Computer Graphics 8 - Texture mapping, bump mapping and antialiasing

Tom Thorne

Slides courtesy of Taku Komura
www.inf.ed.ac.uk/teaching/courses/cg

Overview

- **Texture mapping and bump mapping**
- Anti-aliasing

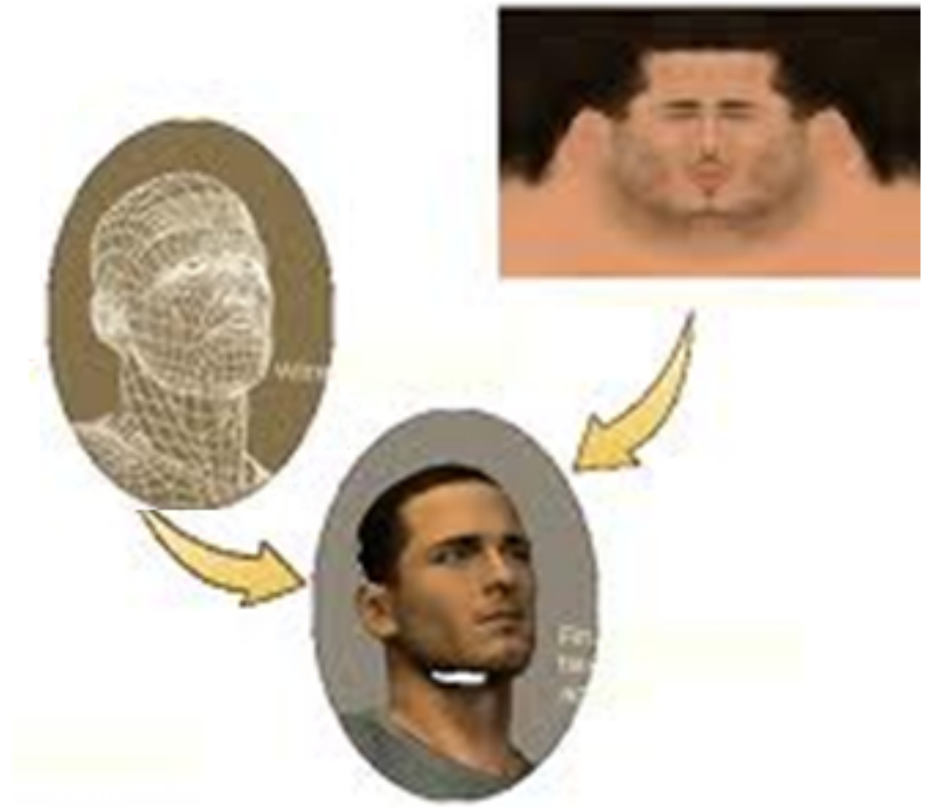
Texture mapping

- Adding detail to polygon meshes to build high resolution models is computationally expensive
- In realtime applications these are not practical to use

by Rami Ali Al-ashqar

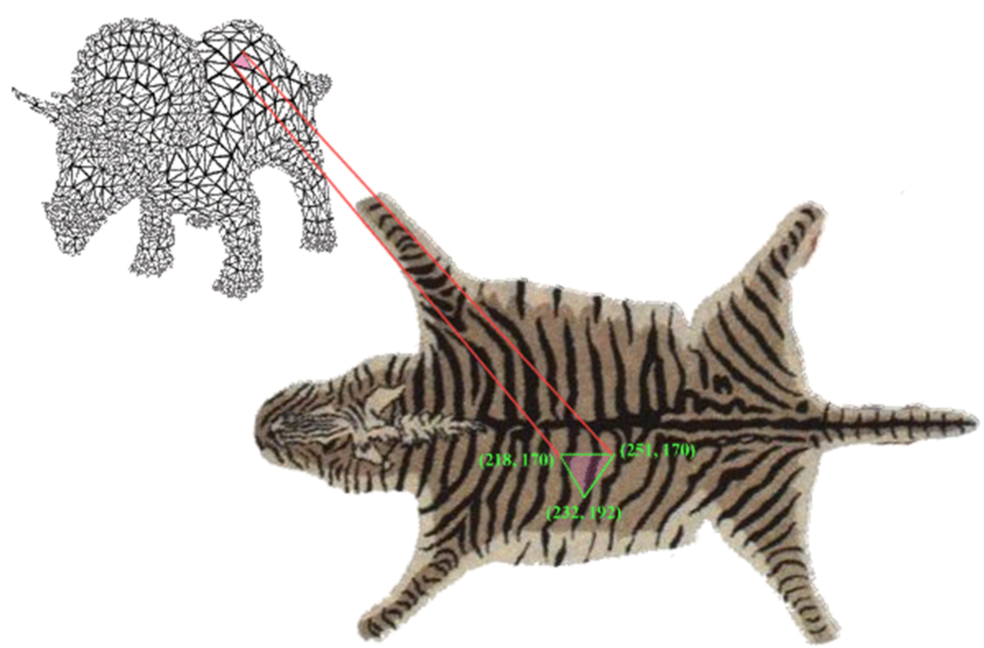
Texture mapping

- We can improve the appearance of polygons by mapping images onto their surface
- This is done during rasterisation



Overview of texture mapping

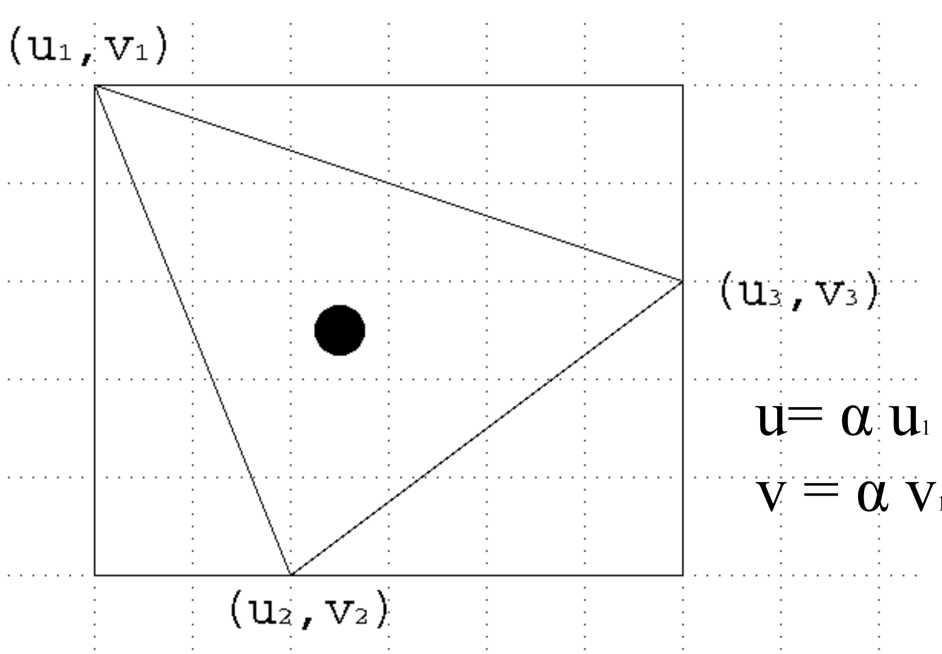
- Assign each triangle in the mesh a region of the image
- During rasterisation, colour the surface of the triangle based on the pixels in the image
- Typically done by assigning triangle vertices coordinates (uv) within the image



Interpolation of uv coordinates

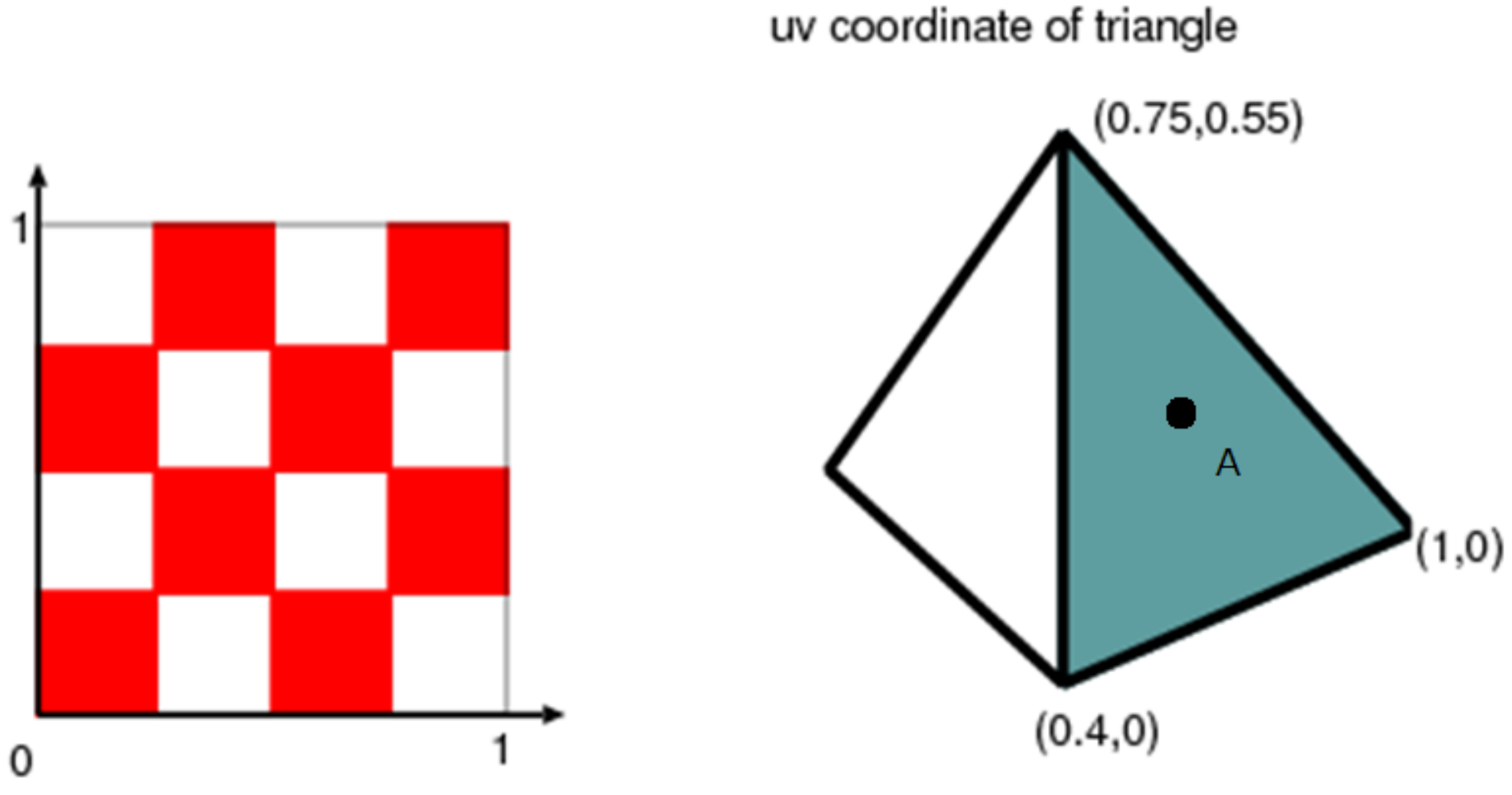
- To calculate which pixel in the image corresponds to a point inside the triangle, we interpolate uv coordinates between vertices

