

CS 428: Fall 2009

Introduction to Computer Graphics

Texture mapping and filtering

Topic overview

- Image formation and OpenGL
- Transformations and viewing
- **Polygons and polygon meshes**
 - **3D model/mesh representations**
 - **Piecewise linear shape approximations**
 - **Illumination and polygon shading**
- Modeling and animation
- Rendering

Topic overview

- Image formation and OpenGL
- Transformations and viewing
- **Polygons and polygon meshes**
 - 3D model/mesh representations
 - Illumination and polygon shading
 - **OpenGL rasterization: hidden surface removal, interpolation, texturing (all image-space!)**
- Modeling and animation
- Rendering

Topic overview

- Image formation and OpenGL
- Transformations and viewing
- Polygons and polygon meshes
 - Programmable pipelines
- Modeling and animation
- **Rendering**
 - **OpenGL rasterization: hidden surface removal, interpolation, texturing (some sampling theory)**
 - **Raytracing and radiosity**

Topic overview

- Image formation and OpenGL
- Transformations and viewing
- Polygons and polygon meshes
 - Programmable pipelines
- Modeling and animation
- **Rendering**
 - **Object space hidden surface removal, bump mapping and other texture tricks**
 - **Raytracing and radiosity**

Textures

- Reality provide a wide spectrum of geometric shapes and physical materials
 - Structures in wood, marble, wallpaper, clouds
 - Far-away background scenery, buildings and trees
- Modeling the exact geometric form of these objects is too complex and not necessary
 - No/small parallax effects for distant scenery + only cover a few pixels
 - Can be performed in a fragment shader (more later...)

Textures

- Make simple objects appear significantly more complex
- Model wall, mirror or window as planar surface
- Apply an image to the window to model the distant scenery



Texture mapping

- 2D-textures are functions that map points (u,v) in texture space to (r,g,b) colors

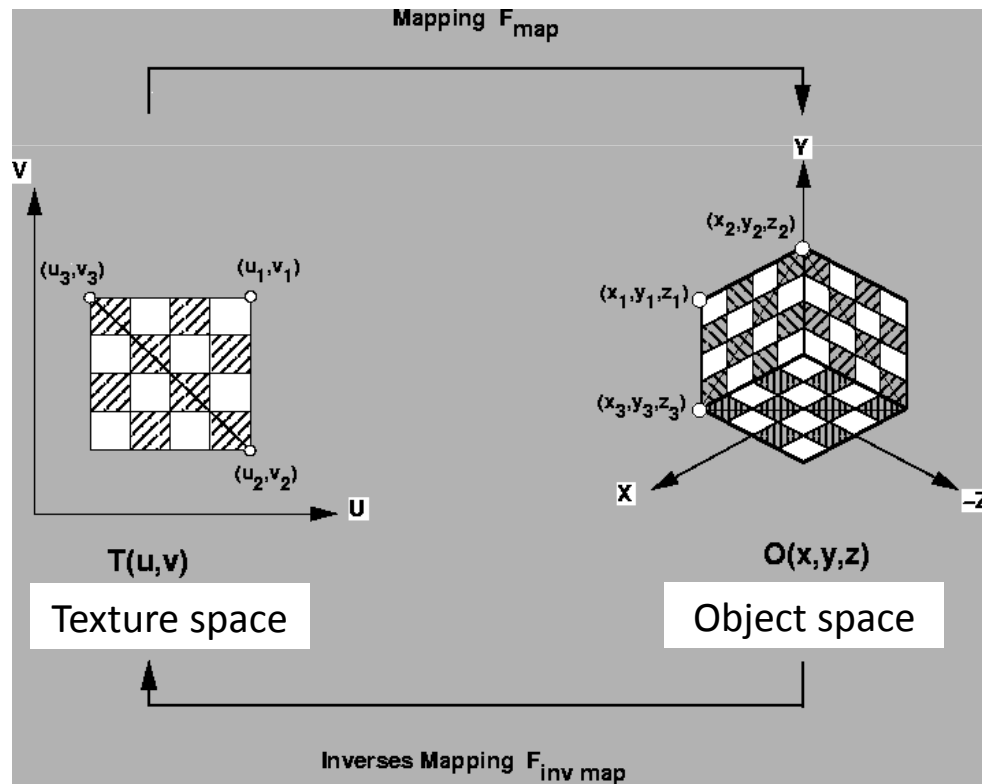
$$(r, g, b) = C_{tex}(u, v)$$

- The mapping describes how to decorate the surface
- For rendering, we need the inverse mapping from known (x,y,z) coordinates to (u,v) points

$$(u, v) = F_{inv\ map}(x, y, z)$$

Texture mapping

- Texture mapping uses barycentric coordinates
 - (u,v) provided at vertices `glTexCoordXX (...)`



Texture mapping

- Mathematically, this is described by concatenation of two mappings

$$(r, g, b) = C_{tex} \left(\overbrace{F_{inv\ map}^{2D}(x, y, z)} \right)$$

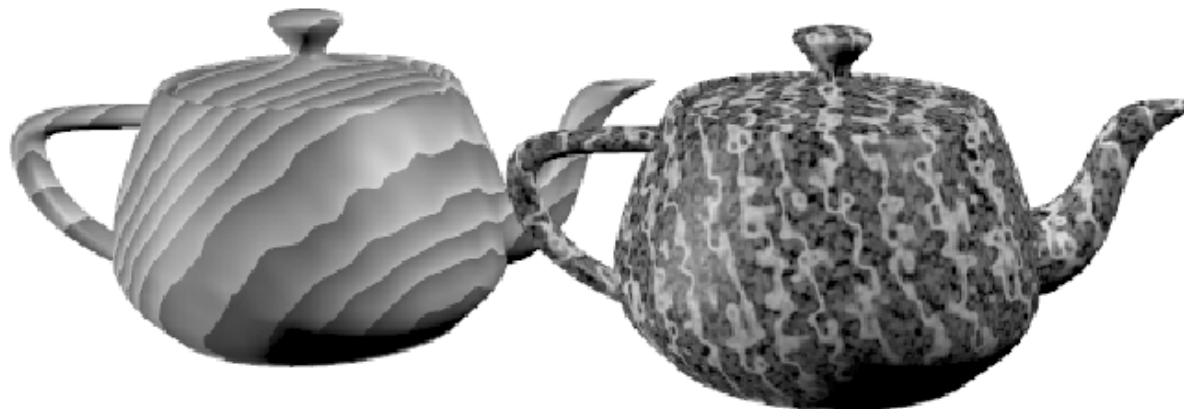
- 3D-textures are functions that map points of (u, v, w) space to (r, g, b) colors

