Texture Mapping

Wolfgang Heidrich

Wolfgang Heidrich

Course News

Assignment 3

- Project
- Handout will be up on Wednesday

Homework 5

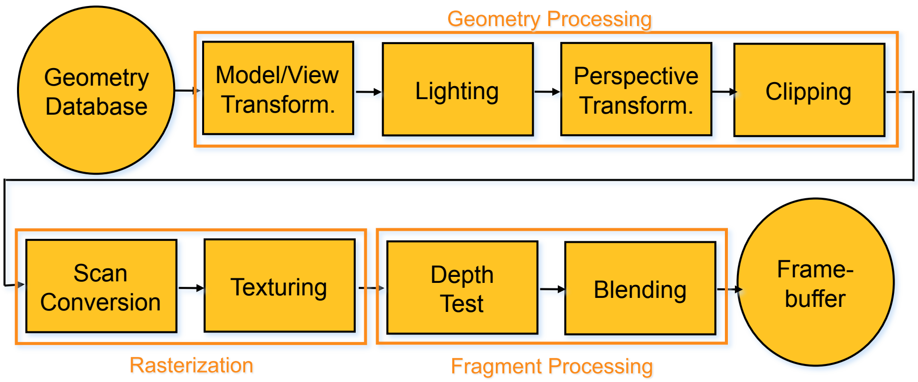
- Out later today (this time for real)
- Remember that these are good practice for the exams!

Reading

- Chapter 11 (Texture Mapping)

Wolfgang Heidrich

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

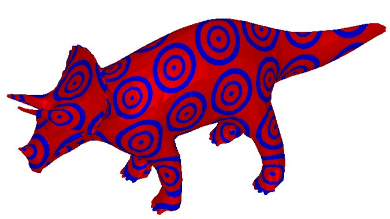
Wolfgang Heidrich

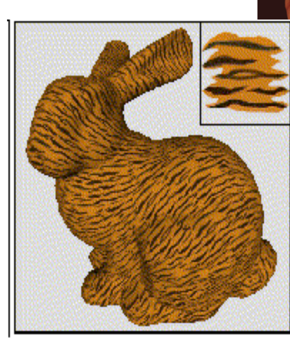
Color Texture Mapping

Define color (RGB) for each point on object surface

Two approaches

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





Wolfgang Heidrich

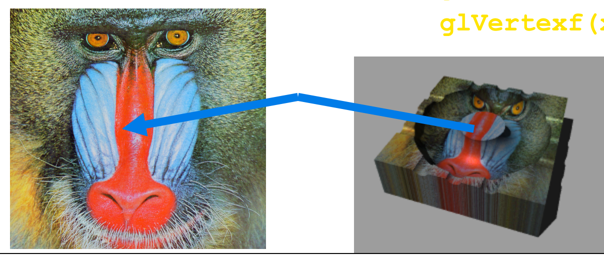
Surface (2D) Textures: Texture Coordinates

Texture map: 2D array of color (texels)

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

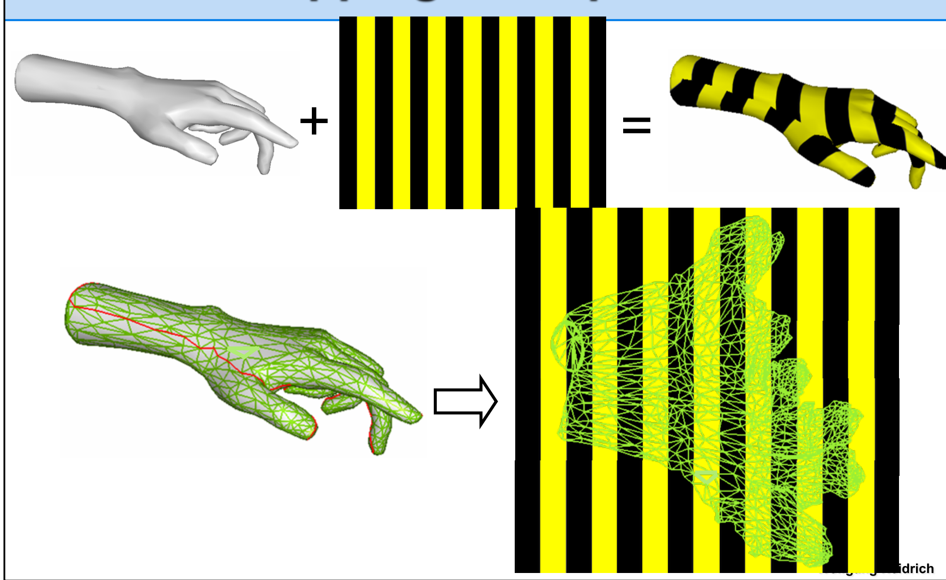
- Use interpolated (s,t) for texel lookup at each pixel
- Use value to modify a polygon's color
- Specified by programmer or artist

