This lecture

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

- Co-located Visualization
  - Multiple view dependent perspectives on one physically co-located computer generated scene.
  - Ways to accomplish co-located visualization
    - Augmented Reality
    - Volumetric Displays
    - Multiple viewer displays
Texture mapping in my research

- Textures based on geospatial data
- 3D texture mapping
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...) (lecture 4)
- to fake lighting and shadow effects (lecture 4)
What to use textures for?

- Transparency handling
  - alpha masking
  - alpha blending
- Camera back drop (= scene background texture)
- Shadow mapping
- Video- and picture animations
- Rendering applications in texture buffer.
Usage of textures

- Transparency handling
  - Alpha blending
  - VRT_SetTextureModulationMode( int texture_modulation )
    - VRT_TEXTURE_MODULATION_DECAL – replaces color of geometry
    - VRT_TEXTURE_MODULATION_MODULATE - multiples color of geometry

- Alpha masking
  - In VRT - use DECAL mode set texture alpha value to zero for alpha masking
Usage of Textures (cont.)

- Camera Backdrop
Usage of Textures (cont.)

- Camera Backdrop

Birds eye view on scene

Camera view on scene
Usage of Textures (cont.)

- Shadow mapping (see lecture 4)
  - 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
Usage of Textures (cont.)

- Video- and picture animations
  - Picture animations (see lecture 4)
  - Example of video animation in textures is the backdrop you will work with in laboration 5
Usage of Textures (cont.)

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

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
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

1. **Modeling**
   - World Coordinate System

2. **Geometric Processing**
   - Model Transformations
   - Lighting Calculation
   - Projection -> 2D

3. **Rasterization**
   - Screen Coordinate System
     - Drawing Pixels/Texels
     - Depth Test
     - Alpha Blending

4. **Display**

Diagram:
- Lights
- Viewing Position
- Objects
- Coordinates
- Colors, Transparency
- Textures
- Surface Normal Vectors
- Specularity
- Frame Buffer
- Surface Normal Vectors
Applying a texture?

- For each point in and 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 \( a, b, c \).
Texturing

\[ A = p_3 - p_1 \]
\[ B = p_2 - p_1 \]

\( w \) can then be expressed as a linear combination of \( A \) and \( B \).

\[ w = \alpha A + \beta B \quad \alpha, \beta \in [0,1] \]

The following system can be set up:

\[
\begin{bmatrix}
  w_x \\
  w_y \\
\end{bmatrix}
= \begin{bmatrix}
  A_x & B_x \\
  A_y & B_y \\
\end{bmatrix}
\begin{bmatrix}
  \alpha \\
  \beta \\
\end{bmatrix}
\]

Solving it results in the following expressions for \( \alpha \) and \( \beta \):

\[
\alpha = \frac{B_y w_x - B_x w_y}{A_x B_y - A_y B_x}, \quad \beta = \frac{A_x w_y - A_y w_x}{A_x B_y - A_y B_x}.
\]

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 $p_1 = \begin{bmatrix} 5 \\ 5 \end{bmatrix}$, $p_2 = \begin{bmatrix} 10 \\ 20 \end{bmatrix}$, $p_3 = \begin{bmatrix} 30 \\ 25 \end{bmatrix}$ and corresponding texture coordinates $\begin{bmatrix} 0 \\ 128 \\ 0 \\ 128 \end{bmatrix}$, compute the texel to use for the point $w_{\text{global}} = \begin{bmatrix} 18 \\ 17 \end{bmatrix}$ inside the triangle.
Texturing example

Given a texture of size 128x128 and a triangle defined by the vertices \( p_1 = \begin{bmatrix} 5 \\ 5 \end{bmatrix}, \quad p_2 = \begin{bmatrix} 10 \\ 20 \end{bmatrix}, \quad p_3 = \begin{bmatrix} 30 \\ 25 \end{bmatrix} \) and corresponding texture coordinates \( \begin{bmatrix} 0 \\ 0 \\ 0 \\ 128 \\ 0 \\ 128 \end{bmatrix} \), Compute the texel to use for the point \( w_{global} = \begin{bmatrix} 18 \\ 17 \end{bmatrix} \) inside the triangle.

Solution:
\[
A = p_2 - p_1 = \begin{bmatrix} 10 \\ 20 \end{bmatrix} - \begin{bmatrix} 5 \\ 5 \end{bmatrix} = \begin{bmatrix} 5 \\ 15 \end{bmatrix}, \quad B = p_3 - p_1 = \begin{bmatrix} 30 \\ 25 \end{bmatrix} - \begin{bmatrix} 5 \\ 5 \end{bmatrix} = \begin{bmatrix} 25 \\ 20 \end{bmatrix}
\]

Translate \( w_{global} \) to the space spanned by \( AB \):
\[
w = w_{global} - p_1 = \begin{bmatrix} 18 \\ 17 \end{bmatrix} - \begin{bmatrix} 5 \\ 5 \end{bmatrix} = \begin{bmatrix} 13 \\ 12 \end{bmatrix}.
\]
\[
\alpha = \frac{B_yw_x - A_yw_y}{A_xB_y - A_yB_x} = \frac{20*13 - 25*12}{5*20 - 15*25} = \frac{-40}{-275} = \frac{40}{275}.
\]
\[
\beta = \frac{A_xw_y - A_yw_x}{A_xB_y - A_yB_x} = \frac{5*12 - 15*13}{5*20 - 15*25} = \frac{-135}{-275} = \frac{135}{275}.
\]

Look up texel at
\[
u = \alpha \times 128 = \frac{40}{275} \times 128 \approx 19
\]
\[
v = \beta \times 128 = \frac{135}{275} \times 128 \approx 63
\]
Addressing texels in a texture map

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

⇒ texture size: 128 * 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 \((u,v) = t_v \cdot \text{width} \cdot \text{bpt} + t_u \cdot \text{bpt} \]
\[ = (t_v \cdot \text{width} + t_u) \cdot \text{bpt} \]

ex.: \((92 \cdot 128 + 98) \cdot 1 = 11874 \]
\[ \cdot 3 = 35622 \]
\[ 35623 \]
\[ 35624 \]

bpt = bytes per texel

if width $2^n \Rightarrow 2^n \leftrightarrow \text{shl} \ n$
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.

-> How should the color of a pixel then be calculated?
Practical problems:

- Texture map is bigger than the rendered polygon ⇒ sub-sampling
- Texture map is smaller than the rendered polygon ⇒ 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

\[ f(x, y) = \frac{\sum_{i=1}^{n} f(u_i, v_i)}{n} \]
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 is 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”);
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);

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!
Practical issues of texture mapping

By Ronny Tucker

Step1: Editing and modeling the 3D object
Practical issues of texture mapping

By Ronny Tucker

Step2: Decompose object into sub-surfaces

Step3: Flatten sub-surfaces to get 2D template

Apply simple u/v texture co-ordinates
Practical issues of texture mapping

http://www.ronnie.net.demon.co.uk/Tutorials/Texture%20Mapping/TextureMapping.htm
By Ronny Tucker

Step 4: Import template into a paint program and draw texture.

Step 5: Apply texture to u/v mapped vertices in the original model.
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?
Inverse mapping based on ray tracing

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 observers 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
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.
3D Texture Mapping

Many 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 don’t 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, pertubating the object surface normal, giving the illusion of a ‘bumpy’ surface.

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