$$(r, g, b) = C_{tex}(u, v, w)$$

Texture mapping

- 3D textures are known as **solid textures**
- Examples: wood and marmorite
- Interpretation: the shape is carved out of the (u,v,w) texture space

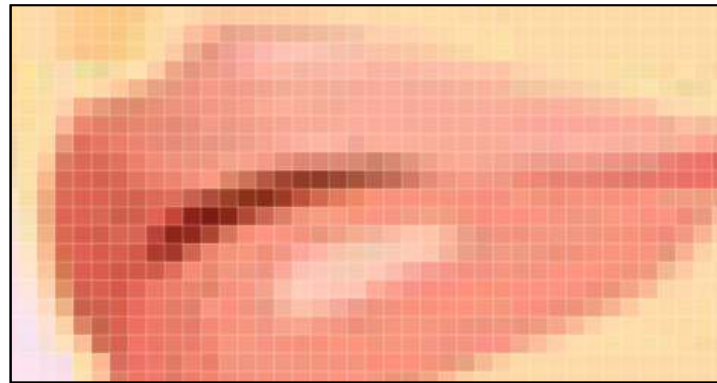
$$(r, g, b) = C_{tex}(u, v, w)$$



Texture representation

Discrete textures

- Grid of color values (texels)
- Three/four scalar quantities per texel (rgba)
- $n \times m$ texture is simply a pixel image with 3(4) tuples stored at integer coordinates

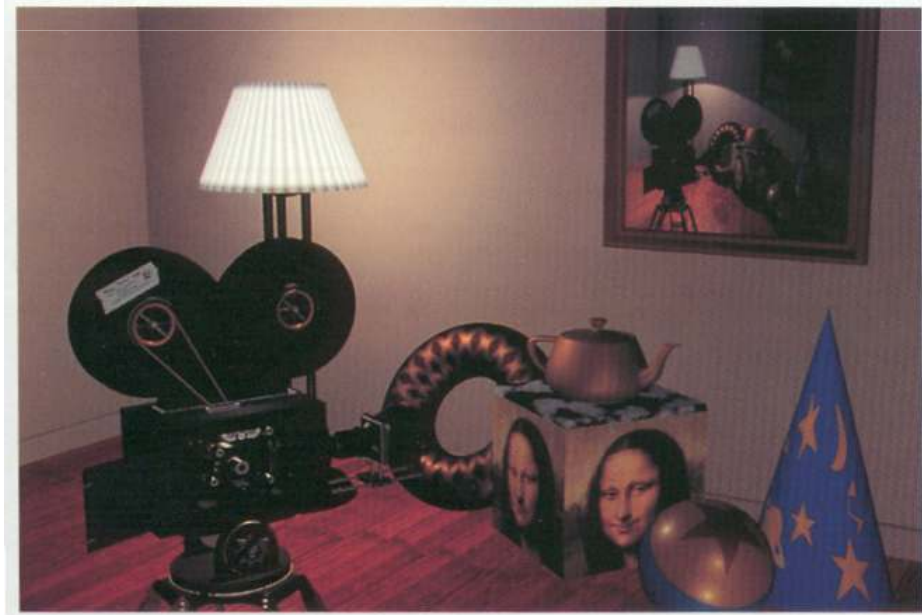


$$\{C[i, j] \mid 0 \leq i < n, 0 \leq j < m\}$$

Discrete textures

Advantages

- Easy to acquire with cameras, scanners etc.
- Generating complex textures using cameras and procedural methods simplifies photorealism



Discrete textures

Problems

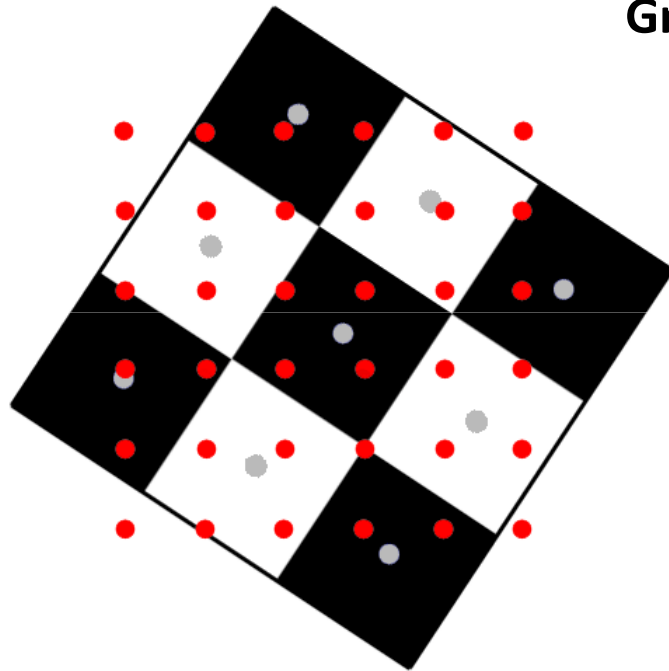
- Large texture memory consumption
- Magnifying textures leads to blurriness
- Tiling a surface with textures is complicated
- The texture lighting is not necessarily coherent with the lighting in the scene
- Texture mapping on surfaces is in general not free of distortion (parameterization, CS 523)
- Texture values $C_{\text{tex}}(u,v)$ at positions (u,v) need to be reconstructed, interpolated and filtered