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

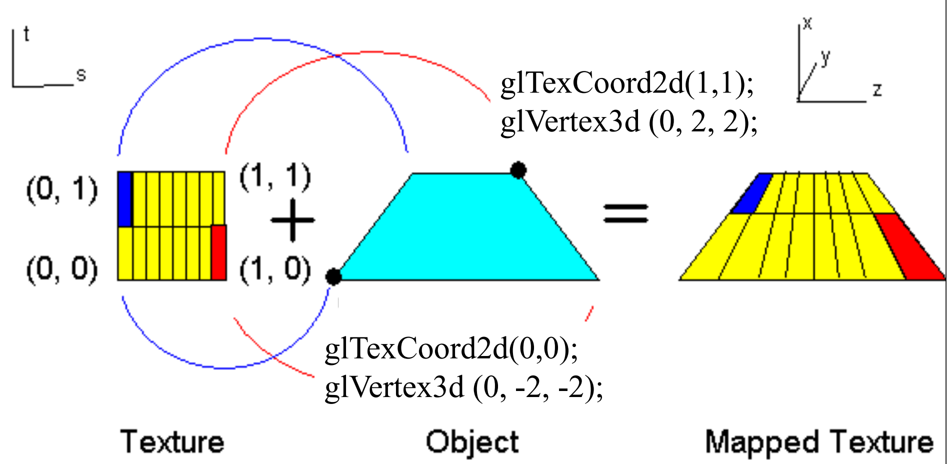


lfgang Heidrich

Texture Mapping Example

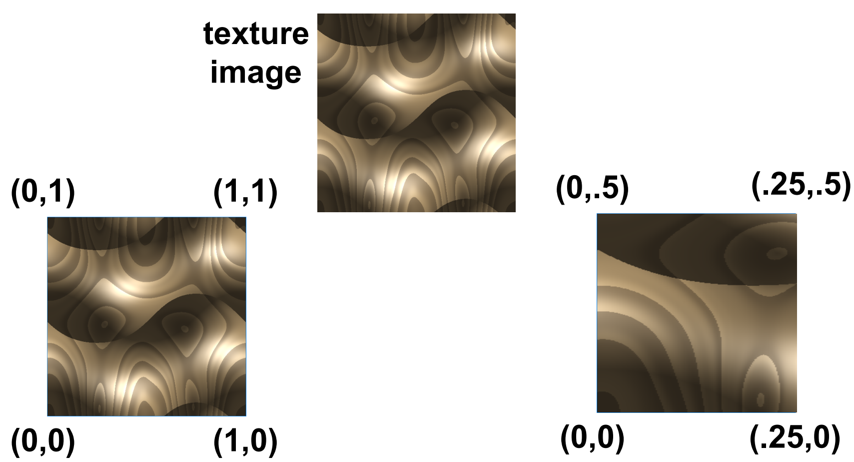


Example Texture Map



Wolfgang Heidrich

Fractional Texture Coordinates



Wolfgang Heidrich

Texture Lookup: Tiling and Clamping

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

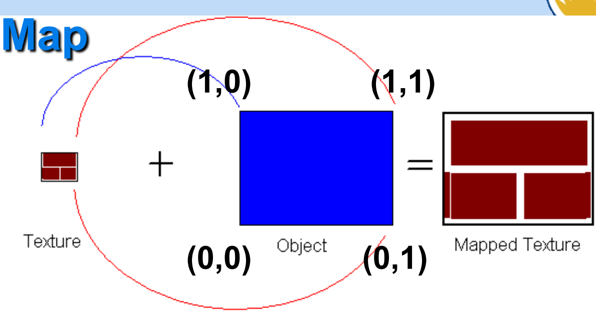
Multiple choices

- 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, ...)

