

## Texture Mapping: Solid Texturing



CSE 681

## Texture Mapping

Visual complexity on demand

Vary display properties over object

Visible **pixel** maps to **location** on object

**Location** on object  
used to **lookup** display **attributes**

Or  
as **function parameters** to generate **attributes**

CSE 681

## Solid Texture Mapping

Object is 'carved' out of textured volume

Use x,y,z location of pixel

Use location in simple procedure to generate, e.g.

- Material color to be used in shading calculation
- Ambient, diffuse, or specular reflection coefficient
- Opacity
- Final color

World space coordinates v. object space coordinates?

CSE 681

## Solid Texture Map Coordinates

If world space

Ok in static scenes

Object moves through texture if object animated

If object space

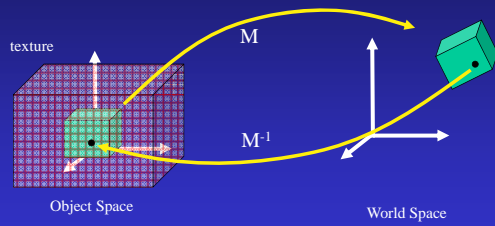
Texture is 'fixed' to object

need to inverse transform intersection

or need to trace inverse ray in object space

CSE 681

## Solid Texture Map Coordinates



CSE 681

## Space Filling Stripes

Uses: modulo divisor %

```
jump(x,y,z) = ((int)(x))%2
if (jump == 0) color = yellow
else if (jump == 1) color = red
```

0.....1.....0



0...s.x...2\*s.x...3\*s.x

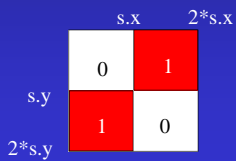
```
jump(x,y,z) = ((int)(A + x/s.x))%2
if (jump == 0) color = yellow
else if (jump == 1) color = red
```



CSE 681

## Space Filling 2D Checkerboard

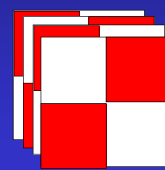
```
jump(x,y,z) = ((int)(A+x/s.x)+(int)(A+y/s.y))%2
if (jump == 0)
color = yellow
Else if (jump == 1)
color = red
```



CSE 681

## Space Filling 3D Checkerboard

```
jump(x,y,z) = ((int)(A+x/s.x)+(int)(A+y/s.y)+(int)(A+z/s.z))%2
if (jump == 0)
color = yellow
Else if (jump == 1)
color = red
```



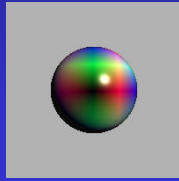
CSE 681

## Cube of Smoothly Varying Colors

Uses  $\text{fract}(x) = x - (\text{floor})(x)$

$\text{Texture}(x,y,z) = (1 - |2*\text{fract}(x)-1|, 1-|2*\text{fract}(y) - 1|, 1-|2*\text{fract}(z)-1|)$

0....1....0



CSE 681

## Rings

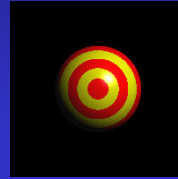
$\text{rings}(r) = (\text{int}(r)) \% 2$

$r = \text{sqrt}(x^2+y^2)$ ;

$\text{rings}(x,y,z) = D + A * \text{rings}(r/M)$

M - thickness

D & A  
scale and translate into  
arbitrary values



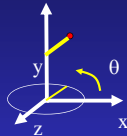
Or, as before, map 0 & 1  
into yellow and red

CSE 681

## Wood Grain

Twist:

Rotate texture around y-axis by  $\theta$



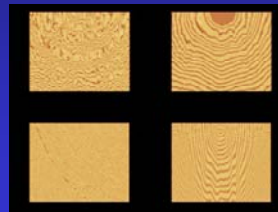
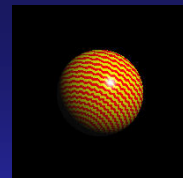
Implement by rotating point by  $-\theta$  around y-axis

