Picking (cont)
Texture Mapping

Wolfgang Heidrich

Course News

Assignment 2

- Due today!

Assignment 3

- Project
- Handout will be up on Wendesday

Reading

- Chapter 11 (Texture Mapping)

OpenGL Picking

"Render" image in picking mode

- Pixels are never written to framebuffer
- Only store IDs of objects that would have been drawn

Procedure

- Set unique ID for each pickable object
- Call the regular sequence of glBegin/glVertex/glEnd commands
  - *If possible, skip glColor, glNormal, glTexCoord etc. for performance*

Select/Hit

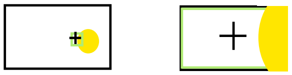
OpenGL support

- Use small region around cursor for viewport
- Assign per-object integer keys (names)
- Redraw in special mode
- Store hit list of objects in region
- Examine hit list

Viewport

Small rectangle around cursor

- Change coord sys so fills viewport



Why rectangle instead of point?

- People aren't great at positioning mouse
  - *Fitts's Law: time to acquire a target is function of the distance to and size of the target*
- Allow several pixels of slop

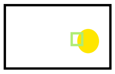
Viewport

Tricky to compute

- Invert viewport matrix, set up new orthogonal projection

Simple utility command

- gluPickMatrix(x,y,w,h,viewport)
  - *x,y: cursor point*
  - *w,h: sensitivity/slop (in pixels)*
- Push old setup first, so can pop it later



Render Modes

glRenderMode(mode)

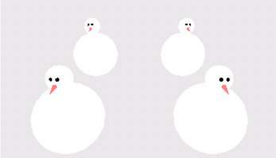
- GL_RENDER: normal color buffer
  - *default*
- GL_SELECT: selection mode for picking
- (GL_FEEDBACK: report objects drawn)

Name Stack

- “names” are just integers
  glInitNames()
- flat list
  glLoadName(name)
- or hierarchy supported by stack
  glPushName(name), glPopName
  - *Can have multiple names per object*
  - *Helpful for identifying objects in a hierarchy*

Hierarchical Names Example

```
for(int i = 0; i < 2; i++) {
  glPushName(i);
  for(int j = 0; j < 2; j++) {
    glPushMatrix();
    glPushName(j);
    glTranslatef(i*10.0,0,j * 10.0);
      glPushName(HEAD);
      glCallList(snowManHeadDL);
      glLoadName(BODY);
      glCallList(snowManBodyDL);
      glPopName();
    glPopName();
    glPopMatrix();
  }
  glPopName();
}
```



http://www.lighthouse3d.com/opengl/picking/

Hit List

- glSelectBuffer(int buffersize, GLuint *buffer)
  - *Where to store hit list data*
- If object overlaps with pick region, create hit record
- Hit record
  - *Number of names on stack*
  - *Minimum and maximum depth of object vertices*
    - Depth lies in the z-buffer range [0,1]
    - Multiplied by 2^32 -1 then rounded to nearest int
  - *Contents of name stack (bottom entry first)*

Using OpenGL Picking

Example code:

```
int numHitEntries;
GLuint buffer[1000];
glSelectBuffer( 1000, buffer );
glRenderMode( GL_SELECT );
drawStuff(); // includes name stack calls
numHitEntries= glRenderMode( GL_RENDER );
// now analyze numHitEntries different hit records
// in the selection buffer
...
```

Integrated vs. Separate Pick Function

Integrate: use same function to draw and pick

- Simpler to code
- Name stack commands ignored in render mode

Separate: customize functions for each

- Potentially more efficient
- Can avoid drawing unpickable objects

Select/Hit

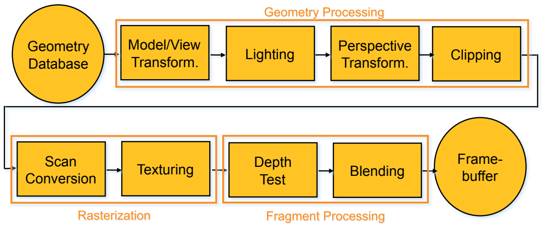
Advantages

- Faster
  - *OpenGL support means hardware acceleration*
  - *Only do clipping work, no shading or rasterization*
- Flexible precision
  - *Size of region controllable*
- Flexible architecture
  - *Custom code possible, e.g. guaranteed frame rate*