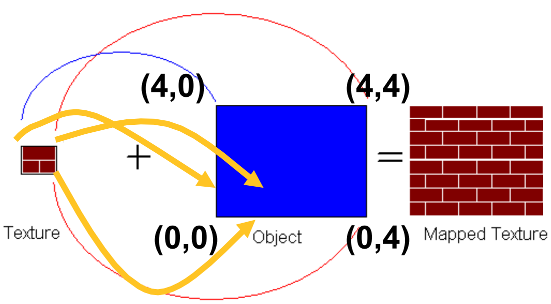
Wolfgang Heidrich

Tiled Texture Map

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



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



Wolfgang Heidrich

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

(x, y, z) Object position (-2.3, 7.1, 17.7) → (s, t) Parameter space (0.32, 0.29) → (s', t') Transformed parameter space (0.52, 0.49) →

Texel space (81, 74) → Texel color (0.9,0.8,0.7) → Final color (0.45,0.4,0.35)

Object color (0.5,0.5,0.5) → Final color

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

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

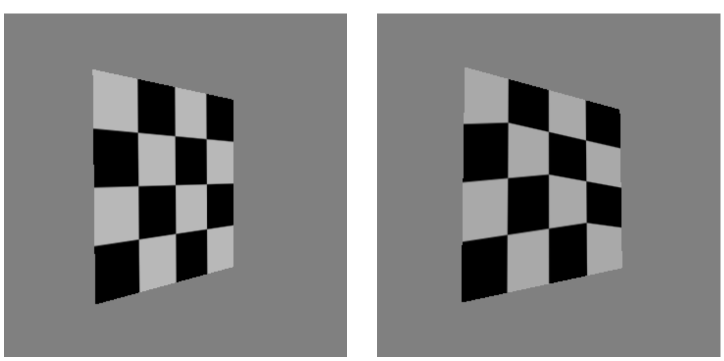
- Common sizes are 32x32, 64x64, 256x256
- But don’t need to be square, i.e. 32x64 is fine
- Smaller uses less memory, and there is a finite amount of texture memory on graphics cards

