

Texture Cube

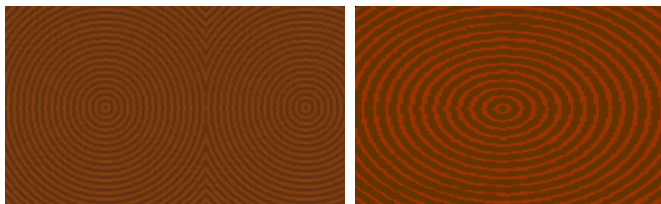
- Define a three dimensional array
- Loop through the size of each dimension
- Specify RGB values for each voxel

- Assign texture coordinates to particular object

Voxel: is a discrete value in our texture volume grid

Texture Coordinates

- Specify for each vertex the placement of our object in the texture cube
- OpenGL function: e.g.:
`glTexCoord3f(0.0, 0.3, 0.4)`
- Be aware of deformations and non-fitting textures



Essential Functions

- `glEnable(GL_TEXTURE_3D)` to enable 3D texture mapping
- `glTexParameteri(...)` to set texture properties
- `glTexImage3D(...)` to specify the voxel values

glTexImage3D

Important parameters:

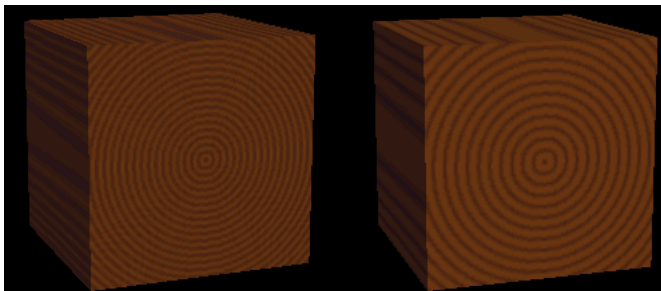
- target: wich kind of texture mapping we would like → `GL_TEXTURE_3D`
- width, height, depth of our texture cube
- type: defines the datatype of one pixel data
- pixels: structured pixel values

Wood Grain

- Woodgrain effects easily simulated using embedded cylinder equations
- Cylinder: $r = \sqrt{(x^2 + y^2)}$
- Perturb (deviate) wood growth:
 $r = r + 2 * \sin(20 * \text{angle} + z/150)$ where angle is $\arctan(x, z)$

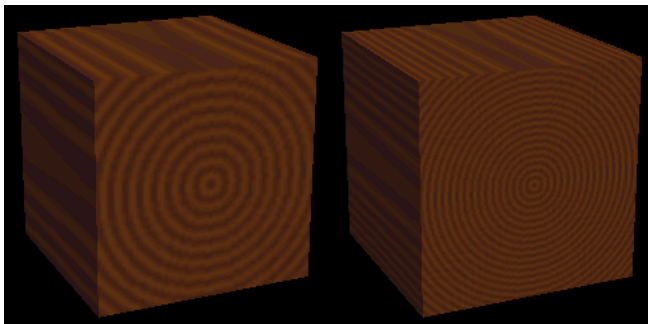
Wood Grain

- Then use the `mod` function to simulate embedded cylinders, and map r size to a colour (eg. dark, light)



Wood Grain

- Different texture cube densities lead to different accuracy



64^3 ca. 0,75MB and 128^3 ca. 6MB voxels

Perlin Noise

- In nature, everything has a random look - mathematical formulas typically don't generate random looking results
- Perlin noise, invented by *Ken Perlin*, uses random numbers to generate natural looking textures

Perlin Noise

- As source for the Perlin noise we need an 3-dimensional array of random values, called `noise[x][y][z]`
- The function `generateNoise()` will fill the array with noise (random numbers between 0 and 1)

Generate Noise

```
void generateNoise()
{
    for (x = 0; x<WIDTH; x++)
        for (y = 0; y<HEIGHT; y++)
            for (z = 0; z<DEPTH; z++)
                noise[x][y][z] = random();
}
```

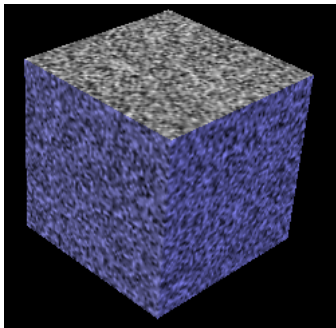
`random()` should create a random number between 0 and 1

Noise Result

If we now say,

$$\text{color}[x][y][z] = 255 * \text{noise}[x][y][z]$$

we will have this result:



Problems with Noise

This noise doesn't look very natural, especially if you zoom in you get something blocky:



When zooming in, we want something smoother
⇒ for that, linear interpolation can be used

Smooth Noise

By using bilinear interpolation on the fractional part, you can make it smoother. For that, a new function, `smoothNoise()`, is introduced:

- 1 Get fractional part of x , y and z
- 2 Calculate neighbor values
- 3 Smooth the noise with bilinear interpolation

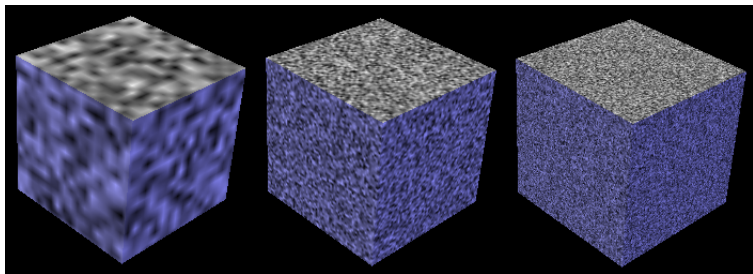
Turbulence

- Turbulence is what creates natural looking features out of smoothed noise
- An example of how this represents nature can be found in a mountain range or a coast



