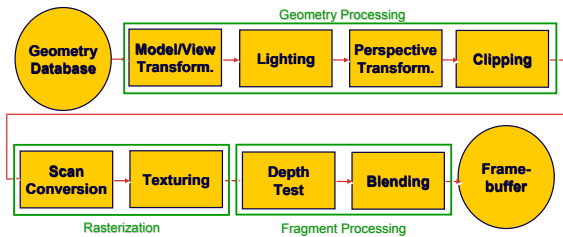


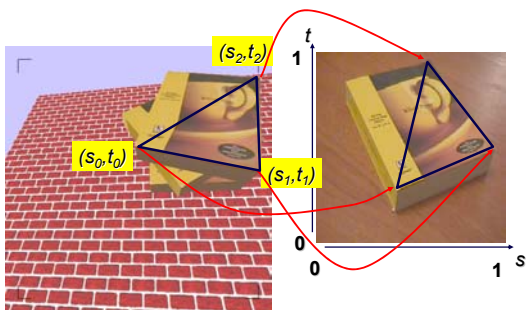
# Texture Mapping

CPSC 414

# The Rendering Pipeline



# Texture Mapping



# Texture Mapping

## Summary

- textures, texture maps
- “texels”: texture elements
- images attached to geometry
- adds visual detail, substitute for geometric detail

# Texture Mapping

## Texture Coordinates

- generation at vertices
  - specified by programmer or artist
 

```
glTexCoord2f (s, t)
```

```
glVertexf (x, y, z)
```
  - generate as a function of vertex coords
 

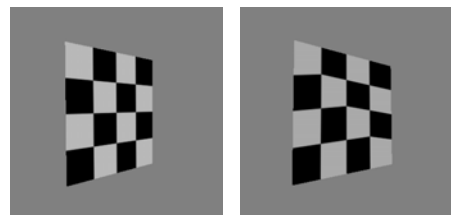
```
glTexGeni (), glTexGenfv ()
```

$$s = a*x + b*y + c*z + d*h$$
- interpolated across triangle (like R,G,B,Z)  
(well, not quite...)

# Texture Mapping

## Texture Coordinate Interpolation

- perspective foreshortening problem
- also problematic for colour interpolation, etc.



## Texture Coordinate Interpolation



### Perspective Correct Interpolation

- $\alpha, \beta, \gamma$ : Barycentric coordinates of a point  $P$  in a triangle
- $s_0, s_1, s_2$ : texture coordinates
- $w_0, w_1, w_2$ : homog coordinates

$$s = \frac{\alpha \cdot s_0 / w_0 + \beta \cdot s_1 / w_1 + \gamma \cdot s_2 / w_2}{\alpha / w_0 + \beta / w_1 + \gamma / w_2}$$

© Wolfgang Heidrich and Michiel van de Pannle

## Texture Mapping



### Texture Coordinate Interpolation

$$P' = \begin{bmatrix} E & 0 & A & 0 \\ 0 & F & B & 0 \\ 0 & 0 & C & D \\ 0 & 0 & -1 & 0 \end{bmatrix} P$$

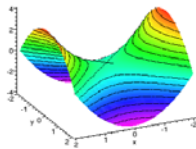
© Wolfgang Heidrich and Michiel van de Pannle

## Texture Mapping



### Textures of other dimensions

- 1D: represent isovalues
    - e.g.: contour lines, temp, ...
- `glTexCoord1f(s)`



© Wolfgang Heidrich and Michiel van de Pannle

## Texture Mapping



### Textures of other dimensions

- 3D: solid textures
    - e.g.: wood grain, medical data, ...
  - 4D: 3D + time, projecting textures
- `glTexCoord3f(s, t, r)`  
`glTexCoord3f(s, t, r, q)`



© Wolfgang Heidrich and Michiel van de Pannle

## Texture Coordinate Transformations



### Motivation:

- Change scale, orientation of texture on an object

### Approach:

- texture matrix stack
- 4x4 matrix stack
- transforms specified (or generated) tex coords

```
glMatrixMode( GL_TEXTURE );
glLoadIdentity();
```

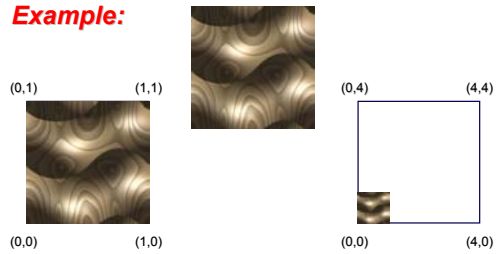
...