Disadvantages

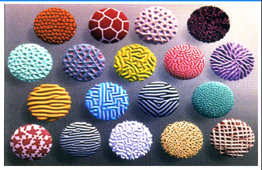
- More complex

The Rendering Pipeline



Texture Mapping

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



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

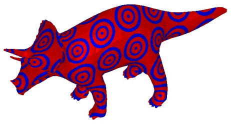
- Point on surface corresponds to a point in texture
- “Paint” image onto polygon

Color Texture Mapping

Define color (RGB) for each point on object surface

Two approaches

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



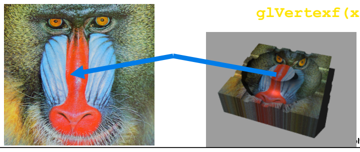
Surface (2D) Textures: Texture Coordinates

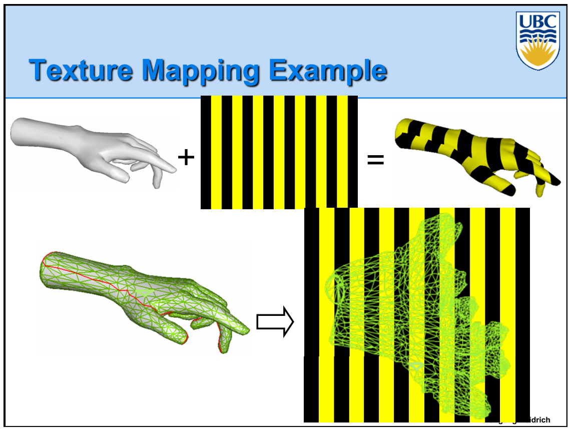
Texture map: 2D array of color (texels)

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

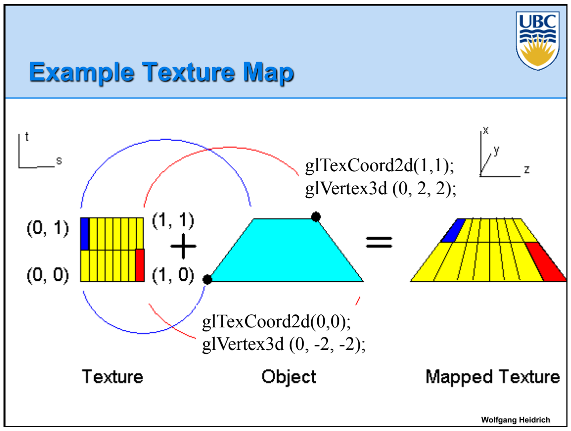
- Use interpolated (s,t) for texel lookup at each pixel
- Use value to modify a polygon’s color
- Specified by programmer or artist