Similarly, rotate  $(x,y,z)$  point around z-axis

CSE 681

## Wood Grain

Add random perturbation to  $(x,y,z)$   
to create jitter wood grain



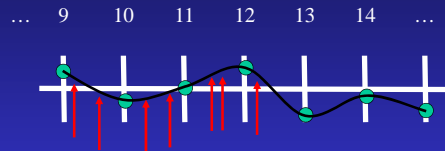
CSE 681

## Noise, Turbulence, Marble

- Define function of random values which is
  - A function of 3D point
  - continuous
  - repeatable
- Use 3D point to retrieve random value
- 3D volume has frequency determined by spacing of random values
- Scale point first to change frequency of noise function

CSE 681

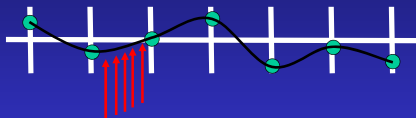
## 1D Noise Example



- Deposit random values at integer locations
- Interpolate through values to get continuous function
- Sample function at intersection points of object with ray

CSE 681

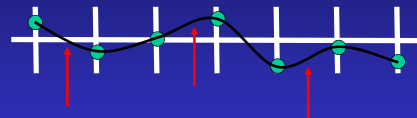
## 1D Noise Example



Sample too frequently - no randomness

CSE 681

## 1D Noise Example

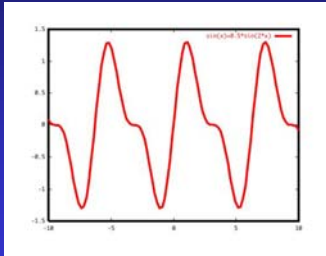


Sample too sparsely - no continuity  
(Nyquist limit)