Discrete textures

Reconstruction

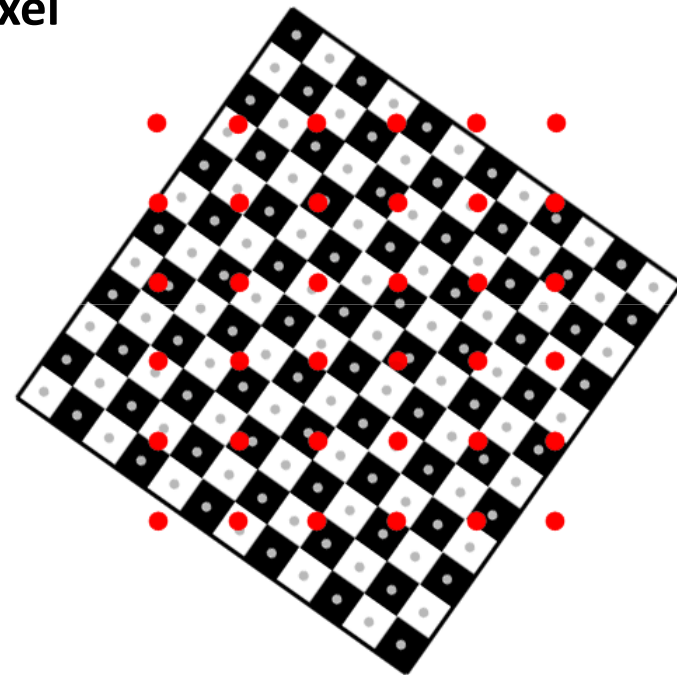
Red: screen pixel

Grey: texel



Magnification

(Oversampling)



Minification

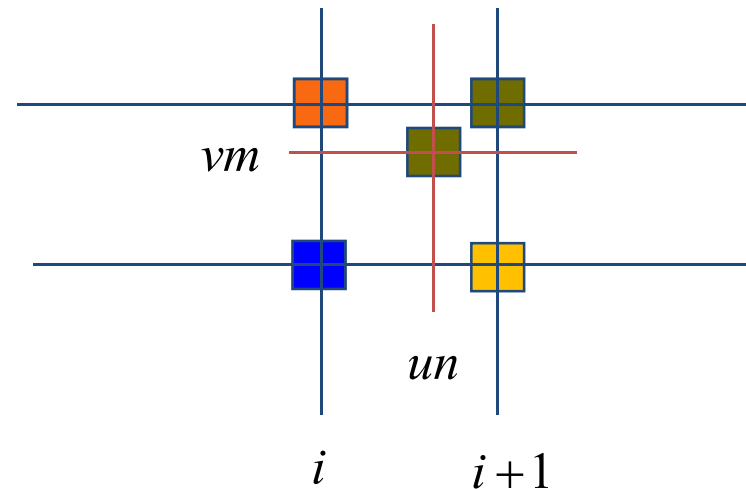
(Undersampling)

Discrete textures

Reconstruction

- 1. nearest neighbor filtering

$$C_{tex}(u, v) = \begin{cases} C[\lfloor un \rfloor, \lfloor vm \rfloor] & : u < 1, v < 1 \\ C[\lfloor n-1 \rfloor, \lfloor vm \rfloor] & : u = 1, v < 1 \\ C[\lfloor un \rfloor, \lfloor m-1 \rfloor] & : u < 1, v = 1 \\ C[\lfloor n-1 \rfloor, \lfloor m-1 \rfloor] & : u = 1, v = 1 \end{cases}$$



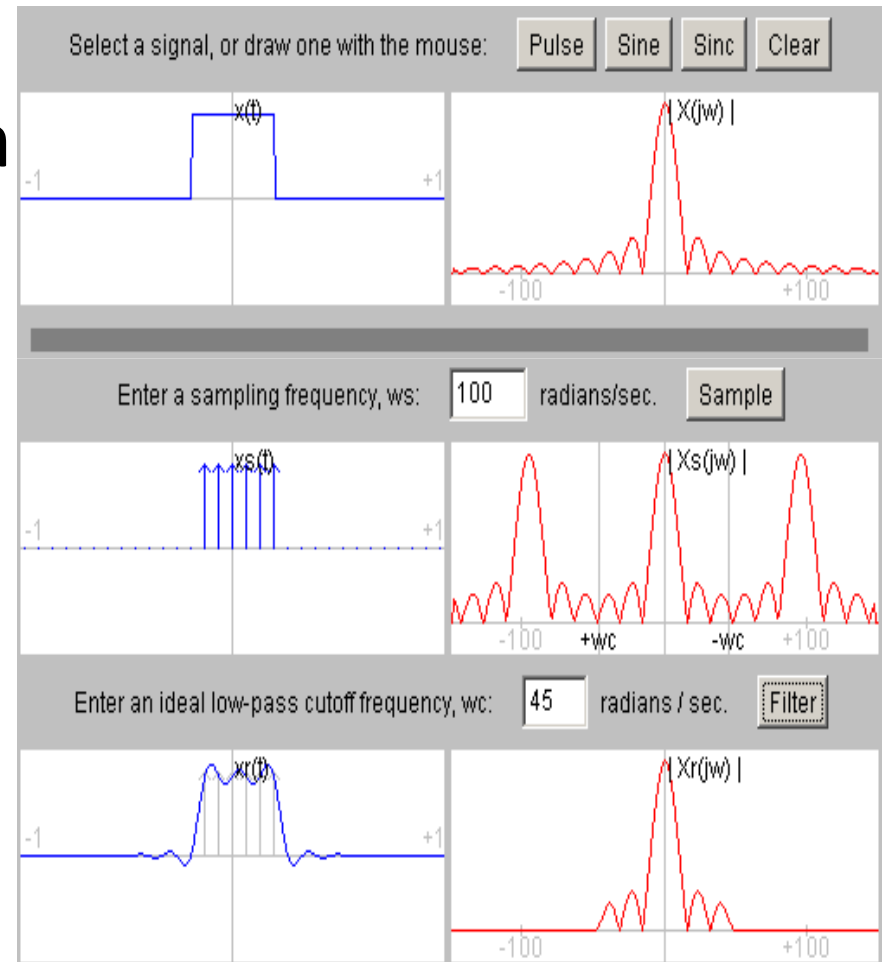
Discrete textures

Reconstruction

- Bilinear interpolation is a good approximation
- Better: convolution with a sinc filter