© Wolfgang Heidrich and Michiel van de Pannle

## Texture Coordinate Transformations



### Example:



`glScalef( 4.0, 4.0, ? );`

© Wolfgang Heidrich and Michiel van de Pannle

## Texture Coordinate Transformations



### Projective Transformations

- can do projective transformations
- tex coord  $(s,t,r,q) : q \leftrightarrow h$

© Wolfgang Heidrich and Michiel van de Panne

## Texture Coordinate Transformations



### Example:



Brabec and Heidrich

© Wolfgang Heidrich and Michiel van de Panne

## Texture Lookup



### Issue:

- What happens to fragments with  $s$  or  $t$  outside the interval  $[0...1]$ ?

### Multiple choices:

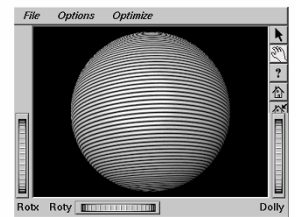
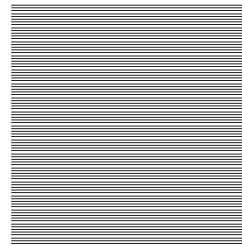
- Take only fractional part of texture coordinates
  - Cyclic repetition of texture to tile whole surface
- Clamp every component to range  $[0...1]$ 
  - Re-use color values from border of texture image

```
glTexParameteri( ..., GL_TEXTURE_WRAP_S,
                 GL_REPEAT )
```

```
glTexParameteri( ..., GL_TEXTURE_WRAP_S,
                 GL_CLAMP )
```

© Wolfgang Heidrich and Michiel van de Panne

## Reconstruction



(image courtesy of Kiriakos Kutulakos, U Rochester)

© Wolfgang Heidrich and Michiel van de Panne

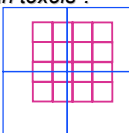
## Reconstruction



- How to deal with:
  - pixels that are much larger than texels ? (apply filtering, "averaging")



- pixels that are much smaller than texels ? (interpolate)

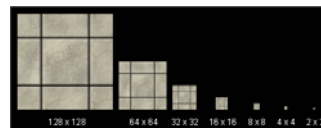


© Wolfgang Heidrich and Michiel van de Panne

## MIP-mapping



Use an "image pyramid" to precompute averaged versions of the texture



Without MIP-mapping



With MIP-mapping

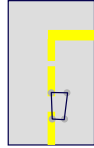
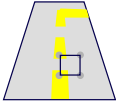
© Wolfgang Heidrich and Michiel van de Panne

## MIP mapping



### **Problem:**

- A MIP-map level selects the same minification factor for both the  $s$  and the  $t$  direction (isotropic filtering)
- In reality, perspective foreshortening (amongst other reasons) can cause different scaling factors for the two directions



© Wolfgang Heidrich and Michiel van de Panne

## MIP mapping



### **Which resolution to choose:**

- MIP-mapping: take resolution corresponding to the smaller of the sampling rates for  $s$  and  $t$ 
  - Avoids aliasing in one direction at cost of blurring in the other direction
- Better: anisotropic texture filtering
  - Also uses MIP-map hierarchy
  - Choose larger of sampling rates to select MIP-map level
  - Then use more samples for that level to avoid aliasing
  - Maximum anisotropy (ratio between  $s$  and  $t$  sampling rate) usually limited (e.g. 4 or 8)

© Wolfgang Heidrich and Michiel van de Panne