Wolfgang Heidrich

Texture Mapping

Texture coordinate interpolation

- Perspective foreshortening problem

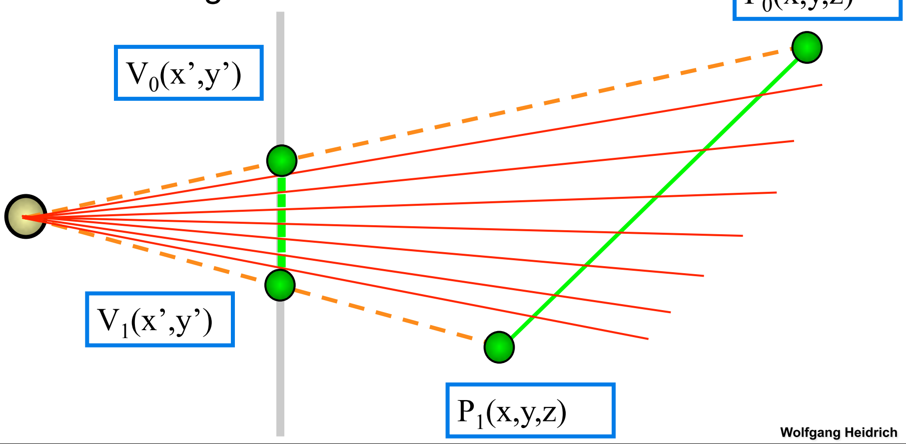


Wolfgang Heidrich

Interpolation: Screen vs. World Space

Screen space interpolation incorrect

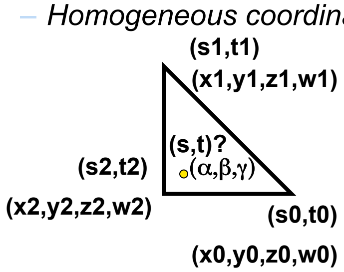
- Problem ignored with shading, but artifacts more visible with texturing



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*

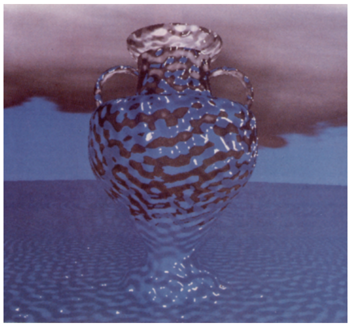


$$s = \frac{\alpha \cdot s_0 / w_0 + \beta \cdot s_1 / w_1 + \gamma \cdot s_2 / w_2}{\alpha / w_0 + \beta / w_1 + \gamma / w_2}$$

Texture Parameters

In addition to color can control other material/object properties

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



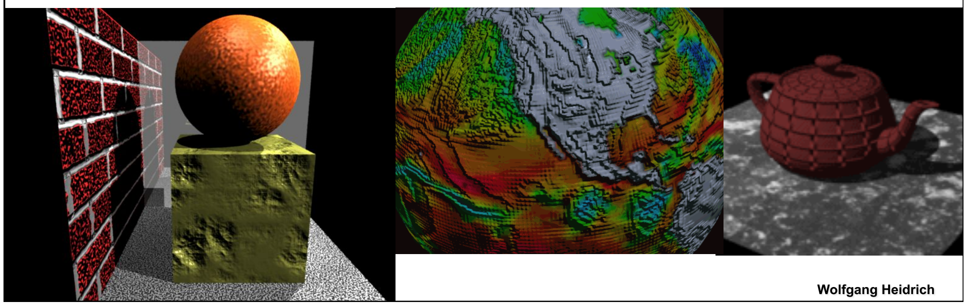
Wolfgang Heidrich

Bump Mapping: Normals As Texture

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

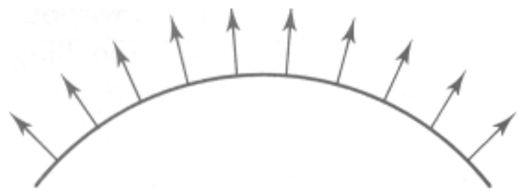
Can control shape "effect" by locally perturbing surface normal

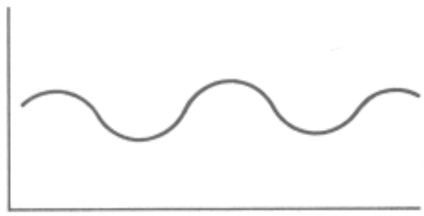
- Random perturbation
- Directional change over region