$$\text{sinc}(x) \equiv \begin{cases} 1 & \text{for } x = 0 \\ \frac{\sin x}{x} & \text{otherwise,} \end{cases}$$

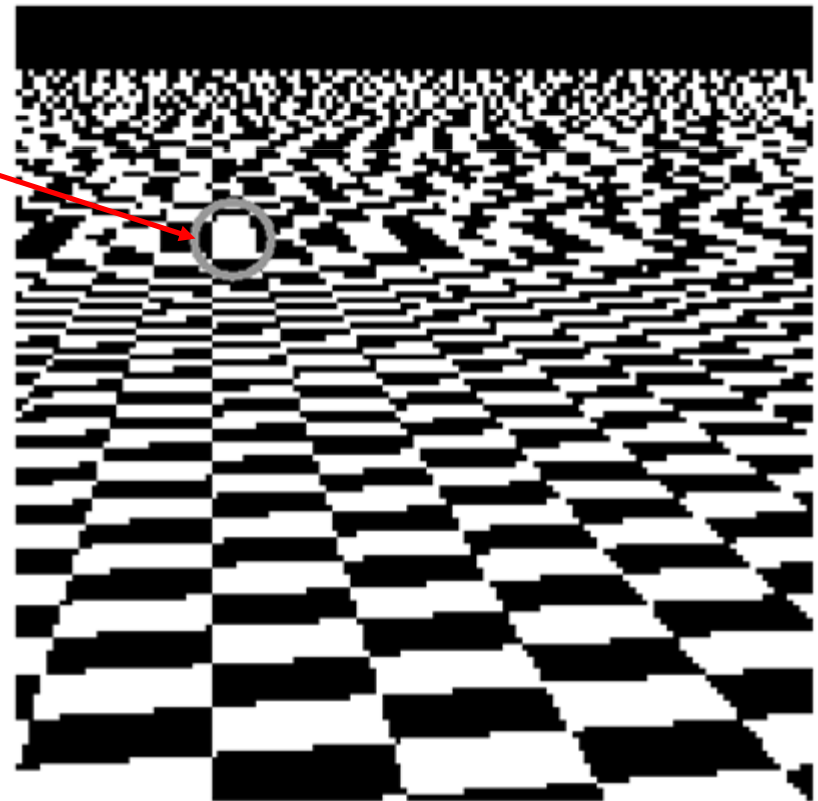
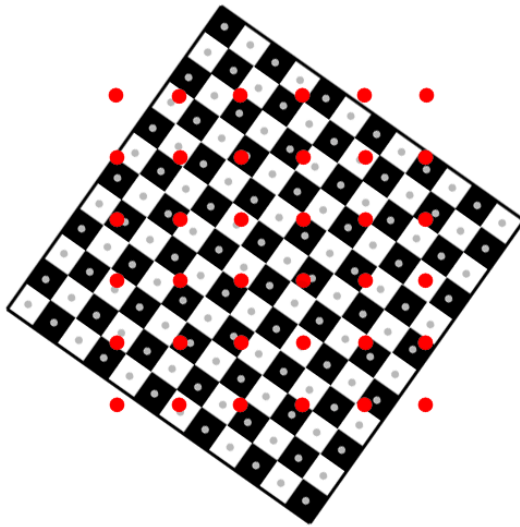
- See
 - <http://www.jhu.edu/~signals/sampling/>



Discrete textures

Filtering

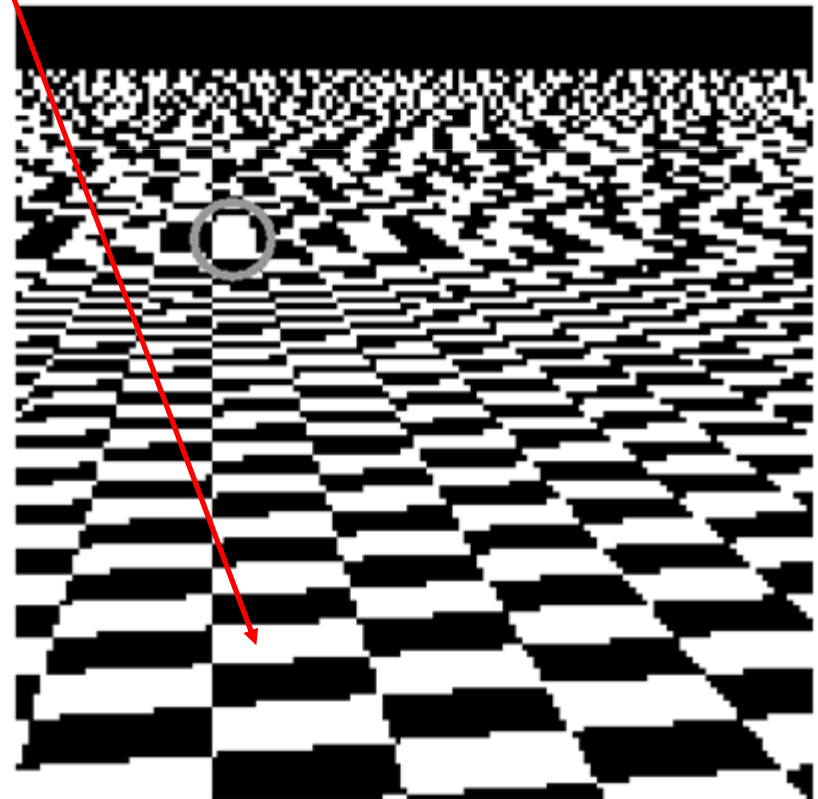
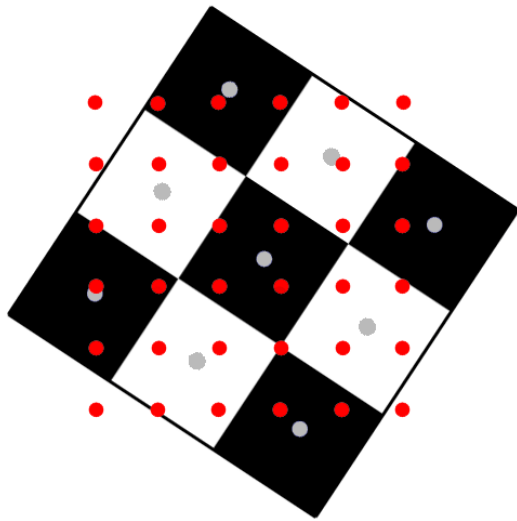
- Nearest neighbor filtering is insufficient for texture minification
- Undersampling errors create aliasing artifacts



