Computer Graphics

Lecture 8

Antialiasing, Texture Mapping

Today

- Texture mapping
- Antialiasing
- Antialiasing-textures

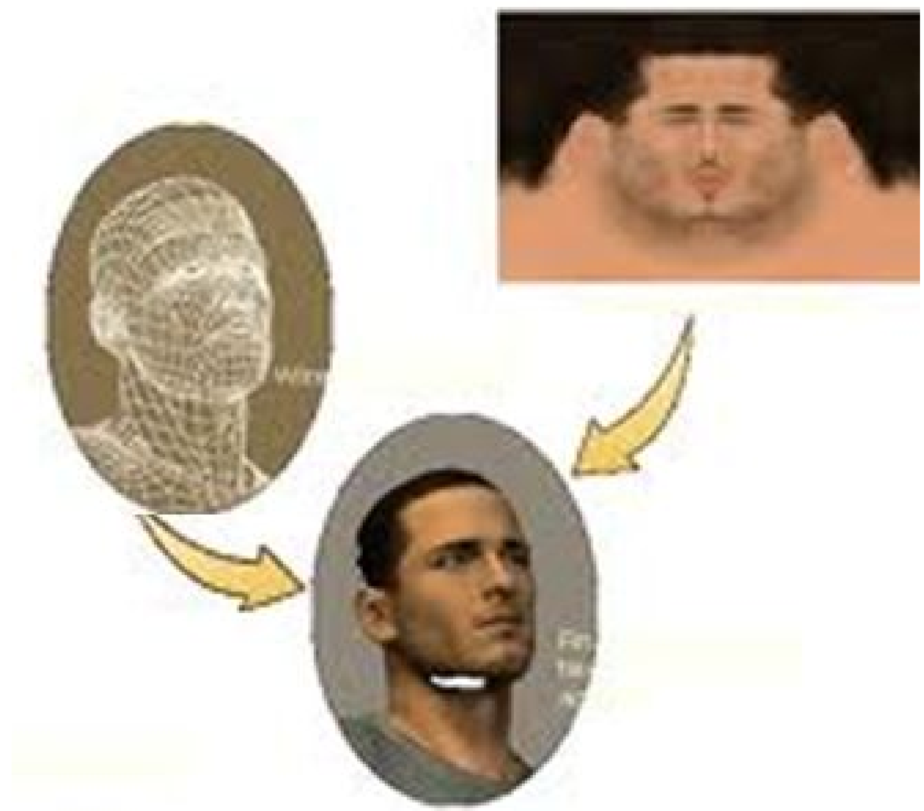
Texture Mapping : Why needed?

- Adding details using high resolution polygon meshes is costly
- Not very suitable for real-time applications

by Rami Ali Al-ashqar

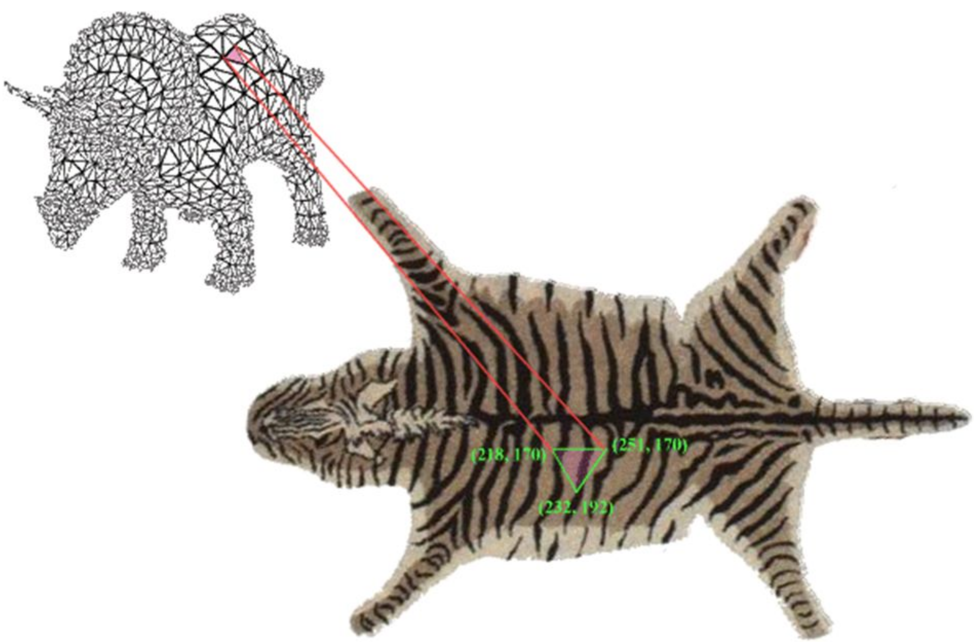
Texture mapping.

- Method of improving surface appearance by mapping images onto the surface
- Done at the rasterization stage



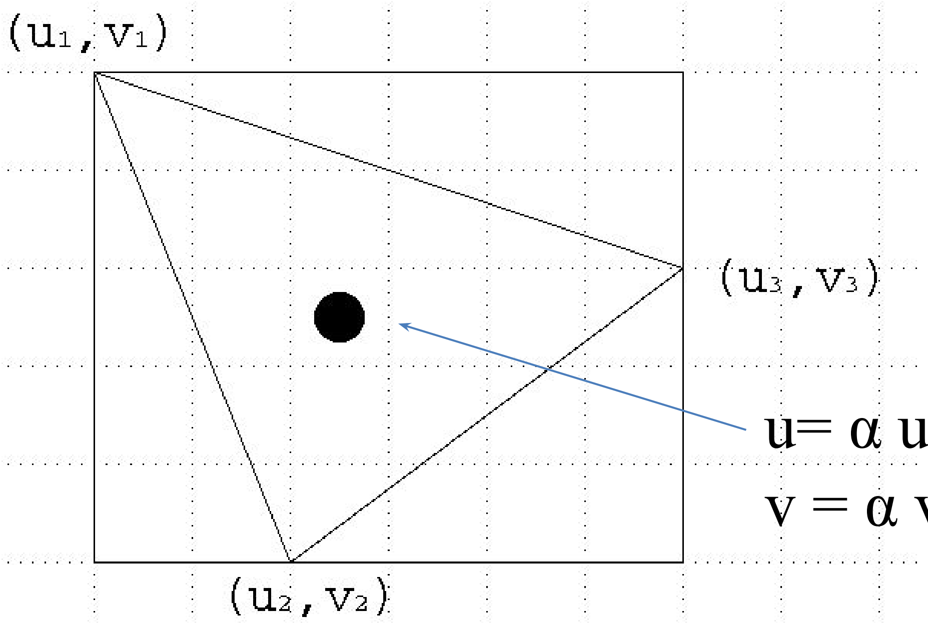
Principles of Texture Mapping

- For each triangle in the model, establish a corresponding region in the texture image
- Image has much higher resolution than mesh
- During the rasterization, color the surface by that of the texture
- UV coordinates: the 2D coordinates of the texture map



Interpolating the uv coordinates

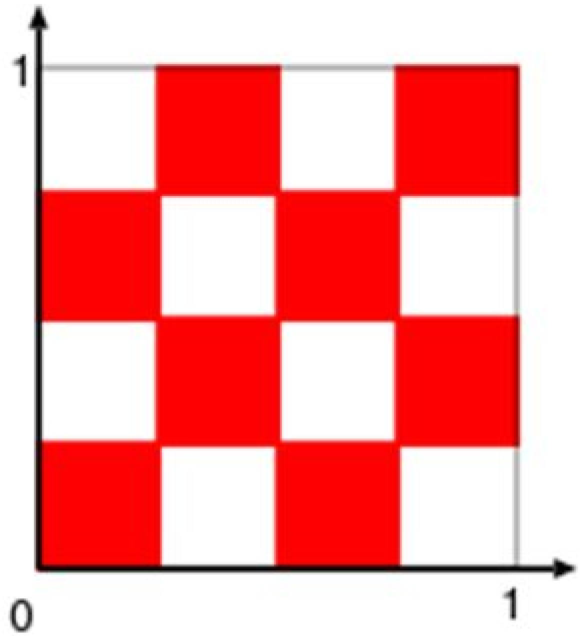
- To compute the UV inside the triangle, interpolate those at the vertices
- Done at the rasterization stage
- Use barycentric coordinates