Wolfgang Heidrich

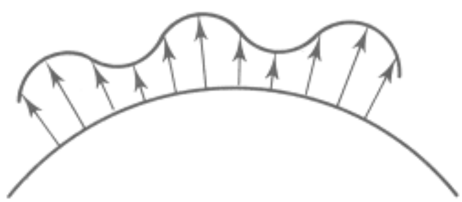
Bump Mapping

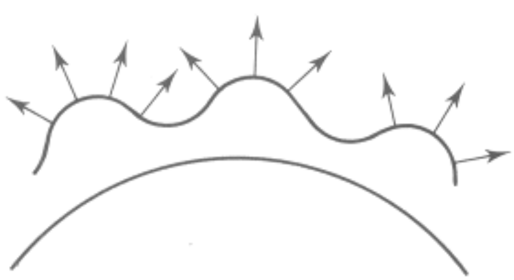




Wolfgang Heidrich

Bump Mapping





Wolfgang Heidrich

Displacement Mapping

Bump mapping gets silhouettes wrong

- Shadows wrong too

Change surface geometry instead

- Need to subdivide surface

GPU support

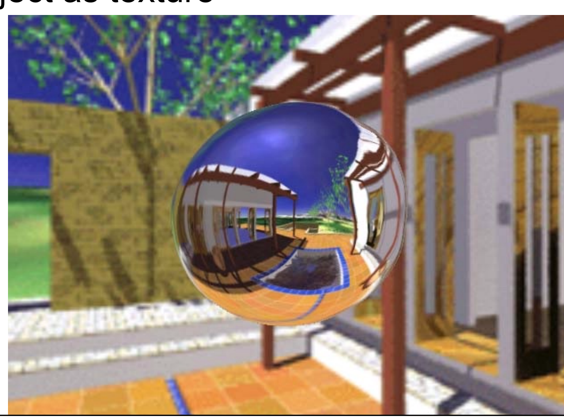
- Bump and displacement mapping not directly supported: require per-pixel lighting
- However: modern GPUs allow for programming both yourself