CSE 681

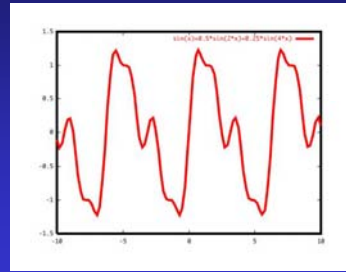


## 1D Turbulence Example



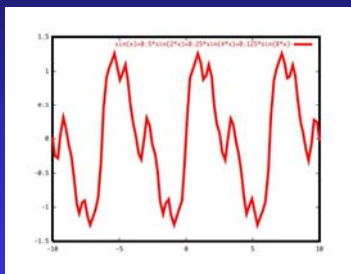
CSE 681

## 1D Turbulence Example



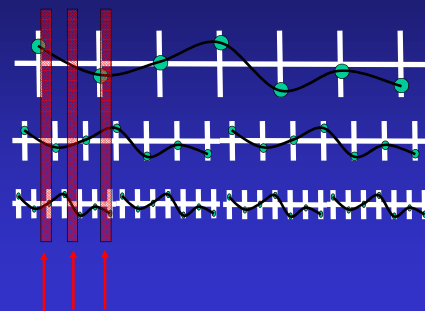
CSE 681

## 1D Turbulence Example

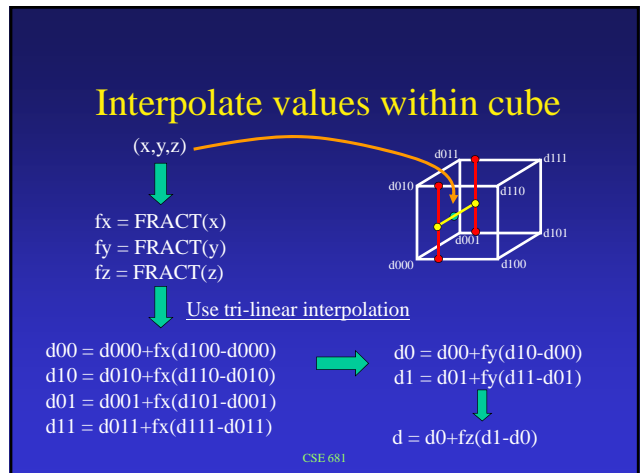
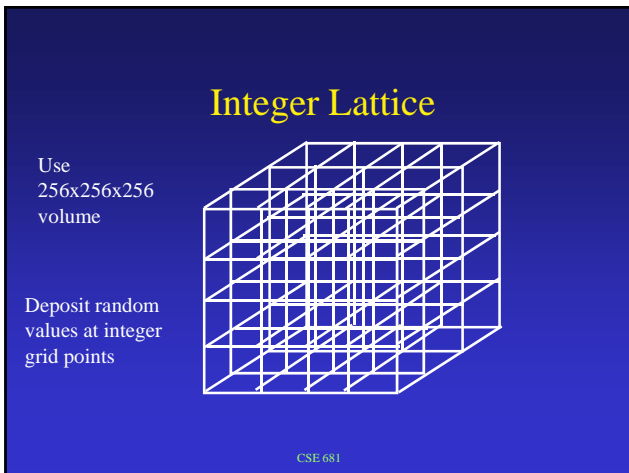
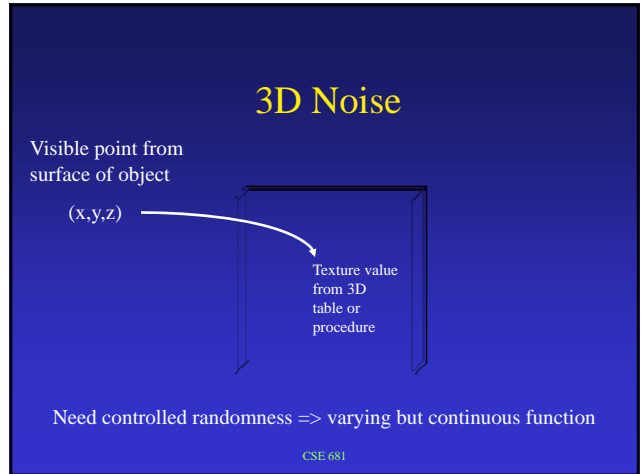
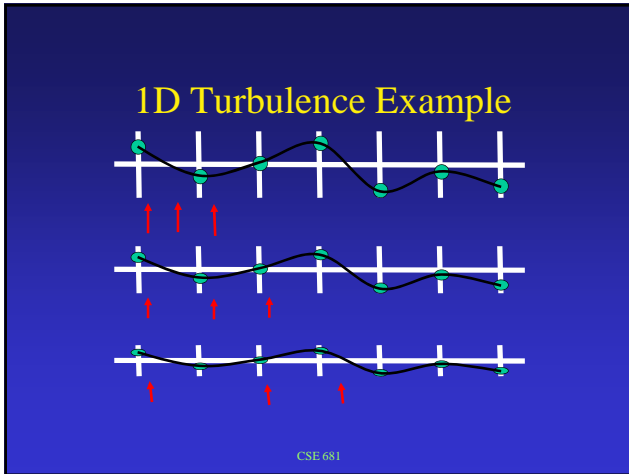


CSE 681

## 1D Turbulence Example



CSE 681



## Implementation notes

NoiseTable[256]: random values [0, 1]

Index[256]: random permutation of values 0:255

```
#define PERM(x) index[x & 255]
#define INDEX(ix,iy,iz) PERM( ix + PERM(iy + PERM(iz)))
```

```
Float latticeNoise(i,j,k)
Return NoiseTable[INDEX(i,j,k)]
```

CSE 681

## Turbulence implementation

Noise(s,x,y,z)

Scale point by s, add 1000 to each coordinate

Get integer (ix,iy,iz) and fractional parts (fx,fy,fz)

Get cell lattice noise values

d000,d001,d010,d011, d100,d101,d110,d111

Do the trilinear interpolation by fx,fy,fz

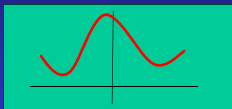
$\text{Turb}(s,x,y,z,k) = (1/2) \sum (1/2^k) \text{noise}(2^k,x,y,z)$

Where k is the number of frequencies

CSE 681

## Marble Texture

Undulate(x) - basic ripple in x



$\text{Marble}(x,y,z) = \text{undulate}(\sin(2\pi xyz + A * \text{turb}(s,x,y,z,k)))$

Parameters: amplitude, scale, number of frequencies

CSE 681

## Marble Texture

See examples

[www.cse.ohio-state.edu/~parent/classes/681/AU09/Labs/Lab4/solidTexture.html](http://www.cse.ohio-state.edu/~parent/classes/681/AU09/Labs/Lab4/solidTexture.html)

CSE 681