Johns Hopkins
Engineering for Professionals

605.767 Applied Computer Graphics

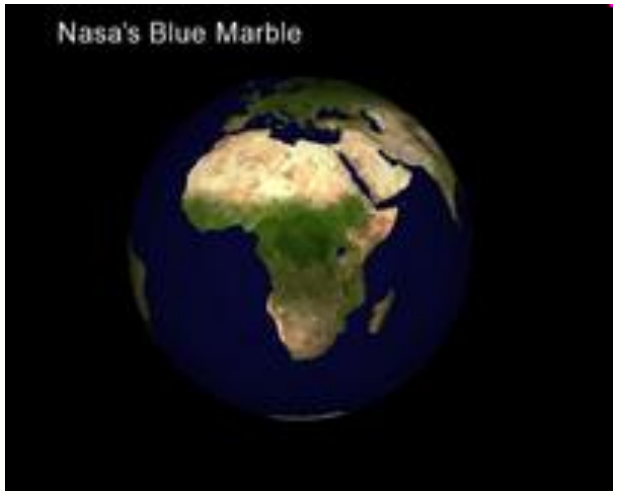
Brian Russin

Module 9A
Texture Review Part 1 - Principles

Texture Mapping

- General discussion
  - Uses of texture mapping
- Mapping texture onto objects
- Anti-aliasing
  - MipMaps
- Texture mapping in OpenGL
  - Loading textures
  - Assigning texture coordinates
    - Clamping and wrapping textures
  - Modulation and blending
  - MipMaps and filtering
- Using Textures with Shaders

Texture Mapping



https://www.openscenegraph.com/index.php/gallery

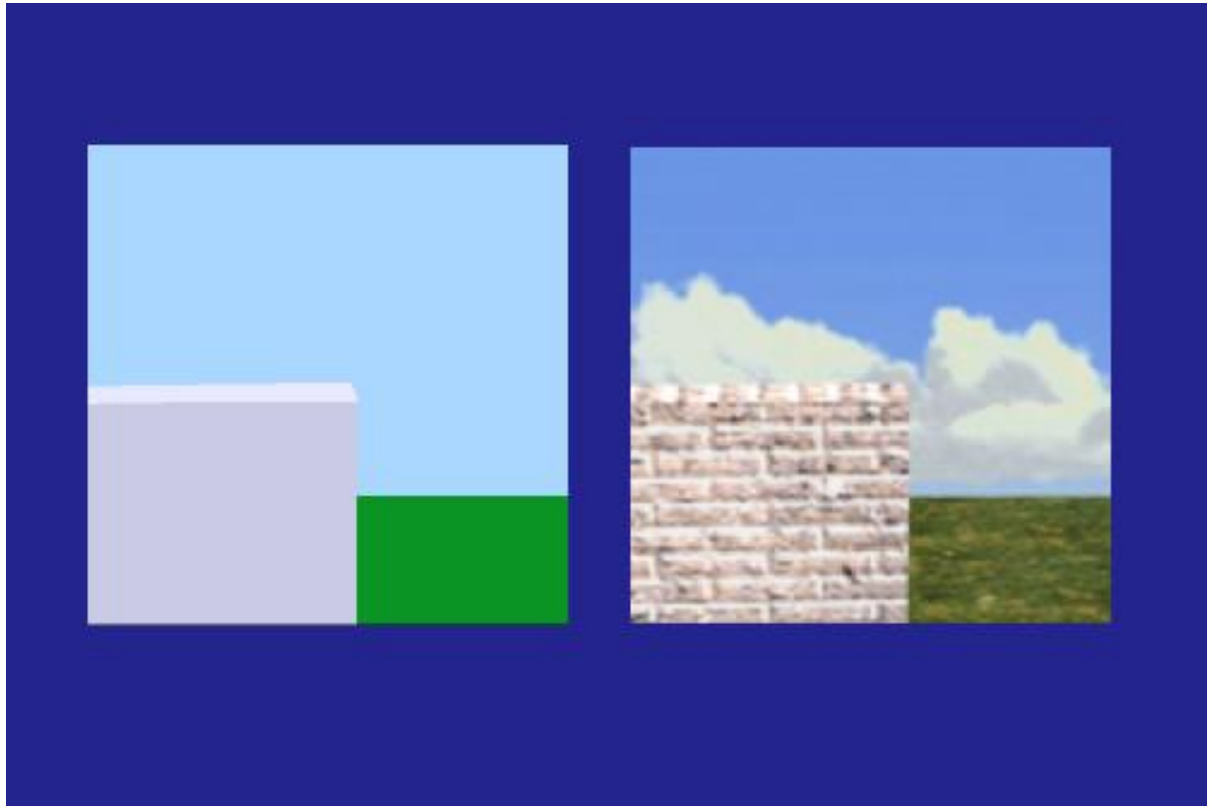
Texture Mapping

- Texture mapping maps a 2D (usually) image to a 3D object
  - Common method for adding surface detail to objects
    - Hardware acceleration in even inexpensive graphics hardware
  - Most texture methods alter surface color using an image or a procedural method
    - Enables objects to exhibit different surface properties
      - Real objects exhibit both color and surface modulation
- Considerations
  - What attributes are to be modulated to produce the desired effect?
  - Mapping between texture domain (usually 2D) and object space (3D)
    - Usually left to the modeling process
  - Inverse mapping of polygon interior back to texture space
    - During rendering phase
  - Texture mapping produces serious aliasing effects
    - Generally requires special anti-aliasing techniques