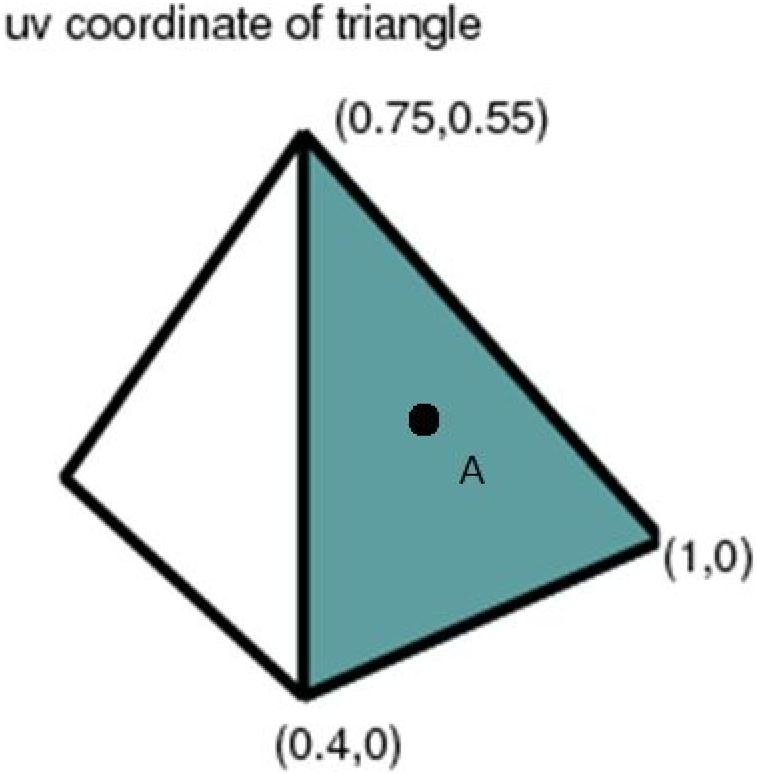


$$u = \alpha\, u_1 + \beta\, u_2 + \gamma\, u_3$$

$$v = \alpha\, v_1 + \beta\, v_2 + \gamma\, v_3$$

1. What is the color of the 3 vertices of the triangle?
2. How will the textured triangle look like?





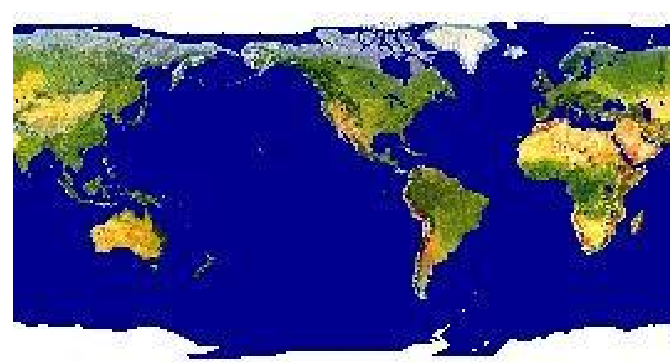
How to Produce a UV Mapping?

- Use common mappings
  - Orthogonal, cylindrical, spherical
- Capturing the real data
  - Obtain the color as well as the 3D shape
- Manually specify the correspondence
  - Using graphical tools
  - Using automatic segmentation, unfolding

Common Texture Coordinate Mappings

- Orthogonal
- Cylindrical
- Spherical





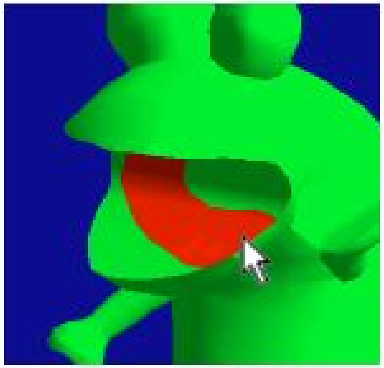
Capture Real Data

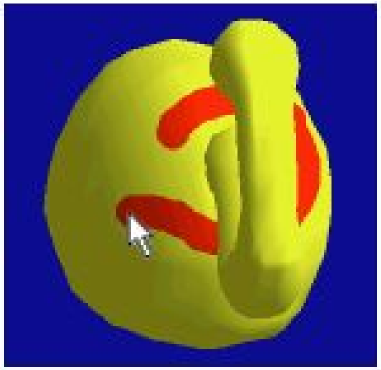
- Capture the depth as well as the color

http://www.3dface.org/media/images.html

Manually specifying the mapping

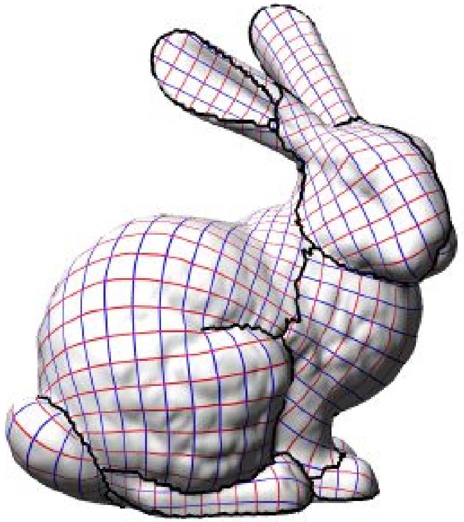
- There are good tools to map the texture to the surface, for example,
  - Directly painting on 3D geometry
  - Manually align unfolded data on the image

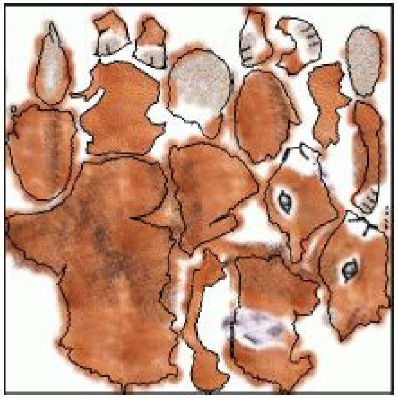




Unfolding the geometry

- Segmenting the body into a set of charts
- Unfolded onto a 2D plane
- Packing: The charts are gathered in texture space
- The artist can then draw the texture on a 2D plane





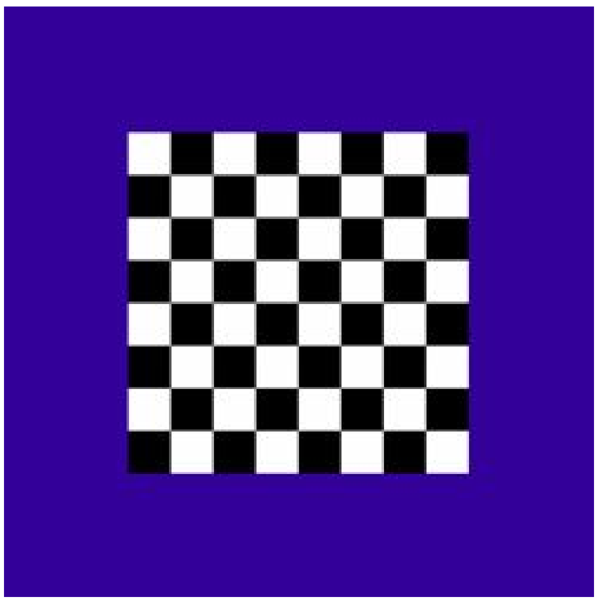
Creating a UV map on Maya

Levy et al. SIGGRAPH 2002

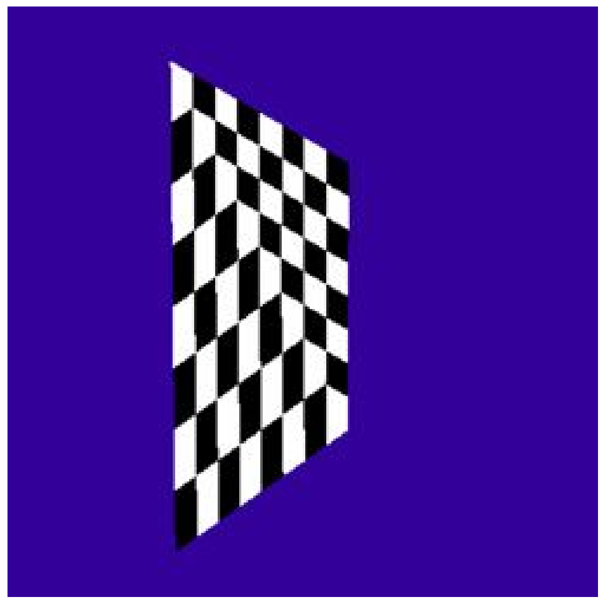
http://www.youtube.com/watch?v=HQKDADeuTFc