Discrete textures

Filtering

- Magnification (pixelation, blurring) is generally preferred over aliasing
- Achieved by reducing original texture res.

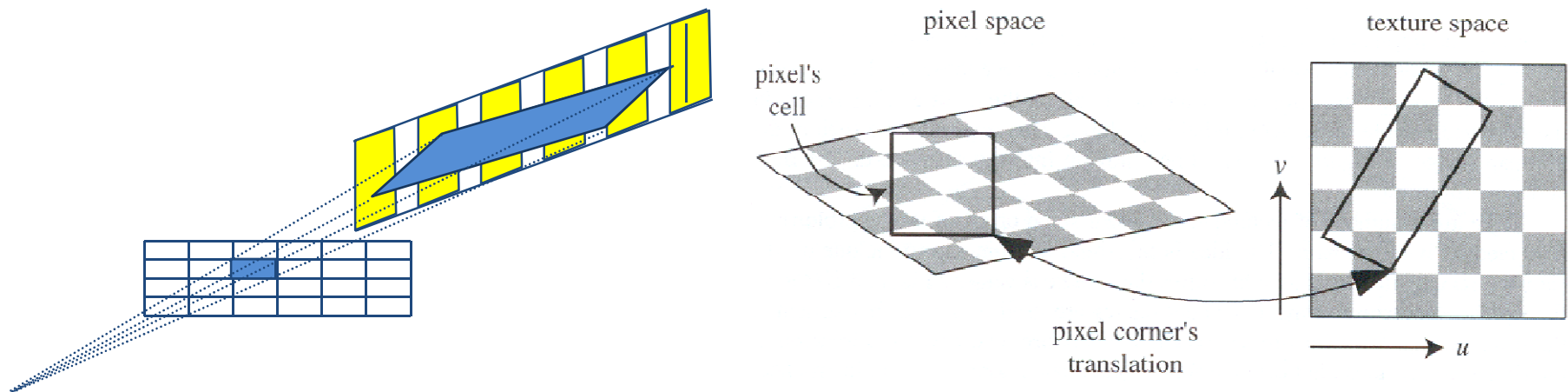


Discrete textures

Filtering

- **Footprint:** projection of the cell associated with a screen pixel (x,y) to texture (u,v) space
- Approximated by a parallelogram, spanned by the vectors

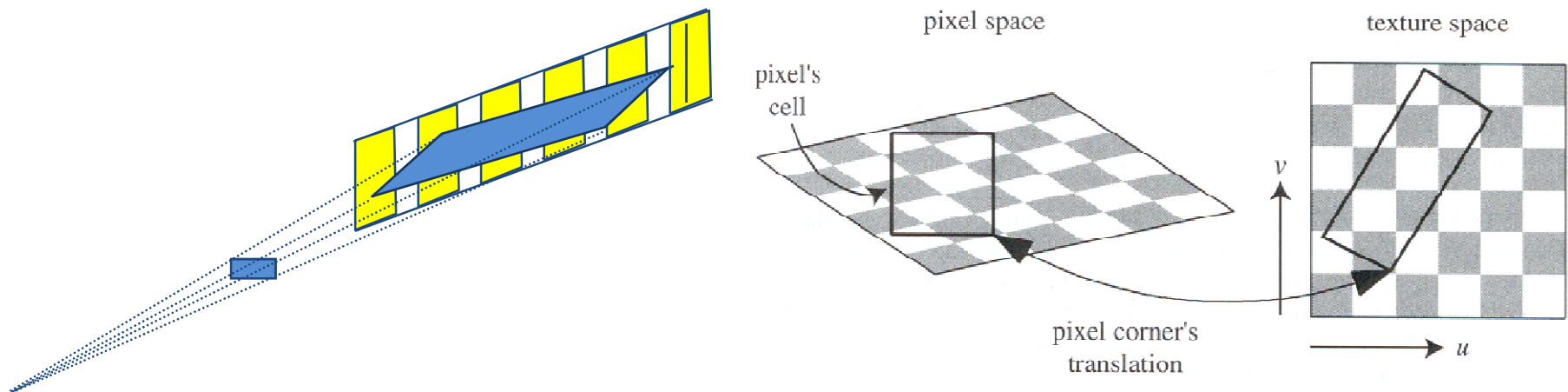
$$r_1 = \left(\frac{\partial u}{\partial x}, \frac{\partial v}{\partial x} \right)^t, \quad r_2 = \left(\frac{\partial u}{\partial y}, \frac{\partial v}{\partial y} \right)^t$$