Uses of Texture Mapping

- Primary use is to apply surface detail
- Benefits and sample uses of texture mapping
  - Reduce polygon count by providing apparent detail to objects
    - Vegetation / terrain texture over a coarse grid of polygons
    - Distant objects can be effectively modeled as flat surfaces with texture
      - Building fronts
    - Common use in computer games, simulations to get higher frame rates
  - Mapping satellite imagery over terrain
    - Popular simulation technique
  - Orienting textured flat surfaces towards viewpoint to provide a 3D effect with minimal geometry
    - "Billboarding" of partially transparent textures
      - Trees and other complex 3D objects
      - SGI Performer and other demos use this frequently

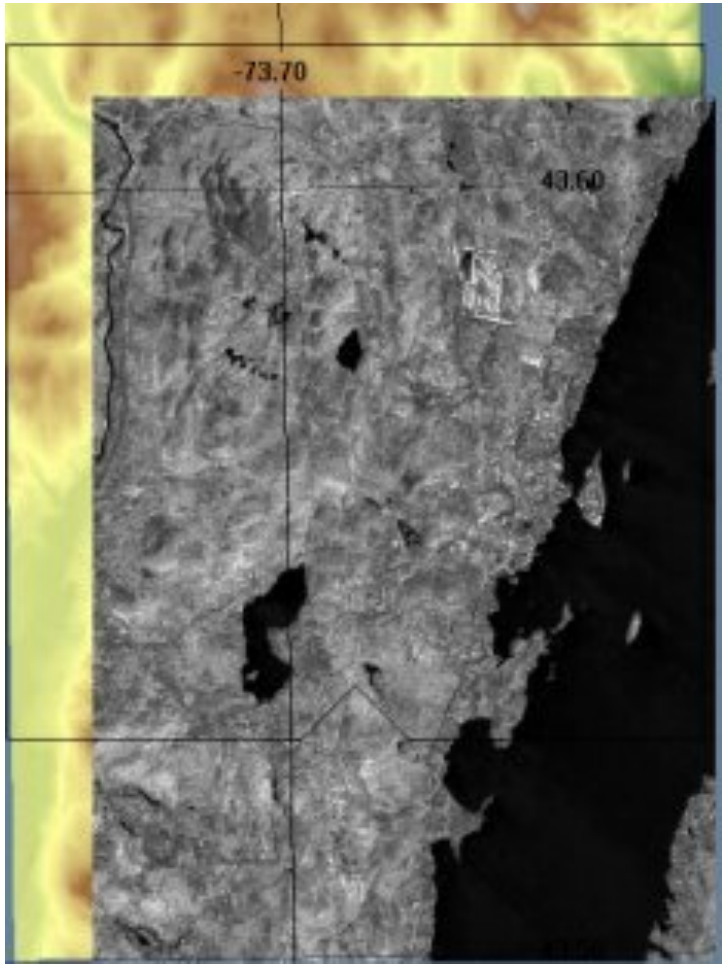
Applying Surface Detail

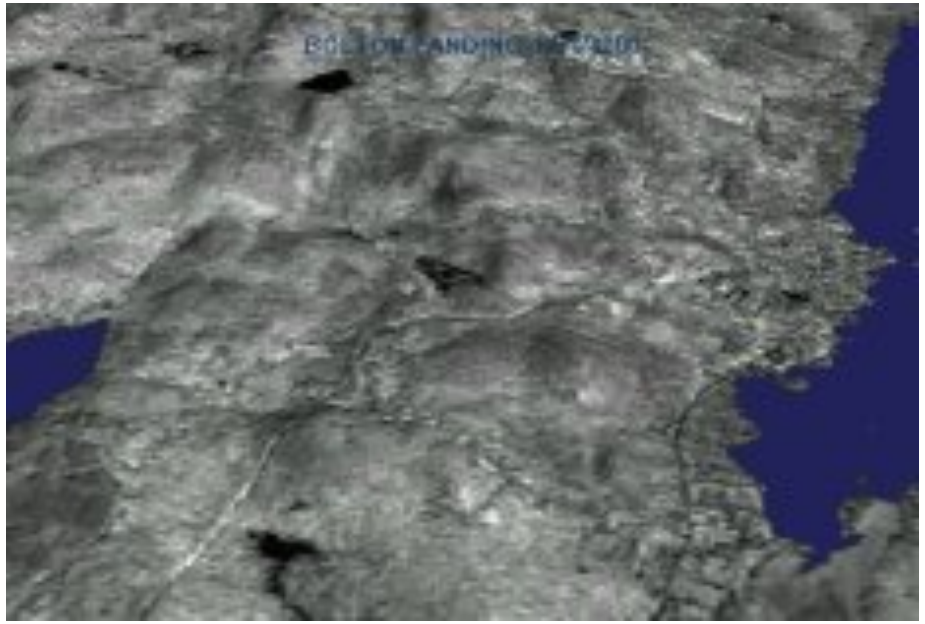


http://www.siggraph.org/education/materials/HyperGraph/mapping/r_wolfe/r_wolfe_mapping_1.htm [link inactive]

Teaching Texture Mapping Visually
Rosalee Wolfe, DePaul University

Image Mapping onto Terrain





http://www.terrainmap.com/rm18.html#ikonosoverlay

IKONOS imaging texture mapped onto USGS DEM (Digital Elevation Model)

Texture Uses (cont.)