- This can be done using barycentric coordinates



$$u = \alpha u_1 + \beta u_2 + \gamma u_3$$
$$v = \alpha v_1 + \beta v_2 + \gamma v_3$$

Example



Producing uv mappings

- Generated based on the mesh - cylindrical, spherical, orthogonal
- Captured from real objects by scanning
- Manually specifying coordinates

Common uv mappings

- Orthogonal
- Cylindrical
- Spherical

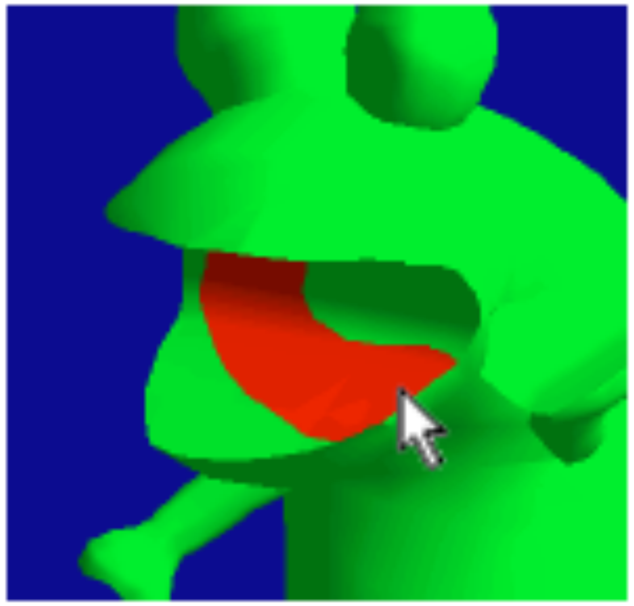
Capturing real data

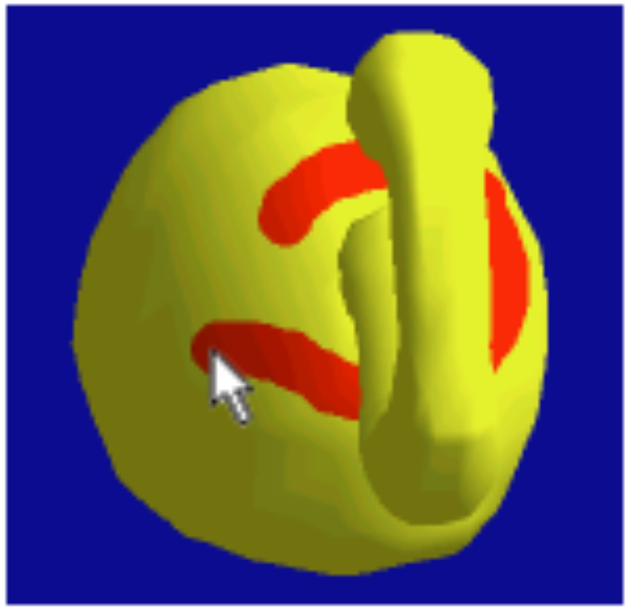
- Scanning depth and colour of objects

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

Manual specification

- Painting on the geometry (Z-brush)
- Manually aligning an unfolded model on the image





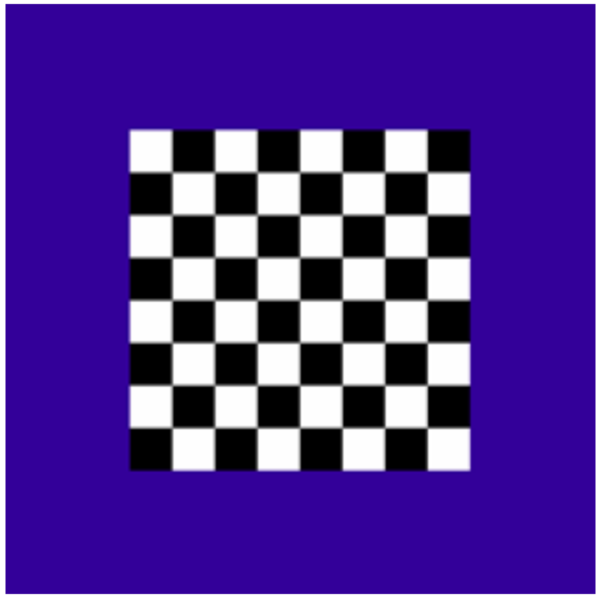
Geometry unfolding

- Segment the mesh into regions that are arranged in a 2D plane
- Draw on these regions in 2D to texture the mesh

Levy et al. SIGGRAPH 2002

Linear interpolation

- Linearly interpolating uv coordinates does not produce the expected results

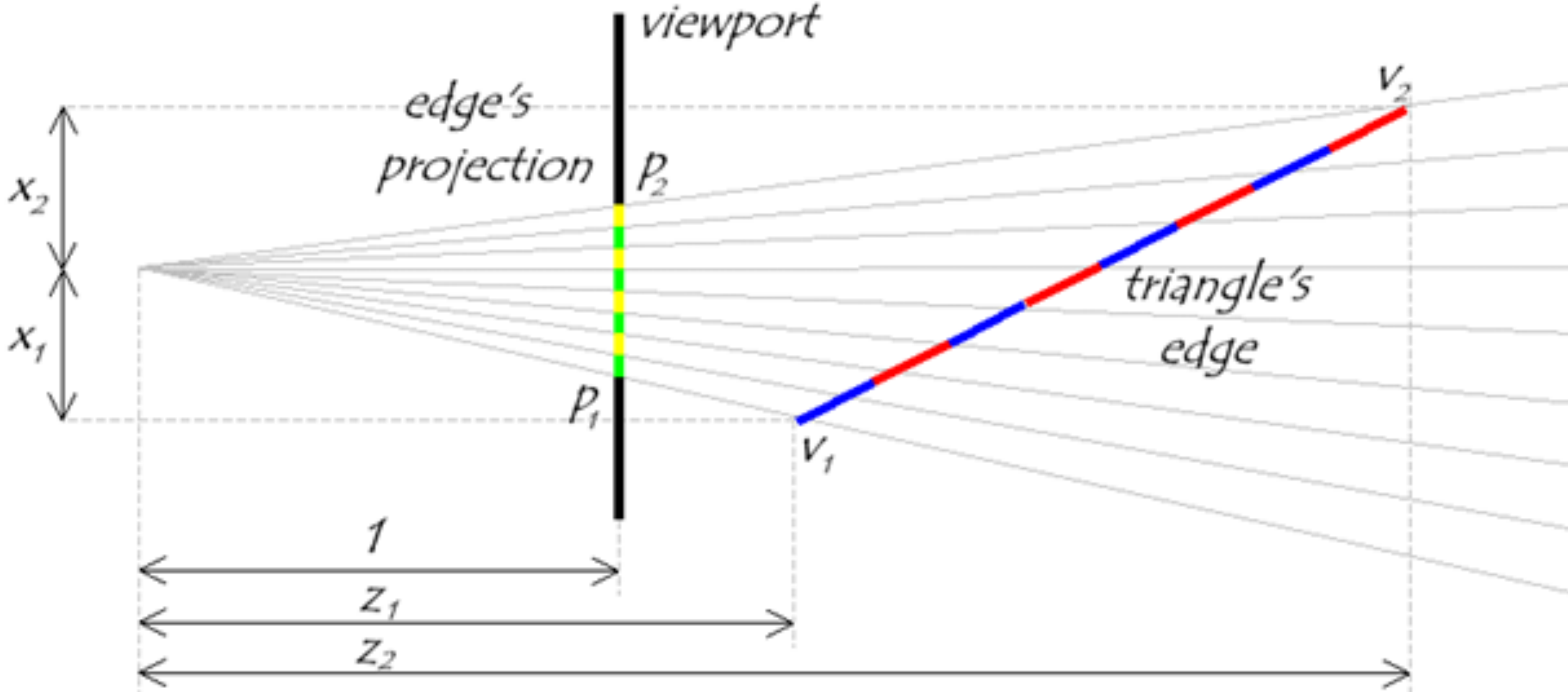

texture source


what we get|


what we want

Why does this happen?