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





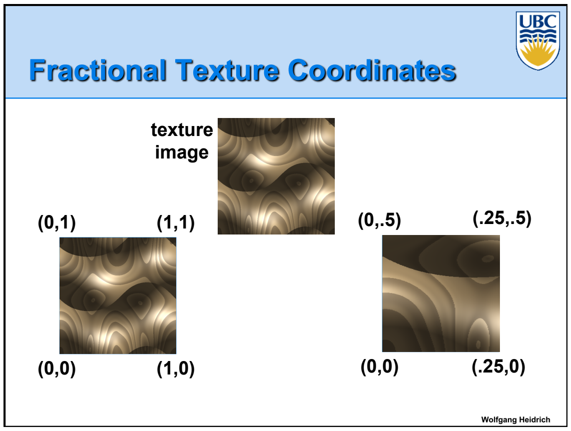
Texture Mapping Example



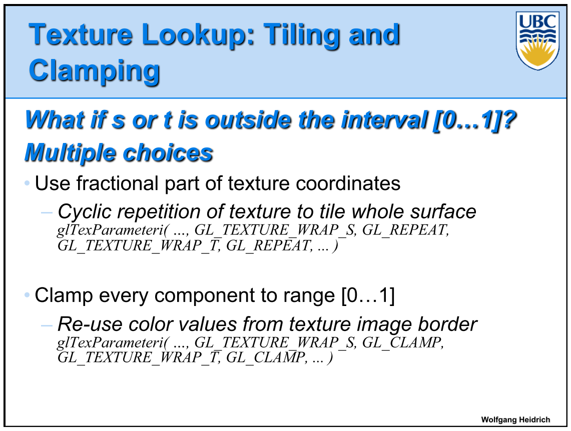
Example Texture Map

glTexCoord2d(1,1);
glVertex3d (0, 2, 2);

glTexCoord2d(0,0);
glVertex3d (0, -2, -2);



Fractional Texture Coordinates



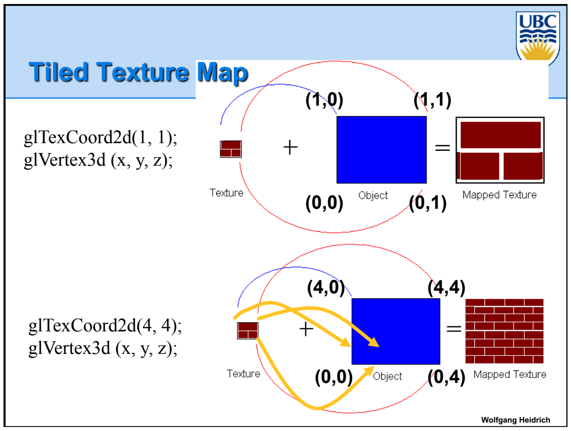
Texture Lookup: Tiling and Clamping

What if s or t is outside the interval [0…1]?

Multiple choices

- Use fractional part of texture coordinates
  - *Cyclic repetition of texture to tile whole surface*
    glTexParameteri(…, GL_TEXTURE_WRAP_S, GL_REPEAT, GL_TEXTURE_WRAP_T, GL_REPEAT, ...)
- Clamp every component to range [0…1]
  - *Re-use color values from texture image border*
    glTexParameteri(…, GL_TEXTURE_WRAP_S, GL_CLAMP, GL_TEXTURE_WRAP_T, GL_CLAMP, ...)

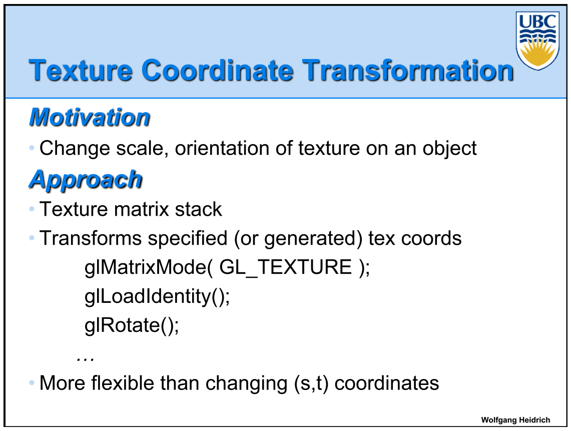
Wolfgang Heidrich



Tiled Texture Map

glTexCoord2d(1, 1);
glVertex3d (x, y, z);

glTexCoord2d(4, 4);
glVertex3d (x, y, z);



Texture Coordinate Transformation

Motivation

- Change scale, orientation of texture on an object

Approach

- Texture matrix stack
- Transforms specified (or generated) tex coords

```
    glMatrixMode( GL_TEXTURE );
    glLoadIdentity();
    glRotate();
  …
```