Wolfgang Heidrich

Environment Mapping

Cheap way to achieve reflective effect

- Generate image of surrounding
- Map to object as texture

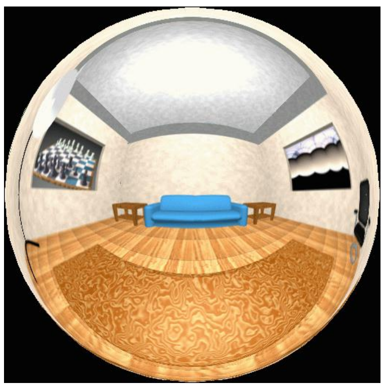


Wolfgang Heidrich

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

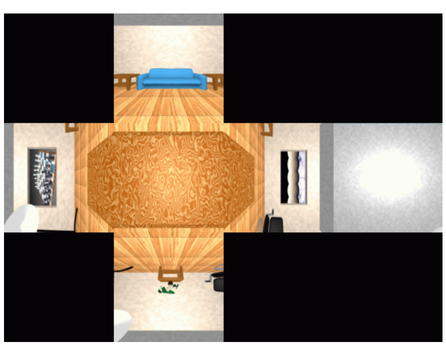


Wolfgang Heidrich

Cube Mapping

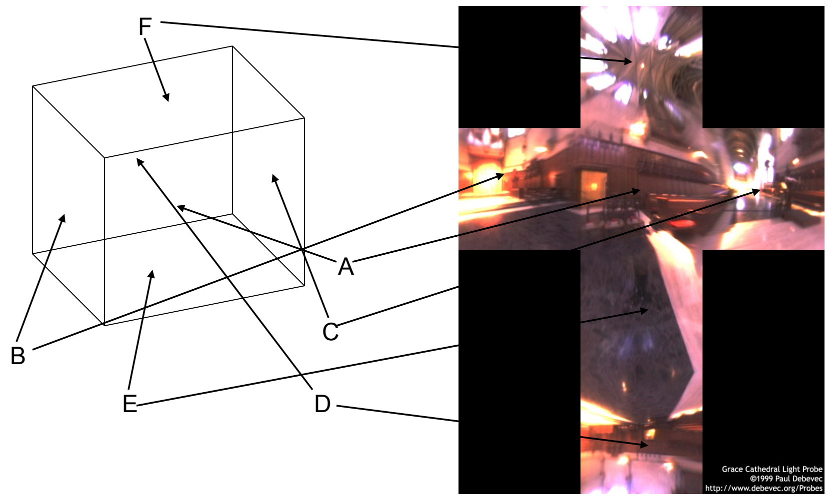
6 planar textures, sides of cube

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



Wolfgang Heidrich

Cube Mapping



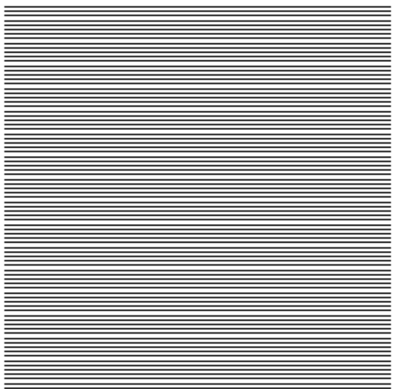
Cube Mapping

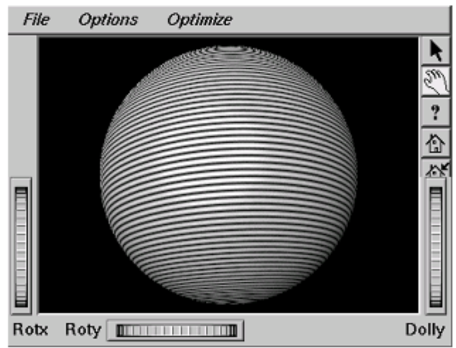
Direction of reflection vector r selects the face of the cube to be indexed

- 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

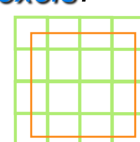




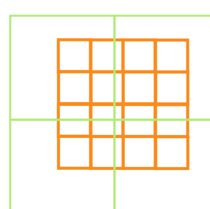
Wolfgang Heidrich

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”

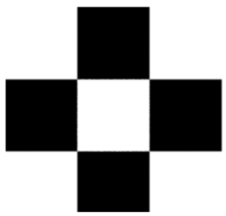
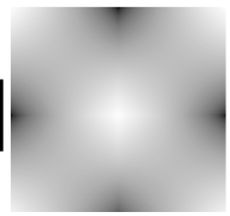


Wolfgang Heidrich

Magnification: Interpolating Textures

- Nearest neighbor
- Bilinear
- Hermite (cubic)

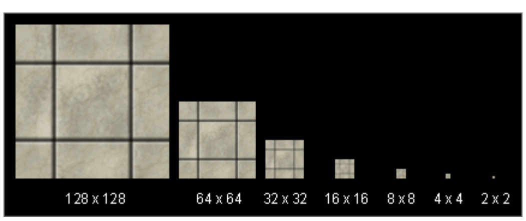
1	0	1
0	1	0
1	0	1

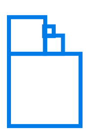

Wolfgang Heidrich

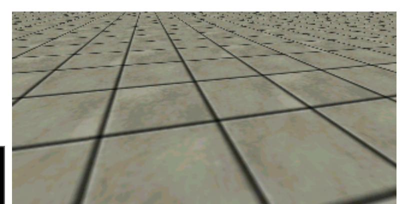
Minification: MIPmapping

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

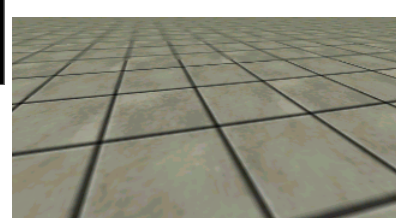


store whole pyramid in single block of memory





Without MIP-mapping



With MIP-mapping

Wolfgang Heidrich

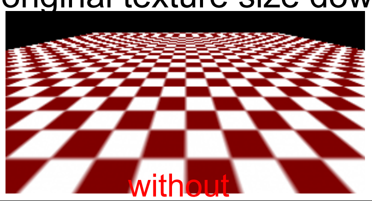
MIPmaps

Multum in parvo

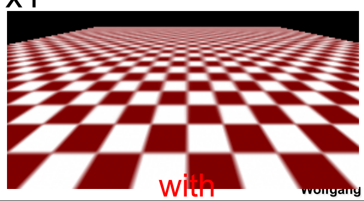
- "many things in a small place"
- Series of prefiltered texture maps of decreasing resolutions
- Avoid shimmering and flashing as objects move

