

Texture

- Pattern of Intensity and color.
 - Can be generalized to 3D texture.
- How do we get them?
 - Take pictures.
 - Write a program (procedural textures).
 - Synthesize from examples
- How do we apply them? (Texture mapping)
 - Specify a mapping from texture to object.
 - Interpolate as needed.
 - This can be a challenging problem, but we'll consider simpler version.

No Textures

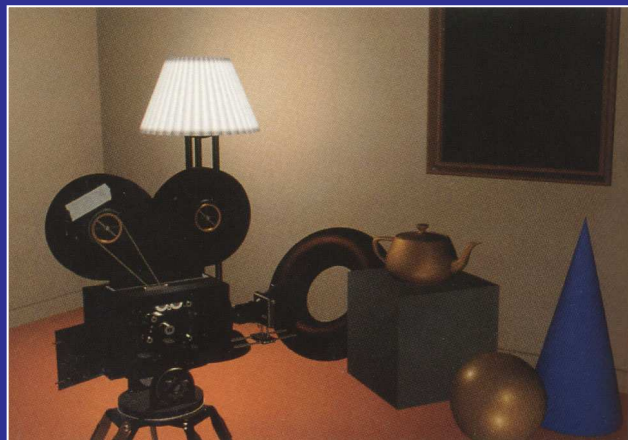


Image courtesy, Foley, van Dam, Feiner, Hughes

With Textures



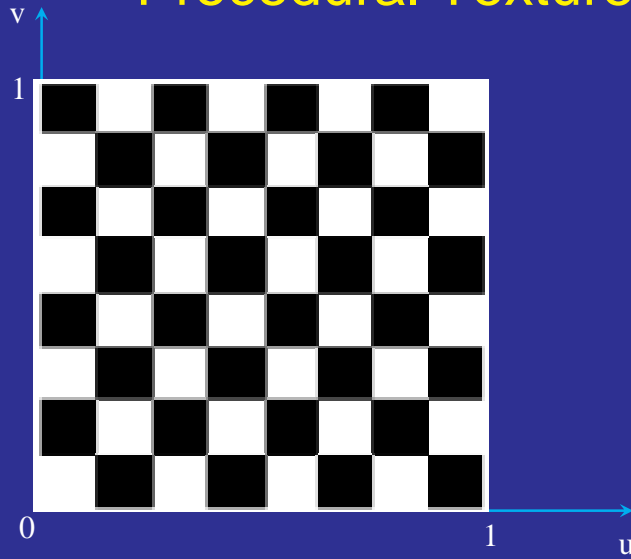
Image courtesy, Foley, van Dam, Feiner, Hughes

