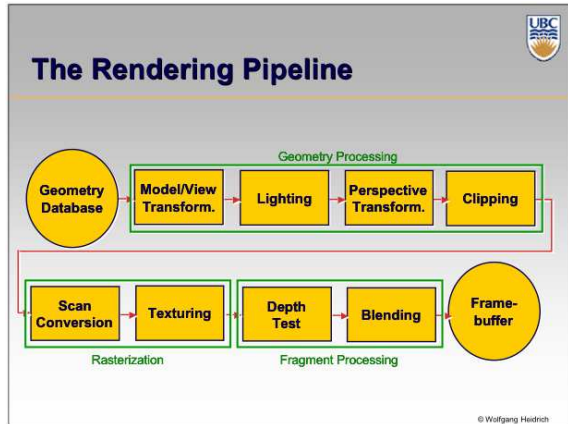





Texture Mapping

CPSC 314


© Wolfgang Heidrich






Texture Mapping

- Real life objects have nonuniform colors, normals
- To generate realistic objects, reproduce coloring & normal variations = **texture**
- Can often replace complex geometric details



© Wolfgang Heidrich



Texture Mapping

Introduced to increase realism

- Lighting/shading models not enough


Hide geometric simplicity

- Images convey illusion of geometry
- Map a brick wall texture on a flat polygon
- Create bumpy effect on surface

Associate 2D information with 3D surface

- Point on surface corresponds to a point in texture
- "Paint" image onto polygon

© Wolfgang Heidrich

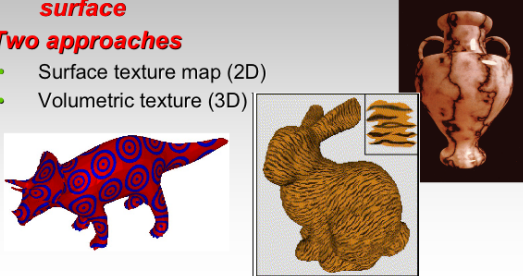


Color Texture Mapping


Define color (RGB) for each point on object surface

Two approaches

- Surface texture map (2D)
- Volumetric texture (3D)



© Wolfgang Heidrich



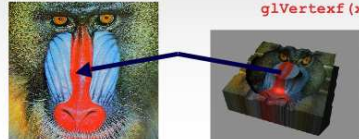
Surface (2D) Textures: Texture Coordinates

Texture Image: 2D array of color values (texels)