- Uniform steps in 2D screen space do not correspond to uniform steps over the surface of the triangle



Hyperbolic interpolation

- To interpolate between vertices P and Q, rather than interpolating between u(P) and u(Q), we interpolate u(P)/z, u(Q)/z and 1/z

- Considering a point M between P and Q and its projection M':

$$P' = -P/z_P, Q' = -Q/z_Q$$
$$M' = \alpha P' + (1 - \alpha) Q'$$

Hyperbolic interpolation

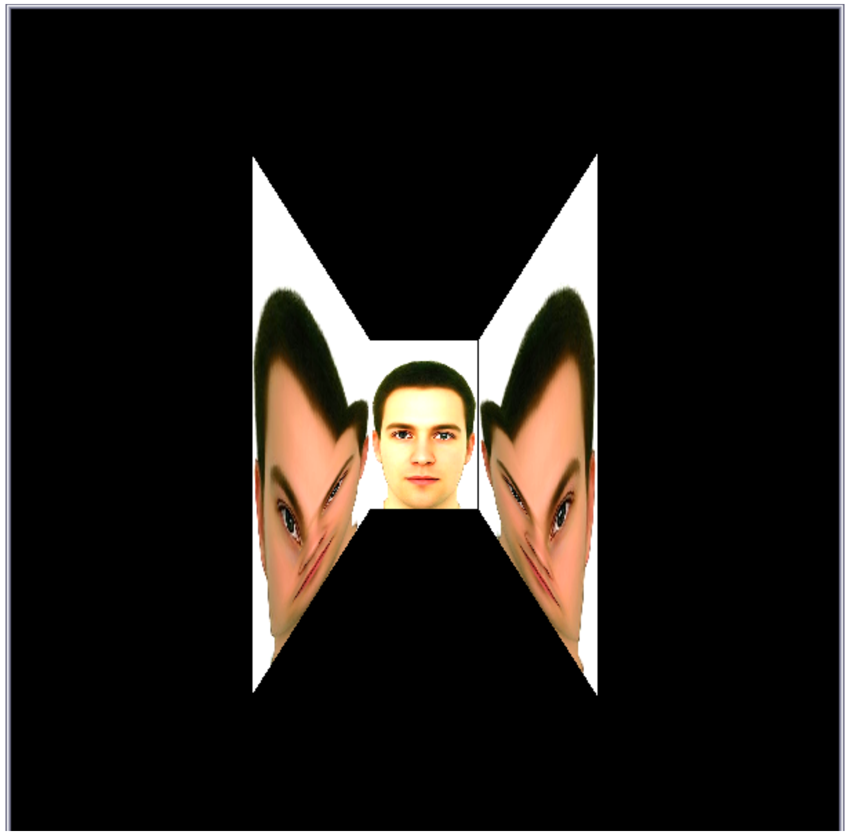
- For a point M the perspective correct interpolated u(M) is then:

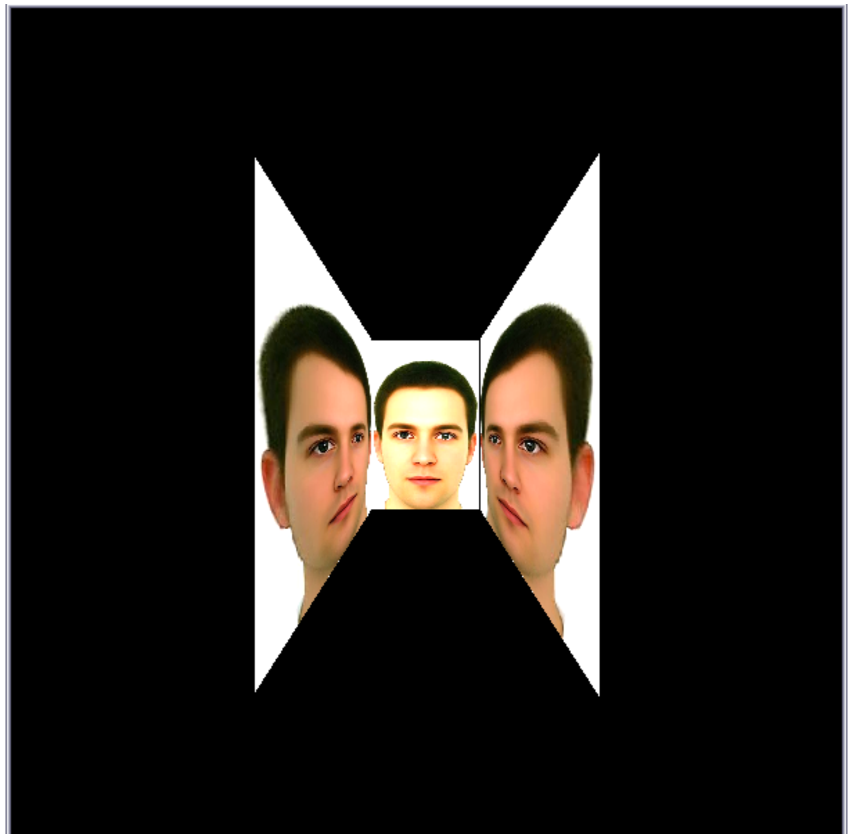
$$\frac{u(M)}{-z_M} = \alpha \frac{u(P)}{-z_P} + (1-\alpha) \frac{u(Q)}{-z_Q}$$

$$u(M) = \frac{\alpha \frac{u(P)}{-z_P} + (1-\alpha) \frac{u(Q)}{-z_Q}}{\alpha \frac{1}{-z_P} + (1-\alpha) \frac{1}{-z_Q}}$$

Example

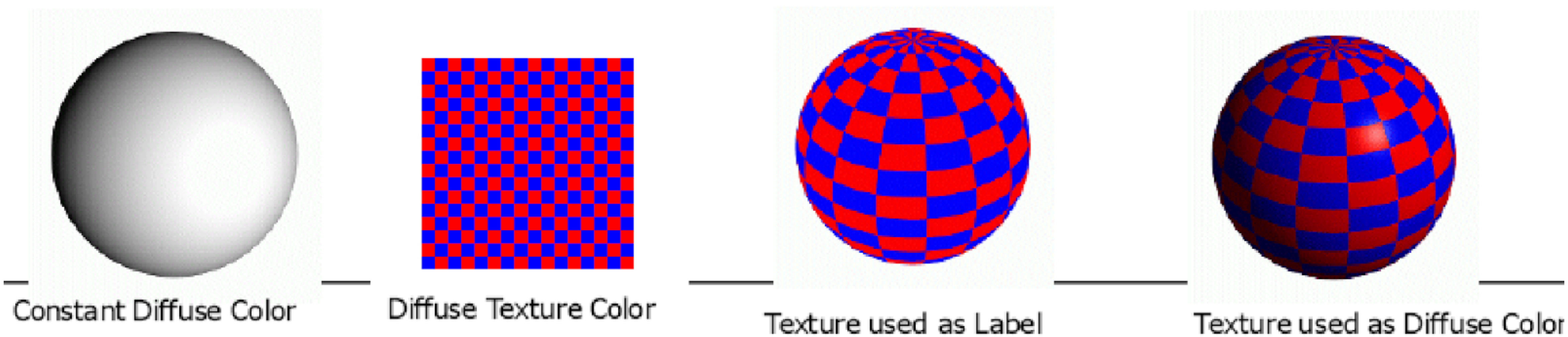
- Linear interpolation vs hyperbolic interpolation





Texture mapping and illumination

- We can use texture information to alter illumination

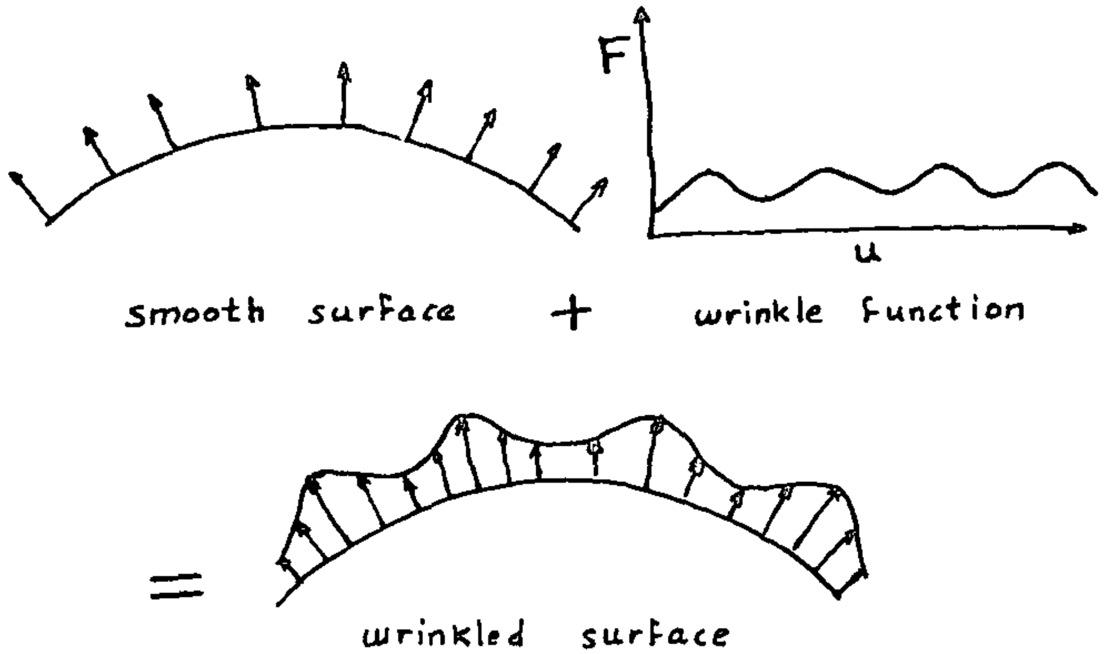


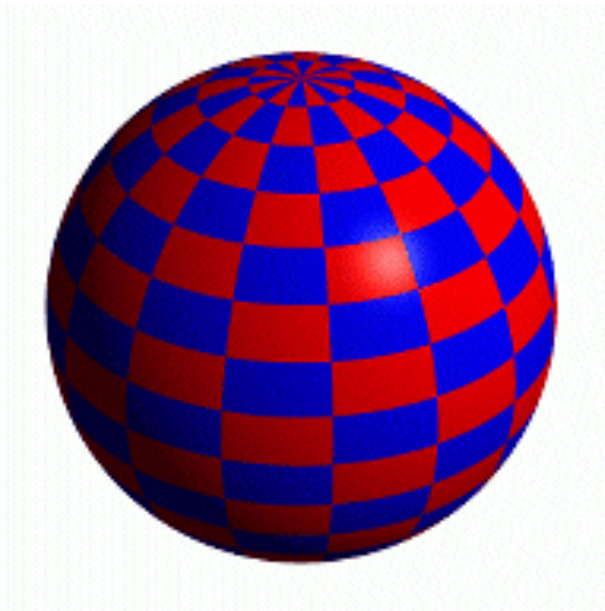
Bump mapping

- Texture mapping leaves the surface looking flat
- Adding extra mesh detail can be computationally too expensive