Texture Image

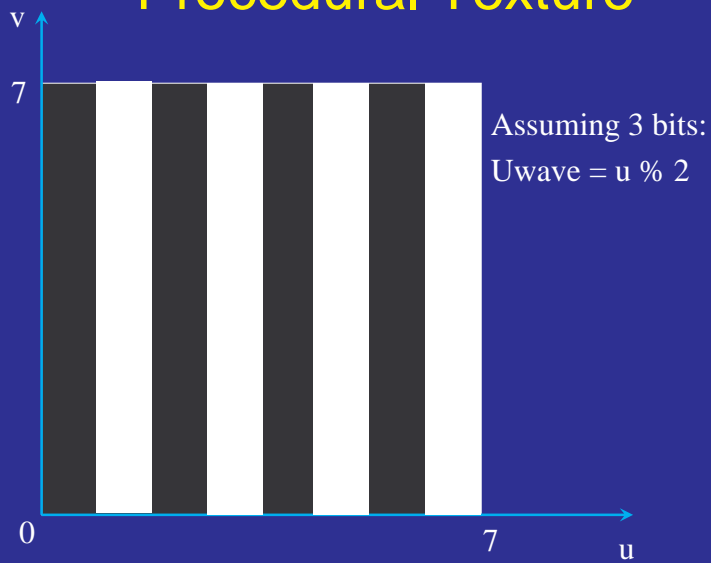


Courtesy, <http://titan.spaceports.com/~seymor>

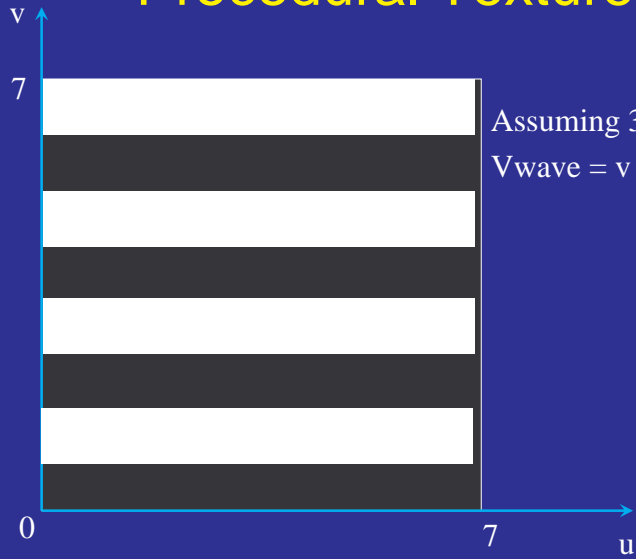
Procedural Texture



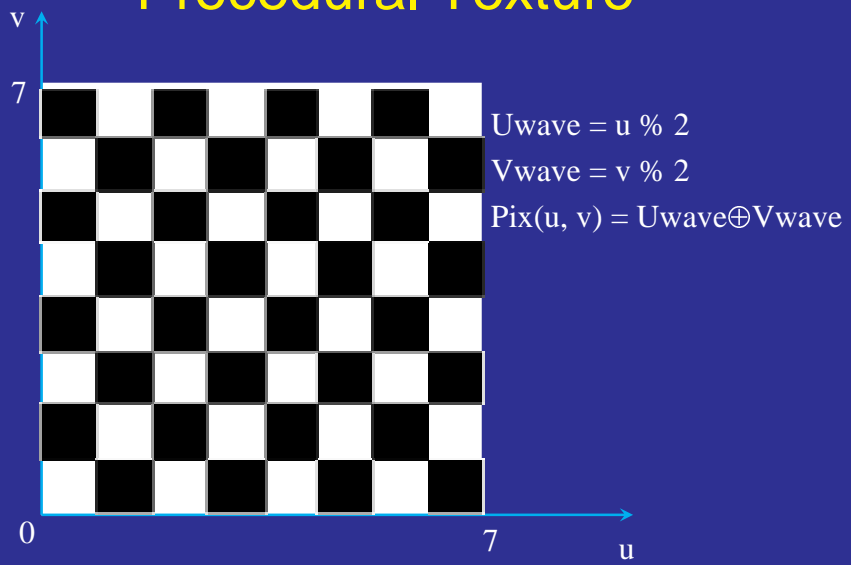
Procedural Texture



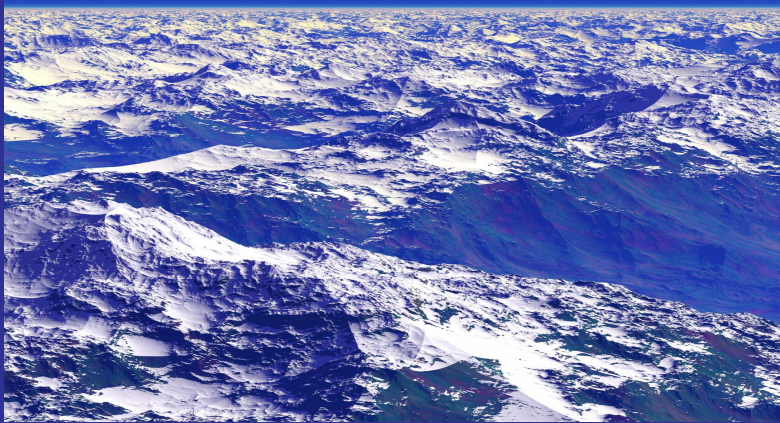
Procedural Texture



Procedural Texture



Example Procedural Models



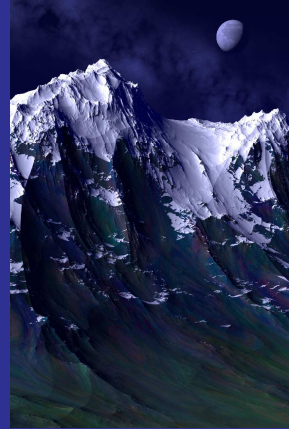
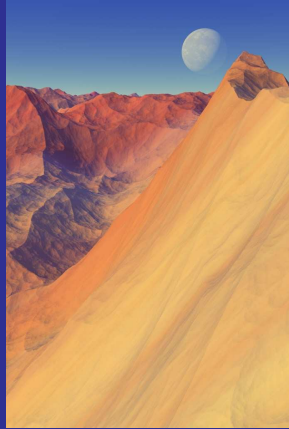
Images from Texturing and Modeling: A Procedural Approach
By Ebert, Musgrave, Peachey, Perlin, and Worley

Example Procedural Models



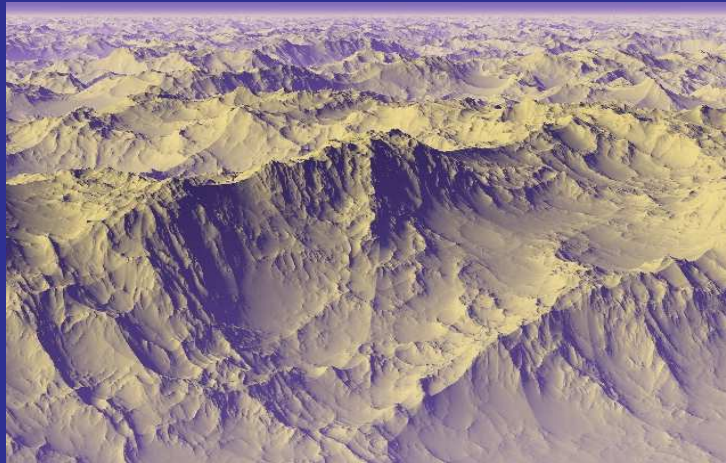