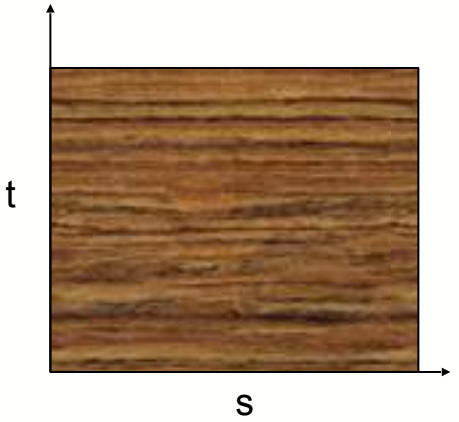
- Reflection
  - A technique called environment mapping can be used to simulate reflections
- Animated textures and video textures
- Projective textures
  - Can be used to generate shadows (shadow map technique)
  - Can act like a film projector
- Contouring
  - Can use textures to draw contour curves on an object
    - Could represent height above some plane
- Light interactions within large polygons
  - Use textures to produce specular highlights, spotlight effects
    - Overcome the deficiencies of Gouraud shading of large polygons
  - Lightmaps
- Shadows in a static scene

Object Attributes Modulated by Texture

- Surface color or diffuse reflection coefficients
  - Most common
  - Often called color mapping
- Specular reflection
  - Environment mapping
- Normal vector perturbation
  - Bump mapping
- Transparency
  - Can be used to generate objects like clouds

Two Dimensional Texture Mapping

- Most common form of texture mapping
- Two dimensional texture domain T(s,t)
  - Some texts use (u,v) as texture coordinates
  - Image is called a texture map
  - Individual elements are called texels (texture pixels)
- At each rendered pixel, selected texels either substitute or scale object material properties
  - Note: a pixel is often covered by a number of texels
    - Leads to aliasing
- Texture mapping consists of two parts
  - Surface parameterization
    - Applies the 2D texture to an object
  - Object - screen space mapping



Texture Domain

Sources of Texture Maps

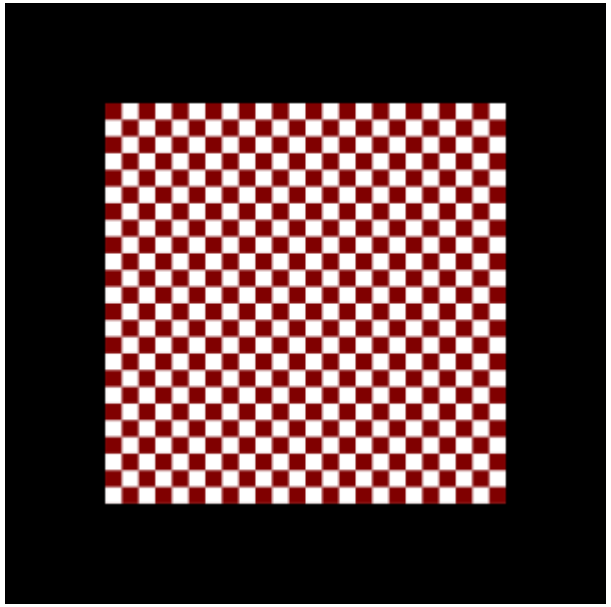
- Image files
  - Photo digitization, clip art, satellite imagery, etc.
  - Can be a previously generated computer graphics image
    - Render to texture methods are becoming common
  - Many different formats available
    - .bmp, .gif, .jpg, .png, etc.
  - Methods to read image file format and load into OpenGL texture formats
- Computed methods
  - Example: checkerboard pattern
- Procedural methods
  - Can generate textures with mathematical functions and procedures
    - Example: using turbulence functions to generate marble patterns

Texture Mapping and Object Representation

- General texture mapping pastes a 2D texture onto a 3D object
  - In effect becomes part of the object database
  - As the object is moved the texture moves with it
- Standard procedure is to associate each polygon vertex with a texture map coordinate (s,t)
  - s,t values associated with each vertex become part of the object database
  - Derive a mapping function: (s,t) = F(x,y,z)
    - Placing s,t values onto the object can be regarded as a modeling function
- Legacy OpenGL
  - glTexCoord2f(s, t)
    - Precedes the call to glVertex*
    - Sets current texture coordinates to (s,t)
    - Attaches the texture coordinate at the subsequent vertex coordinate
  - **glTexCoordPointer – add texture coordinates to vertex arrays**