Bump mapping

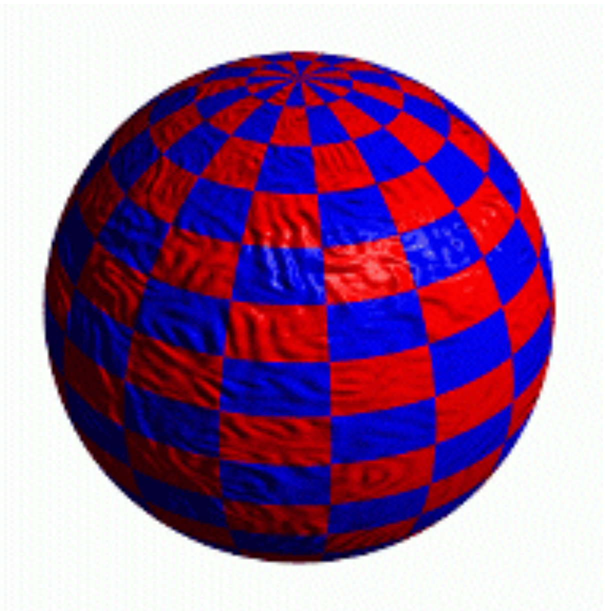
- Uses a texture to alter the surface normal to change the illumination
- Applied during rasterisation





Sphere w/Diffuse Texture

Swirly Bump Map



Sphere w/Diffuse Texture & Bump Map

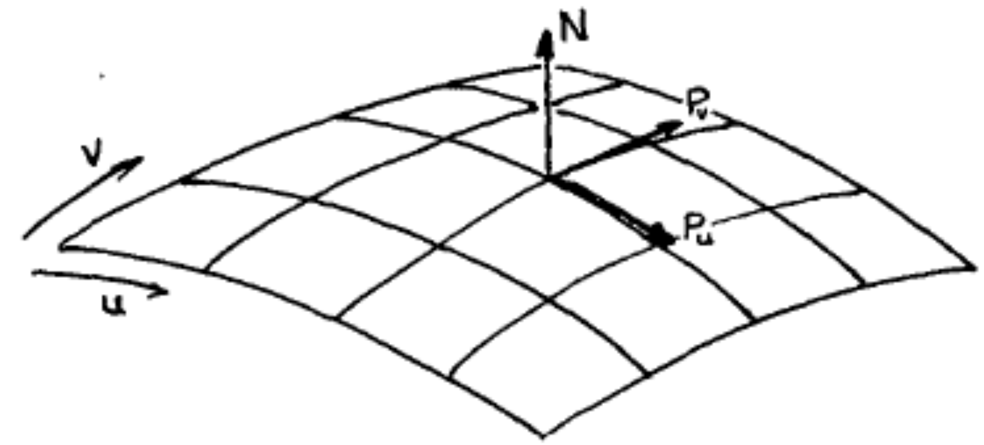
Bump mapping

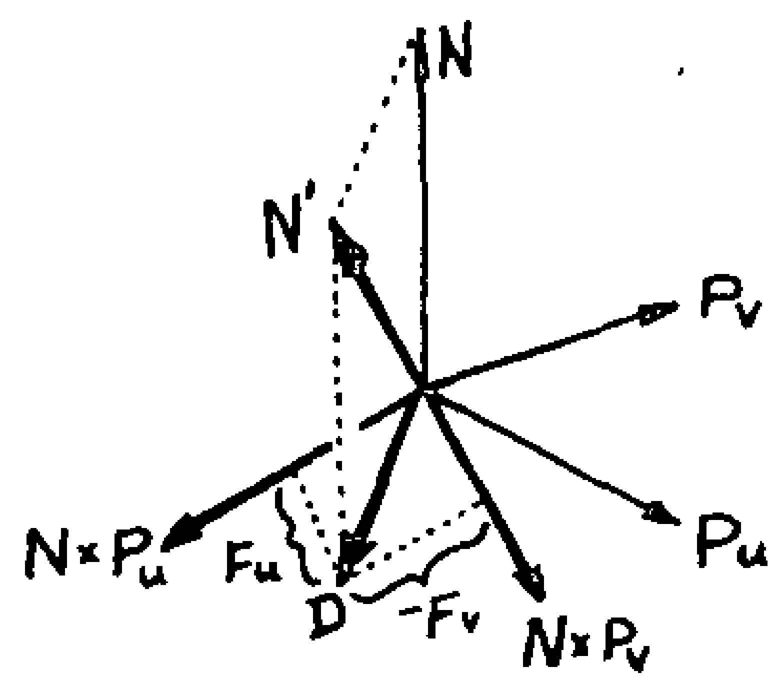
- Calculate finite differences from the bump map

$$F_u = \frac{dF}{du}, F_v = \frac{dF}{dv}$$

- Calculate a perturbed normal vector

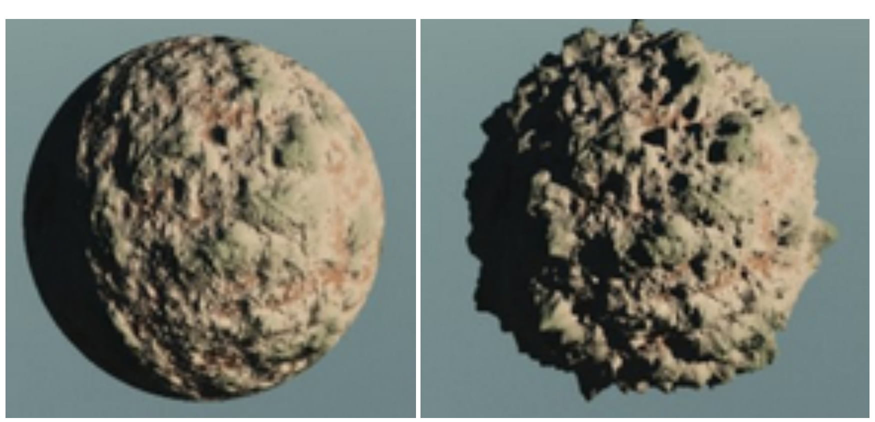
$$\mathbf{n}' = \mathbf{n} + \frac{F_u(\mathbf{n} \times \mathbf{P_v}) - F_v(\mathbf{n} \times \mathbf{P_u})}{||\mathbf{n}||}$$

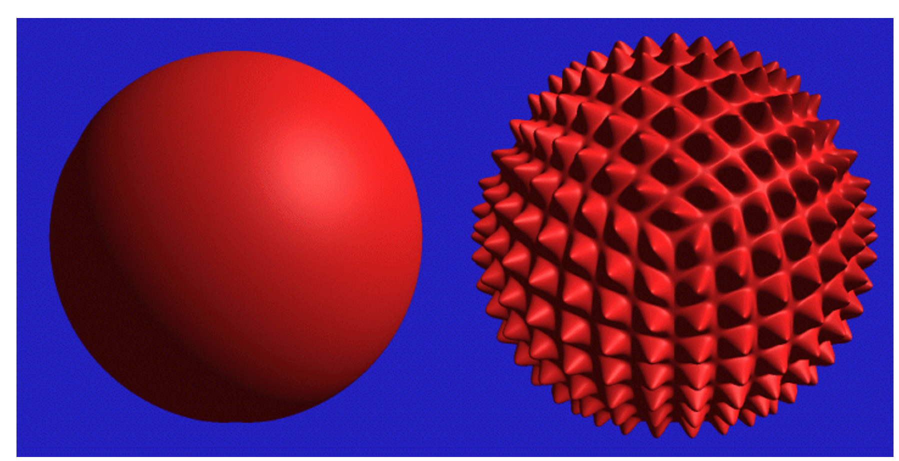




Displacement mapping

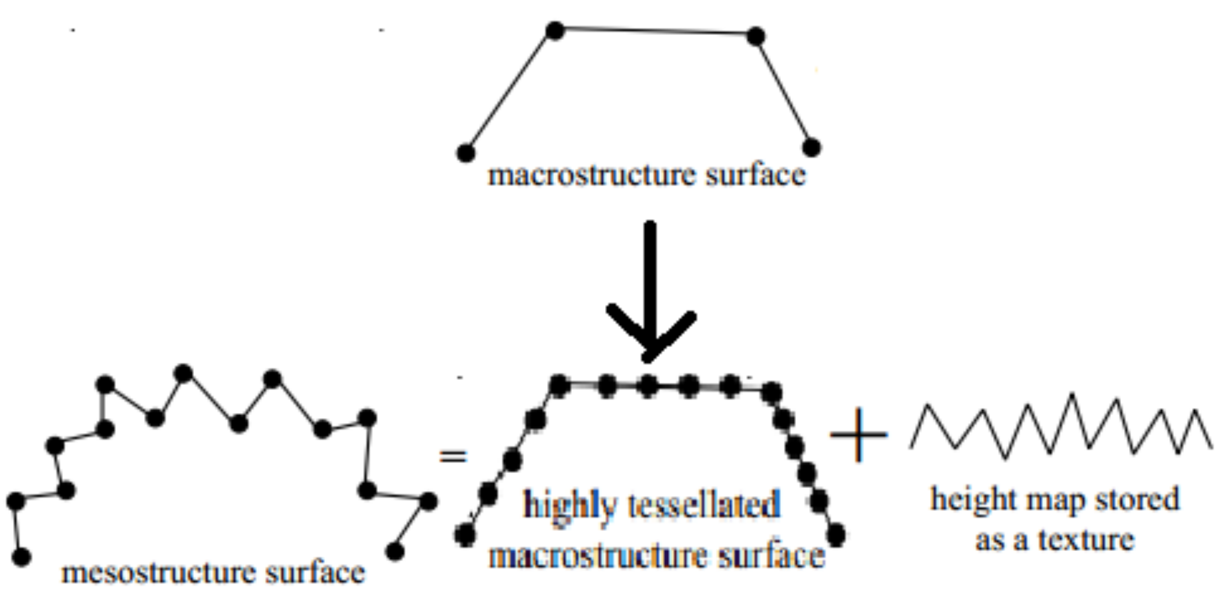
- Bump mapping only changes shading of the surface, not the actual geometry
- Displacement mapping alters the surface using the texture as a mesh defined in uv coordinates