Images from Texturing and Modeling: A Procedural Approach
By Ebert, Musgrave, Peachey, Perlin, and Worley

Example Procedural Models



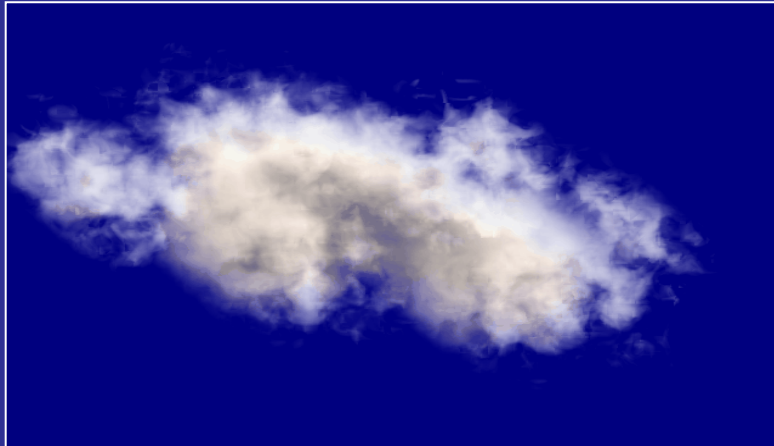
Images from Texturing and Modeling: A Procedural Approach
By Ebert, Musgrave, Peachey, Perlin, and Worley

Example Procedural Models



Images from Texturing and Modeling: A Procedural Approach
By Ebert, Musgrave, Peachey, Perlin, and Worley

Example Procedural Textures



Images from *Texturing and Modeling: A Procedural Approach*
By Ebert, Musgrave, Peachey, Perlin, and Worley

What is *Texture*?