- More flexible than changing (s,t) coordinates

Wolfgang Heidrich

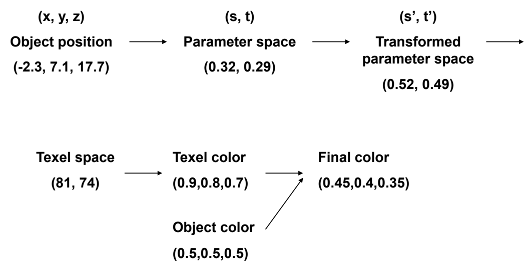
Texture Functions

Given value from the texture map, we 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*

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 w/ 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*
- glTexParameteri(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 OpenGLTexturing (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

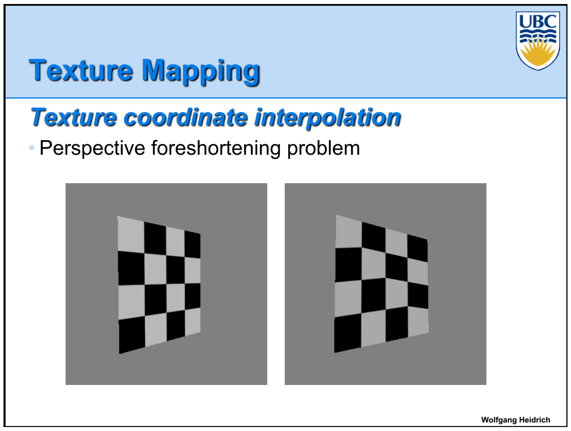
Large range of functions for controlling layout of texture data