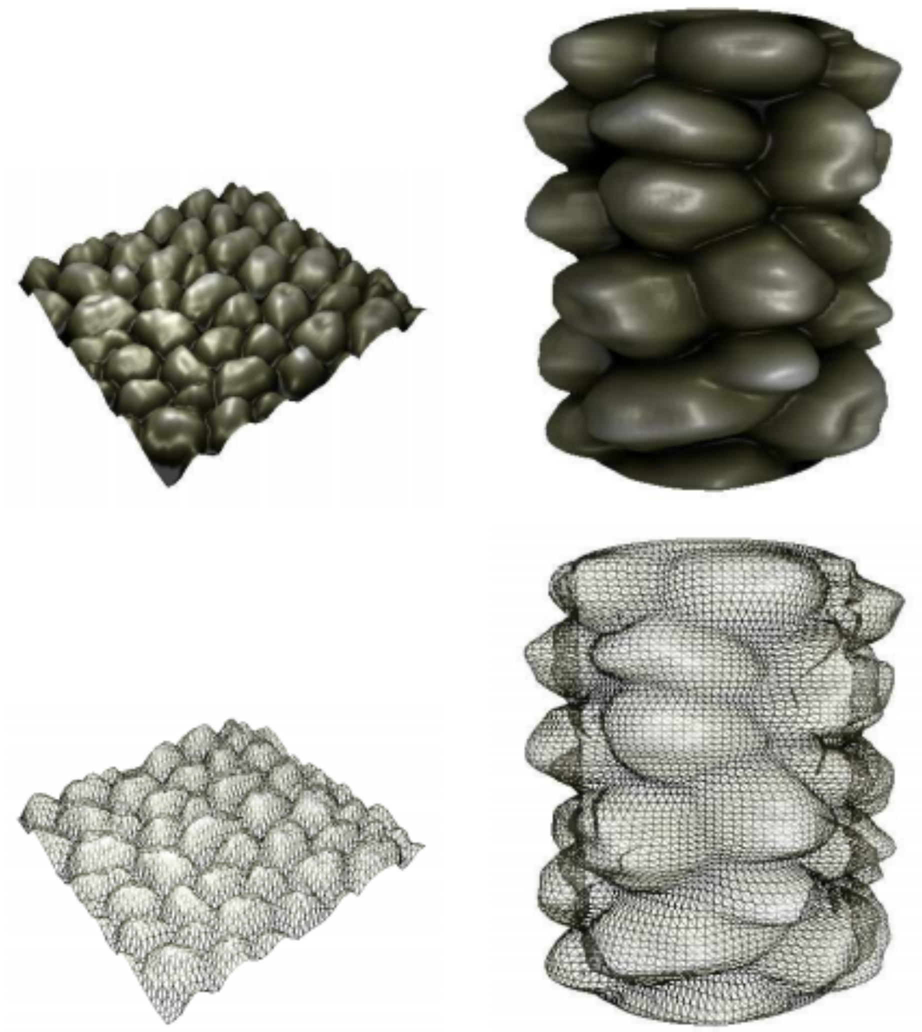




Displacement mapping

- Subdivide the surface to resolution of texture
- Displace vertices in normal direction of surface by height in displacement map





Overview

- Texture mapping
- Bump mapping
- **Anti-aliasing**

Anti-aliasing

- Aliasing is the distortion produced by representing a high resolution signal at a lower resolution

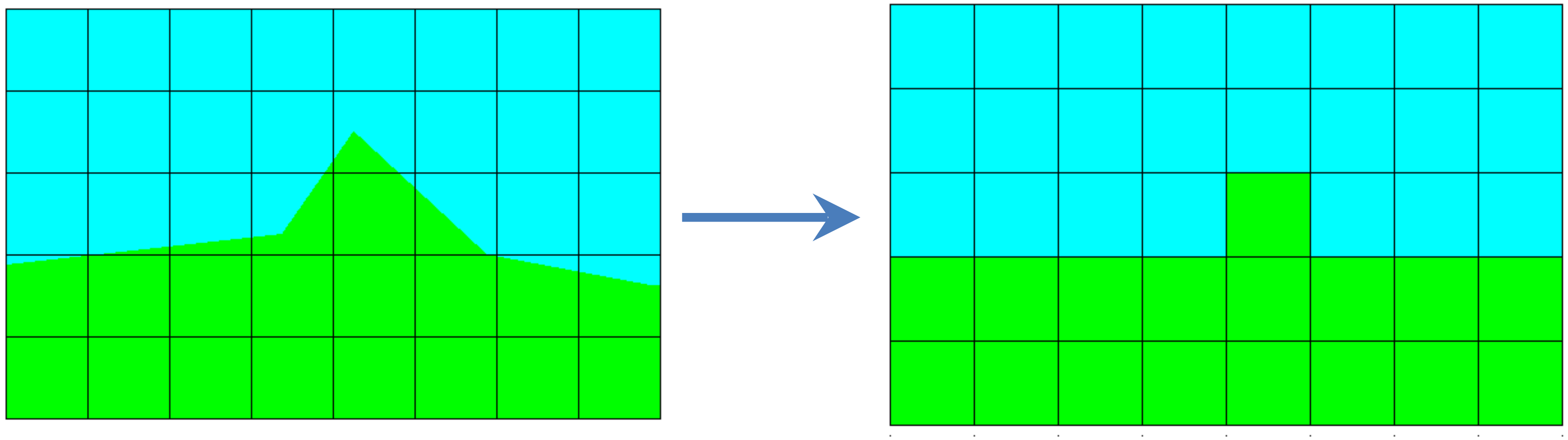
- Anti-aliasing aims to remove this distortion

Aliased polygons (jagged edges)

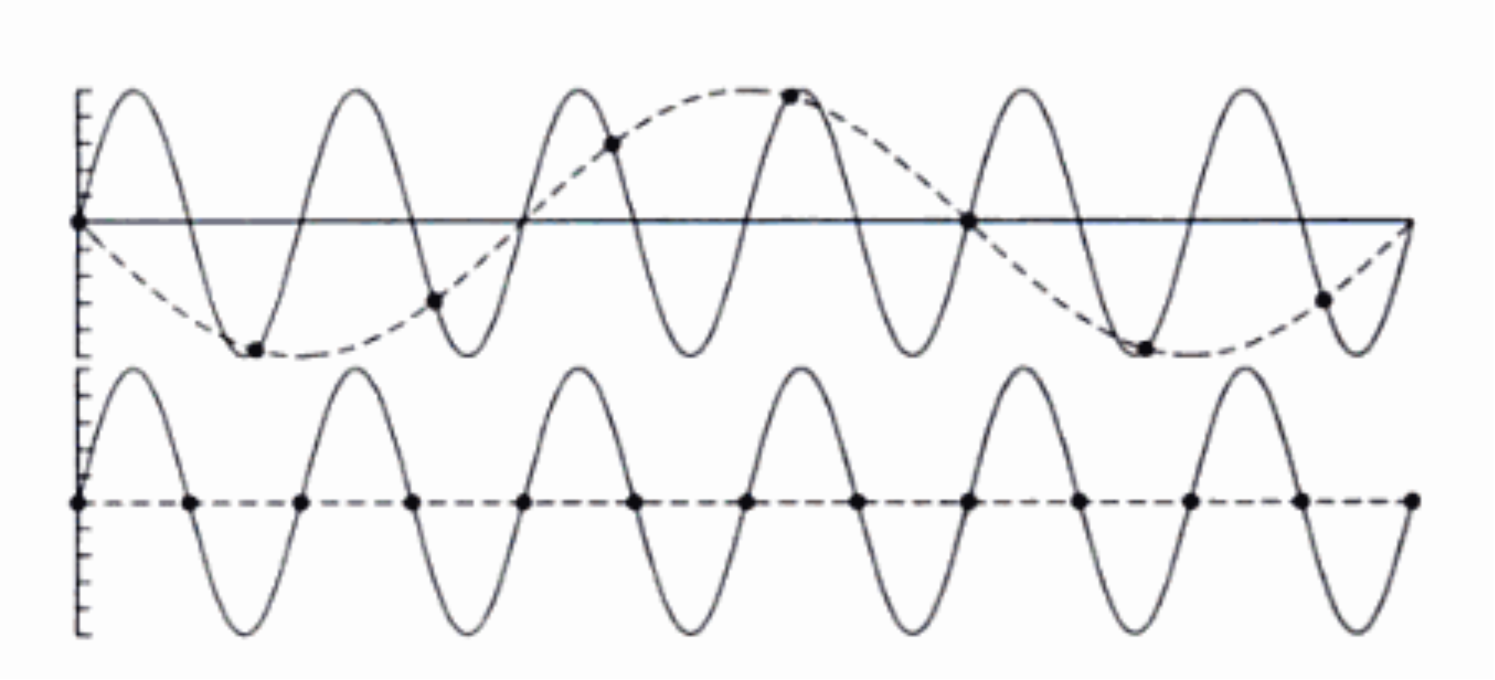
Anti-aliased polygons

Why does aliasing happen?

- When the sampling frequency is too low to represent the signal



Nyquist limit



$$f_{signal} = 0.8 f_{sample}$$

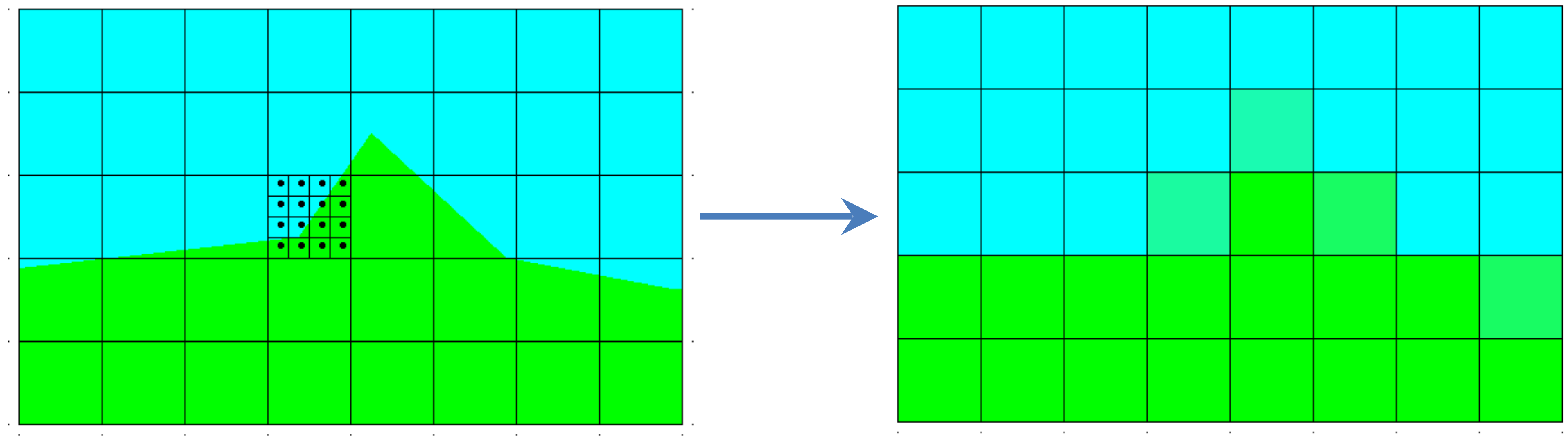
$$f_{signal} = 0.5 f_{sample}$$

- To reproduce a signal, the sampling frequency should be twice the signal frequency