Problems with Linear Interpolation of UV Coordinates

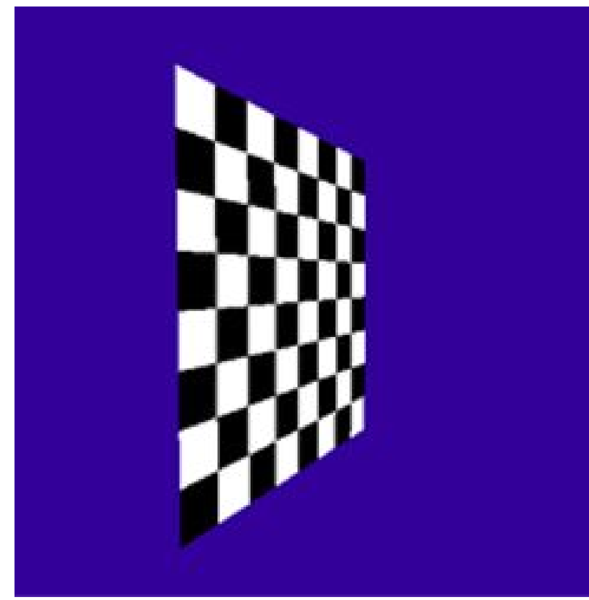
- Linear interpolation in screen space:



texture source



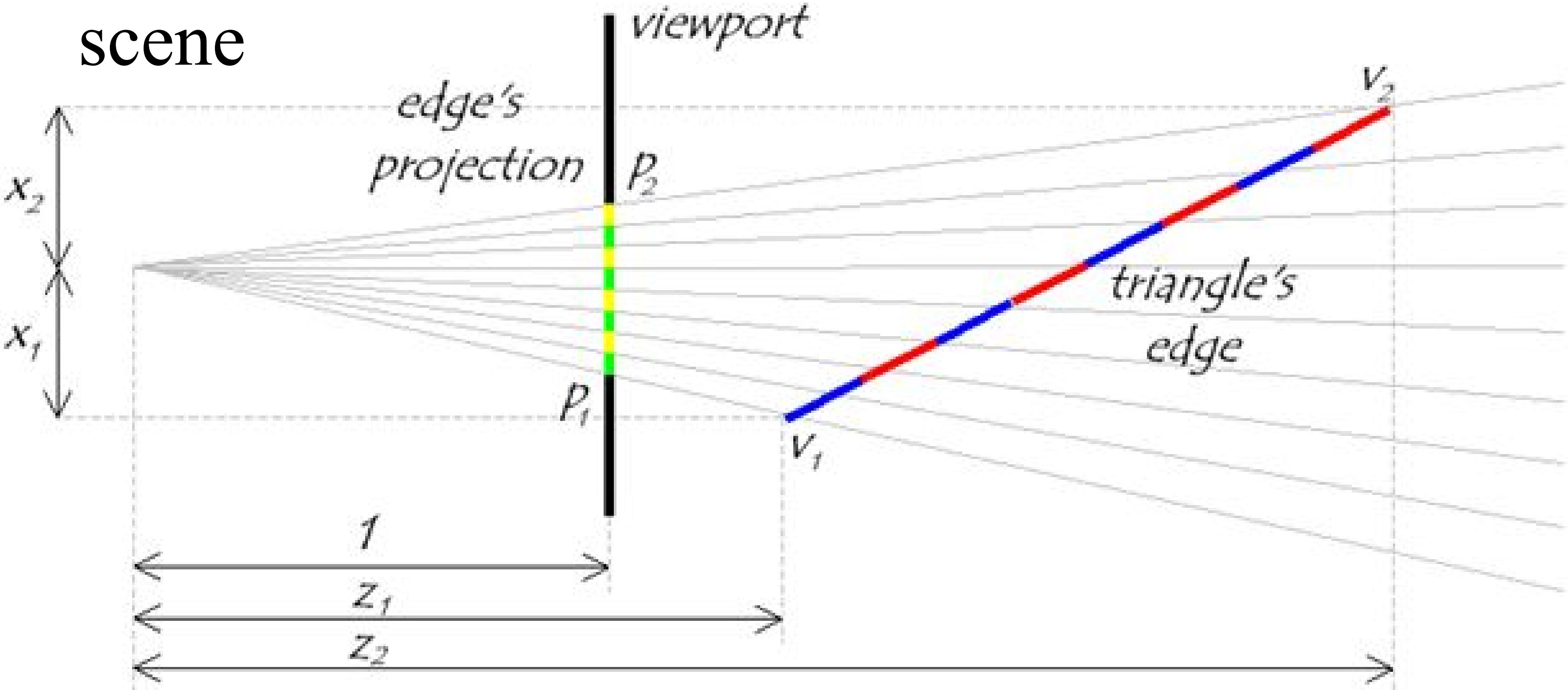
what we get|



what we want

Why Does it Happen?

- Uniform steps on the image plane does not correspond to uniform steps in the original 3D scene

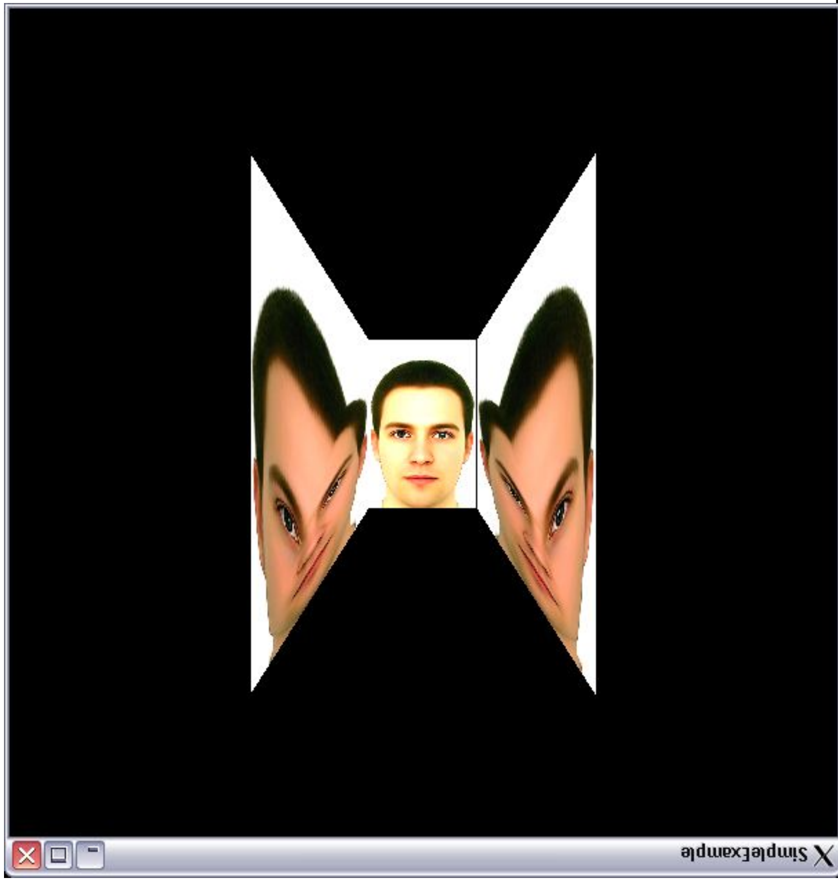


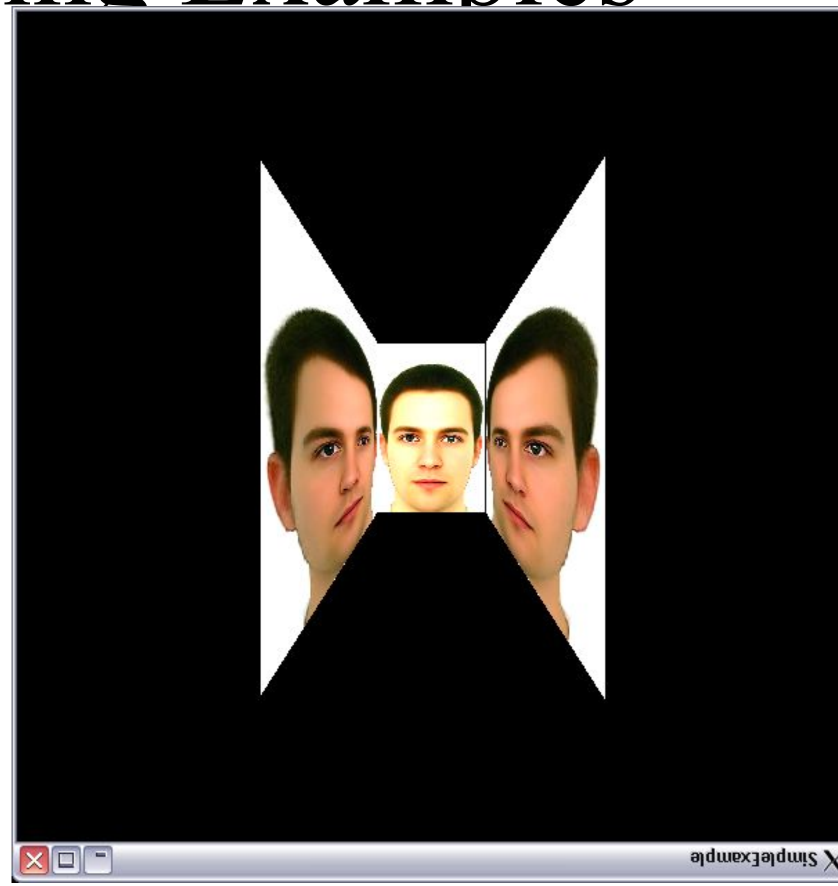
Solution : Hyperbolic Interpolation

- *(u,v)* cannot be linearly interpolated, but *1/w* and (*u/w, v/w*) can
- Compute *(u,v)* by dividing the interpolated (*u/w, v/w*) by the interpolated *1/w*
- *w* is the last component after the canonical view transformation
- A *w* is computed for every 3D vertex

$$\begin{bmatrix} x_p \\ y_p \\ z_p \\ w \end{bmatrix} = \begin{bmatrix} \frac{2n}{r-l} & 0 & -\frac{r+l}{r-l} & 0 \\ 0 & \frac{2n}{t-b} & -\frac{t+b}{t-b} & 0 \\ 0 & 0 & \frac{f+n}{f-n} & \frac{-2fn}{f-n} \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Texture Mapping Examples