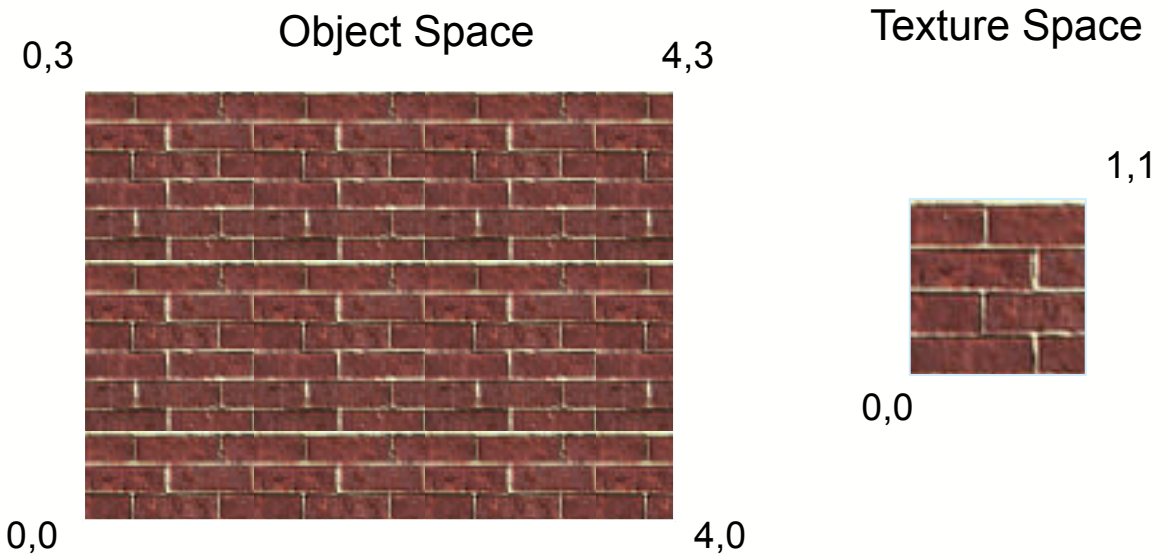
Mapping / Projector Functions

- Common mapping functions
  - Planar
  - Cylindrical
  - Shrink-wrap method
  - Spherical projection
- Texture coordinates can lie outside [0,1]
  - Repeat or wrap vs. clamping to range
  - Often use textures designed to repeat over large region
    - e.g., brick pattern or checkerboard pattern



Texture Mapping a Flat Surface

- Planar mapping
  - Scale and translate
  - Need to worry about object scale and texture scale
    - Example: scaling texture for a brick wall such that the bricks aren't too large or too small

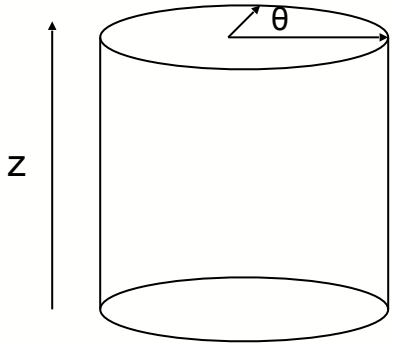


Repeating texture: Texture s,t Equal Plane x,y Coordinates

Cylindrical Mapping

- Cylindrical mapping
  - A point on the curved surface of a cylinder of height h and radius r:
    - $(r\cos\theta, r\sin\theta, hz) \qquad 0 < \theta < 2\pi \qquad 0 < z < 1$
  - Can associate texture values with a point on the cylinder by using cylindrical coordinates

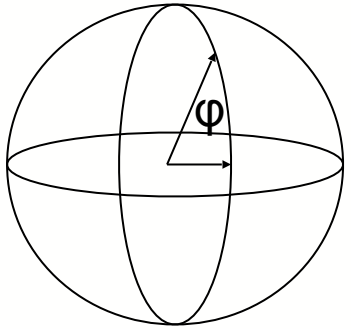
$$(s,t) = \left(\frac{\theta}{2\pi}, z\right) \qquad s,t \in [0,1]$$

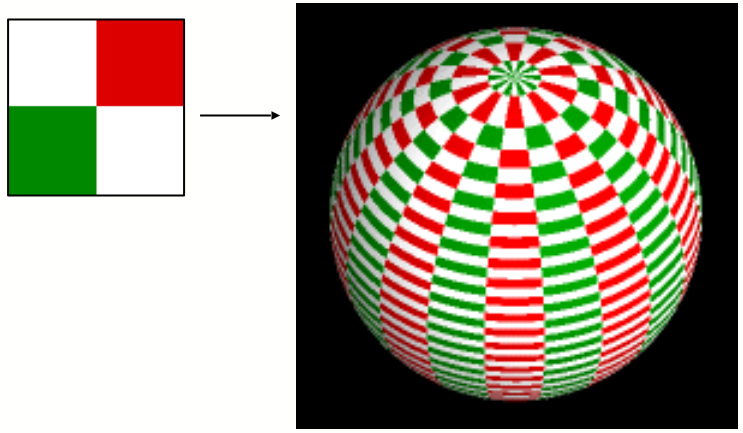


Spherical Mapping

- Spherical mapping
  - Mapping a plane onto a sphere produces unbounded distortion at the poles
  - Problem is inverse of cartographic projections (Mercator, oblique etc.)
- Could map based on latitude, longitude $(r\cos\theta\sin\varphi, r\sin\theta\sin\varphi, r\cos\varphi)$
  - Normalized to 0,1 range
- Consider mapping onto part of a sphere
  - Example: sphere parameterized as
    - $0 < \theta < \frac{\pi}{2} \qquad \frac{\pi}{4} < \varphi < \frac{\pi}{2}$
  - Mapping is

$$(s,t) = \left(\frac{\theta}{\frac{\pi}{2}}, \frac{\frac{\pi}{2} - \varphi}{\frac{\pi}{4}} \right)$$