- 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

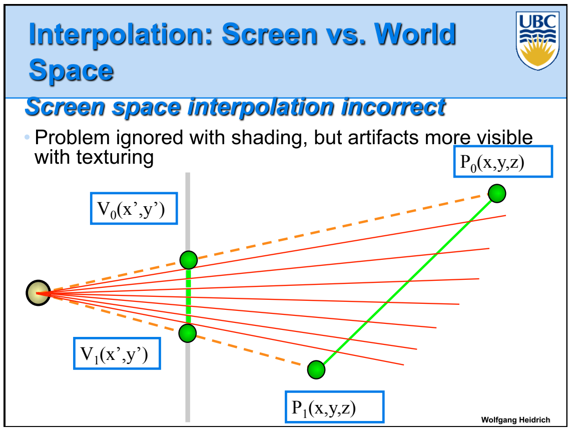
Wolfgang Heidrich



Texture Mapping

Texture coordinate interpolation

- Perspective foreshortening problem

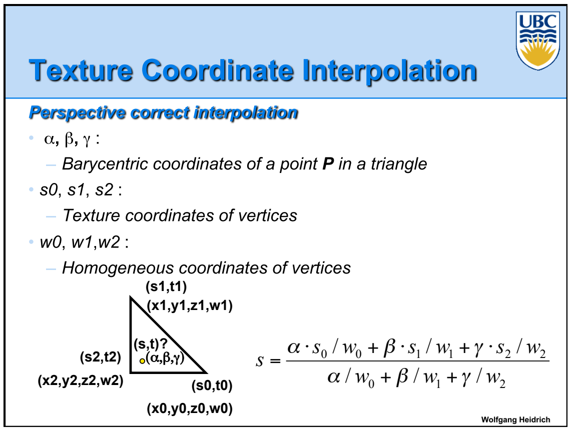


Interpolation: Screen vs. World Space

Screen space interpolation incorrect

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

$P_0(x,y,z)$
$V_0(x',y')$
$V_1(x',y')$
$P_1(x,y,z)$



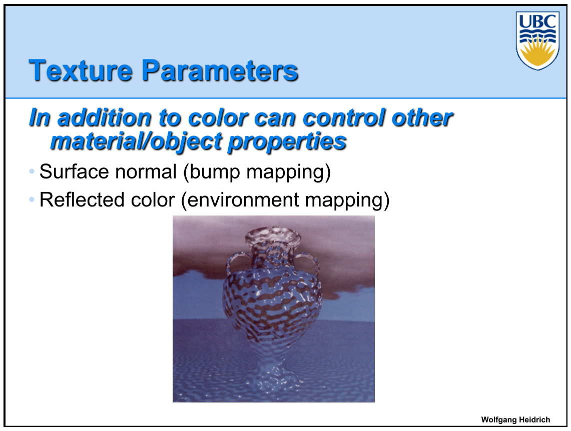
Texture Coordinate Interpolation

Perspective correct interpolation

- α, β, γ :
  - *Barycentric coordinates of a point **P** in a triangle*
- *s0, s1, s2* :
  - *Texture coordinates of vertices*
- *w0, w1,w2* :
  - *Homogeneous coordinates of vertices*

(s1,t1)
(x1,y1,z1,w1)
(s,t)?
(α,β,γ)
(s2,t2)
(x2,y2,z2,w2)
(s0,t0)
(x0,y0,z0,w0)

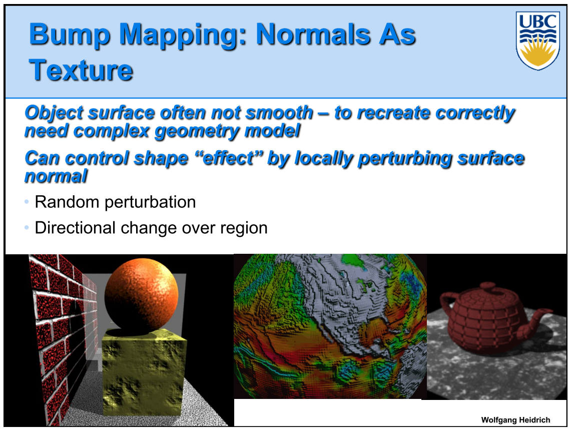
$$s = \frac{\alpha \cdot s_0 / w_0 + \beta \cdot s_1 / w_1 + \gamma \cdot s_2 / w_2}{\alpha / w_0 + \beta / w_1 + \gamma / w_2}$$



Texture Parameters

In addition to color can control other material/object properties

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

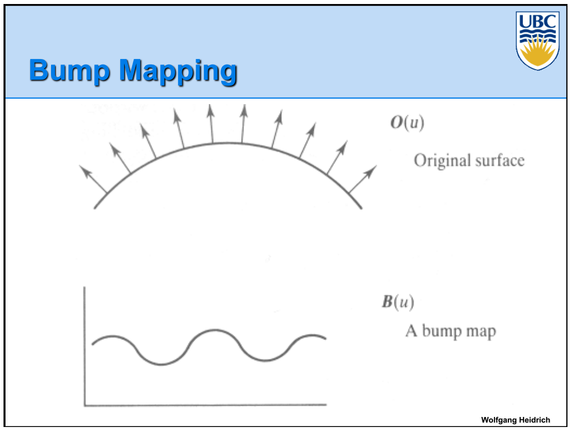


Bump Mapping: Normals As Texture

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

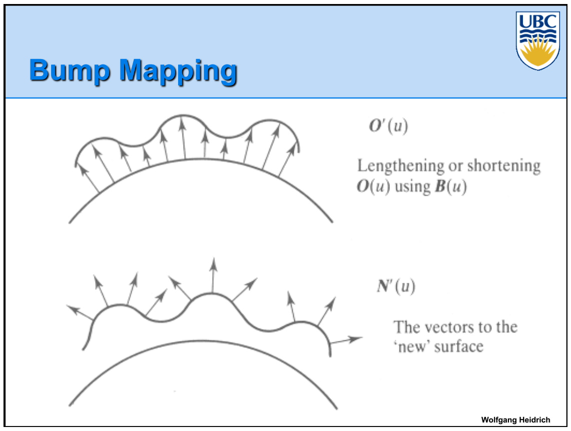
Can control shape "effect" by locally perturbing surface normal