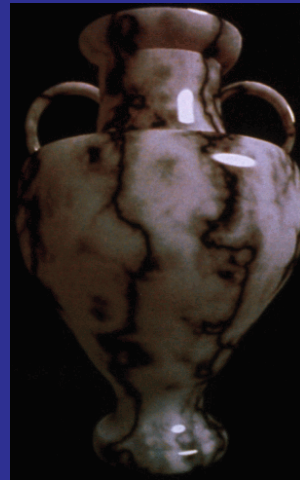
- Something that repeats (or continues, or has structure) with variation.
- Must separate what repeats and what stays the same.
- Model as repeated trials of a random process
 - The probability distribution stays the same.
 - But each trial is different.
 - This may be true (eg., pile of objects)
- Can be added structure (general shape of mountain, plus rocky variation).

Perlin Noise

- Natural phenomenon derives its richness from variations
- But we don't want *white* noise!
- Almost all movies use some form of Perlin noise:
 - James Cameron Movies (Abyss, Titanic,...)
 - Animated Movies (Lion King, Moses,...)
 - Arnold Movies (T2, True Lies, ...)
 - Star Wars Episode I, Star Trek Movies
 - Batman Movies
 - Refer noisemachine.com for details

Perlin Noise

- Reproducibility
- No repeatability
- Band limited (smoothly changing)
- User control



Strategy

- Generate smooth noise.
 - Generate noise with particular frequency
 - Pick random values
 - Interpolate to connect them smoothly.
 - Sum noise over many frequencies
- Map real numbers to intensity or color in interesting way.
 - Turn numbers into color.
 - Distort numbers spatially
 - Turbulence
 - Non-linear mappings.

1D Perlin Noise

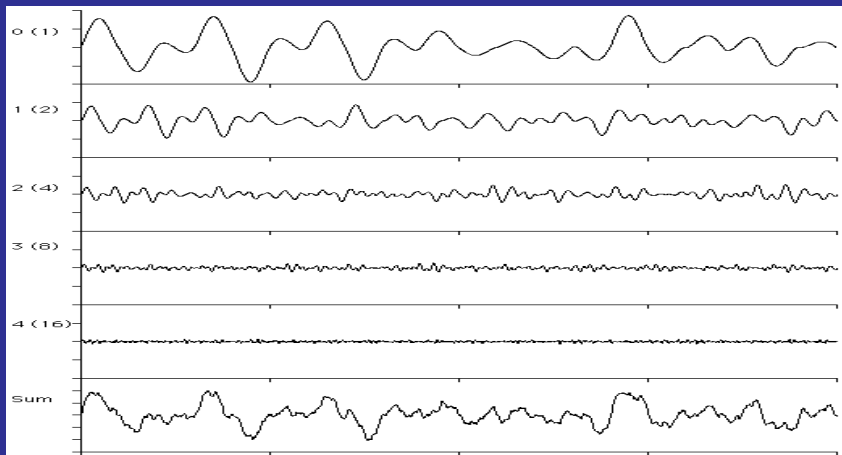
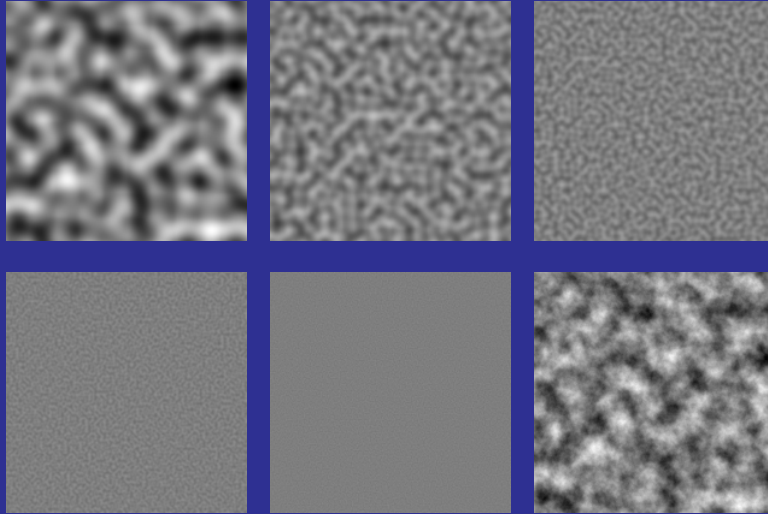


