CS 563 Advanced Topics in Computer Graphics

Textures – Image, Solid and Procedural

by Alex White

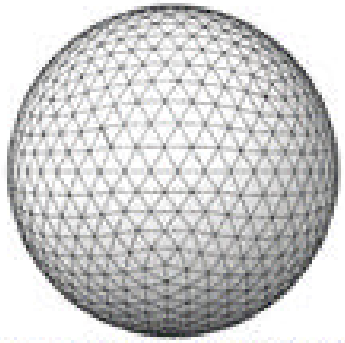
Outline

- Image Textures
  - MIP Map
  - Filter Methods
- Solid and Procedural Textures
  - Avoiding Aliasing
  - Averaging
  - Supersampling

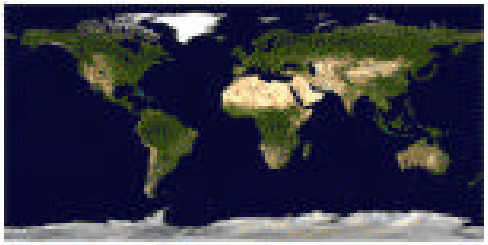
Image Textures

- Images are used to enhance objects
- Unit name: Texels
- Texture images ambiguously termed “texture maps”

Texture Mapped Images



Sphere with no texture



Texture image

Sphere with texture

Physically Based Rendering - Figure 11.8 *Wikipedia*

PBRT Implementation

- Image maps = lots of memory
  - Why especially in default for PBRT?
- Texture caching alleviates some load
  - However…the filter parameters must match

- Implemented in PBRT as ImageTexture class
- Give it filename and three simple parameters
- If it cannot open or read the file, a map with a single sample of 1 is created

PBRT Implementation

- ImageTexture::Evaluate handles the coordinate transformation

- Takes a DifferentialGeometry
- Uses _____Mapping2D::Map
- Computes s,t and associated partial derivatives
- Performs lookup on the created MIP Map
- Returns a value for that point

MIP Map

- The guts of Image Textures
- Handles reconstruction and filtering
- Trivia: MIP from Latin “multum in parvo”

- Texture sampling vs. image (geometry) sampling
  - No ray tracing required
  - Only an array lookup
  - Highest frequency already known
  - No surprises

Prefiltering

- Many more texels than texture space sampling rate
- However, this sampling rate varies spatially
- No single prefilter fits all

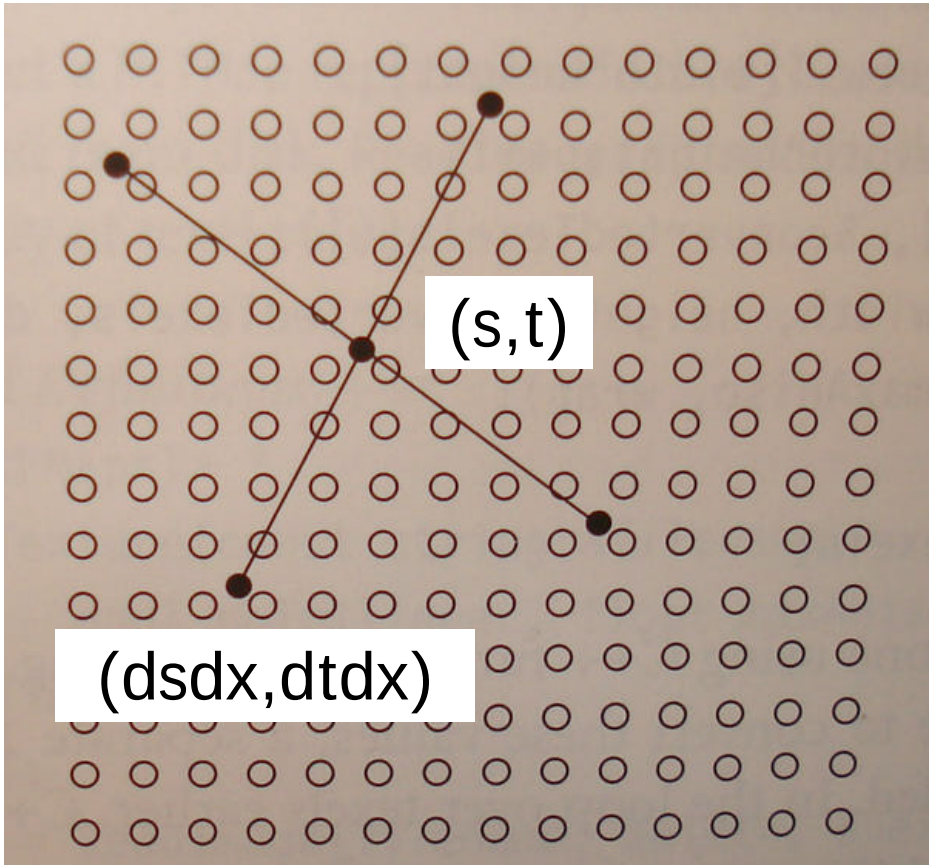
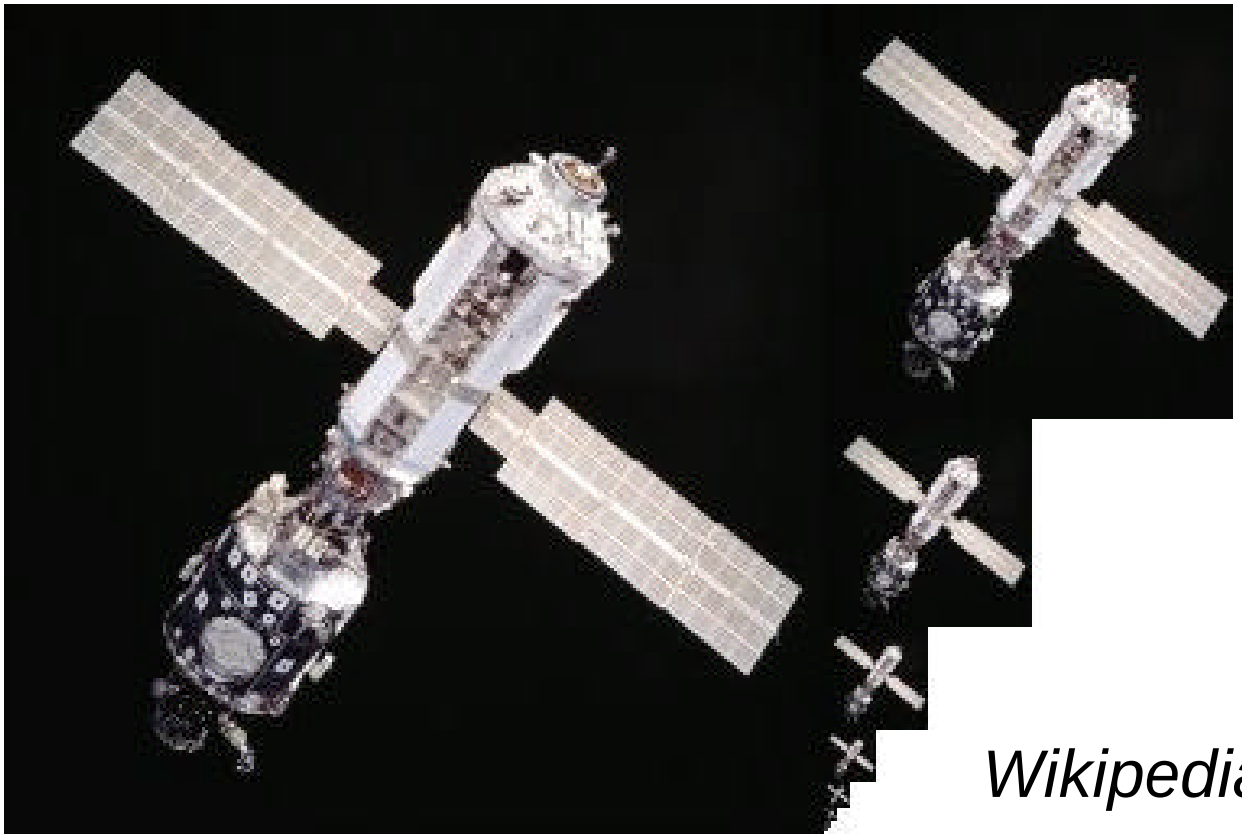


