

11 Texture Mapping

- Assuming a *uniform reflectance* for a diffuse surface is not efficient for objects with variation in reflectance
- Common technique to handle variation of reflectance is to store reflectance as a function of a pixel-based image and map it onto a surface
- The function or image is called a *texture map*
- The process of controlling reflectance properties is called *texture mapping*
- Texture mapping can be classified by
 - **dimensionality** of the texture function
 - **correspondence** between points on the surface and points in the texture function
 - whether texture function is **procedural** or a **table look-up**

11.1 3D Texture Mapping

- Specify texture in space
- Also called *solid texture* or *volume texture*
- For object without a solid color, replace c_r with a function $c_r(\mathbf{p})$ which maps 3D points to RGB colors
- Create a 3D texture but only call it for points \mathbf{p} on the surface

3D Stripe Textures

- Intuitive approach: with two given colors c_0 and c_1

RGB stripe (point \mathbf{p})

if ($\sin(x_p) > 0$) **then**

return c_0

else

return c_1

- With stripe's width controllable

RGB stripe (point \mathbf{p} , real w)

if ($\sin(\pi x_p / w) > 0$) **then**

return c_0

else

return c_1

- Interpolating smoothly between stripes

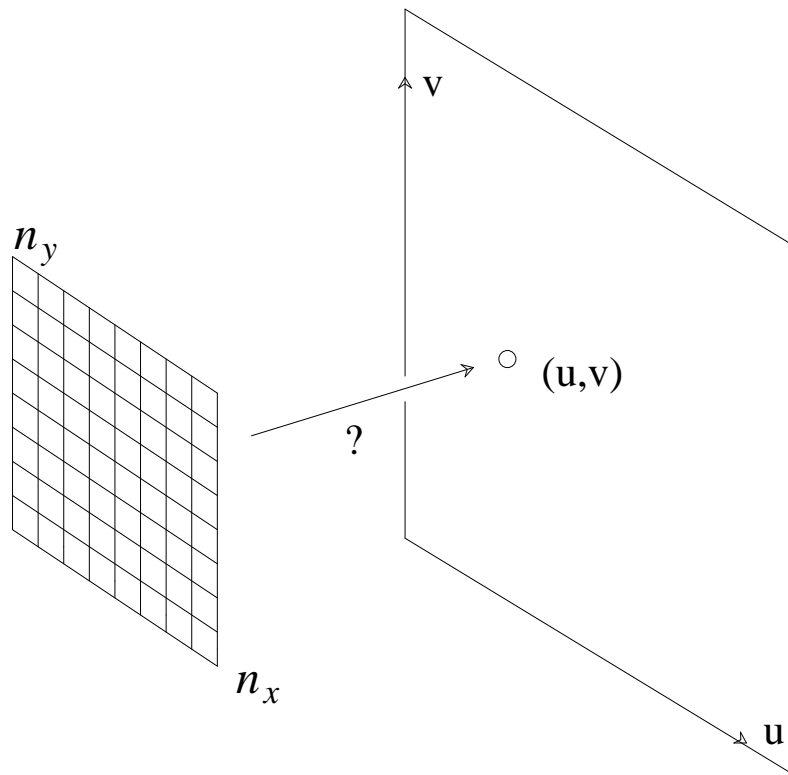
RGB stripe (point \mathbf{p} , real w)

$t = (1 + \sin(\pi x_p / w)) / 2$

return $(1 - t)c_0 + tc_1$

3D Texture Arrays

- How to do this in 2D? i.e., for each (u, v) , how to compute $c(u, v)$?



- **Intuitive approach:** constant color for each tile

Remove integer portion of (u, v) ;

$$i = [n_x u] ;$$

$$j = [n_y v] ;$$

$$c(u, v) = c_{ij} ;$$

- **Bilinear interpolation:** to avoid discontinuity in adjacent texture colors

Remove integer portion of (u, v) ;