Image courtesy of Paul Bourke

2D Perlin Noise



Images courtesy of Paul Bourke

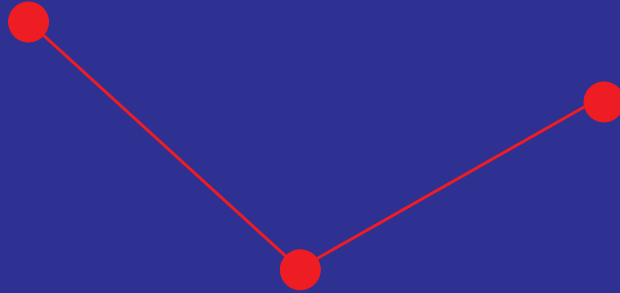
Smooth Noise

1. Populate an integer grid with random values.
2. Interpolate in between to get values at non-integer points
 - This creates a kind of smoothing

Advantages:

1. Randomness at grid points
2. Band limited

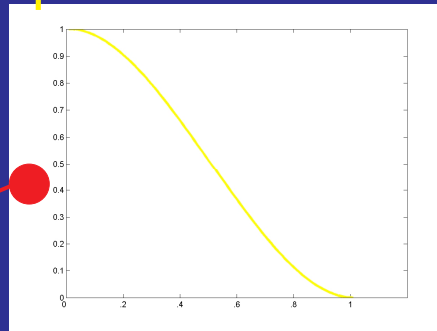
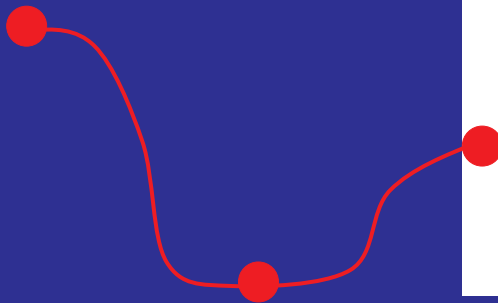
Linear Interpolation



$$f(x) = \text{floor}(x) \cdot (x - \text{floor}(x)) + \text{ceiling}(x) \cdot (\text{ceiling}(x) - x)$$

- Average of neighboring points weighted by distance to them.

Cubic Interpolation



Instead of weighting by distance, d , weight by:

$$1 - 3d^2 + 2|d|^3$$

- Smooth
- Symmetric

Suppose $0 \leq x \leq 1$, and a function f is defined on $f(0), f(1)$. We want to define it for $f(x)$ so that $f(x)$ is smooth.

If we do this by averaging neighbors, we have:

$f(x) = g(x)f(0) + g(1-x)f(1)$. Then we want a function g that is smooth, and in which $g(0) = 1$ and $g(1) = 0$, and in which g is symmetric so that $g(x) + g(1-x) = 1$.

With linear interpolation $g(x) = 1-x$. This fits the second two criteria, but this g is not smooth. There is a discontinuity at $f(0)$, since we suddenly switch between averaging $f(0)$ and $f(1)$ and averaging $f(0)$ and $f(-1)$

So instead, we want $f(x)$ near $f(0)$ to be based mostly on the value of $f(0)$, and only to gradually average in $f(1)$ as we get closer to it.