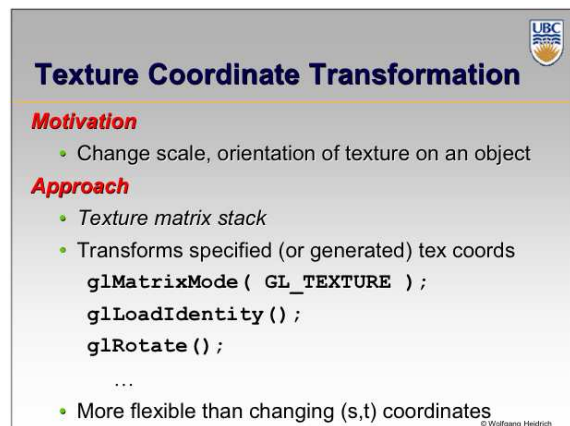
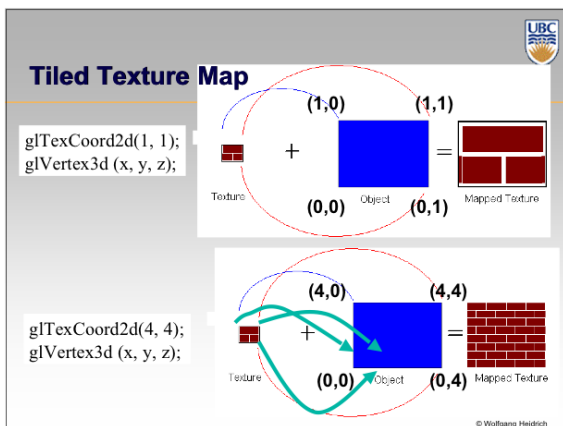
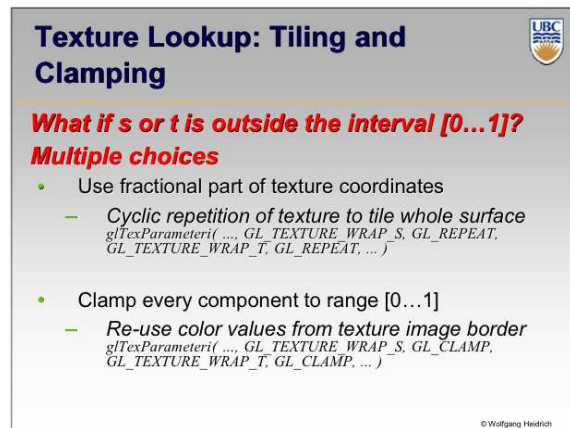
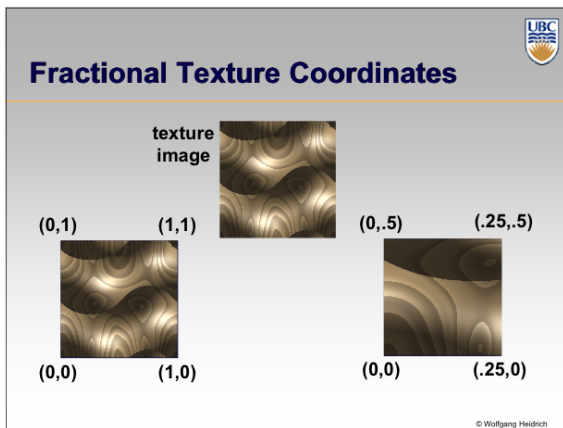
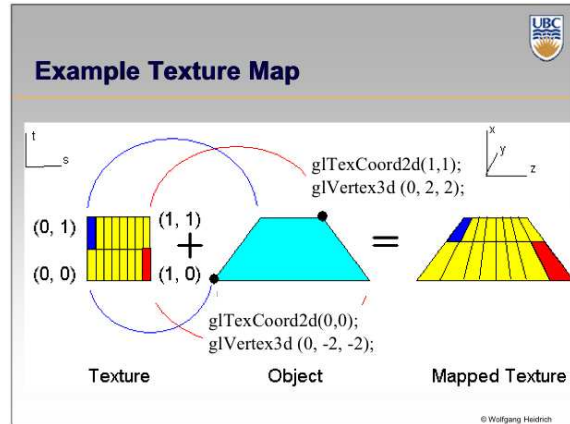
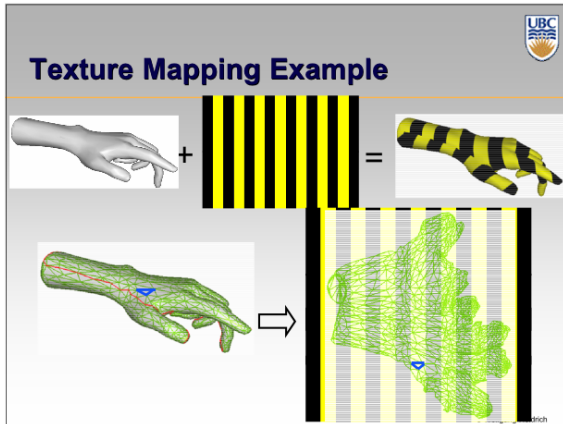
Assigning texture coordinates (s,t) at vertex with object coordinates (x,y,z,w)

- Use interpolated (s,t) for texel lookup at each pixel
- Use value to modify a polygon's color
- Or other surface property
- Specified by programmer or artist

`glTexCoord2f (s, t)`
`glVertexf (x, y, z, w)`



© Wolfgang Heidrich



Texture Functions

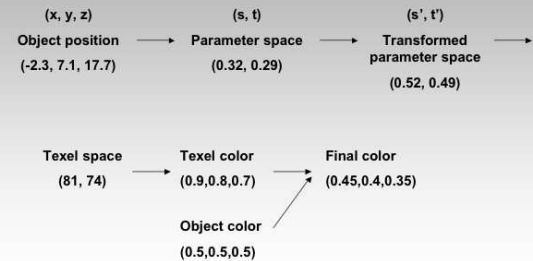
Once you have value from the texture map, can:

- Directly use as surface color: `GL_REPLACE`
 - Throw away old color, lose lighting effects
- Modulate surface color: `GL_MODULATE`
 - Multiply old color by new value, keep lighting info
 - Texturing happens **after** lighting, not relit
- Use as surface color, modulate alpha: `GL_DECAL`
 - Like replace, but supports texture transparency
- Blend surface color with another: `GL_BLEND`
 - New value controls which of 2 colors to use