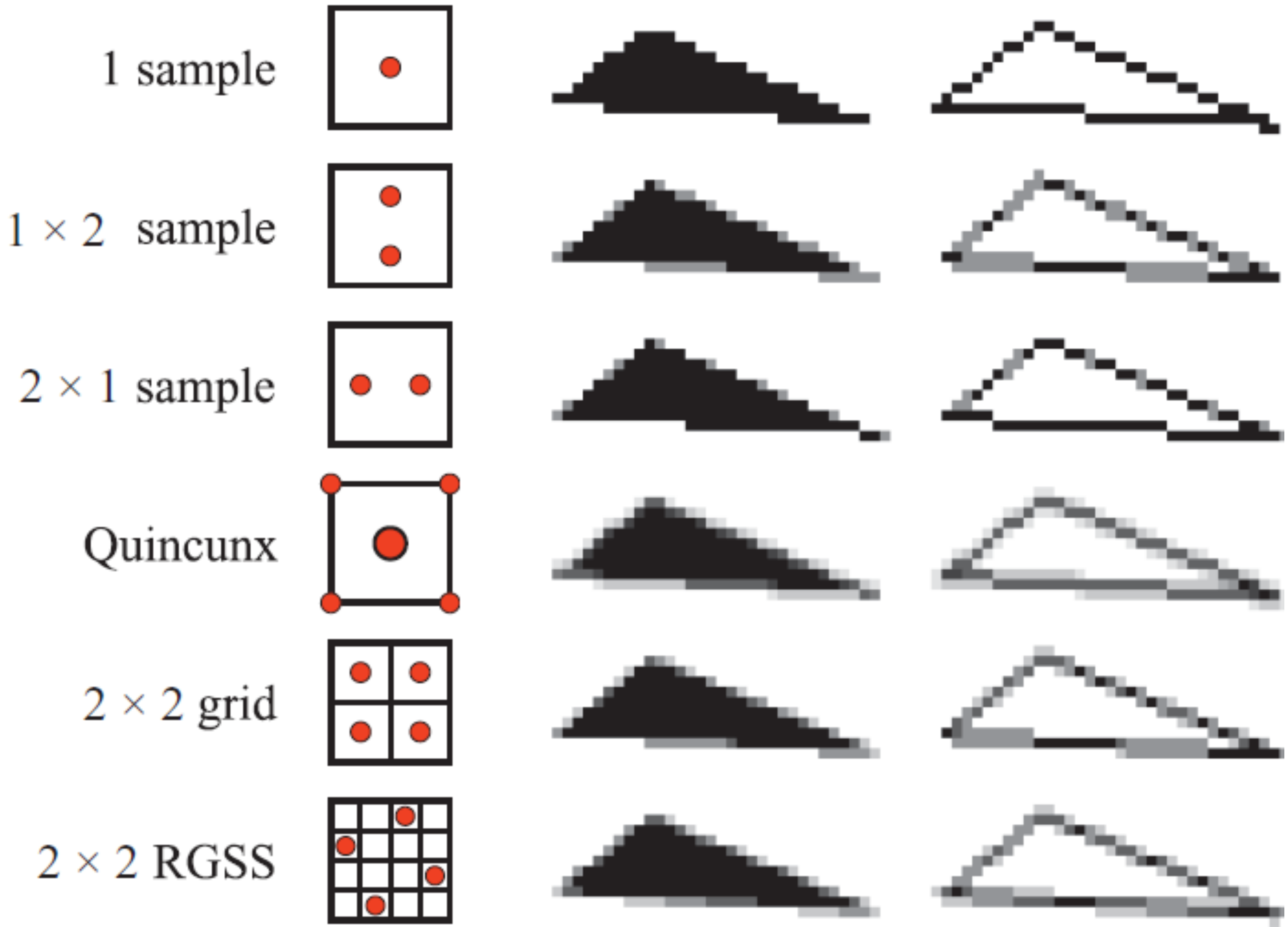
$$f_{signal} < 0.5 f_{sample}$$

Anti-aliasing by subsampling

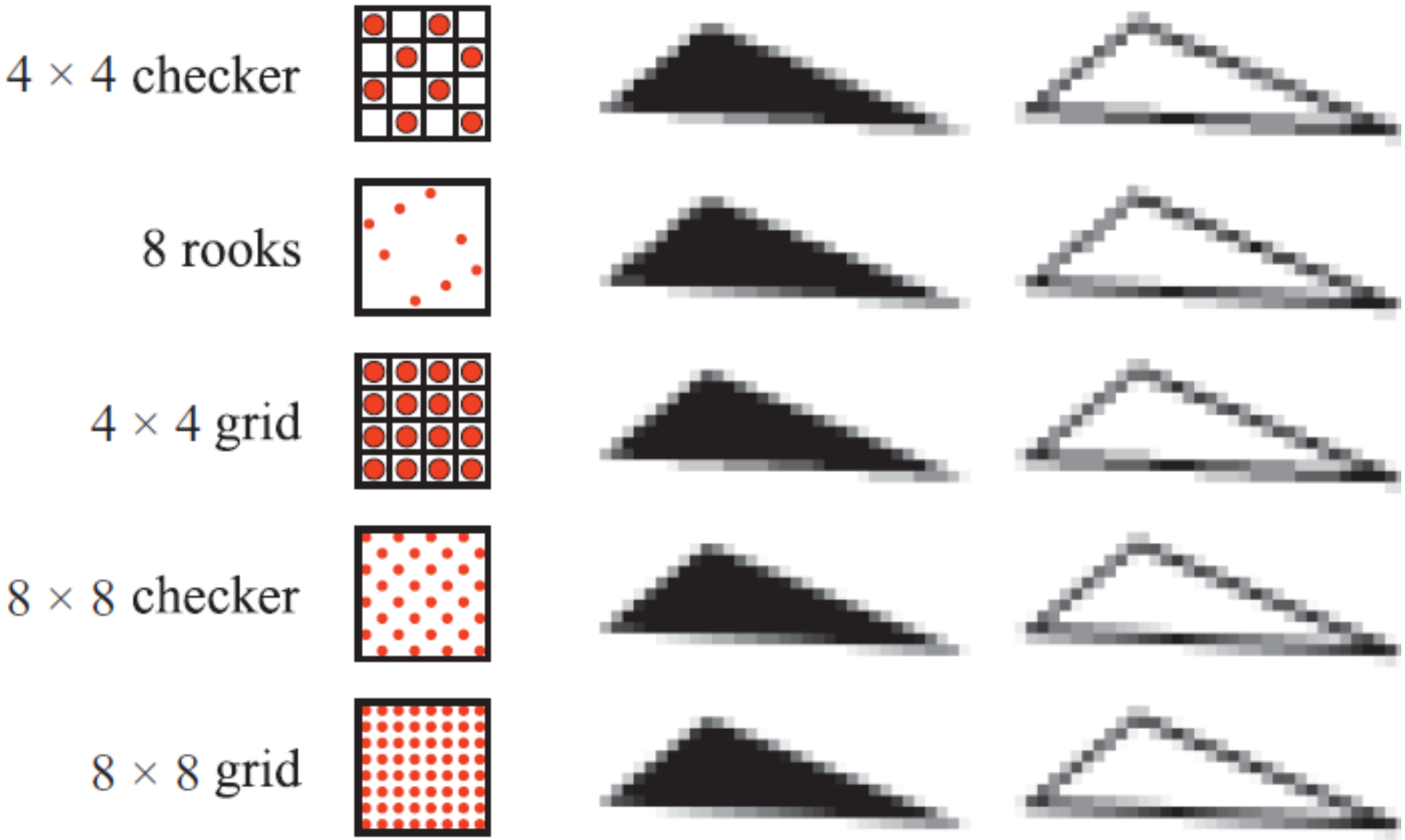
- Subdivide each pixel into n regions
- Colour each sub-pixel
- Calculate average colour



Subsampling schemes

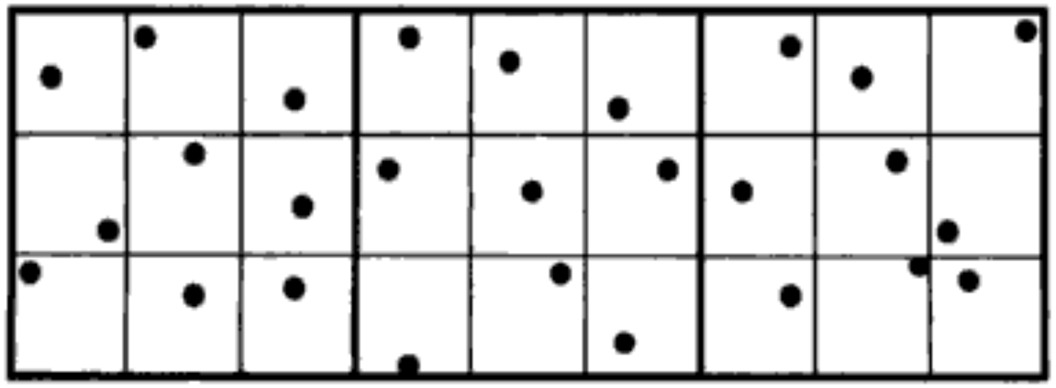


Subsampling schemes



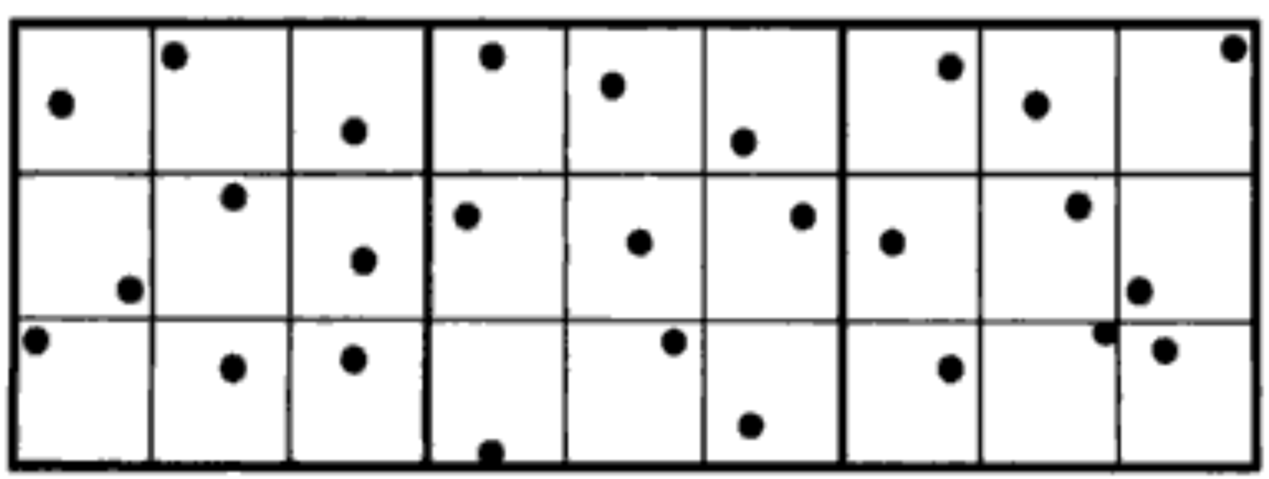
Stochastic sampling

- Regular patterns still exhibit some aliasing for very small details
- Random sampling causes higher frequency signals to appear as noise rather than aliasing
- Our eyes are more sensitive to aliasing than noise