A nice function that does this is $1 - 3x^2 + 2x^3$

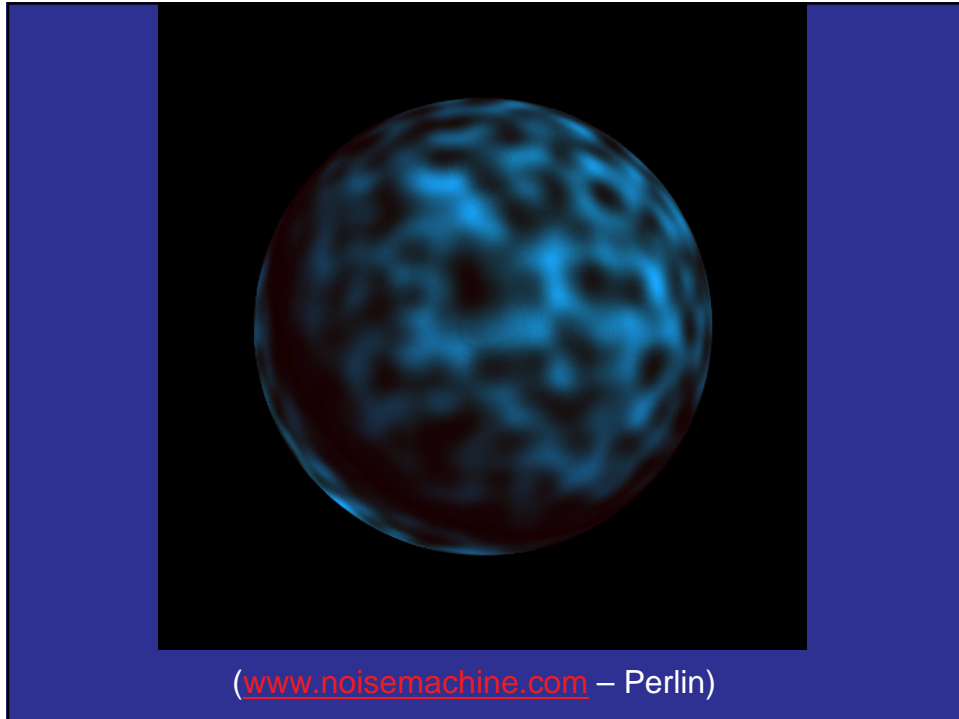
Note that $g(1-x) = 1 - 3(1-x)^2 + 2(1-x)^3$

$$= 1 - 3 + 6x - 3x^2 + 2 - 6x + 6x^2 - 2x^3 = 3x^2 - 2x^3$$

$$= 1 - (1 - 3x^2 + 2x^3)$$

Gradient (Perlin) Noise

- Generate unit vectors (a, b) at each integer lattice point (ix, iy)
 - Use zero intensity at each lattice point.
- For a non-integer evaluation point (x, y) determine the 4 nearest integer grid points.
- For each integer grid point (ix, iy) , find the fractional value of the vector (fx, fy) from (ix, iy) to (x, y, z) and take its dot product with the gradient vector at that integer point (ix, iy) .
- Interpolate the 4 dot-product values to compute the noise value at the grid point
- This has more high-frequency energy than value noise.