Figure 11.9

MIP Maps Ease the Pain

- MIP Map creates an *image pyramid*
- Instead of filtering over 50+ samples per ray
- Store images (each as ¼ size)



Wikipedia

MIP Map Setup

- Image dimensions need to be a power of 2
- What do we do?
  - Upsample – (Lanczos works well)
  - Important not to lose information
  - Since a continuous function, we can reconstruct effectively
- Get scaled images using a lowly box filter
  - Average the 4 texels from more detailed level
- Store in memory-access-efficient BlockedArray

Isotropic Triangle Filter

- Not high quality, but efficient
- Takes (s,t) coordinate and a filter width
  - Yields a square to filter over
  - Isotropic – does not support non-square or non-axis aligned filter extents
- Performs trilinear interpolation over nearest MIP Map levels

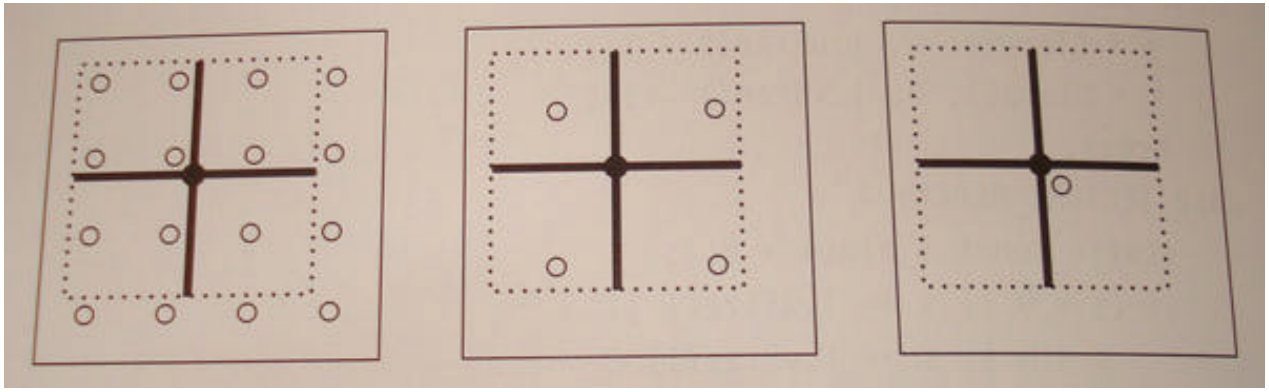


Figure 11.13

Elliptically Weighted Average

- EWA regarded as one of best texture filtering algorithms
- Carefully derived from sampling theory
- Utilizes Gaussian filter
- Anisotropic – can adapt to sampling rates

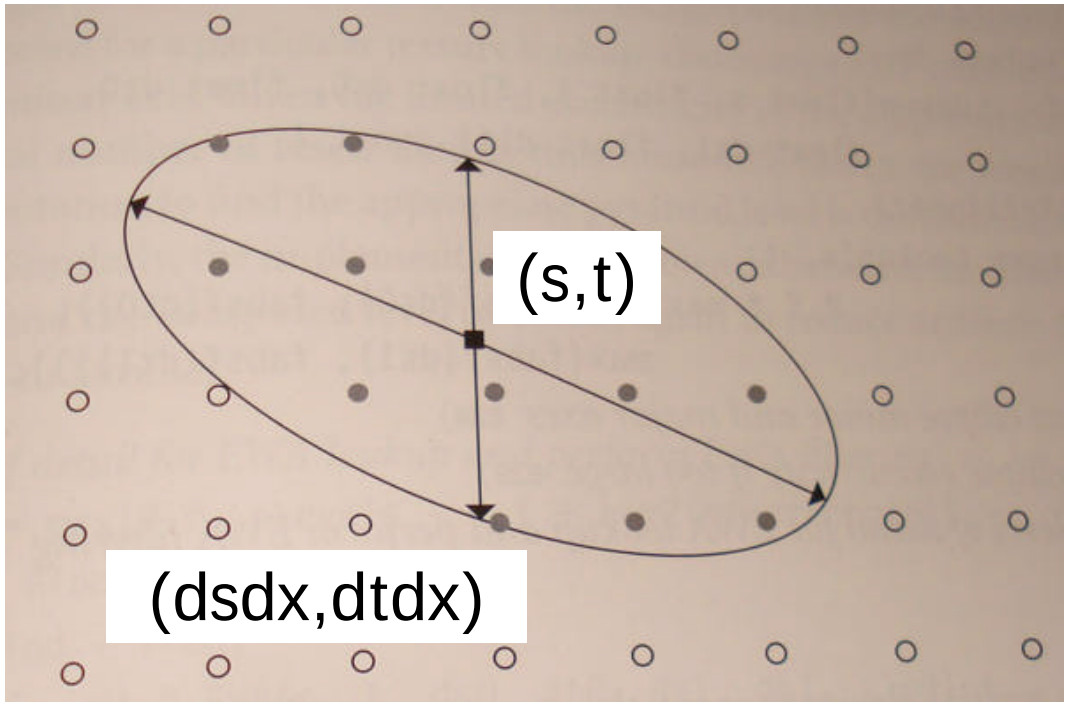


Figure 11.15

How is EWA Computed?