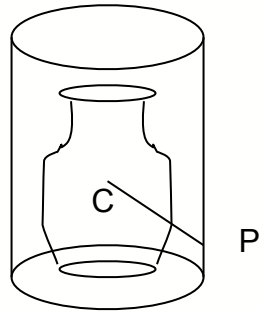


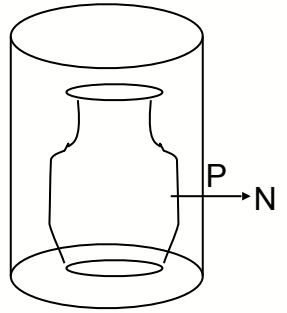


http://local.wasp.uwa.edu.au/~pbourke/ [link inactive]

Shrink Wrapping

- Useful technique to map a texture onto a surface of revolution
- Each quadrilateral has four vertices:
  - $P(u_i, v_i)$, $P(u_{i+1}, v_i)$, $P(u_i, v_{i+1})$, $P(u_{i+1}, v_{i+1})$
- Texture coordinate assignment to a surface of revolution:
  - s varies with rotation angle
  - t varies with height
  - Called “shrink-wrapping”
  - 2 alternatives using a “bounding cylinder”
    - Draw line from object centroid through the object vertex
      - Use the texture coordinate at the intersection with bounding cylinder
    - Use normal vector at object vertex to find intersection with cylinder



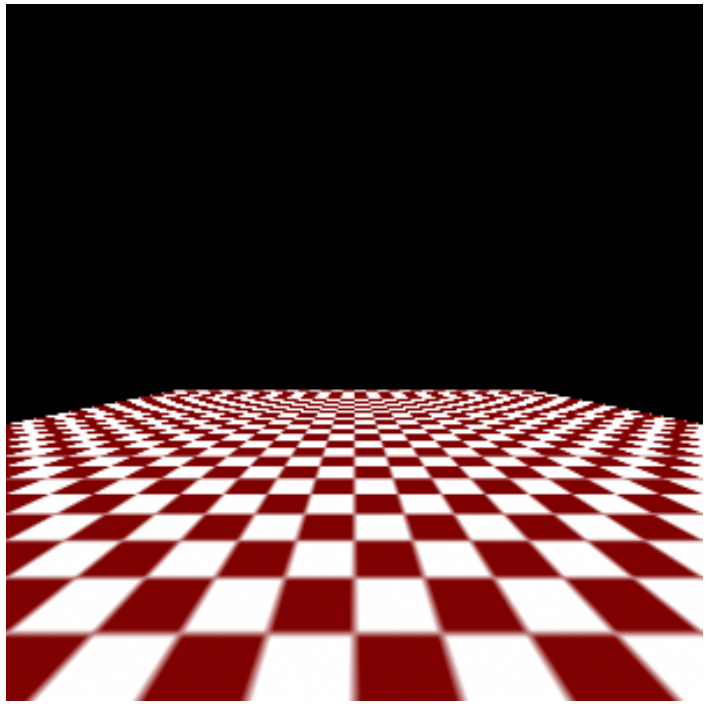


Rendering Polygons with Textures

- How does the rendering engine deal with interior polygon points?
  - As the polygon gets clipped and scan converted
- Simple solution: pass s,t coordinates to the rendering engine
  - As object gets clipped, interpolate s,t just as colors and normals are interpolated
  - Interpolate s,t along clipped edges and in the interior
    - Efficient hardware implementation possible
- However!
  - s,t do not change linearly with x,y when perspective projection is used
    - Need to perform **perspective correction**
  - Must determine a single texture value for each pixel to pass into the shading algorithm
    - Anti-aliasing considerations
      - Bilinear and **trilinear filtering** are common
      - More complete solution: Find the **pre-image** of the pixel in the texture domain by inverse mapping of the four corners of the pixel

Anti-aliasing and Texture Mapping

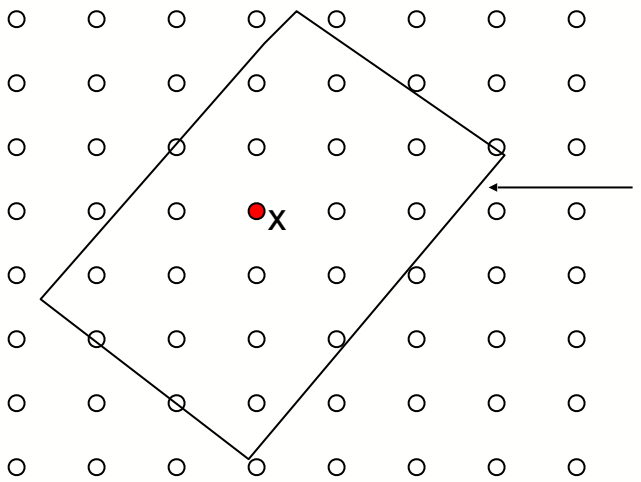
- Aliasing artifacts are very problematic in texture mapping
  - Texture mapping is generally unsuccessful unless some form of anti-aliasing is performed
    - Regular textures may break up and form moiré patterns
    - Textures may shimmer as viewpoint and/or object are moved
    - Particularly noticeable in regular texture patterns
- Point sampling of the texture during inverse mapping will lead to errors unless some filtering is performed
  - Large number of texels may map into a single pixel
- Anti-aliasing in texture mapping is the process of integrating information from the texture map
  - Return a single value from the texture domain to apply at the pixel