gluBuild2DMipmaps

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



without

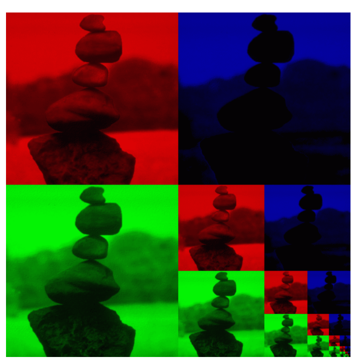


with

Wolfgang Heidrich

MIPmap storage

Only 1/3 more space required



Wolfgang Heidrich

Sampling & Reconstruction

CPSC 314

Wolfgang Heidrich

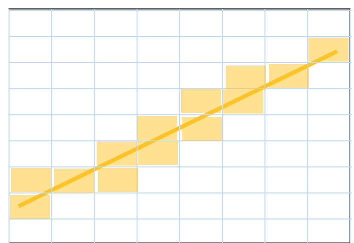
Samples

- Most things in the real world are continuous
- Everything in a computer is discrete
- The process of mapping a continuous function to a discrete one is called sampling
- The process of mapping a discrete function to a continuous one is called reconstruction
- The process of mapping a continuous variable to a discrete one is called quantization
- Rendering an image requires both sampling and quantization
- Displaying an image involves reconstruction

Wolfgang Heidrich

Line Segments

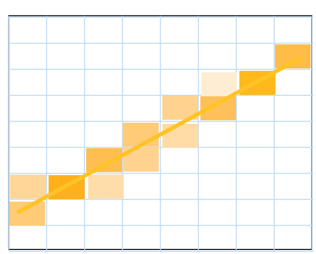
- We tried to sample a line segment so it would map to a 2D raster display
- We quantized the pixel values to 0 or 1
- We saw stair steps, or jaggies



Wolfgang Heidrich

Line Segments

- Instead, quantize to many shades
- But what sampling algorithm is used?



Wolfgang Heidrich

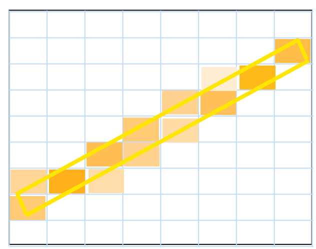
Unweighted Area Sampling

Shade pixels wrt area covered by thickened line

Equal areas cause equal intensity, regardless of distance from pixel center to area

- Rough approximation formulated by dividing each pixel into a finer grid of pixels

Primitive cannot affect intensity of pixel if it does not intersect the pixel



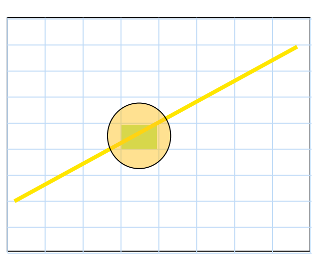
Wolfgang Heidrich

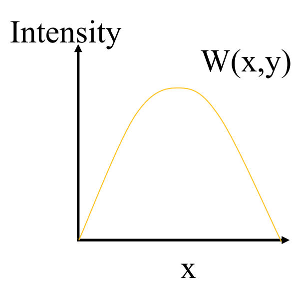
Weighted Area Sampling

Intuitively, pixel cut through the center should be more heavily weighted than one cut along corner

Weighting function, W(x,y)

- Specifies the contribution of primitive passing through the point (x, y) from pixel center





Wolfgang Heidrich

Images

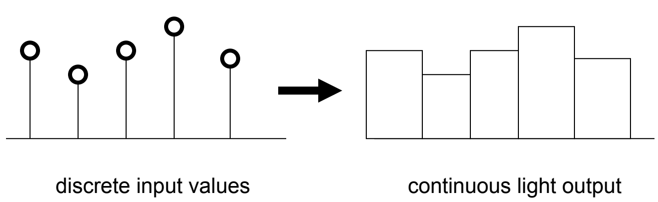
An image is a 2D function I(x, y)

- Specifies intensity for each point (x, y)
- (we consider each color channel independently)