- Use the partial derivatives to create major and minor axes
  - Defines the ellipse
- Compute the eccentricity
  - Why?
  - But end result may end up blurry
  - The third parameter, maxAnisotropy, is used as a scaling factor - higher values = less clamping
  - Graphics cards implement 1x – 16x (though in a different fashion - parallelogram vs ellipse)

How is EWA Computed? (2)

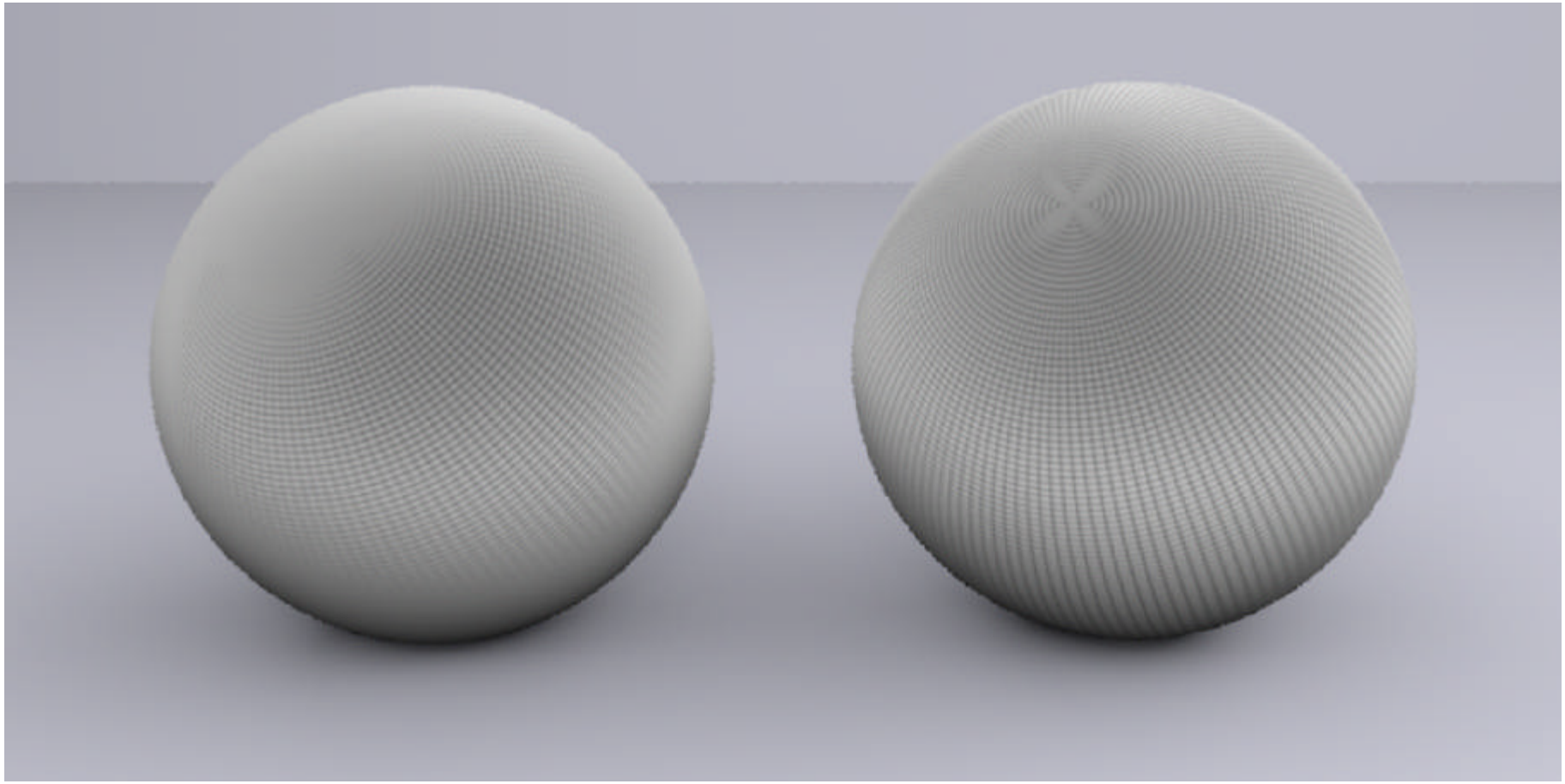
- Like in the Triangle Filter, EWA takes advantage of the MIP Map pyramid
- This time based on the length of the minor axis
- The implicit equation for an ellipse:

$$e(s,t) = As^2 + Bst + Ct^2 < F$$

How is EWA Computed? (3)

- Loop over candidate texels in bounding box
- Weight inside texels by Gaussian centered in the middle of the ellipse
  - Done with a Look-Up-Table
  - Independent of coordinate space
  - Exponential term computationally expensive
- Sum each texel's weight