Specify desired behavior with `glTexEnvf(GL_TEXTURE_ENV, GL_TEXTURE_ENV_MODE, <mode>)`

© Wolfgang Heidrich

Texture Pipeline



© Wolfgang Heidrich

Texture Objects and Binding

Texture object

- An OpenGL data type that keeps textures resident in memory and provides identifiers to easily access them
- Provides efficiency gains over having to repeatedly load and reload a texture
- You can prioritize textures to keep in memory
- OpenGL uses least recently used (LRU) if no priority is assigned

Texture binding

- Which texture to use right now
- Switch between preloaded textures

© Wolfgang Heidrich

Basic OpenGL Texturing

Create a texture object and fill it with texture data:

- `glGenTextures(num, &indices)` to get identifiers for the objects
- `glBindTexture(GL_TEXTURE_2D, identifier)` to bind
 - Following texture commands refer to the bound texture
- `glTexParameterf(GL_TEXTURE_2D, ..., ...)` to specify parameters for use when applying the texture
- `glTexImage2D(GL_TEXTURE_2D, ..., ...)` to specify the texture data (the image itself)

© Wolfgang Heidrich

Basic OpenGL Texturing (cont.)

Enable texturing:

- `glEnable(GL_TEXTURE_2D)`

State how the texture will be used:

- `glTexEnvf(...)`

Specify texture coordinates for the polygon:

- Use `glTexCoord2f(s, t)` before each vertex:
 - `glTexCoord2f(0, 0);`
 - `glVertex3f(x, y, z);`

© Wolfgang Heidrich

Low-Level Details

Large range of functions for controlling layout of texture data

- State how the data in your image is arranged
- e.g.: `glPixelStorei(GL_UNPACK_ALIGNMENT, 1)` tells OpenGL not to skip bytes at the end of a row
- You must state how you want the texture to be put in memory: how many bits per "pixel", which channels,...

Textures must have a size of power of 2

- Common sizes are 32x32, 64x64, 256x256
- But don't need to be square, i.e. 32x64 is fine
- Smaller uses less memory, and there is a finite amount of texture memory on graphics cards

© Wolfgang Heidrich

Texture Mapping

Texture coordinate interpolation

- Perspective foreshortening problem

© Wolfgang Heidrich

Interpolation: Screen vs. World Space

Screen space interpolation incorrect

- Problem ignored with shading, but artifacts more visible with texturing

© Wolfgang Heidrich

Texture Coordinate Interpolation

Perspective correct interpolation

- α, β, γ :
 - Barycentric coordinates of a point P in a triangle
- s_0, s_1, s_2 :
 - Texture coordinates of vertices
- w_0, w_1, w_2 :
 - Homogeneous coordinates of vertices

$$s = \frac{\alpha \cdot s_0 / w_0 + \beta \cdot s_1 / w_1 + \gamma \cdot s_2 / w_2}{\alpha / w_0 + \beta / w_1 + \gamma / w_2}$$

© Wolfgang Heidrich

Reconstruction

© Wolfgang Heidrich

Reconstruction

- How to deal with:
 - Pixels that are much larger than texels?**
 - Apply filtering, "averaging"
 - "Minification"
 - Pixels that are much smaller than texels?**
 - Interpolate
 - "Magnification"

© Wolfgang Heidrich

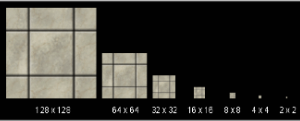
Magnification: Interpolating Textures

- Nearest neighbor
- Bilinear
- Hermite (cubic)


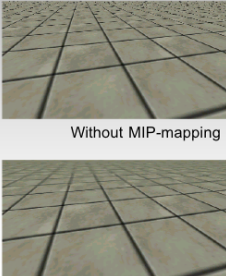
© Wolfgang Heidrich

Minification: MIPmapping

use "image pyramid" to precompute averaged versions of the texture



store whole pyramid in single block of memory

Without MIP-mapping

With MIP-mapping

MIPmaps


Multum in parvo -- many things in a small place

- Prespecify a series of prefiltered texture maps of decreasing resolutions
- Requires more texture storage
- Avoid shimmering and flashing as objects move

gluBuild2DMipmaps

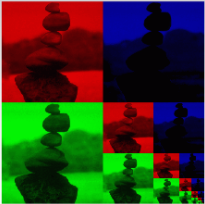
- Automatically constructs a family of textures from original texture size down to 1x1

without with



MIPmap storage

only 1/3 more space required




© Wolfgang Heidrich

Texture Parameters

In addition to color can control other material/object properties

- Surface normal (bump mapping)
- Reflected color (environment mapping)



