Texture Mapping and Sampling
The Rendering Pipeline

- Geometry Processing
- Model/View Transform.
- Perspective Transform.
- Clipping
- Texturing
- Blending
- Depth Test
- Rasterization
- Scan Conversion
- Frame-buffer
- Geometry Database

© W. Heldrich and M. van de Panne
Texture Mapping

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

W. Heidrich and M. van de Panne
Texture Mapping

- "Paint" image onto polygon
- Point on surface corresponds to a point in texture
- Associate 2D information with 3D surface
- Create bumpy effect on surface
- Map a brick wall texture on a flat polygon
- Images convey illusion of geometry
- Hide geometric simplicity
- Lighting/shading models not enough
- Introduced to increase realism

© W. Heidrich and M. van de Panne
Color Texture Mapping

Two approaches

- Volume texture
- Surface texture map

Define color (RGB) for each pt on surface
Assigning texture coordinates \((x', y', z', w)\) at vertex with object coordinates \((x, y, z, w)\)

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

Specified by programmer or artist

Or other surface property

- Use value to modify a polygon's color
- Use interpolated \((s, t)\) for texel lookup at each pixel
Textures of other dimensions

1D: Represent isovalue 1D: Represent Isovalues

e.g.: Contour lines, temp

\( \text{GLuint} \) texcoord1f(s)
Texture Mapping Example
Example Texture Map
Fractional Texture Coordinates

Image
Texture

(0, 0) (0, 0) (0, 0) (0, 0)
(0, 0.5) (0.25, 0.5) (0.75, 0.5) (1, 0.5)
(0.25, 0) (0.25, 1) (0.75, 0) (0.75, 1)
Texture Tiling and Texture Lookup: Tiling and Clamping

- Image border
- Re-use color values from texture
- Clamp every component to range [0...1]
- GL_REPEAT
- GL_REPEAT, GL_REPEAT, GL_REPEAT, GL_REPEAT, GL_REPEAT, GL_REPEAT, GL_REPEAT
- Cylindrical repetition
- Use fractional part of texture coordinates
- Multiple choices [0...1]
- What if s or t is outside the interval

Clamping

© W. Hedditch and M. van de Panne
glTexCoord2d(0, 0);
vertex3d (x, y, z);

Tiled Texture Map

glTexCoord2d(4, 0);
vertex3d (x, y, z);

Tiled Texture Map

glTexCoord2d(0, 1);
vertex3d (x, y, z);

Tiled Texture Map

glTexCoord2d(1, 1);
vertex3d (x, y, z);

Tiled Texture Map
Texture Coordinate Transformation

Motivation
• More flexible than changing (s,t) coordinates

... 

glLoadIdentity();

• More flexible than changing (s,t) coordinates

Approach

• Change scale, orientation of texture on an object

• Transforms specified (or generated) tex cords

• Texture matrix stack

Texture Coordinate Transformation

© W. Heidrich and M. van de Panne

Heidrich and M. van de Panne

W. ©

© Heidrich and M. van de Panne

W.
Texture Functions

Ways of applying texture colour:

- GL_REPLACE
  - Directly use as surface colour; no lighting effects
- GL_MODULATE
  - Modulate surface colour: multiply old color by new value
  - Like replace, but modulate alpha to support transparency
- GL_DECAL
  - Like replace, but modulate alpha to support transparency
  - Texturing happens after lighting, not relit
- GL_BLEND
  - Blend surface color with existing on-screen colour
Texture Pipeline

(s', t')

Transformed parameter space

(0.52, 0.49)

Object color (0.5, 0.5, 0.5)

(0.32, 0.29)

Texel space

(0.9, 0.8, 0.7)

Texel color

(0.45, 0.4, 0.35)

Final color

(0.5, 0.5, 0.5)

Object color
Texture Objects and Binding

Switch between preloaded textures
Which texture to use right now

Texture binding assigned
OpenGL uses least recently used (LRU) if no priority is assigned

You can prioritize textures to keep in memory
Reload a texture
Provides efficiency gains over having to repeatedly load and
and provides identitiers to easily access them
An OpenGL data type that keeps textures resident in memory

Texture Object
Create a texture object and fill it with texture data:

- `glTexImage2D(GL_TEXTURE_2D, ...)` to specify the texture data (the image itself)
- `glTexParameteri(GL_TEXTURE_2D, ...)` to specify parameters for use when applying the texture
- `glTexParameterf(GL_TEXTURE_2D, ...)` to specify parameters for use when applying the texture
- Following texture commands refer to the bound texture
  - `bindTexture(GL_TEXTURE_2D, identifier)` to bind the texture
  - `getTexImage()` to get identifiers for the objects
- Create a texture object and fill it with texture data.
Enable texturing:

* Use `glTexCoord2d(\(s', t\))` before each vertex:
  - Specify texture coordinates for the polygon:
    * `glTexCoord2d(0, 0)`
    * `glTexCoord2d(1, 0)`
    * `glTexCoord2d(0, 1)`
    * `glTexCoord2d(1, 1)`
* State how the texture will be used:
  * `glEnable(GL_TEXTURE_2D)`
  * `glEnable(GL_TEXTURE_3D)`
  * `glVertex3f(x, y, z)`

Basic OpenGL Texturing (cont.)
Textures must have a size of power of 2...many bits per "pixel", which channels...you must state how you want the texture to be put in memory: how OpenGL not to skip bytes at the end of a row

state how the data in your image is arranged

data

Large range of functions for controlling layout of texture

•

•

•

•

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...
• Texture coordinate interpolation

• Perspective foreshortening problem

Texture Mapping
Screen space interpolation incorrect

Problem ignored with shading, but artifacts more visible with texturing.
Texture Coordinate Interpolation

\[
\frac{z_w / w + l w / g + o w / k}{z_w / w \cdot l + l w / s \cdot g + o w / o \cdot k} = s
\]

Homogeneous coordinates of vertices

- \((x_0, y_0, z_0, w_0)\)
- \((x_1, y_1, z_1, w_1)\)
- \((x_2, y_2, z_2, w_2)\)

Texture coordinates of vertices

- \((s_0, t_0)\)
- \((s_1, t_1)\)
- \((s_2, t_2)\)

Barycentric coordinates of a point \(p\) in a triangle

- \(a, b, c\)

Perspective Correct Interpolation
Reconstruction

(image courtesy of Kiriakos Kutulakos, U Rochester)
Reconstruction

• How to deal with:

  - Interpolate pixels that are much smaller than texels?
  - Apply filtering, "averaging" pixels that are much larger than texels?

© W. Heidrich and M. van de Panne
Interpolating Textures

- Nearest neighbor
- Bilinear
- Hermite
With MIP mapping

Without MIP mapping

single block of memory

store whole pyramid in

averaged versions of the texture

use "Image pyramid" to precompute

MIP mapping
MIPmaps

Multum in parvo -- many things in a small place

Avoid shimmering and flashing as objects move

Requires more texture storage

Prespecify a series of prefiltered texture maps of decreasing resolution

Automatically constructs a family of textures from original

Texture size down to 1x1 without

With MIPmaps

With
MIPmap storage

only 1/3 more space required
In addition to color can control other material/object properties.

- Reflected color (environment mapping)
- Surface normal (bump mapping)
- Texture parameters
Bump Mapping: Normals

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

Can control shape “effect” by locally perturbing surface

Directional change over region

Random perturbation

© W. Heidrich and M. van de Panne
A bump map

Original surface

Bump Mapping

\( (n)B \)

\( (n)O \)
new surface
The vectors to the
(n)\textbf{N}

Lenghening or shortening
(n)\textbf{B} using (n)\textbf{O}

Bump Mapping
Displacement Mapping

**GPU Support**

- **Surface Geometry Instead**
  - Need to subdivide

- **Change Surface**
  - Silhouettes wrong
  - Shadows wrong too

- Bump mapping gets

However: modern GPUs allow for programmable pixel lighting. However, modern GPUs require per-pixel lighting support.

Bump and displacement mapping not directly support GPU subdivision.

Heidrich and M. van de Panne © UBC
Environment Mapping

- Cheap way to achieve reflective effect
- Generate image of surrounding
- Map to object as texture

© W. Heidrich and M. van de Panne
Texture is distorted fish-eye view

Sphere Mapping

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

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

6 planar textures, sides of cube
Cube Mapping
Difficulty in Interpolating across Faces

- e.g., \((-0.2, 0.5, 0.84)\) selects the Z face.
- e.g., \((-0.2, 0.5, 0.38)\) gets mapped to \((0.38, 0.80)\).
- e.g., \((-0.2, 0.5)\) selects the pixel from the face.
- Remaining two coordinates (normalized by the 3rd coordinate) selects the face.

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

Co-ordinate with largest magnitude selects the face of the cube to be indexed.
Volumetric Texture

Volumetric Texture

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

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

Locate texture pattern over 3D domain containing the object

Locate texture pattern over 3D domain containing the object

Texture function can be digitized or procedural

Texture function can be digitized or procedural

For each point on object compute texture from point location in space

For each point on object compute texture from point location in space

Texture pattern over 3D domain - 3D space

Texture pattern over 3D domain - 3D space

Heidrich and M. van de Panne
Procedural Textures

- Allows arbitrary level of detail
- Often saves space
- Loading from disk
- Generate “image” on the fly, instead of

© W. Heidrich and M. van de Panne