Triangle versus EWA



Isotropic Triangle EWA

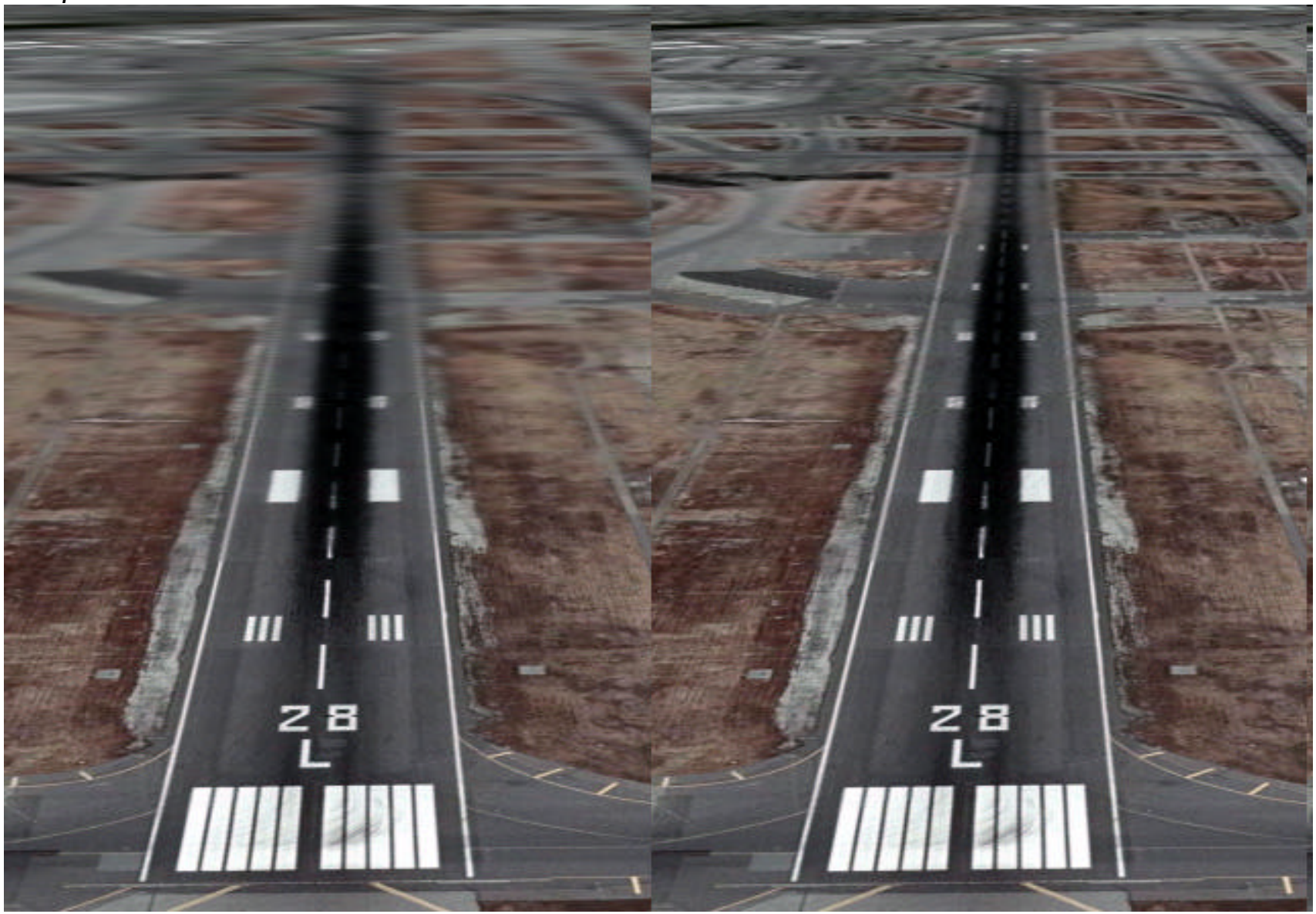
Figure 11.10

Notice the added detail where normals near perpendicularity?

Anisotropic Filtering

(Real-Time Graphics)

Wikipedia

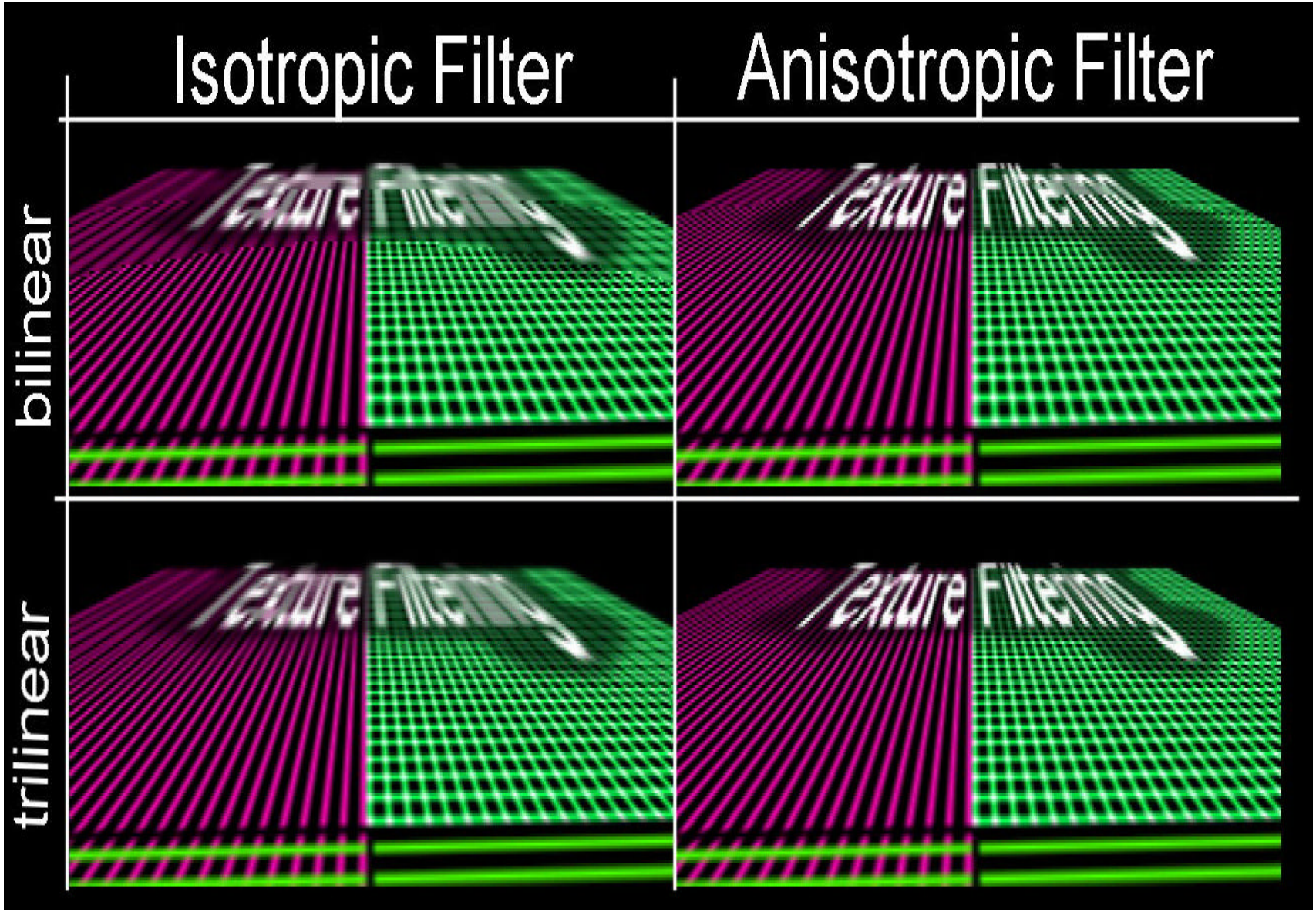


Trilinear MIP Mapping

w/ Anisotropic Filtering

(Real-Time Graphics)

Texture Filtering Comparison



www.extremetech.com - *The Naked Truth About Anisotropic Filtering*

Solid / Procedural Textures

- What do we do for the popular triangle meshes?
  - They don’t have a natural (u,v) parameterization
- The (s,t) image map coordinates can be computed by any arbitrary function
- Solid textures – texture functions are generalized over 3-D domains

Solid / Procedural Textures (2)

- Travis discussed an interface to computer 3D texture coordinates (TextureMapping3D)
- Photos are easy 2D images
- Difficult to acquire 3D representations