- Random perturbation
- Directional change over region



Bump Mapping

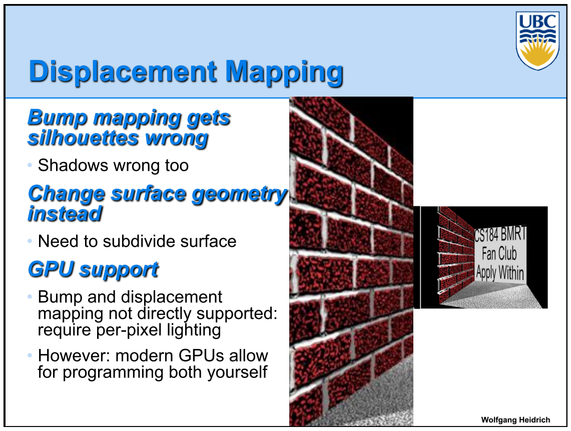
O(u)
Original surface
B(u)
A bump map



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

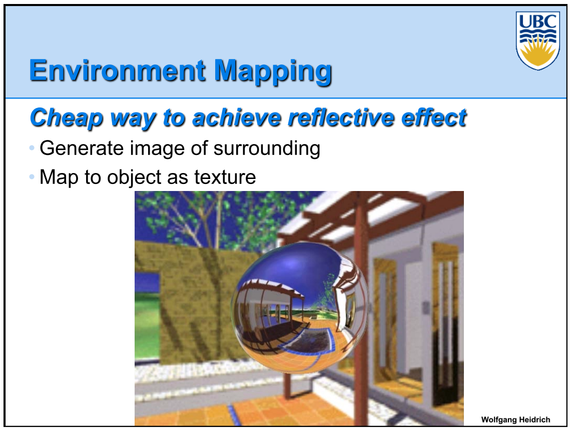
Change surface geometry instead

- 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

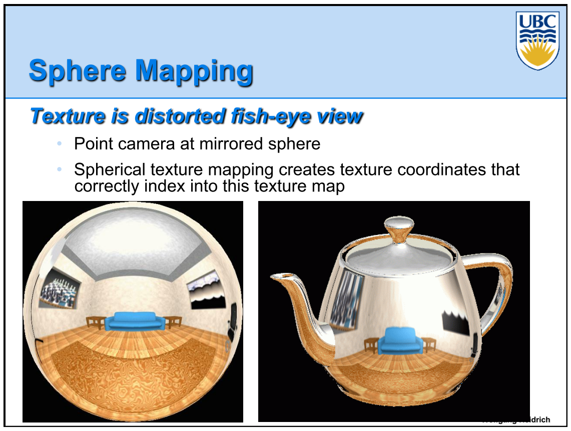


Environment Mapping

Cheap way to achieve reflective effect

- Generate image of surrounding
- Map to object as texture

Wolfgang Heidrich

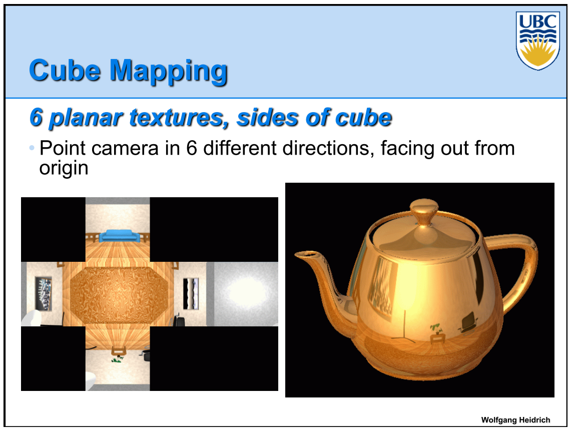


Sphere Mapping

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

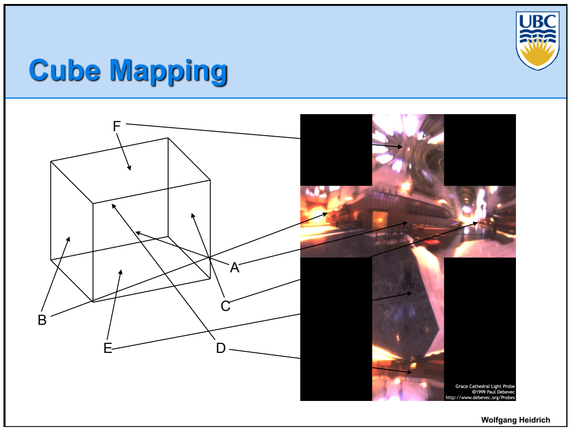


Cube Mapping

6 planar textures, sides of cube

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

Wolfgang Heidrich



Cube Mapping

Grace Cathedral Light Probe
©1999 Paul Debevec
http://www.debevec.org/Probes

Wolfgang Heidrich

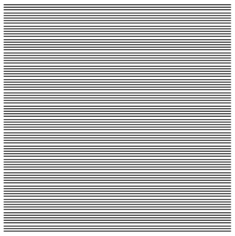
Cube Mapping

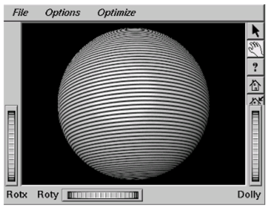
Direction of reflection vector r selects the face of the cube to be indexed

- 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)*
    - Why?

Difficulty in interpolating across faces

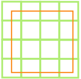
Texture Lookup – Sampling & Reconstruction