Aliasing

Undersampling

- Aliasing results from undersampling texture space
  - In the example below, selecting a single texel to represent the value at a pixel is clearly undersampling
- Many texels may contribute to the color of a single pixel
  - Need a method to account for the contributions of multiple texels
  - Some sort of integration / filtering
    - Performance and quality are 2 issues



Texture Space

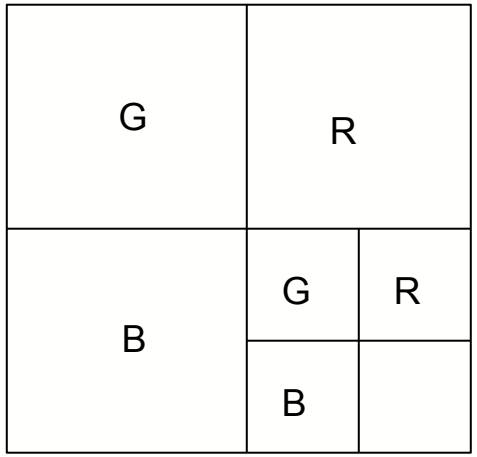
Mapping of the corners of a pixel into texture space

x marks the center of the pixel

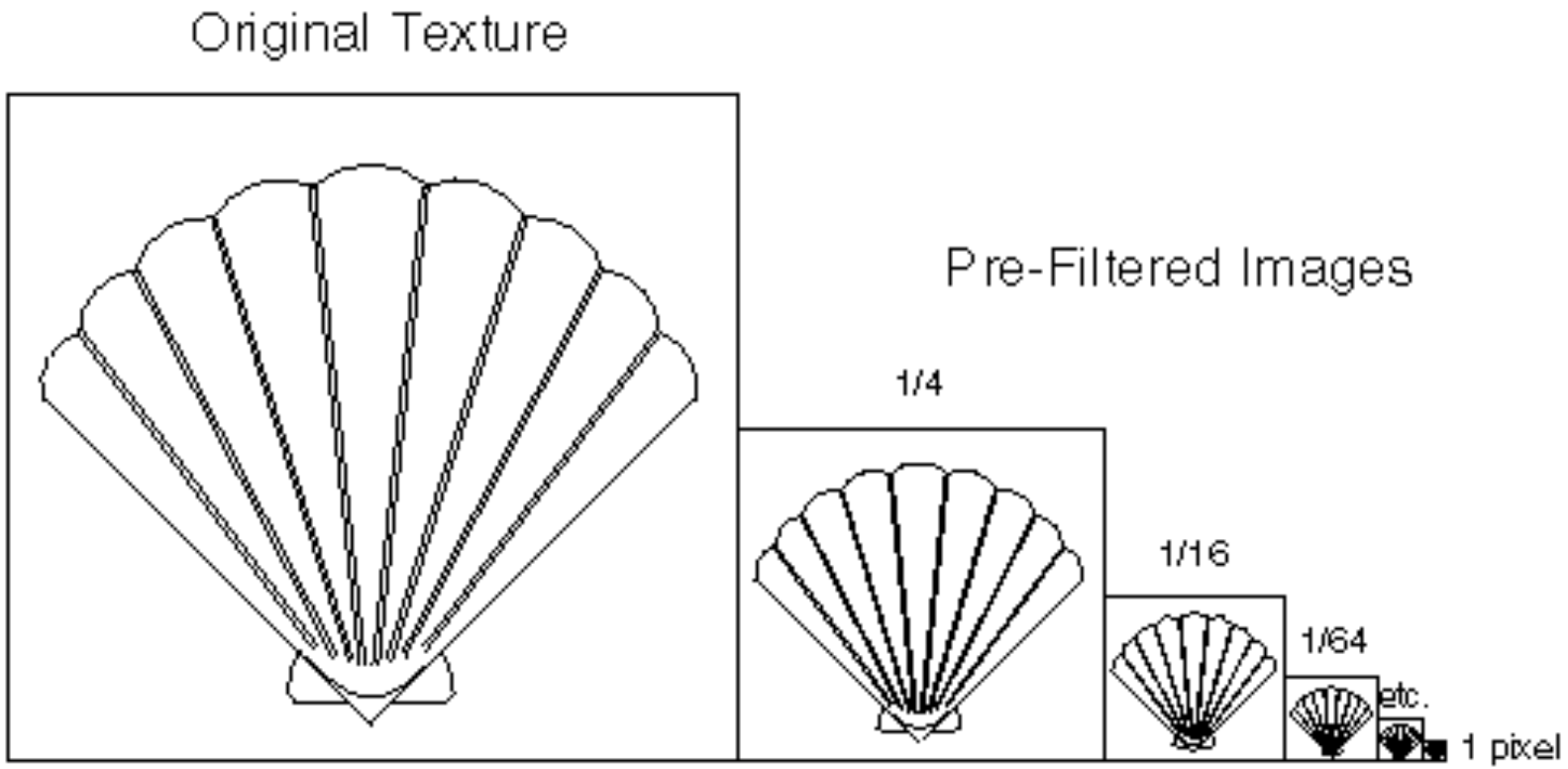
● marks the texel closest to the center of the pixel