- Linear interpolation vs. Hyperbolic interpolation
- Two triangles per square

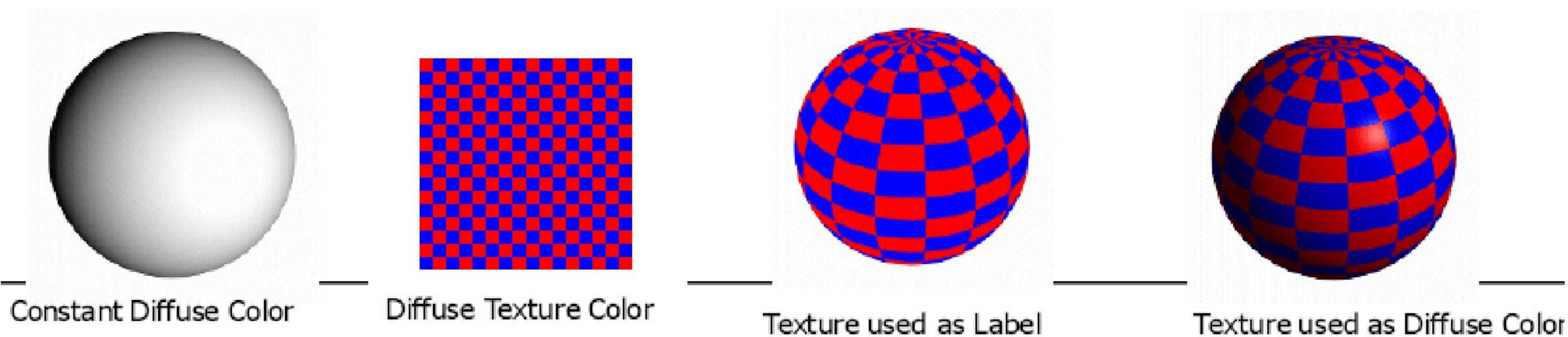
Questions

1. In what case hyperbolic interpolation is needed?
2. Then, how can we solve the problem without using hyperbolic interpolation?

Texture Mapping & Illumination

Texture mapping can be used to alter some or all of the constants in the illumination equation:

- Use the texture color as the diffuse color
  - more natural appearance



Today

- Texture mapping
- Antialiasing
- Antialiasing textures

Anti-aliasing

- Aliasing: distortion artifacts produced when representing a high-resolution signal at a lower resolution.
- Anti-aliasing : techniques to remove aliasing

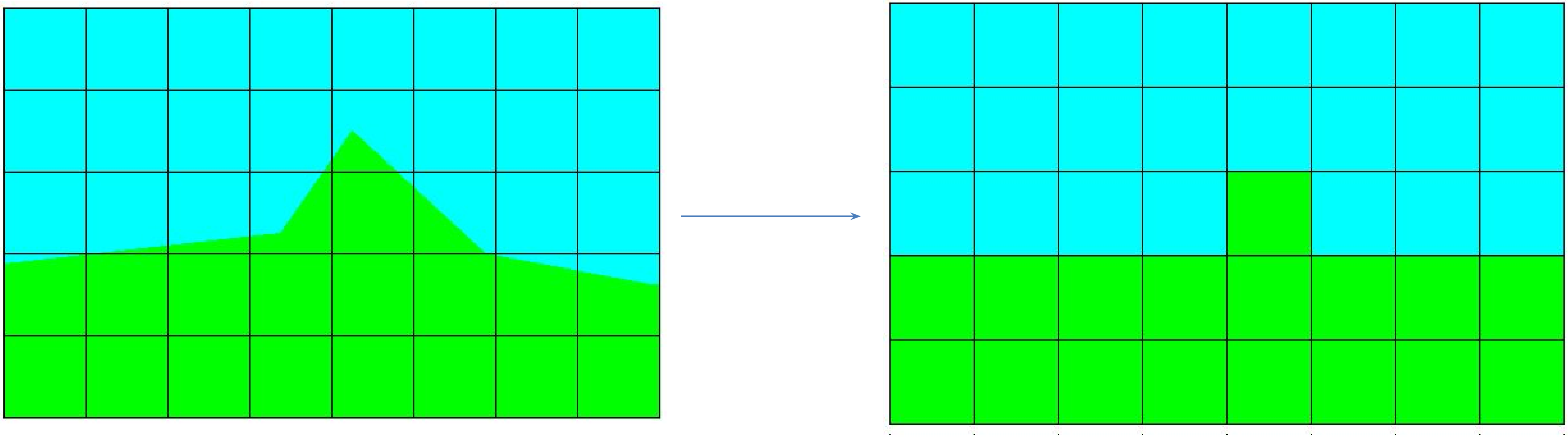
Aliased polygons (jagged edges)

Anti-aliased polygons

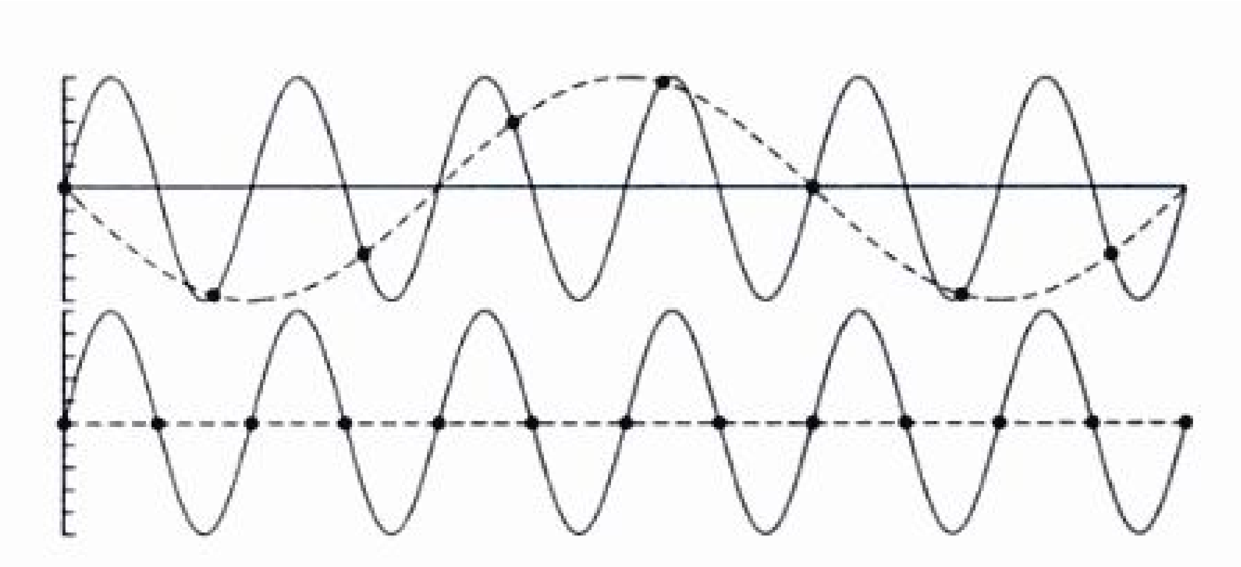
Why Does Aliasing Happen?