Discrete textures

Filtering

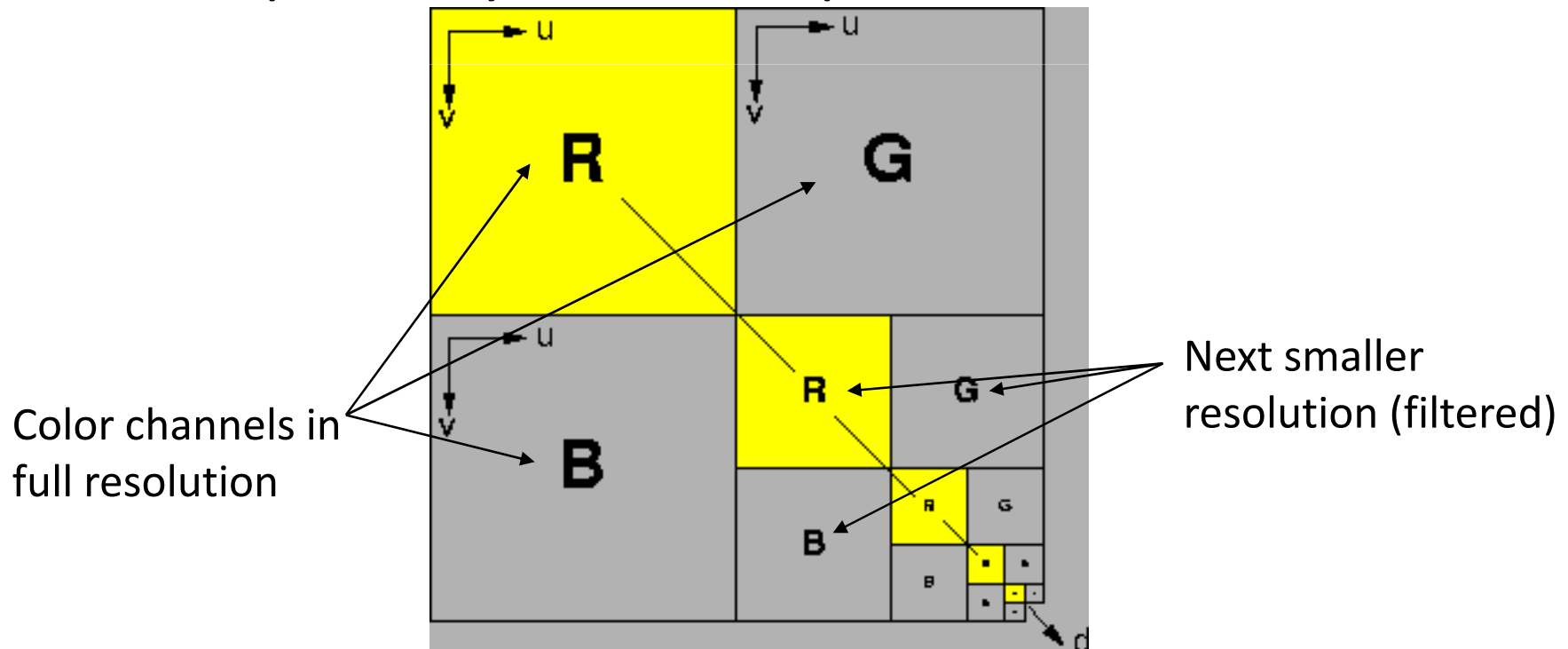
- For real-time texture filtering
 - Replace the actual footprint by a union of simpler surfaces
 - Such that the filtered texture value can be pre-computed



Discrete textures

Filtering: mip-map

- A $2n \times 2n$ mip-map $C_{\text{mip}} [i,j]$ stores a 2D texture $C[i,j]$ of size $n \times n$ where $n = 2^k$
 - sequentially at **half** the previous resolution



Discrete textures

Filtering: mip-map

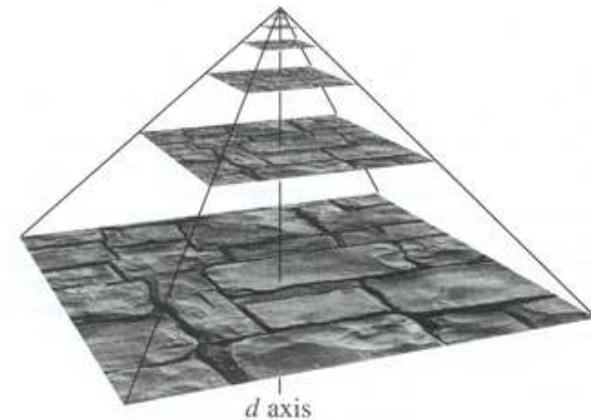
- Level $d=0$ are the original texture values

$$C_{mip}^0[i, j] = C[i, j], \quad 0 \leq i, j < 2^k.$$

- All other levels are constructed by filtering the previous resolution (weighted summation)

$$C_{mip}^d[i, j] = \frac{1}{4} \left(C_{mip}^{d-1}[2i, 2j] + C_{mip}^{d-1}[2i+1, 2j] + C_{mip}^{d-1}[2i, 2j+1] + C_{mip}^{d-1}[2i+1, 2j+1] \right)$$

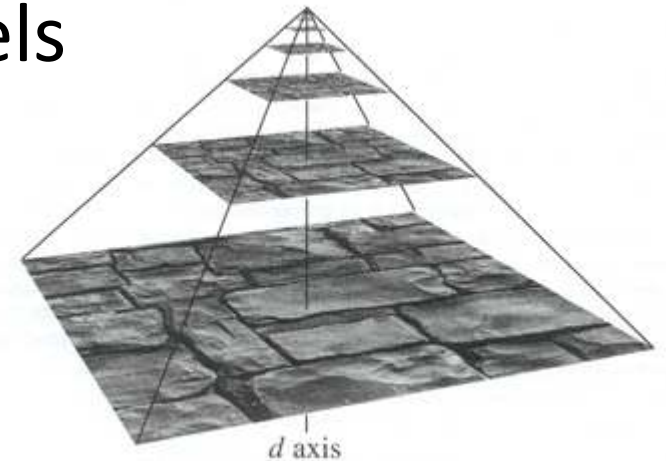
$$1 \leq d < k-1 \text{ and } 0 \leq i, j < 2^{k-d}.$$