Prefiltering Techniques

- Prefiltering techniques produce approaches where cost does not grow in proportion to texture mapped area
  - Computational costs with space-variant filters is high
    - Especially when large texture areas are mapped into small screen areas
- A prefiltering technique known as mip-mapping has become popular
  - Mip is derived from the Latin term multim im parvo - many things in a small place
  - Mip map contains a series of prefiltered texture maps of decreasing resolutions
    - Each image is half the resolution of the previous
  - Mip map occupies 4/3 of the memory of the original texture map
    - 512x512x24 bit true color image will require 1024x1024x8 bit memory area



MipMaps

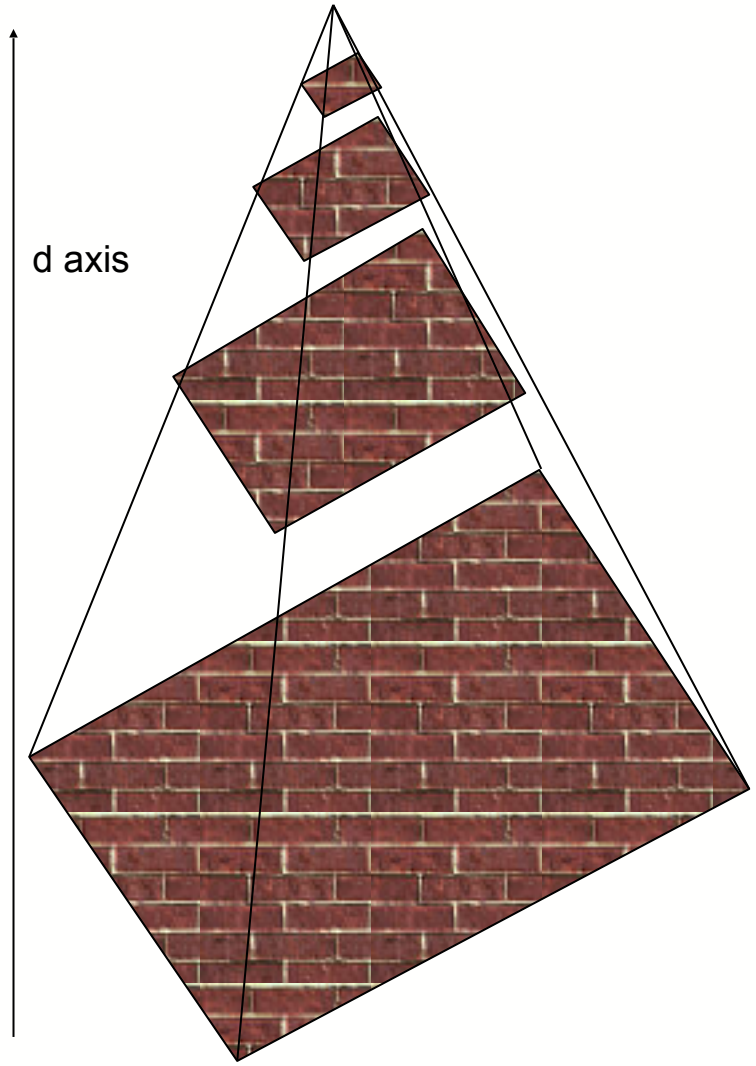


Mip Mapping

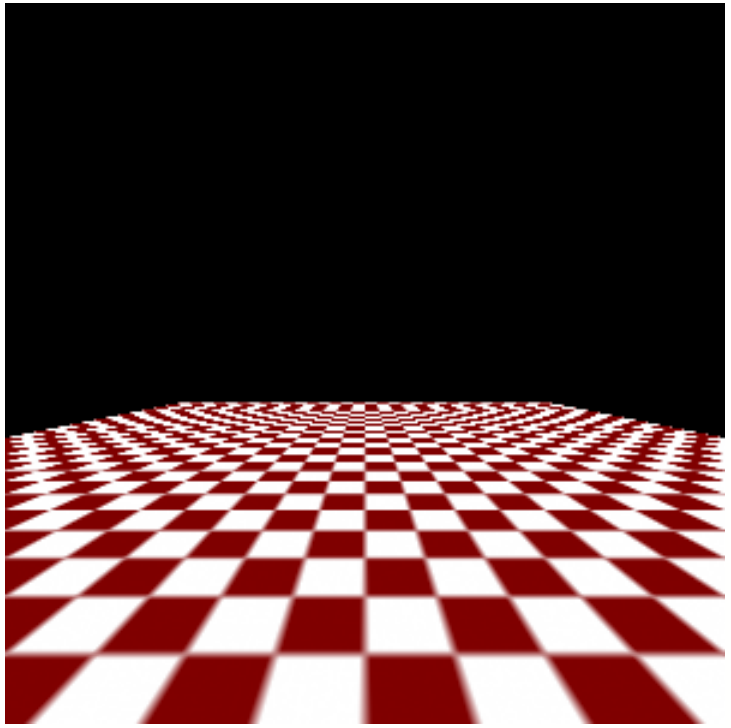
- Instead of filtering a large set of texels at execution time, a pre-filtered set of texels is selected
- Mip-mapping uses a parameter D to choose resolution
  - Selection of D is important
    - Too large and the image will look blurred
    - Too small and aliasing artifacts will still be visible
  - Williams proposed
    - $$D = \max\left[\sqrt{\left(\frac{\delta s}{\delta x}\right)^2 + \left(\frac{\delta t}{\delta x}\right)^2}, \sqrt{\left(\frac{\delta s}{\delta y}\right)^2 + \left(\frac{\delta t}{\delta y}\right)^2}\right]$$
    - Measures of how much texture changes as pixels change in x,y
    - OpenGL calls this value lambda (or level of detail)
- Produces a floating point number
  - Linear interpolation is often performed between the nearest two levels of resolution to further smooth the values
    - To avoid discontinuities between images at varying resolutions
  - We'll see how filtering is done when we discuss OpenGL