Sampling frequency is too low with respect to the signal frequency

In the example below, the pixel size is too large compared to the original scene



Nyquist Limit



- The signal frequency (f_{signal}) should be no greater than half the sample frequency (f_{sample})
- $f_{signal} < 0.5\ f_{sample}$
- In the top, $f_{signal} = 0.8\ f_{sample}$ -> cannot reconstruct the original signal
- In the bottom $f_{signal} = 0.5\ f_{sample}$ -> the original signal can be reconstructed by slightly increasing the sampling rate

Wagon-wheel Effect (temporal aliasing)

Anti-aliasing by Subsampling

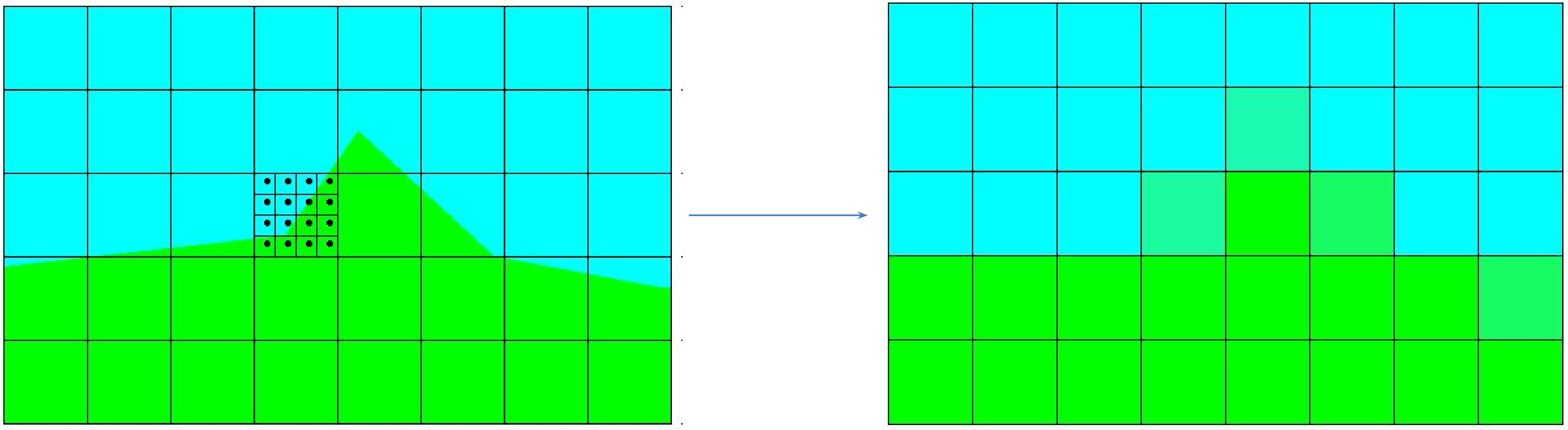
1. Each pixel is subdivided (sub-sampled) into n regions
2. Obtain the color at each subpixel
3. Compute the average color

$$p(x, y) = \sum_{i=1}^{n} w_i c(i, x, y)$$

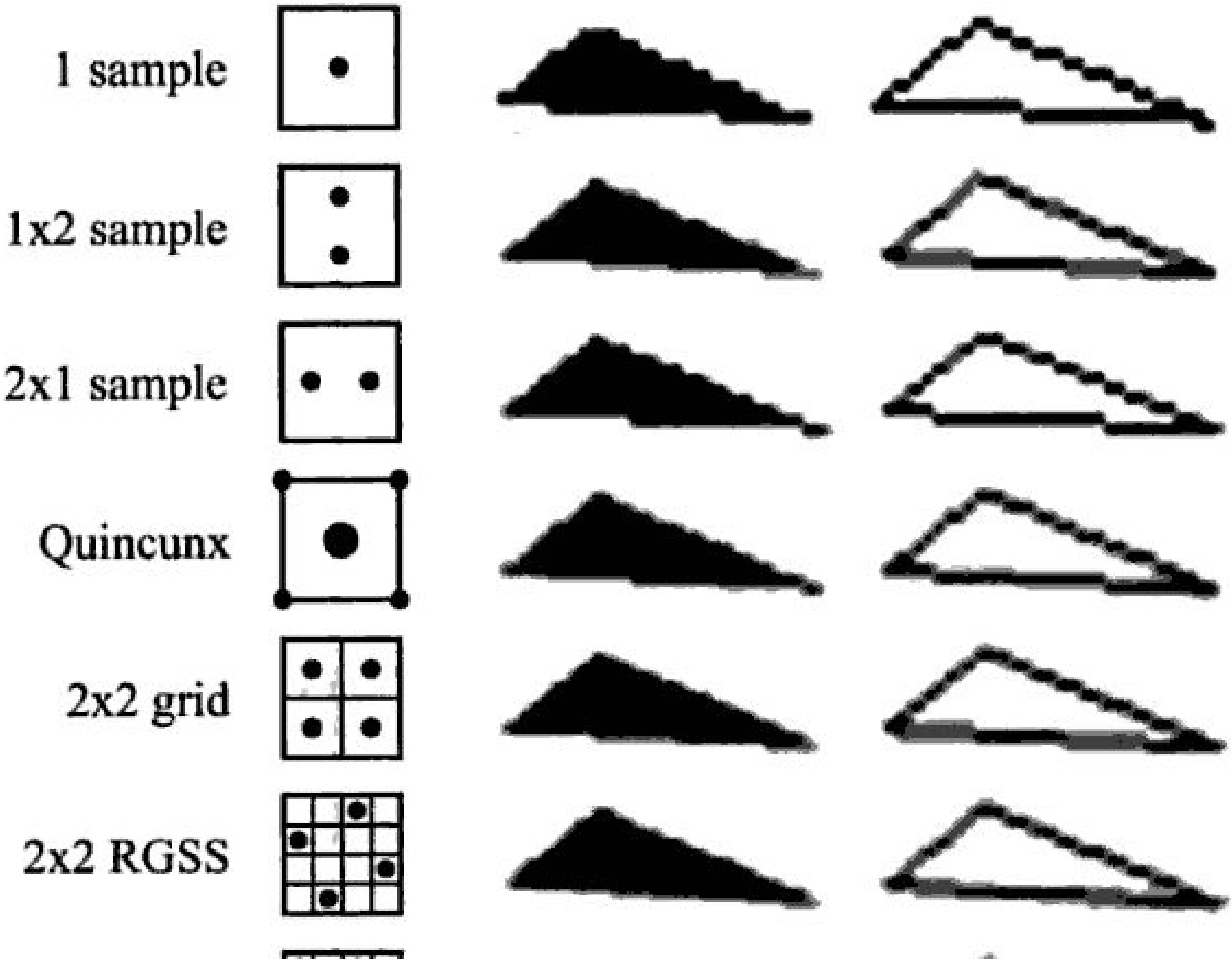
$p(x, y)$: pixel color at (x, y)