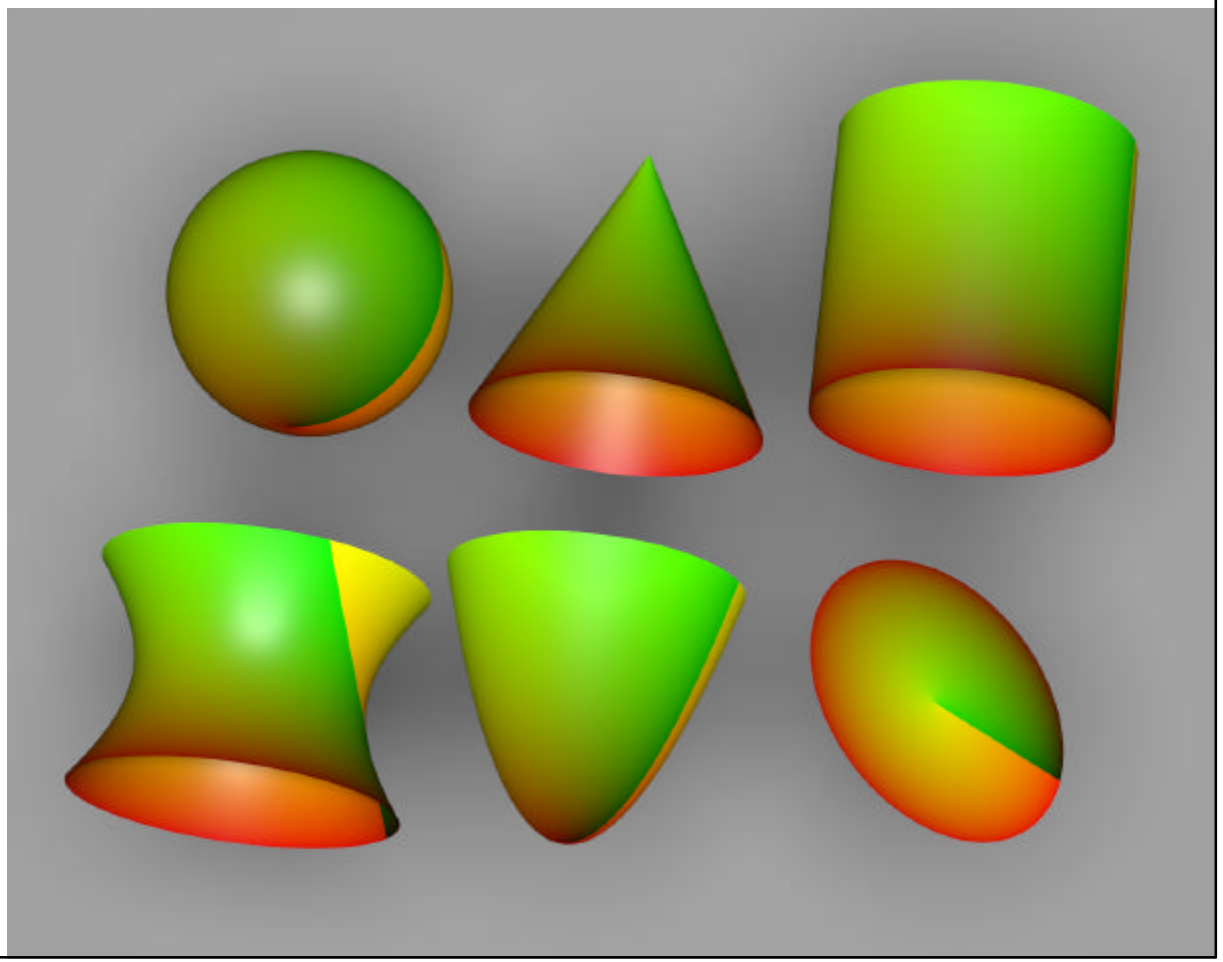
- Procedural texturing – small programs generate texture values are arbitrary positions
- Growing field
- Possibly infinite detail!

The Ultimate Technique?

- Unfortunately, subtle details of appearance can be hard to control
- Antialiasing becomes a problem
- Expensive evaluation

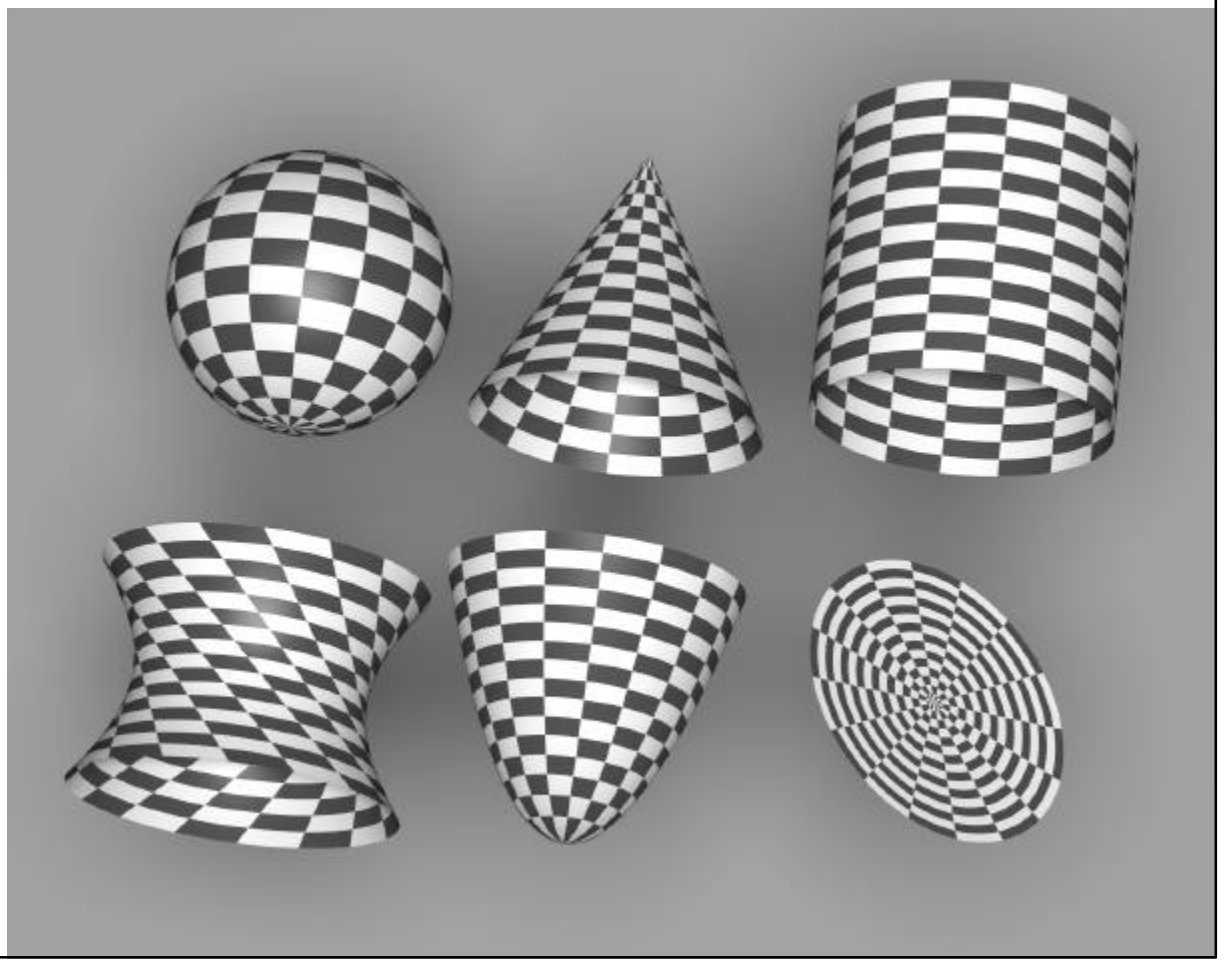
Procedural Textures

- UV Texture
  - Map U to red
  - Map V to green
  - Get a smooth color change
  - Good for debugging



Procedural Textures

- Checkerboard
  - Alternating textures
  - Passed in by user
  - Constant Textures used here