Mip Map as an Image Pyramid

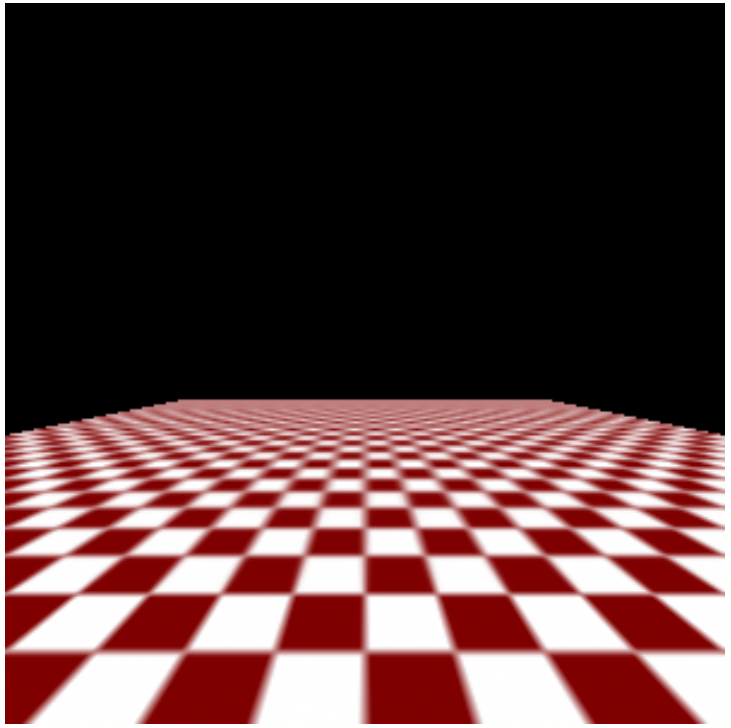
- Can also think of Mip Maps as a pyramid of images
- Base level (0) is full image
- Each successive level is ½ the width and ½ the height
  - Average each 2x2 area into a new texel value
    - Or average nxn area from level 0 texture
- Can interpolate between two levels
  - Trilinear interpolation



Example of MipMapping



No MipMapping



With MipMapping

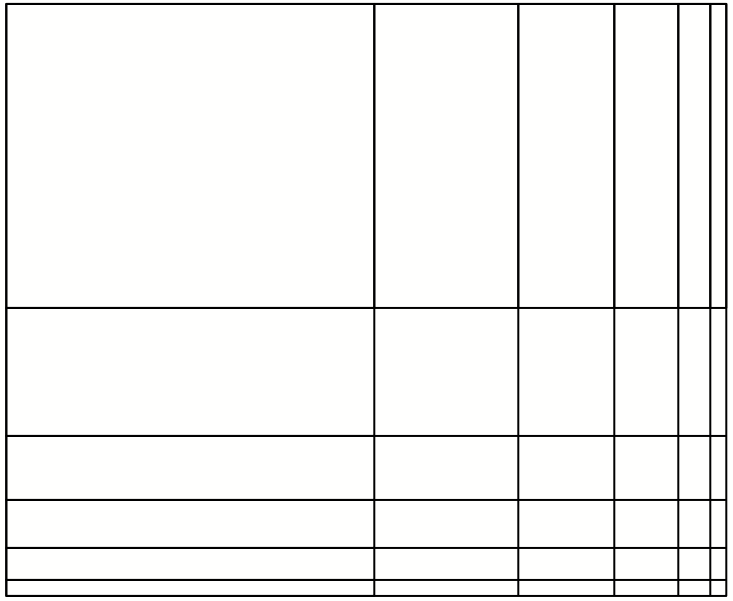
Notice how the texture is distorted in the distance without mipmaps, but with mipmaps the texture is smooth and does not become distorted in the distance.

MipMapping – Advantages and Shortcomings

- MipMaps are widely supported
- Most hardware is optimized to do trilinear filtering
- However, mipmapping tends to result in **overblurring**
  - Filtering is done over square regions
  - Problem if texture covers only a few texels in one direction but many in another
    - Results when a surface is being viewed nearly edge-on
  - A square region of the texture is accessed (for largest measure)
- A couple methods attempt to deal with this
  - Ripmaps
  - Summed area tables
  - Both are **anisotropic filtering** methods
    - Filter over non-square regions but most effective in horizontal and vertical directions

RipMapping

- Ripmaps are an extension of Mipmaps that allow rectangular areas as subtextures
  - Rectangular portions allow sampling of non-square subtextures
    - Accommodates pixels that project to non-square regions
  - Memory intensive
    - Ripmap memory is 3x the original texture
  - Used in high end Hewlett-Packard graphics accelerators in the early 1990s



Ripmap structure
Note that images along diagonal from upper left to lower right are the mipmap subtextures.