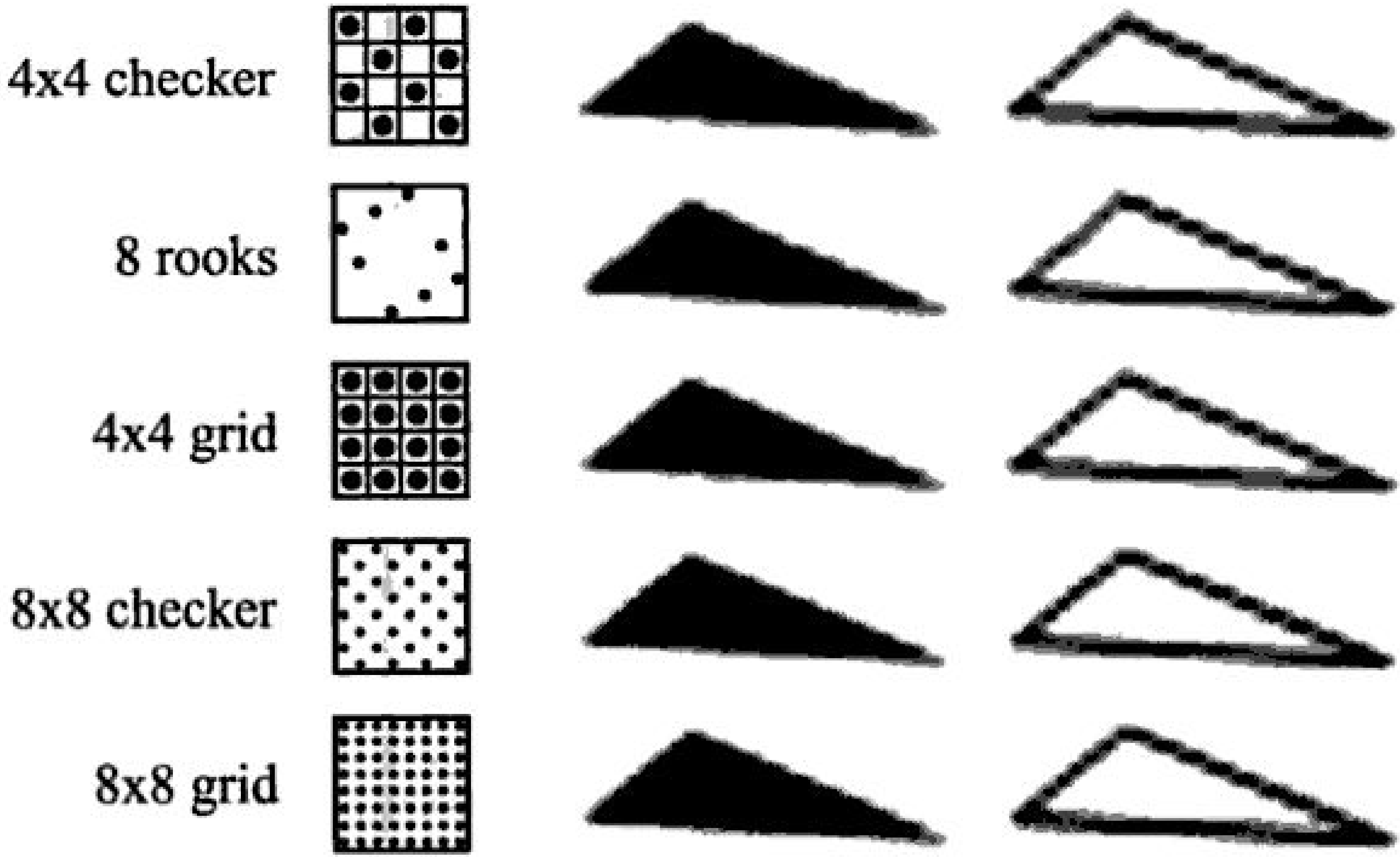
$c(i, x, y)$: sample color

w_i : weight



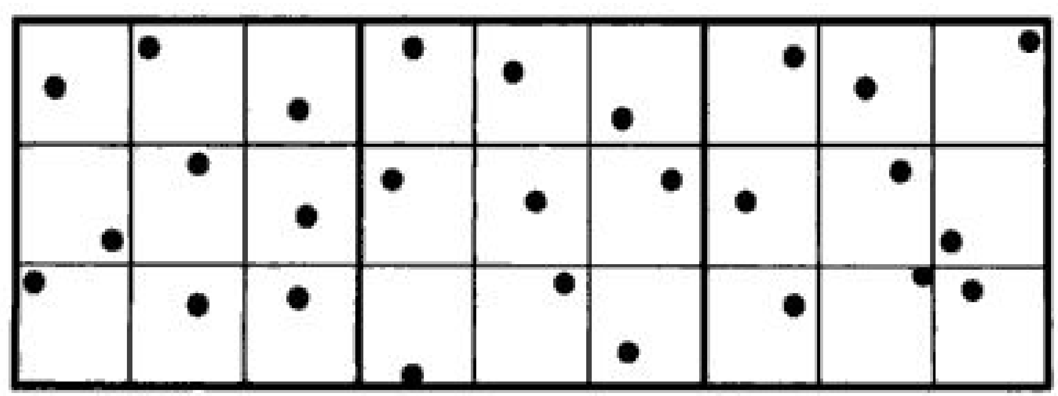
Different Sampling Schemes




4x4 checker
8 rooks
4x4 grid
8x8 checker
8x8 grid

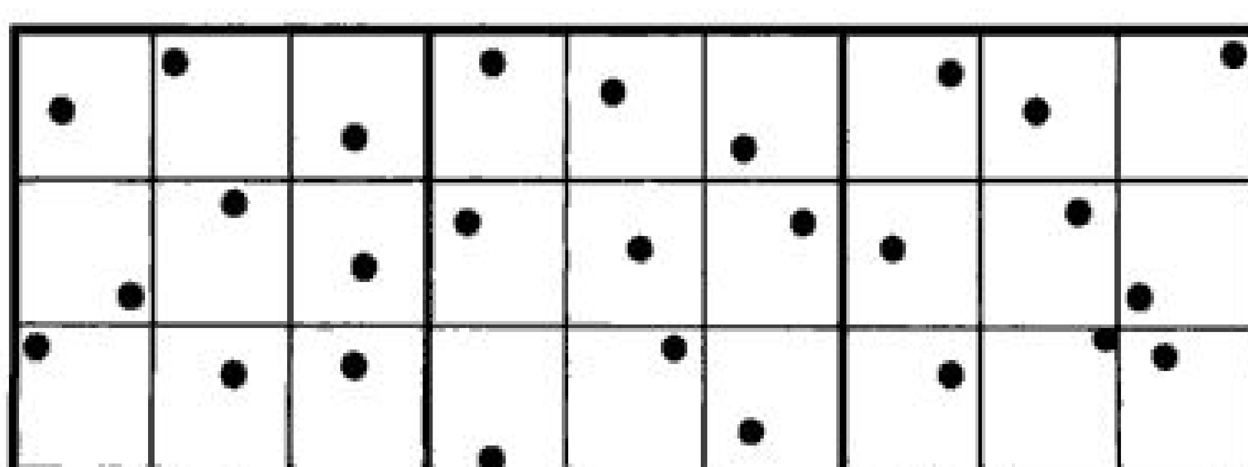
Stochastic Sampling

- A scene can be produced of objects that are arbitrarily small
- A regular pattern of sampling will exhibit some sort of aliasing
- By irregular sampling, the higher frequencies appear in the image as noise rather than aliases
- Humans are more sensitive to aliases than noise



Stochastic Sampling

- One approach to solve this is to randomly sample over the pixels
- Jittering : subdivide into *n* regions of equal size and randomly sample inside each region
- Compute the colour at the sample and average
- Can precompute a table and recycle it or simply jitter on the fly



Comparison

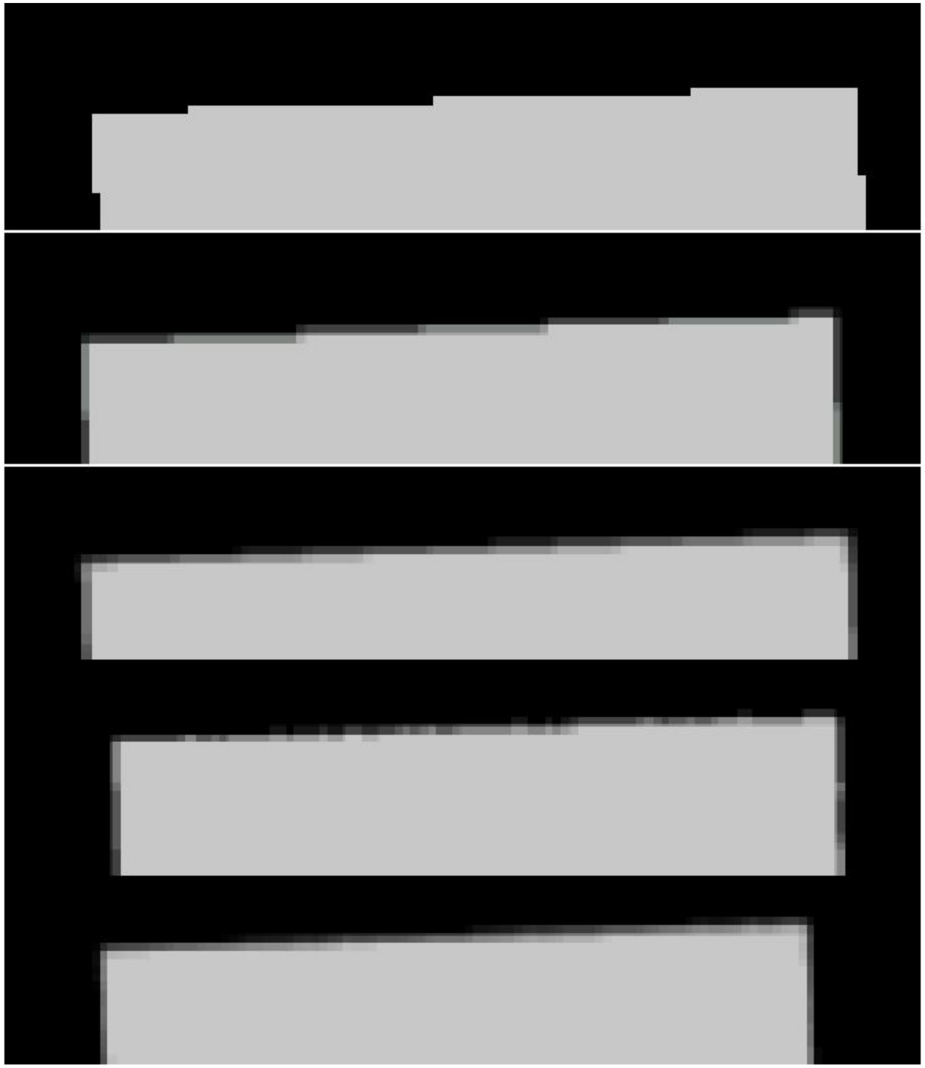
Regular, 1x1

Regular 3x3

Regular, 7x7

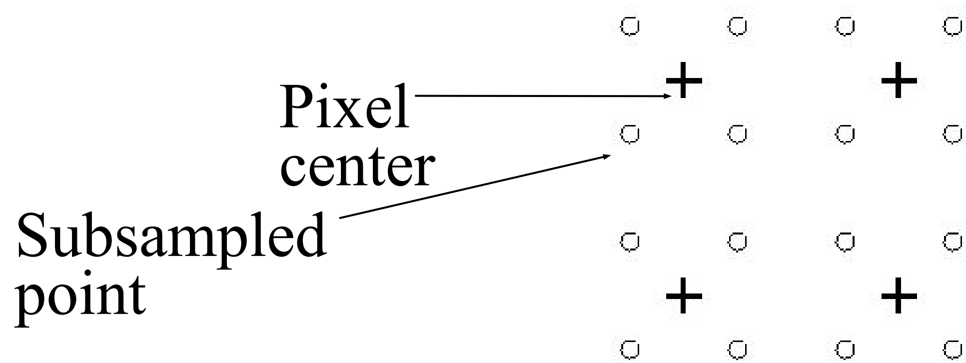
Jittered, 3x3

Jittered, 7x7



Accumulation Buffer (A-Buffer)