Avoiding Aliasing

- Simplest case, filter box lies within one of the textures
  - Compute bounding box on check
  - Check with axis-aligned bounding box

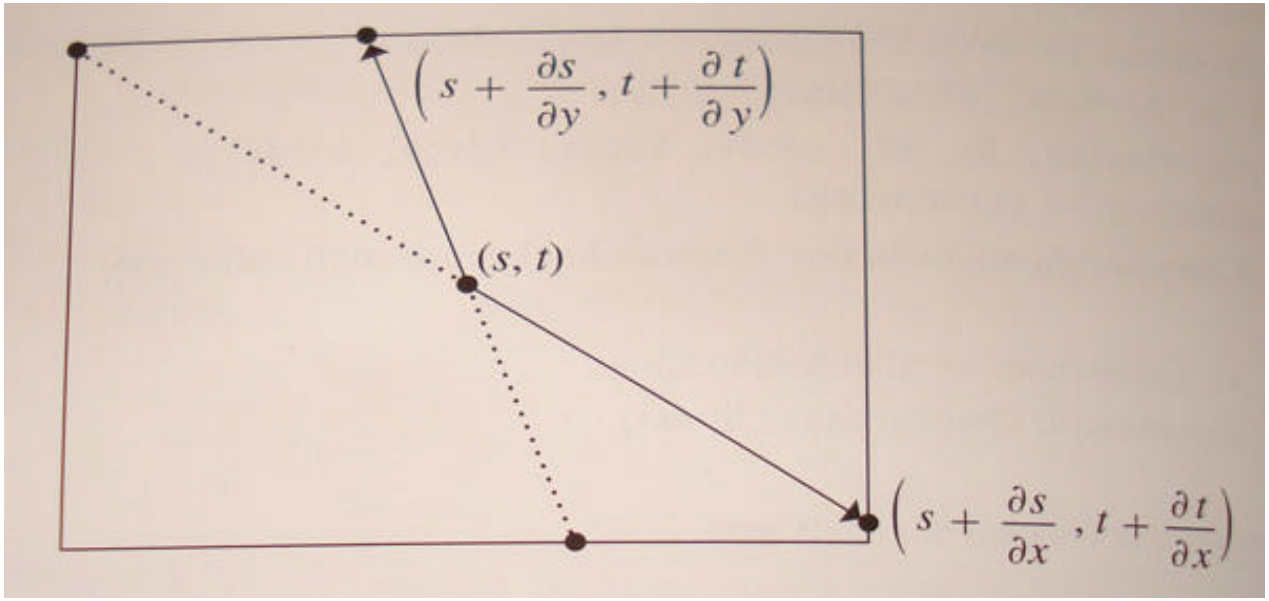


Figure 11.20

Lookup Method

- Blend two subtextures
  - Integrate the function to compute the average value over some extent
  - Complete each 1D direction and average
- However can be incorrect
- Each subtexture evaluates and antialiases as if completely visible

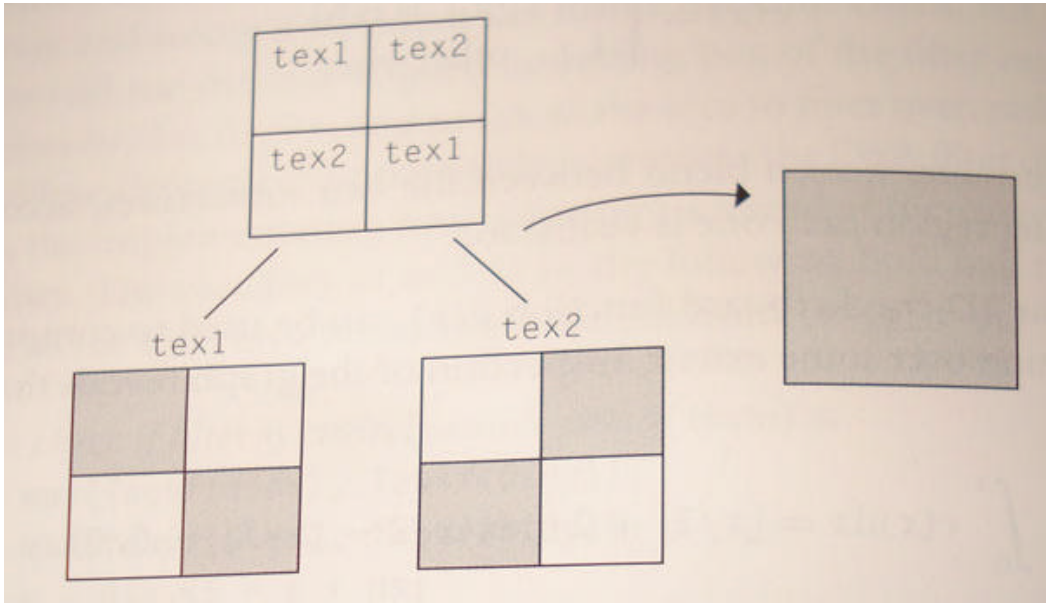


Figure 11.20

Supersampling

- Function is evaluated at a set of random stratified position around (s,t)
- 4 in s direction, 4 in t = 16 samples
- These are jittered to roughly cover the surface area

- Jitter values obtained though StratifiedSample2D()
- Shift by partial derivatives scaled by the jitter
- Finally, weight evaluated texture with a Gaussian filter
- Sum the results