Stochastic sampling

- Subdivide pixels into n regions and randomly sample within those regions
- Calculate colours for each subsample and average
- Either precompute a table of sample positions or calculate them in real time

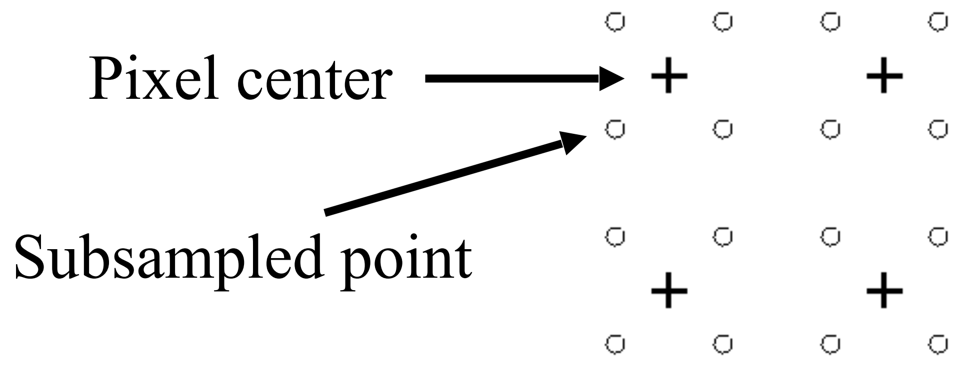


Comparison


Regular, 1x1
Regular 3x3
Regular, 7x7
Jittered, 3x3
Jittered, 7x7

Accumulation buffers

- Use a buffer of the same size as the target image
- Shift the frame buffer around each pixel center
- Accumulate and average

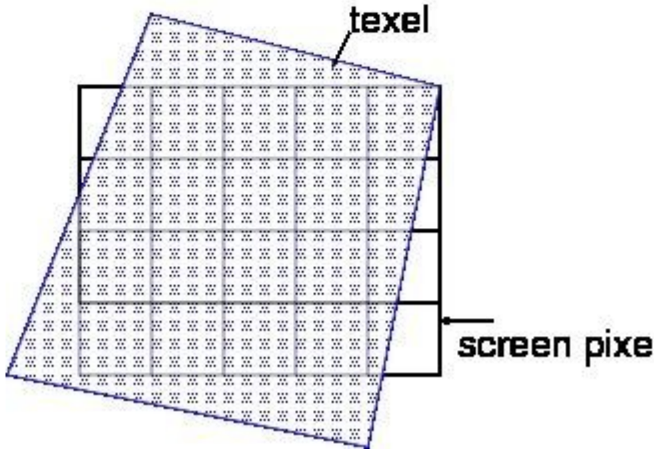


Antialiasing textures

- When textures are zoomed in, individual texture pixels (texels) are clearly visible

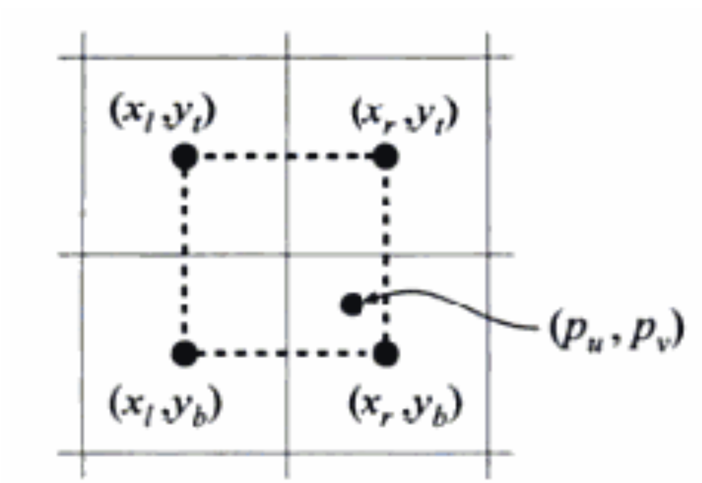
- When zoomed out, several texels could be mapped to a single pixel

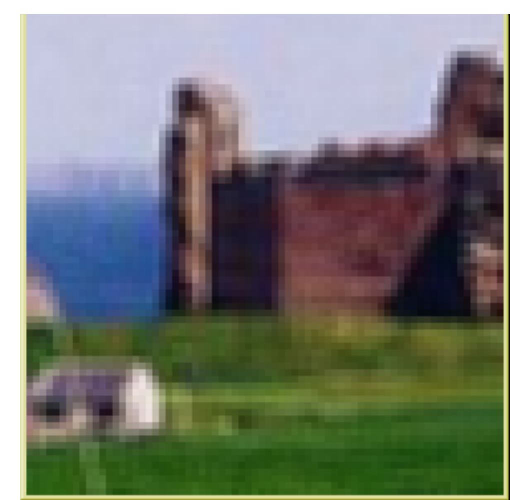
Magnification



Bilinear interpolation

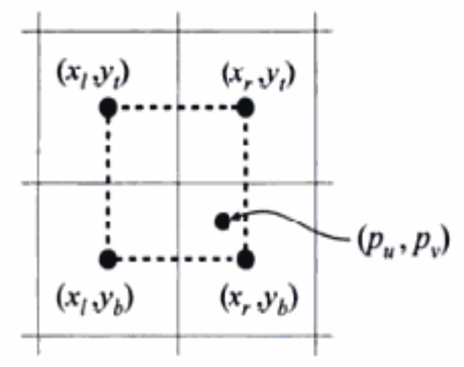
- Averaging the neighbouring texels based on the uv coordinates for the current pixel





Bilinear interpolation

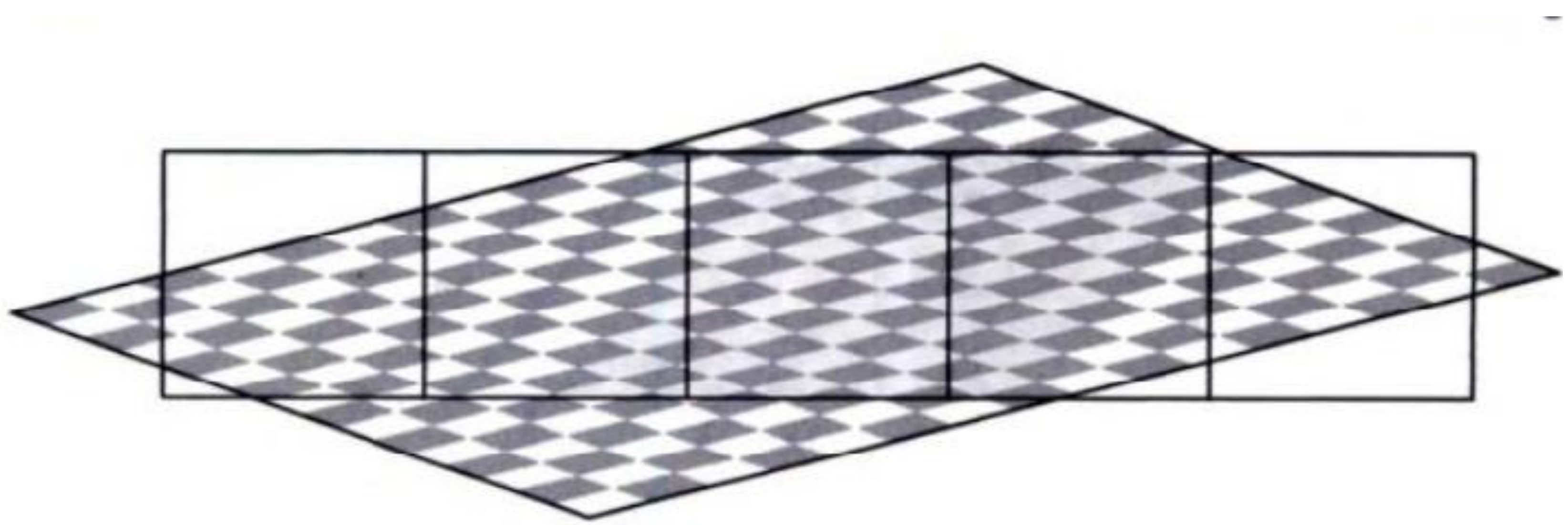
- Colour is calculated from coordinates by interpolation, with u' and v' the distance from the rounded down coordinates



$$
\begin{aligned}
u' =& p_u - (\text{int})p_u, v' = p_v - (\text{int})p_v \\
c(p_u, p_v) =& (1-u')(1-v')t(x_l, y_b) + u'(1-v')t(x_r, y_b) \\
&+ (1-u')v't(x_l, y_t) + u'v't(x_r, y_t)
\end{aligned}
$$