- Use a buffer that has the same resolution as the original image
- To obtain a 2x2 sampling of a scene, 4 images are made by shifting the frame buffer horizontally/vertically for half a pixel
- The results are accumulated and the final results are obtained by averaging
- We can recycle the vertex attributes



Today

- Texture mapping
- Antialiasing
- Antialiasing textures

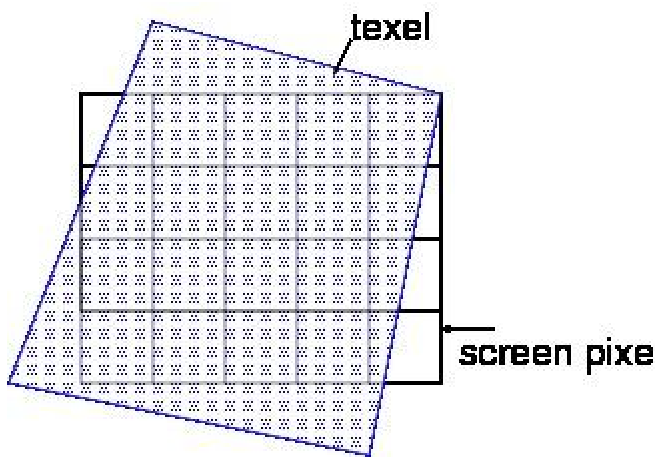
Aliasing of textures

Happens when the camera is zoomed too much into the textured surface (magnification)

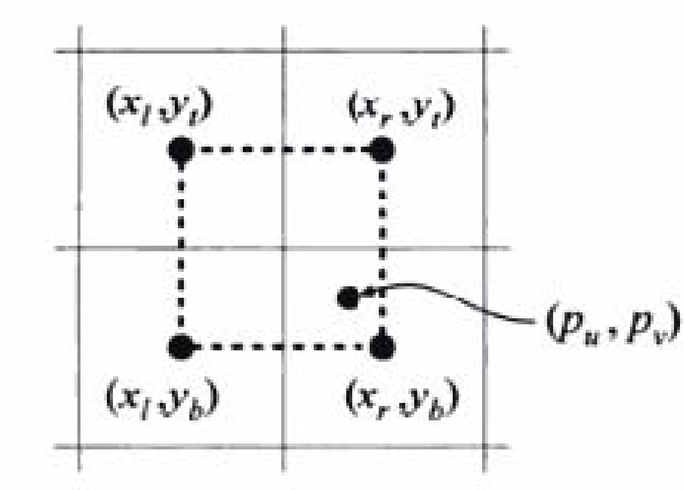
Several texels (pixels of the texture) covering a pixel's cell (minification)

Texture Magnification

Zooming too much into a surface with a texture
One texel covering many pixels



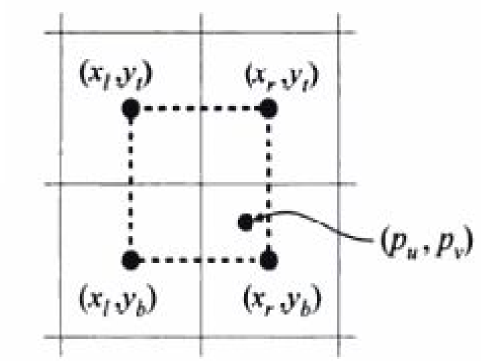
Bilinear Interpolation



- Mapping the pixel centre to the uv coordinates
- Computing the pixel colour by interpolating the surrounding texel values

Bilinear Interpolation - 2

- For (p_u, p_v), compute its (u,v) coordinates by barycentric coordinates
- u = p_u – (int)p_u, v = p_v - (int)p_v



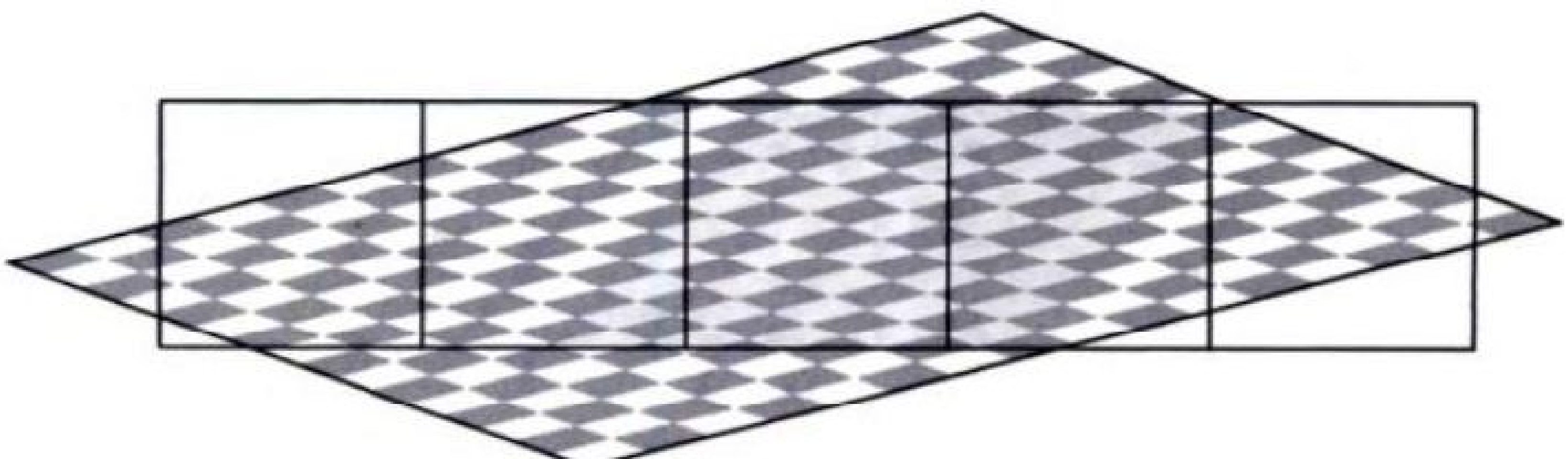
$$\mathbf{b}(p_u, p_v) = (1-u')(1-v')\mathbf{t}(x_l, y_b) + u'(1-v')\mathbf{t}(x_r, y_b) + (1-u')v'\mathbf{t}(x_l, y_t) + u'v'\mathbf{t}(x_r, y_t).$$

- (p_u, p_v) : the pixel centre mapped into the texture space
- b(p_u,p_v) : the colour at point pu, pv
- t(x,y) : the texel colour at (x,y)
- u = p_u – (int)p_u, v = p_v - (int)p_v

Texture Minification

Many texels covering a pixel's cell

Results in aliasing (remember Nyquist limit)
The artifacts are even more noticeable when the surface moves