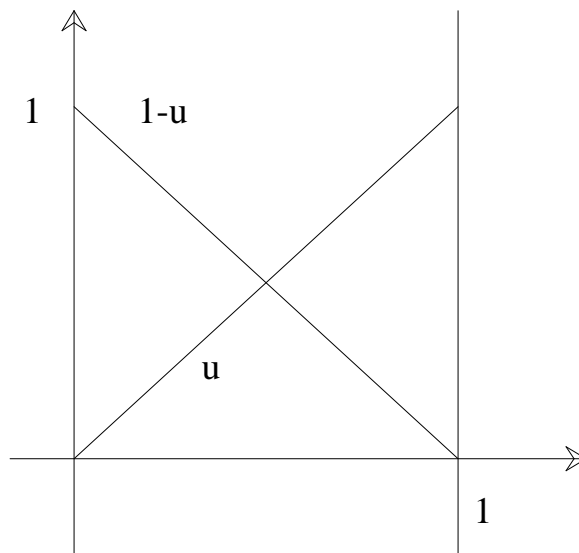
$$i = [n_x u] ;$$

$$j = [n_y v] ;$$

$$u' = n_x u - i ;$$

$$v' = n_y v - j ;$$

$$c(u, v) = (1 - u')(1 - v')c_{ij} + u'(1 - v')c_{(i+1)j} \\ + (1 - u') v' c_{i(j+1)} + u' v' c_{(i+1)(j+1)} ;$$



- **Hermite interpolation:** to avoid Mach band effect

Remove integer portion of (u, v) ;

$$i = [n_x u] ;$$

$$j = [n_y v] ;$$

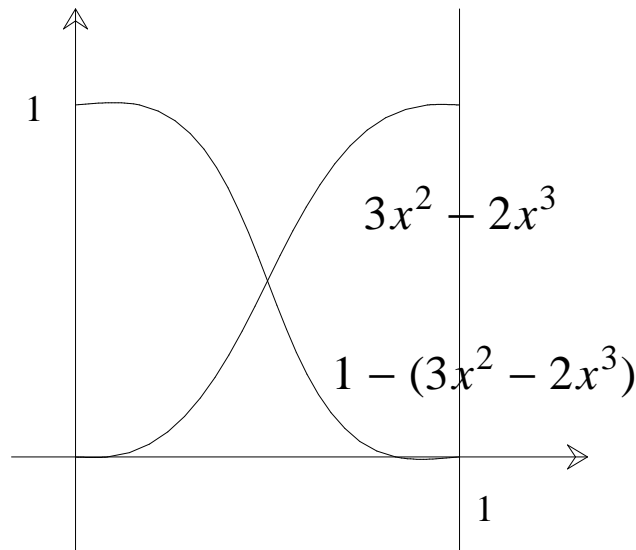
$$u' = n_x u - i ;$$

$$v' = n_y v - j ;$$

$$u'' = 3(u')^2 - 2(u')^3 ;$$

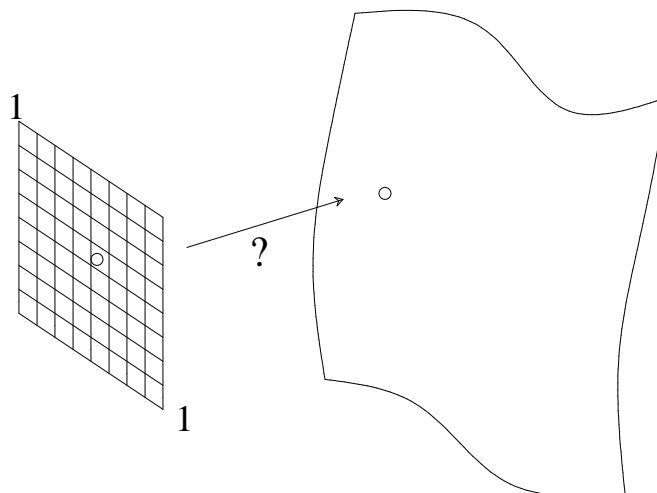
$$v'' = 3(v')^2 - 2(v')^3 ;$$

$$c(u, v) = (1 - u'')(1 - v'')c_{ij} + u''(1 - v'')c_{(i+1)j} \\ + (1 - u'') v'' c_{i(j+1)} + u'' v'' c_{(i+1)(j+1)} ;$$



11.2 2D Texture Mapping

- Basically the process of constructing a map between a 2D image and the surface of the object
- The coordinate system for the 2D image is set to be the unit square
- For (u, v) outside this square, use the fractional parts of the coordinates (resulting in a tiling of the plane)