Discrete textures

Filtering: mip-map

- On level d of the texture hierarchy, 2^{2d} texels of the original texture are represented as a single texel
- In the application, pixel values are interpolated from two levels depending on the size of the **footprint**
- This known as trilinear filtering



Discrete textures

Filtering: mip-map

- For a rectangular footprint, $l = \max(a, b)$
- Too small footprints (e.g. $(a+b)/2$) can lead to aliasing (too large footprints to blurring)
- Texture value at level $d = \log_2(l)$ is

$$C_{tex}(u, v, d) = BiLinInt(C_{mip}^d, u, v)$$

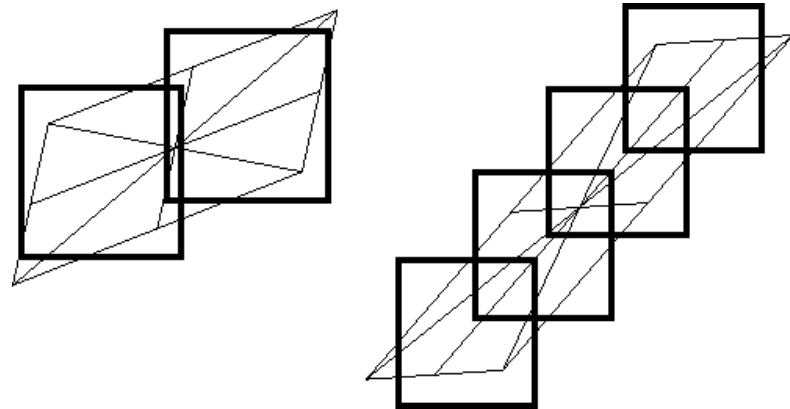
- When d is not an integer value
 - Interpolate between two levels $\lfloor d \rfloor$ und $\lceil d \rceil$

$$C_{tex}(u, v, d) = (d - \lfloor d \rfloor) * BiLinInt(C_{mip}^{\lfloor d \rfloor + 1}, u, v) + (\lfloor d \rfloor + 1 - d) * BiLinInt(C_{mip}^{\lfloor d \rfloor}, u, v).$$

Discrete textures

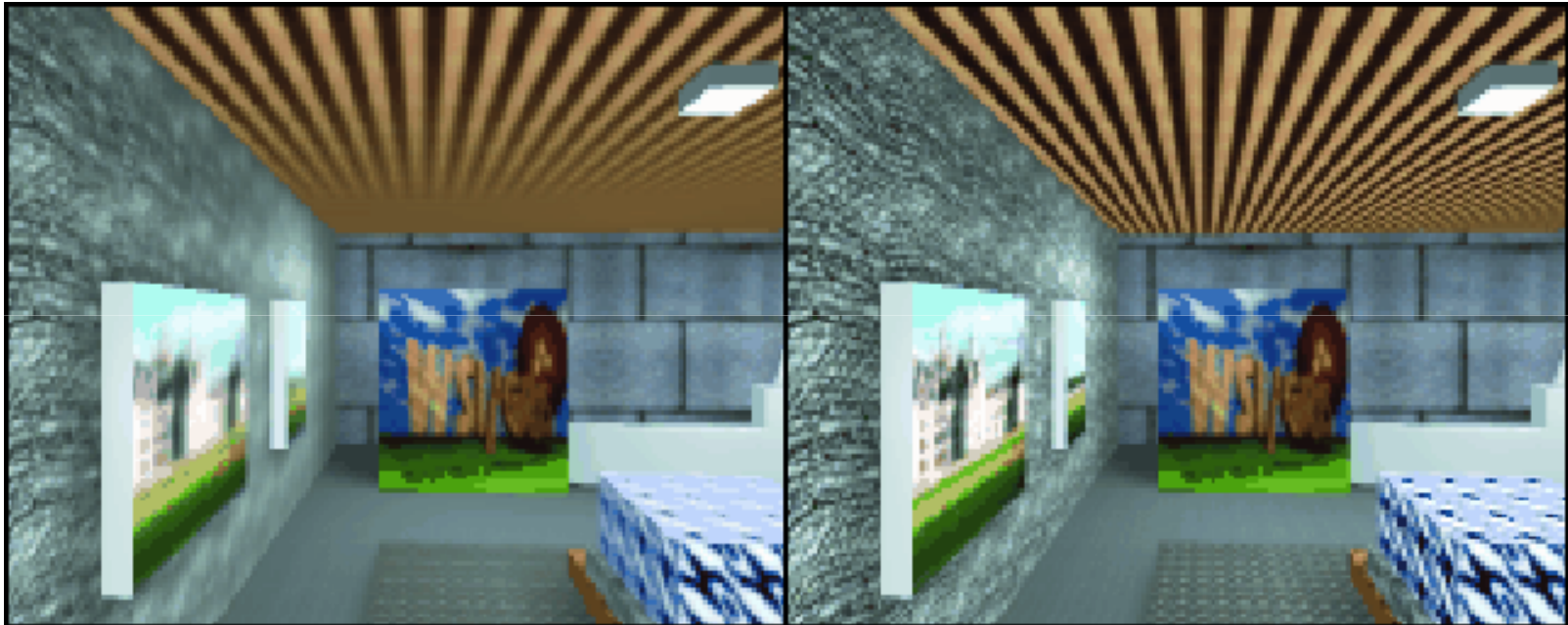
Filtering: footprint assembly

- Square projection ignores anisotropy!
- Idea: use multiple mip-map accesses to cover the footprint parallelogram
- **Anisotropic texture filtering** in real time
 - Number of square parts determines the quality of the anisotropic filter