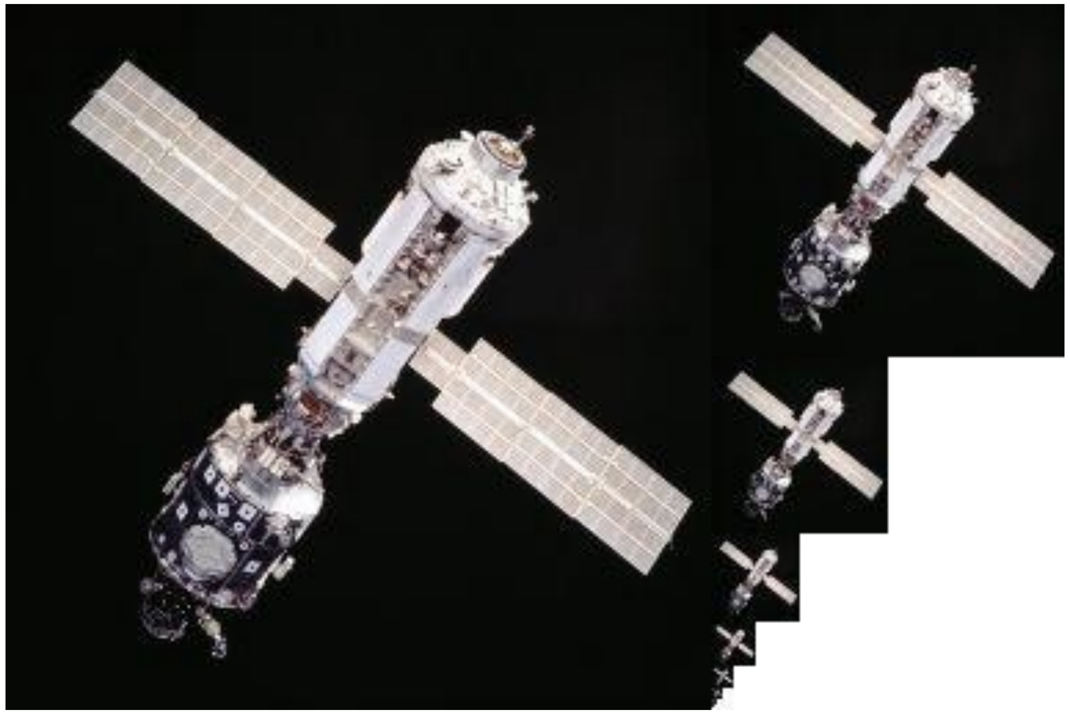
Minification

- When textures are zoomed out, multiple texels fall inside a single pixel. Causes aliasing due to the Nyquist limit



MIP mapping

- Textures are produced at multiple resolutions
- Resolutions are switched depending on number of texels in a pixel
- Select a resolution where ration of texels to pixel is 1:1



MIP mapping

- Map pixel corners to texture space
- Find a resolution where texel size is close to the pixel size
- Alternatively, use trilinear interpolation between multiple resolutions

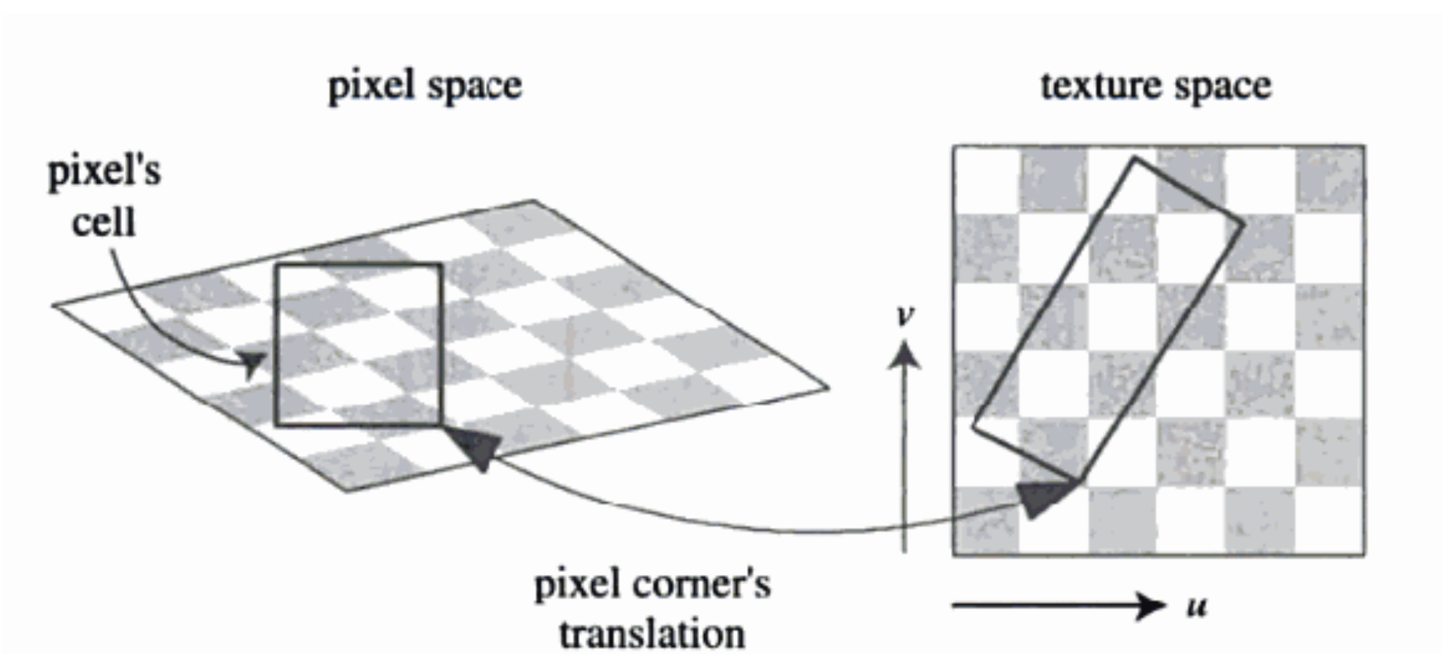
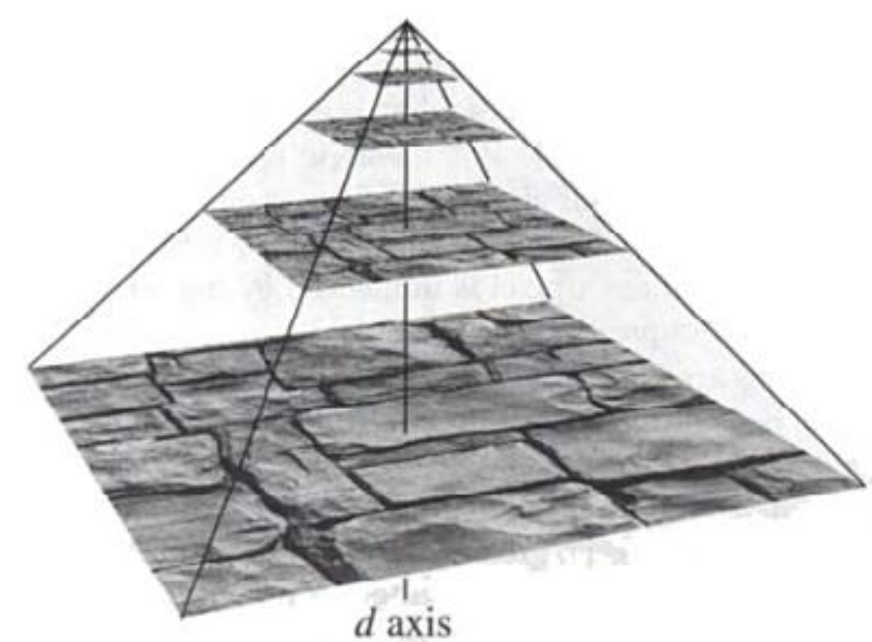
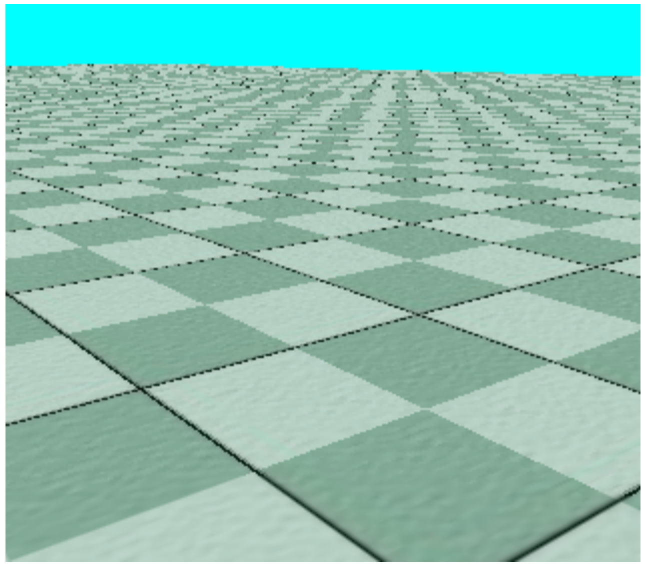


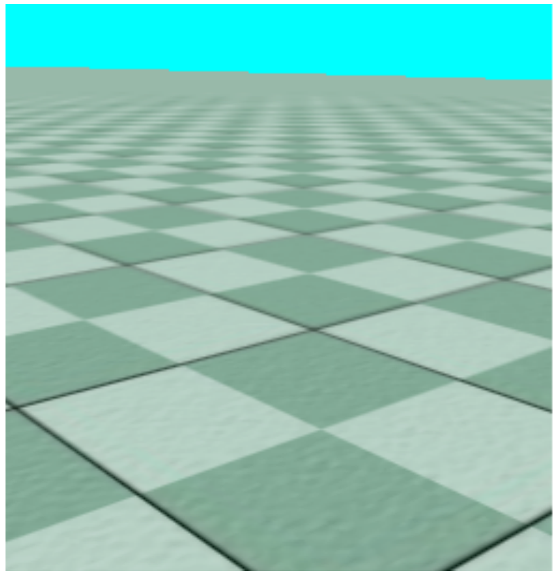
Figure 5.13. On the left is a square pixel cell and its view of a texture. On the right is the projection of the pixel cell onto the texture itself.



Example



Standard



MIP mapping

References

- Texture mapping - Shirley Chapter 11 (Texture mapping)
- Bump mapping - Akenine-Möller Chapter 6.7 (6.7.1) Bump mapping. Blinn, J. F. (1978). Simulation of wrinkled surfaces. *ACM SIGGRAPH Computer Graphics*, *12*(3).
- Anti-aliasing - Shirley Chapter 8.2 (Simple anti-aliasing), Akenine-Möller Chapter 5.6.2 (Screen-based anti-aliasing), Chapter 6.2.1,6.2.2 (Magnification and Minification)