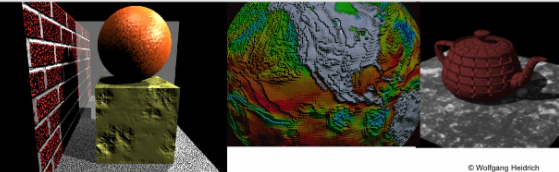
© Wolfgang Heidrich

Bump Mapping: Normals As Texture

Object surface often not smooth – to recreate correctly need complex geometry model

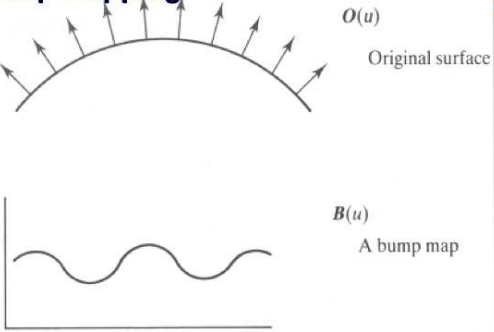
Can control shape "effect" by locally perturbing surface normal

- Random perturbation
- Directional change over region



© Wolfgang Heidrich

Bump Mapping



$O(u)$
Original surface

$B(u)$
A bump map

© Wolfgang Heidrich

Bump Mapping

$O'(u)$
Lengthening or shortening $O'(u)$ using $B(u)$

$N'(u)$
The vectors to the 'new' surface

© Wolfgang Heidrich

Displacement Mapping

Bump mapping gets silhouettes wrong

- Shadows wrong too

Change surface geometry instead

- Need to subdivide surface

GPU support

- Bump and displacement mapping not directly supported: require per-pixel lighting
- However: modern GPUs allow for programming both yourself

© Wolfgang Heidrich

Environment Mapping

Cheap way to achieve reflective effect

- Generate image of surrounding
- Map to object as texture

© Wolfgang Heidrich

Sphere Mapping

Texture is distorted fish-eye view

- Point camera at mirrored sphere
- Spherical texture mapping creates texture coordinates that correctly index into this texture map

© Wolfgang Heidrich

Cube Mapping


6 planar textures, sides of cube

- Point camera in 6 different directions, facing out from origin

© Wolfgang Heidrich

Cube Mapping

© Wolfgang Heidrich




Cube Mapping

Direction of reflection vector r selects the face of the cube to be indexed

- Co-ordinate with largest magnitude
 - e.g., the vector $(-0.2, 0.5, -0.84)$ selects the $-Z$ face
- Remaining two coordinates (normalized by the 3rd coordinate) selects the pixel from the face.
 - E.g., $(-0.2, 0.5)$ gets mapped to $(0.38, 0.80)$.

Difficulty in interpolating across faces

© Wolfgang Heidrich

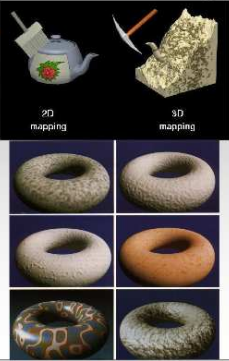


Volumetric (3D) Texture


Define texture pattern over 3D domain - 3D space containing the object

- Texture function can be **sampled**
 - 3D table of texels
- Or **procedural**
 - A function describes the color at each point
 - Implemented in special shading language

Common for natural material/irregular textures (stone, wood, etc...)



© Wolfgang Heidrich




Procedural Textures

Generate "image" on the fly, instead of loading from disk


- Also called **shader**
- Often saves space
- Allows arbitrary level of detail
 - "magnification" not an issue
 - "minification" less so than for sampled representation
- But can be quite slow for complicated shaders

© Wolfgang Heidrich




Volumetric Bump Mapping


Marble



Bump



© Wolfgang Heidrich




Volumetric Texture Mapping

In Hardware:

- Sampled 3D textures supported very much analogously to 2D textures:
 - `glTexCoord3f, glTexImage3f...`
- Procedural textures supported with modern GPUs
 - More in upcoming lectures

© Wolfgang Heidrich



Coming Up...

Thursday:

- Sampling
- A2 due...

Tuesday:

- Quiz 2

© Wolfgang Heidrich