Discrete textures

Filtering: footprint assembly



Mip-mapping

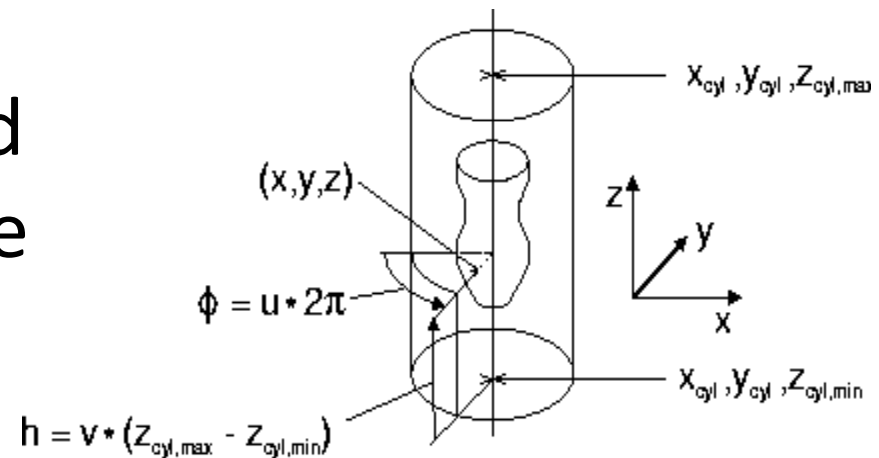
Anisotropic
filtering

Texturing surfaces

- How to determine the texture coordinates for the vertices of a polygonal model?
 - They might be given by a modeler 😊
 - If not, embed the complex object in a simpler object for which a simple (u,v) parameterization exists
 - Project from the polygonal model onto the simpler object
 - Examples: cylinder, sphere, box

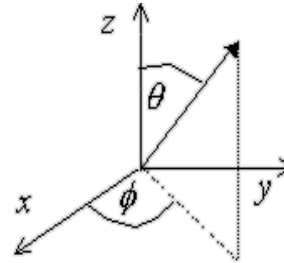
Cylinder mapping

- Parameterization:
rotation angle ϕ and height h
- Points interior to the cylinder are projected orthogonally from the axis to the cylinder surface
 - Interpret (ϕ, h) parameters as (u, v) texture coordinates

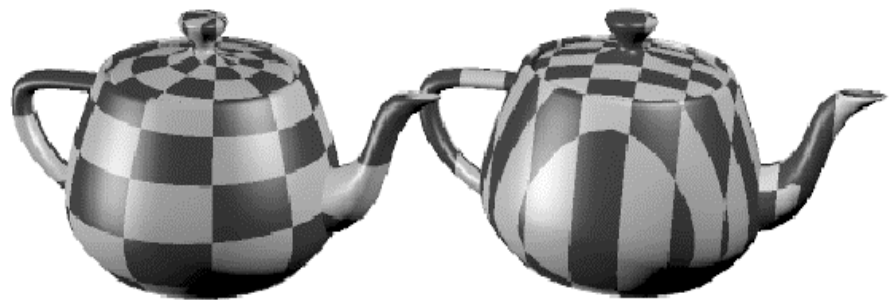


Sphere mapping

- Polar coordinates ϕ and θ
- Project points (x,y,z) on the interior through the midpoint (x_s, y_s, z_s) onto the sphere
- Interpret (ϕ, θ) as (u, v)



$$u = \frac{\pi + \arctan2(y - y_s, x - x_s)}{2\pi}$$
$$v = \frac{\arctan2\left(\sqrt{(x - x_s)^2 + (y - y_s)^2}, z - z_s\right)}{\pi}$$

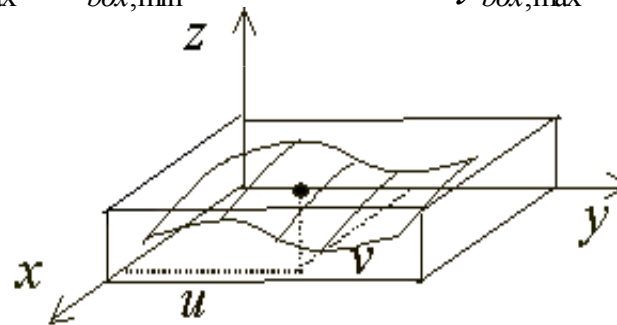


Box mapping

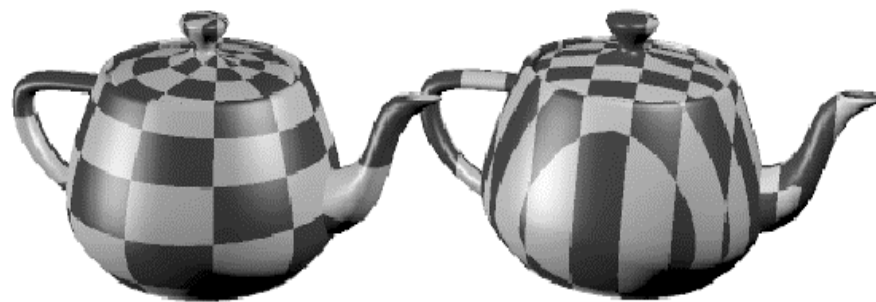
$$(x_{box,max} - x_{box,min}) \geq (y_{box,max} - y_{box,min}) \geq (z_{box,max} - z_{box,min})$$

$$u = \frac{x - x_{box,min}}{x_{box,max} - x_{box,min}}$$

$$v = \frac{y - y_{box,min}}{y_{box,max} - y_{box,min}}$$



Sphere
mapping



Box
mapping