Turbulence

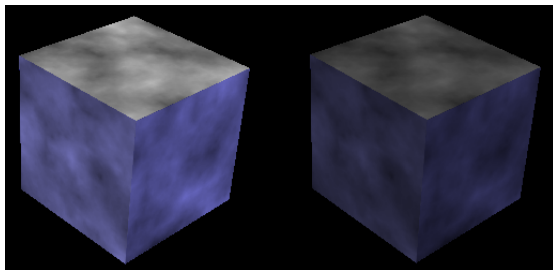
The trick is to add multiple noise textures of different zooming scales together:



Here we have accuracies of 16, 64, and 128

Turbulence

By adding these n images together, and dividing the result through n to get the average, you get a turbulence texture:



Make the images with a smaller zoom darker, so adding them will have less effect

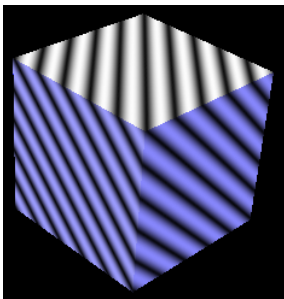
Turbulence

Here's a function that'll automatically do all this for a single pixel:

```
float turb(float x, float y, float z, float size)
{
    float value = 0.0; float initialSize = size;
    while(size >= 1)
    {
        value += smooth(x/size, y/size, z/size)*size;
        size /= 2.0;
    }
    return(128.0*value/initialSize);
}
```

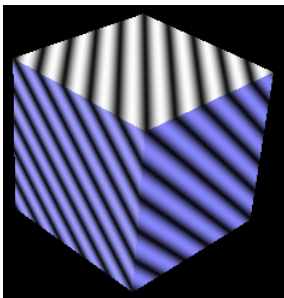
Sine pattern

- It's possible to use Perlin Noise to create a texture that looks like marble
- To do this, a sine pattern is taken as base, a sine pattern looks like this:



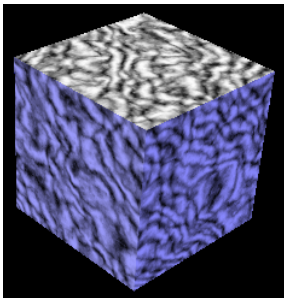
Sine pattern

- The sine texture is generated by giving the pixel at position (x, y, z) the color value $255 * \sin(x + y + z)$
- Change the angle and period (= amount of lines) by multiplying x and y with factors



Marble

- By applying turbulence to these lines by adding a turbulence term in the sine, you get something that looks like the veins of marble:



Marble

- Define repetition of marble lines in x , y and z direction:
`xPeriod = yPeriod = zPeriod = 5.0;`

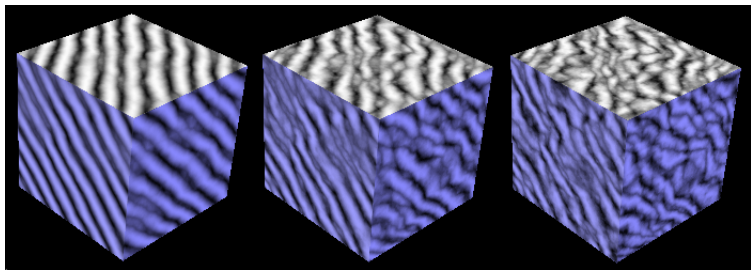
- Make twists
`turbPower = 5.0;`

- Initialize size of the turbulence
`turbSize = 32.0;`

```
xyzValue = x*xPeriod/HEIGHT +  
           y*yPeriod/WIDTH +  
           z*zPeriod/DEPTH +  
           turbPower*turb(x,y,z,turbSize)/256.0;  
sineValue = 256 * fabs(sin(xyzValue));
```

Marble

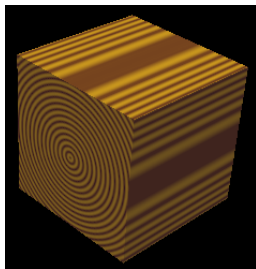
- `xPeriod`, `yPeriod`, `zPeriod` and `turbPower` are parameters that can be changed to get different textures



- Here `turbPower` is set to 1, 2 and 3

Wood

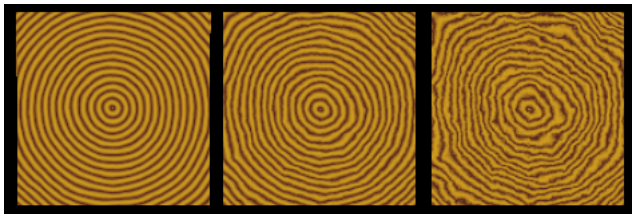
- Natural looking rings of wood can be created by adding turbulence to the following mathematical function:



- To get the pattern above, take the sine of the distance of x, y and z to the center, so the color of the pixel at position (x, y, z) is $256 * \sin(\sqrt{x^2 + y^2})$

Wood

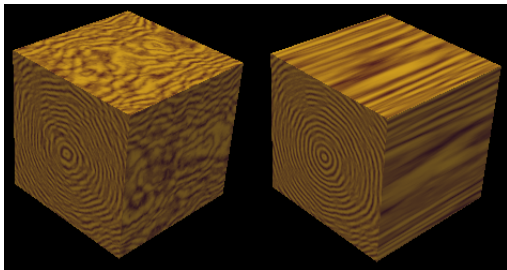
- Add a turbulence term into the sine, and you get natural looking wood



- The rings are supposed to be visible here so, unlike for the marble, turbPower should be small

Wood

- To get a more wood-like object decrease the turbulence for the z-value of each point



- In the right picture the *turbulence* is set to $0.7 * turbulence$

Thanks for your attention