Wolfgang Heidrich

Image Sampling and Reconstruction

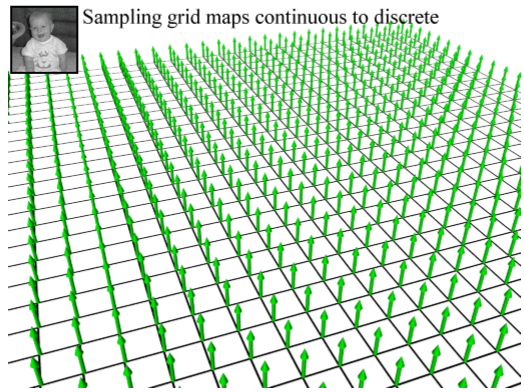
- Convert continuous image to discrete set of samples
- Display hardware reconstructs samples into continuous image
  - *Finite sized source of light for each pixel*



Wolfgang Heidrich

Point Sampling an Image

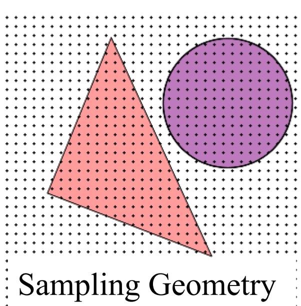
- Simplest sampling is on a grid
- Sample depends solely on value at grid points

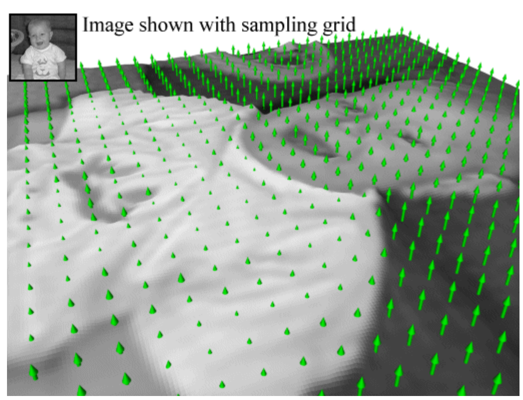


Wolfgang Heidrich

Point Sampling

Multiply sample grid by image intensity to obtain a discrete set of points, or samples.





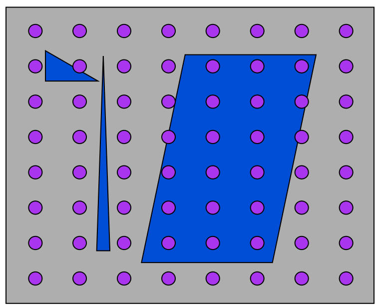
Wolfgang Heidrich

Sampling Errors

Some objects missed entirely, others poorly sampled

- Could try unweighted or weighted area sampling
- But how can we be sure we show everything?

Need to think about entire class of solutions!



Wolfgang Heidrich

Image As Signal

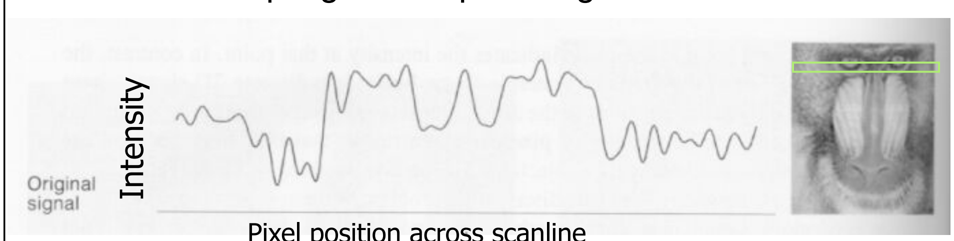
Image as spatial signal

2D raster image

- Discrete sampling of 2D spatial signal

1D slice of raster image

- Discrete sampling of 1D spatial signal



Examples from Foley, van Dam, Feiner, and Hughes

Wolfgang Heidrich

Sampling Theory

How would we generate a signal like this out of simple building blocks?

Theorem

- Any signal can be represented as an (infinite) sum of sine waves at different frequencies

Wolfgang Heidrich

Coming Up:

Friday

- Sampling & reconstruction

Wolfgang Heidrich