One sample per pixel

Antialiasing Comparison

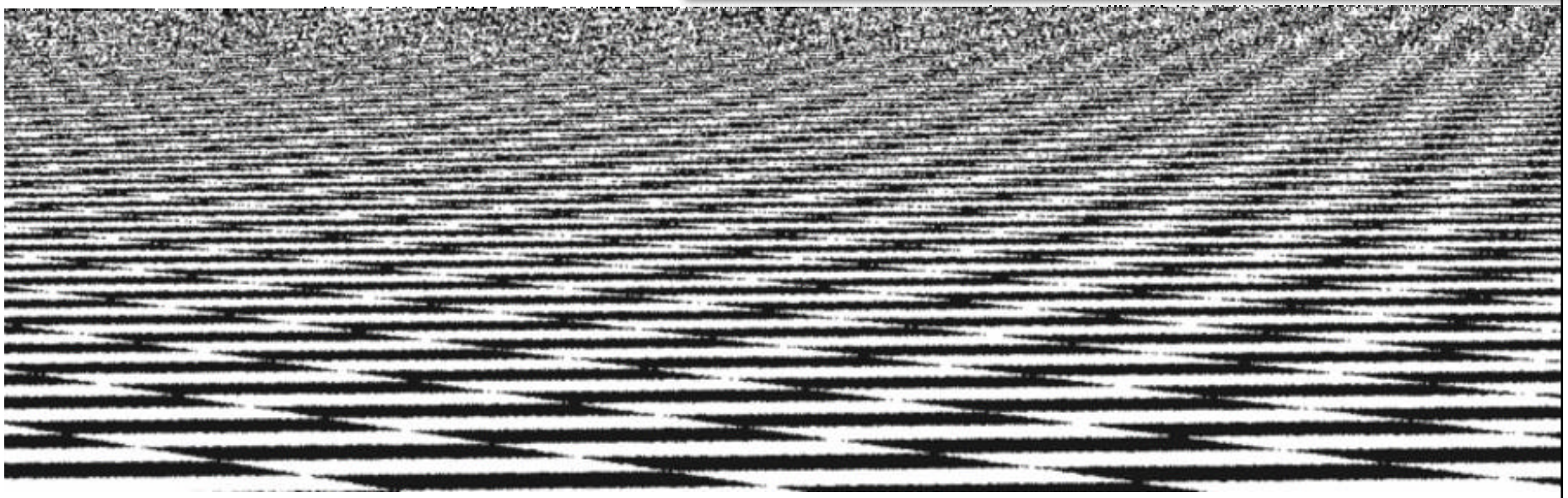


Figure 11.23a: No effort to remove high frequency variation

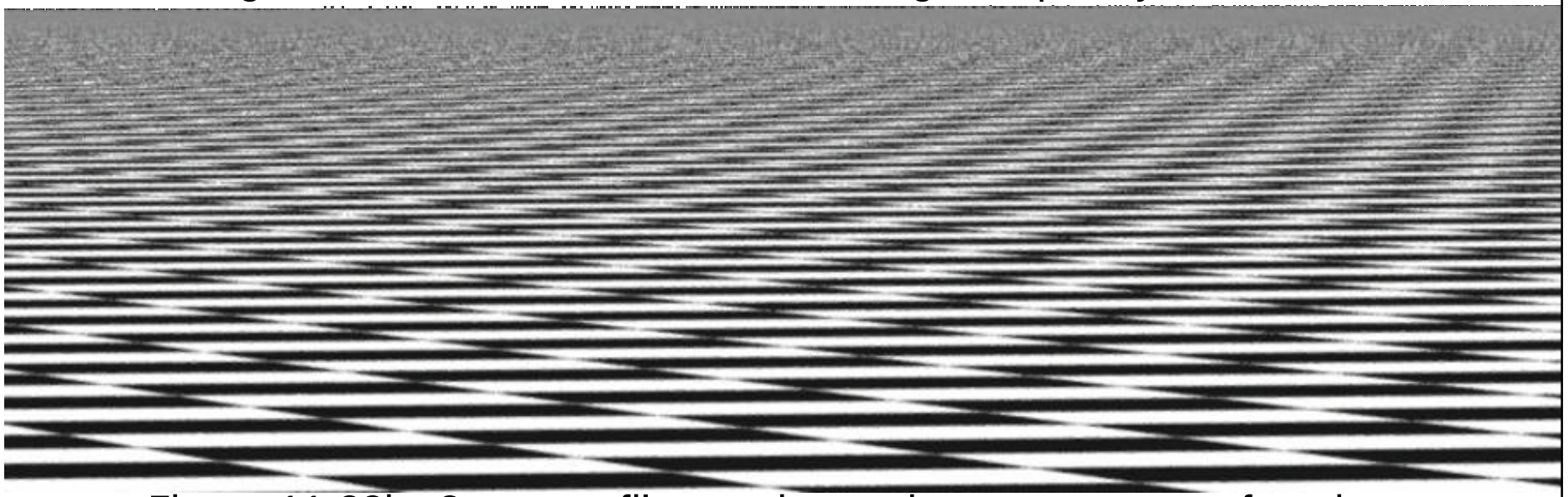


Figure 11.23b: Compute filter region and average texture function

One sample per pixel

Antialiasing Comparison (2)