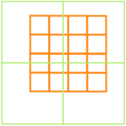


Texture Lookup – Sampling & Reconstruction

- How to deal with:
  - *Pixels that are much larger than texels?*
    - Apply filtering, “averaging”
    - “Minification”



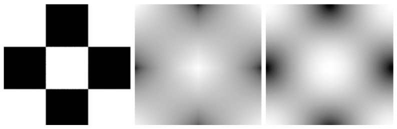
  - *Pixels that are much smaller than texels ?*
    - Interpolate
    - “Magnification”



Magnification: Interpolating Textures

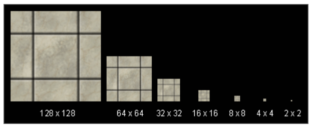
- Nearest neighbor
- Bilinear
- Hermite (cubic)

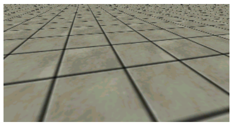
1	0	1
0	1	0
1	0	1



Minification: MIPmapping

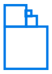
use “image pyramid” to precompute averaged versions of the texture

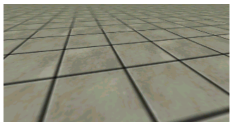




Without MIP-mapping

store whole pyramid in single block of memory





With MIP-mapping

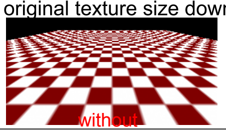
MIPmaps

Multum in parvo

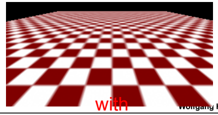
- “many things in a small place”
- Series of prefiltered texture maps of decreasing resolutions
- Avoid shimmering and flashing as objects move

gluBuild2DMipmaps

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



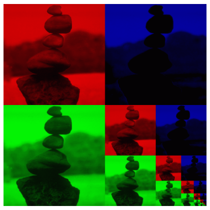
without



with

MIPmap storage

Only 1/3 more space required



Coming Up:

Wednesday / Friday

- More texture mapping
- Sampling & reconstruction