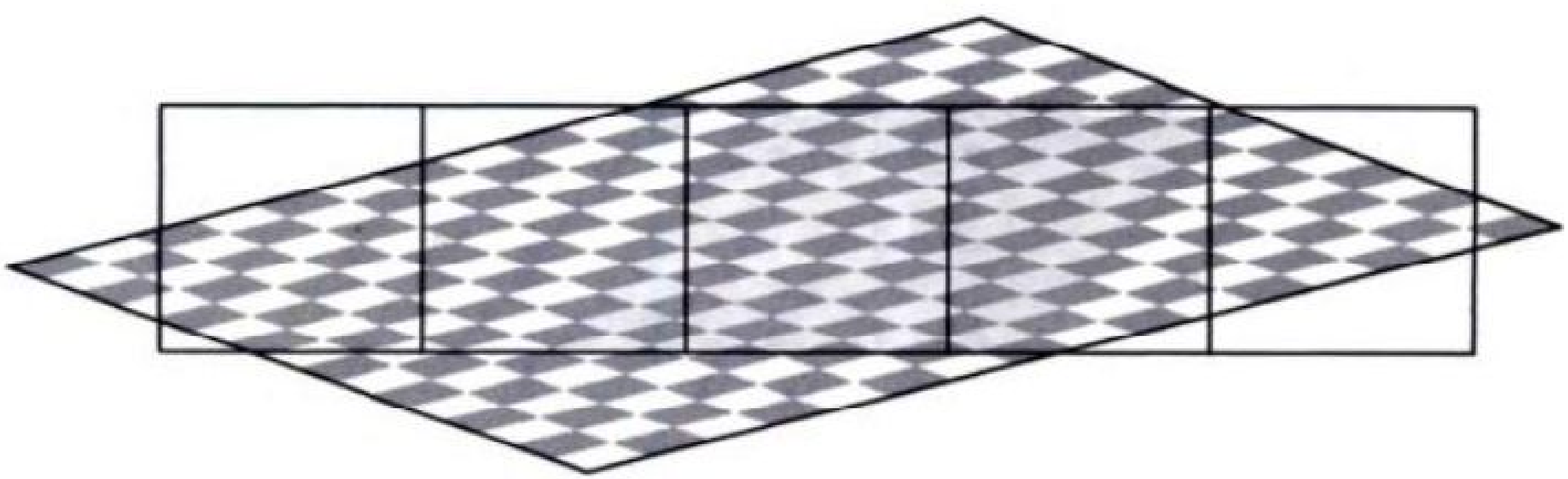


Texture Minification (2)

We can do subsampling as before

But actually we know the texture pattern in advance

→ Mipmapping



MIP map

Multum In Parvo = Many things in a small place

Produce a texture of multiple resolutions
Switch the resolution according to the number of texels in one pixel
Select a level that the ratio of the texture and the pixel is 1:1

Selecting the resolution in Mipmap

Map the pixel corners to the texture space

Find the resolution that roughly covers the mapped quadrilateral

Apply a bilinear interpolation in that resolution,

Or find the two surrounding resolutions and apply a trilinear interpolation (also along the resolution axis)

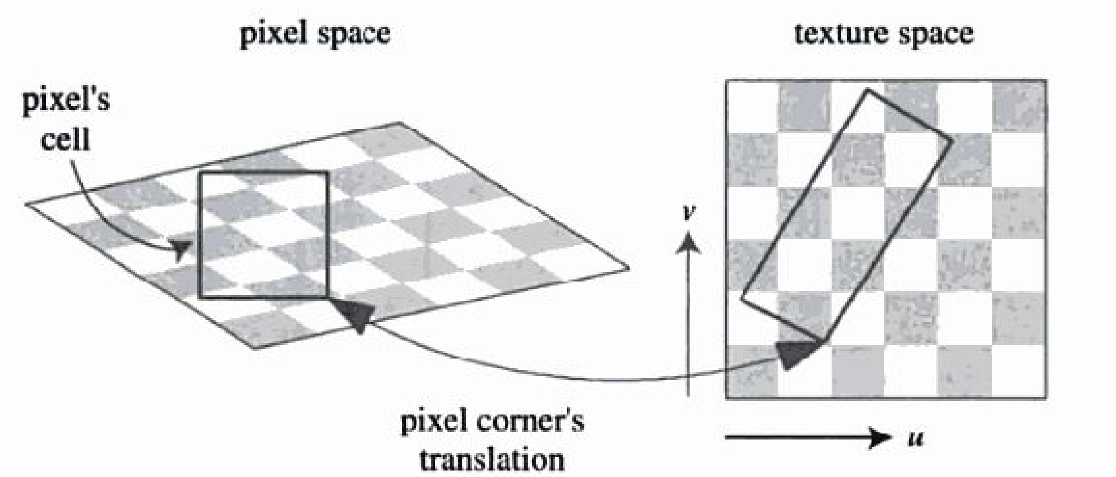
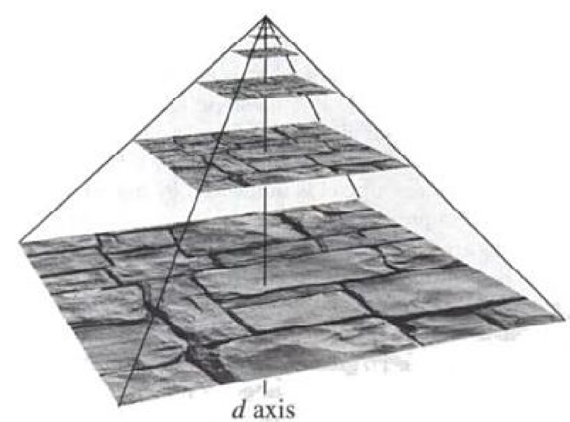
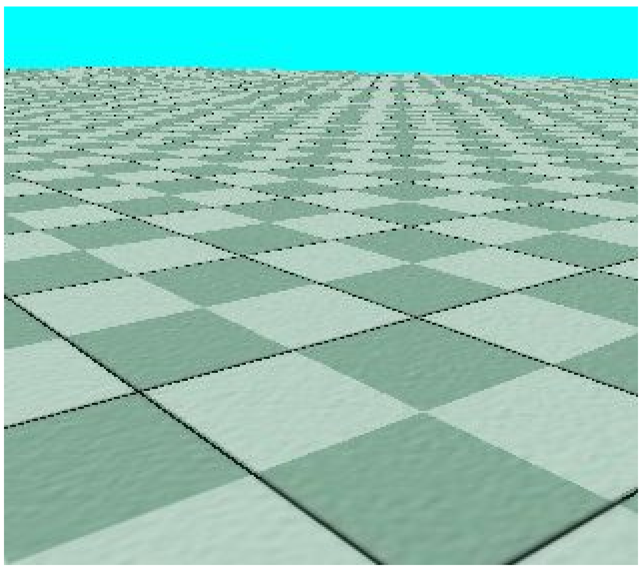
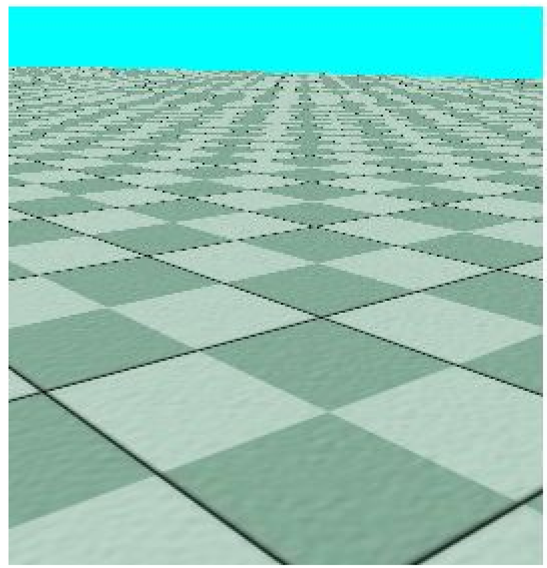
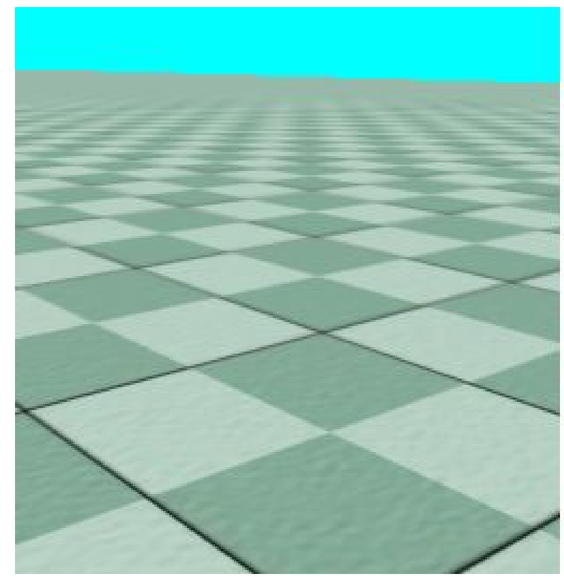


Figure 5.13. On the left is a square pixel cell and its view of a texture. On the right is the projection of the pixel cell onto the texture itself.



Texture Minification

Multiple textures in a single pixel
Solution:

Nearest neighbour Bilinear blending Mipmapping

Summary

1. Texture mapping
   a. Different ways to synthesize the texture maps
   b. Hyperbolic interpolation
2. Antialiasing
   a. Nyquest limit
   b. Subsampling
   c. texture maps
      i. magnification : bilinear interpolation
      ii. minification : mipmapping

Reading List

Shirley et al. Chapter 11.

Akenine-Moller Chapter 5.6, 6.2

http://books.google.co.uk/books?id=V1k1V9Ra1FoC&pg=PA418&dq=real-time+rendering+texture+mapping&hl=ja&sa=X&ei=ErdXUtCBBsjA0QWazYHoAw&redir_esc=y#v=onepage&q=real-time%20rendering%20texture%20mapping&f=false