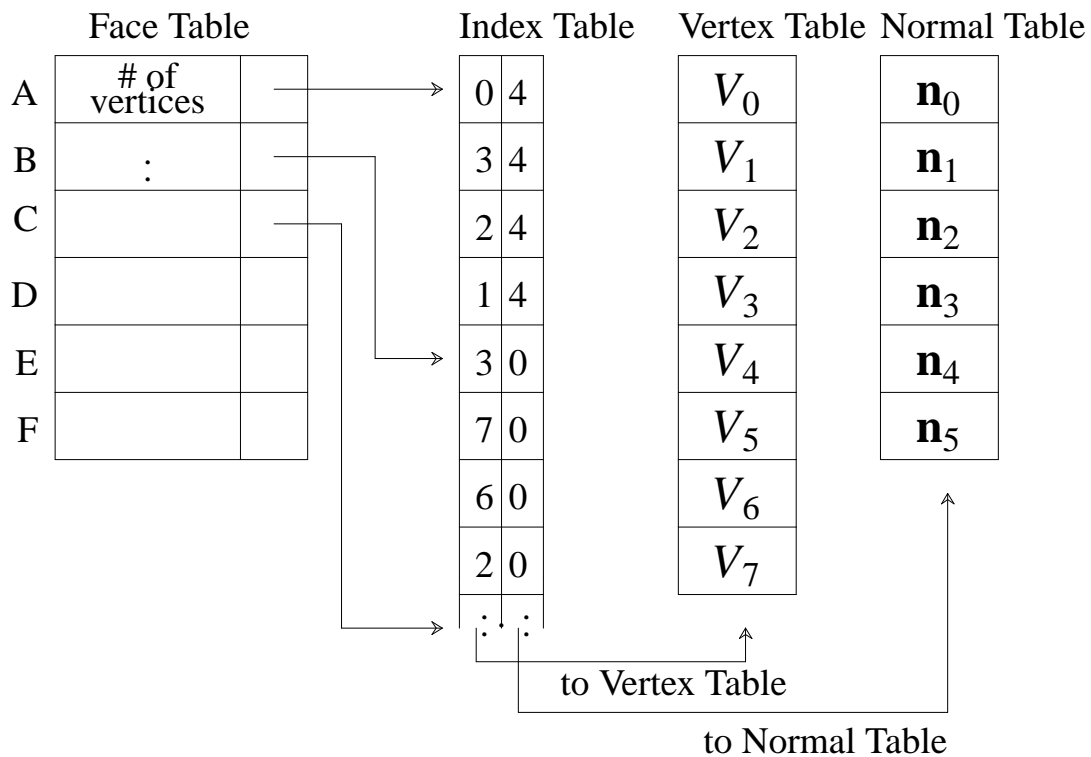


- To wrap a sphere with an image:

$$[0, 1] \times [0, 1] \dashrightarrow [0, \pi] \times [-\pi, \pi]$$

11.3 Tessellated Models

- How should a triangular mesh (or, more general, a polygonal mesh) be represented?
- Use a face table, a vertex table and an index table



- can be implemented with two classes

```
class mesh
material m
array_of_vector3 vertices
```

```
class meshtriangle
pointer_to_mesh meshptr
int i0, i1, i2
```

- Texture parameters (and other parameters such as materials, irradiances, ...) stored at vertices are bilinearly interpolated across the triangle
- If (β, γ) are the barycentric coordinates of a point of the triangle, i.e.,

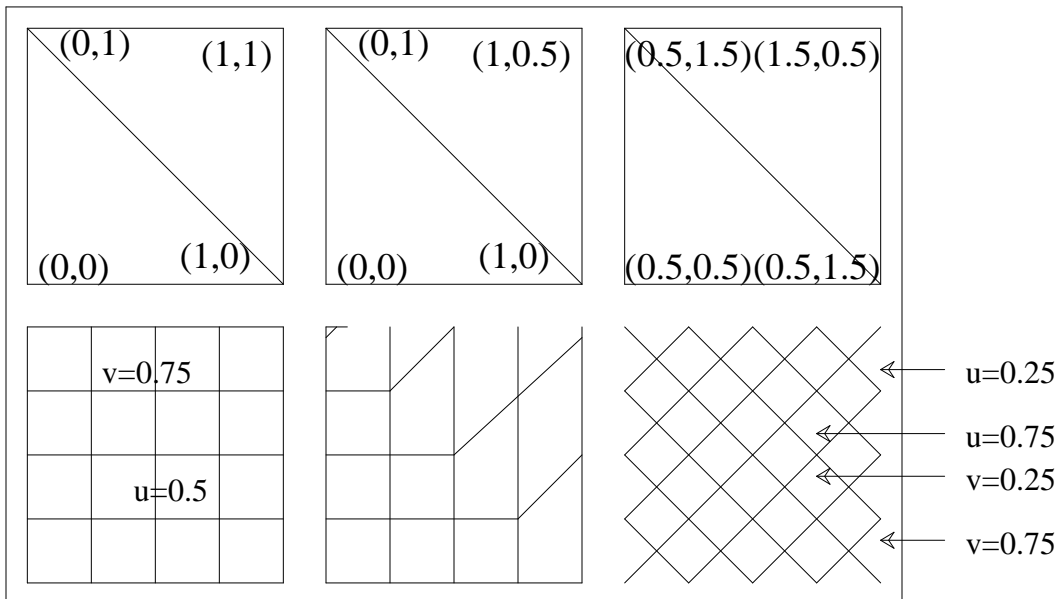
$$\mathbf{p}(\beta, \gamma) = \mathbf{a} + \beta(\mathbf{b} - \mathbf{a}) + \gamma(\mathbf{c} - \mathbf{a})$$

