographic textures of the same object exist one can use View-dependent texture mapping.

Every triangle has several textures associated with it. Before the texturing phase begins a texture is computed from the reference textures.

This will give better results as some details may not be seen in all views.

Most modern graphic cards can in addition to 2D texture mapping also handle 3D textures. The texels has an additonal dimension "t", (u,v,t).
Bump mapping is a technique where the texel values do not correspond to a colour but rather a normal. This normal indicates changes in the object surface.

The changes are taken into account when the shading calculations are performed, perturbing the object surface normal, giving the illusion of a 'bumpy' surface.

Note that this technique doesn’t alter the geometry of the object.