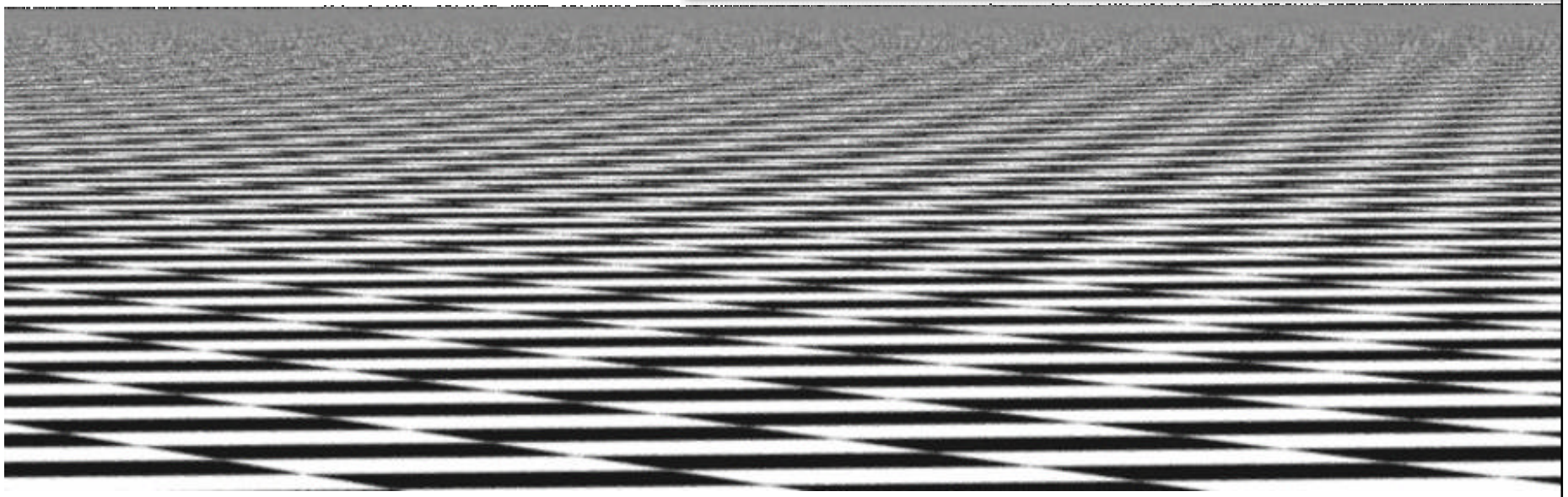


Figure 11.23b: Compute filter region and average texture function

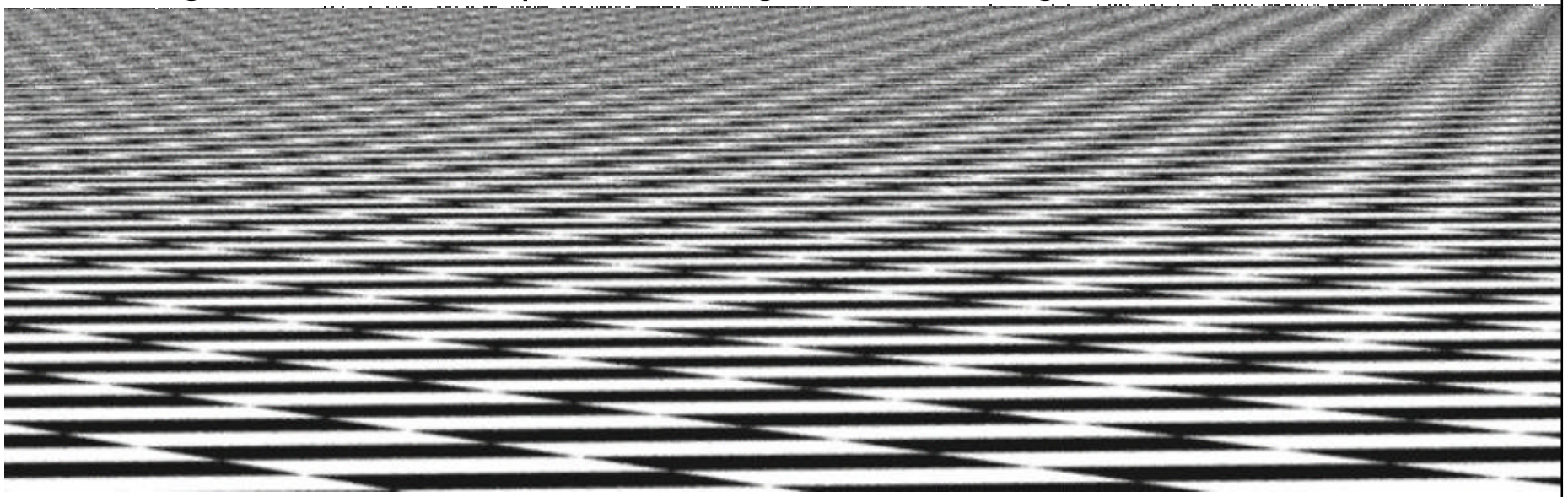


Figure 11.23c: Supersampled 16 times

Solid Checkerboard

- A checkerboard, but in 3D cubes
- Object “carved out of”
- Checkerboard3D

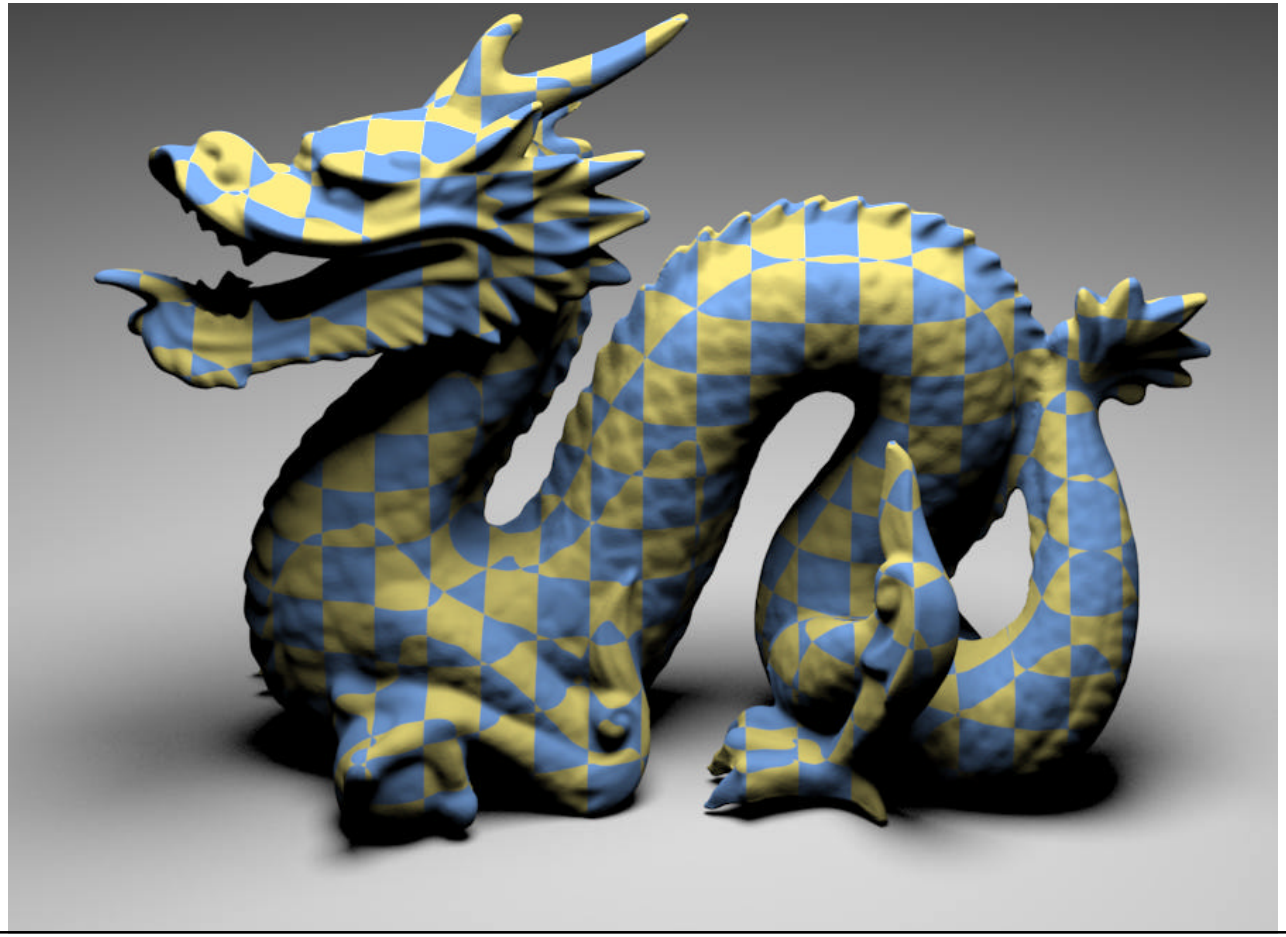


Figure 11.27

Review

- MIP Maps
  - Solve high frequency textures
  - Average relatively small number of samples
  - Only 1/3 more space
- Isotropic Triangle Filter
  - Quick and better than unfiltered
  - Cannot show oblique surfaces in detail
- Elliptically Weighted Average
  - Preserves detail approaching perpendicual normals
  - Anisotropic (user controllable)

Review (2)

- Solid Textures are 3D maps
  - Carved out of, rather than painted on
- Procedual Textures utilize functions to calculate values at positions (2D or 3D)
  - Can be powerful, provide immense detail
  - Reduces memory requirements
  - Expensive to calculate
  - Introduces aliasing

Review (3)

- Procedural Antialiasing Techniques
  - Use average area in box filter to weight
    - Works fine for solid textures
    - Can introduce “incorrectness” for others
    - Guarantees no aliasing due to texture
    - But prone to blurring
  - Supersampling
    - Slower (multiple evaluations)
    - Susceptible to aliasing (poor sampling)
    - Less prone to blurring

References

- *Physically Based Rendering*. Pharr and Humphreys.
- *Wikipedia*. Assorted Entries
- www.extremetech.com.*The Naked Truth About Anisotropic Filtering.*

Questions?