then (u, v) are computed similarly

$$u(\beta, \gamma) = u_a + \beta(u_b - u_a) + \gamma(u_c - u_a)$$

$$v(\beta, \gamma) = v_a + \beta(v_b - v_a) + \gamma(v_c - v_a)$$

- Examples:



11.4 Texture Mapping for Rasterized Triangles

- Just interpolating texture coordinates in screen space results in incorrect images

for all x do

for all y do

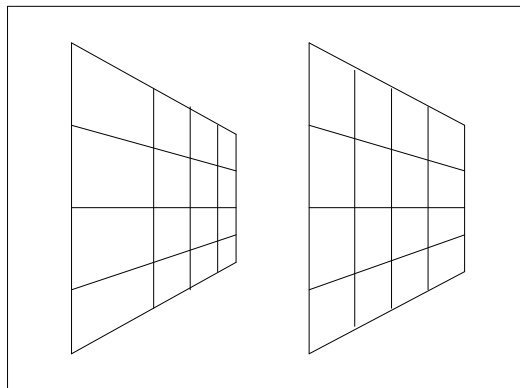
compute (α, β, γ) for (x, y)

if $0 < \alpha, \beta, \gamma < 1$ then

$\mathbf{t} = \alpha\mathbf{t}_0 + \beta\mathbf{t}_1 + \gamma\mathbf{t}_2$

drawpixel (x, y) with color texture(\mathbf{t}) for a solid texture or with texture (β, γ) for a 2D texture

In the following example, left is correct perspective, right is result of interpolation in screen space



- Where is the problem?

Stage of Interpolation. Consider the following flow from world space point \mathbf{q} to homogeneous point \mathbf{r} to homogenized point \mathbf{s} :

$$\begin{bmatrix} x_q \\ y_q \\ z_q \\ 1 \end{bmatrix} \xrightarrow{\text{transform}} \begin{bmatrix} x_r \\ y_r \\ z_r \\ h_r \end{bmatrix} \xrightarrow{\text{homogenize}} \begin{bmatrix} x_r/h_r \\ y_r/h_r \\ z_r/h_r \\ 1 \end{bmatrix} \equiv \begin{bmatrix} x_s \\ y_s \\ z_s \\ 1 \end{bmatrix}$$

Linearly interpolate \mathbf{q} and \mathbf{Q} in WS then transform:

$$\mathbf{s} + \frac{h_R t}{h_r + t(h_R - h_r)} (\mathbf{S} - \mathbf{s}).$$

\mathbf{q} and \mathbf{Q} are transformed then interpolated:

$$\mathbf{s} + u(\mathbf{S} - \mathbf{s}).$$

They represent the same line but with different parametrization.

Perspective Correct Textures

- Perform interpolation in homogenized space on \mathbf{s} and \mathbf{S} , but also on $1/h$
- Note that when t ranges from 0 to 1, we have

$$\frac{1}{h_r} + \frac{h_R t}{h_r + t(h_R - h_r)} \left(\frac{1}{h_R} - \frac{1}{h_r} \right) = \frac{1}{h_r} + t \left(\frac{1}{h_R} - \frac{1}{h_r} \right)$$

- Use this property, we can compute the world space barycentric coordinates of the triangle (β_w, γ_w) in terms of screen space coordinates (β, γ)