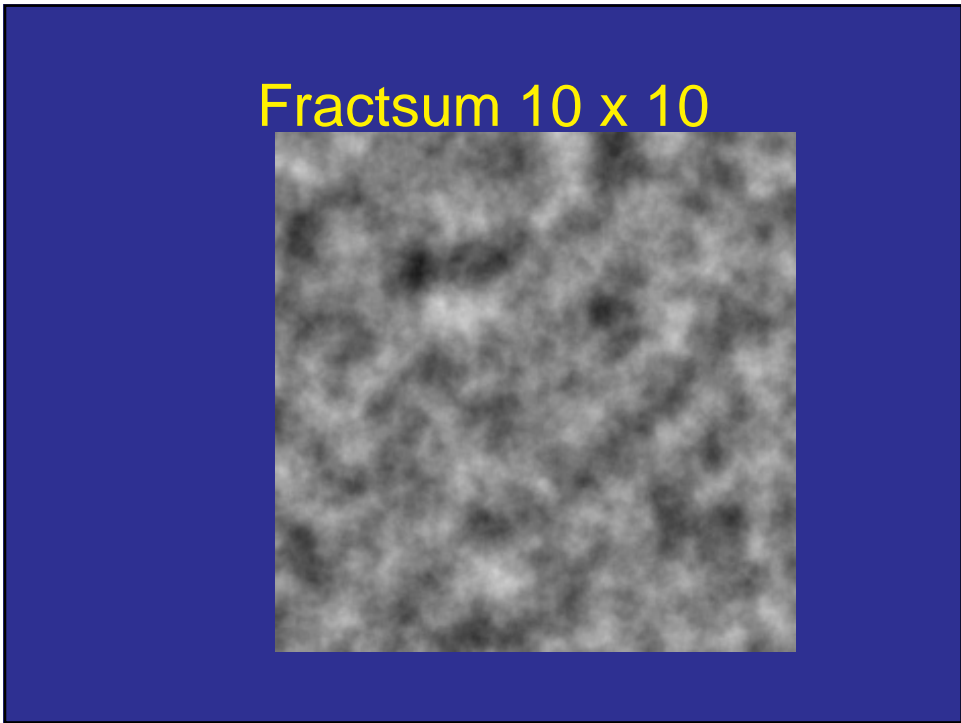
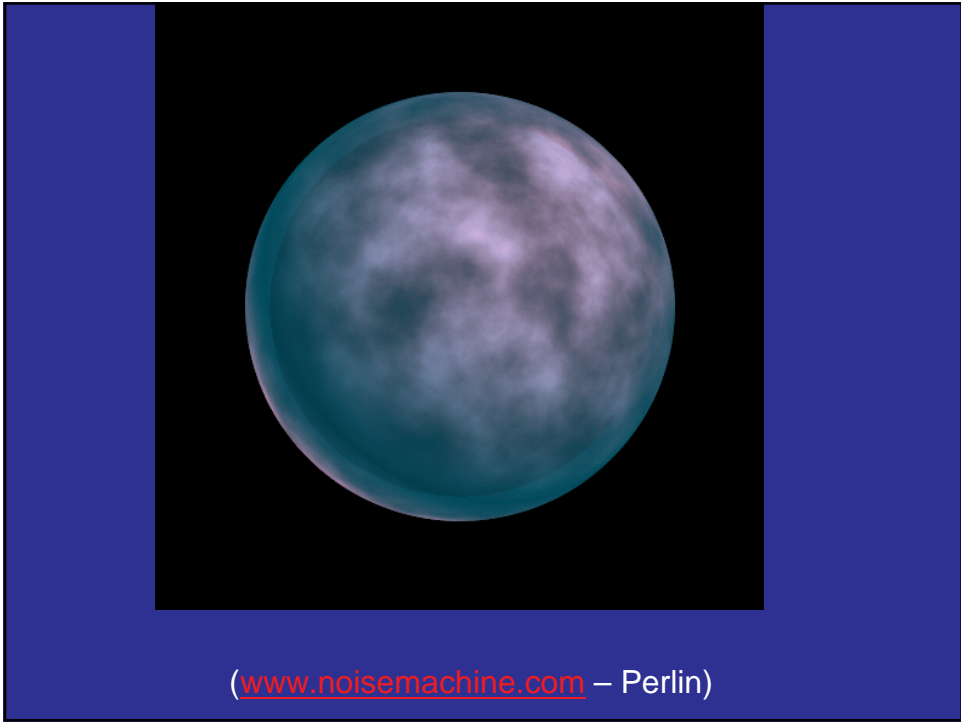
Multiple Octaves

How about if we introduce more frequencies?

```
double fractalsum(double x, double y, double z)
{
    double value = 0;
    double f;

    for(f = MINFREQ; f < MAXFREQ; f *= 2)
        value += gnoise(x * f, y * f) / f;
    return(value);
}
```

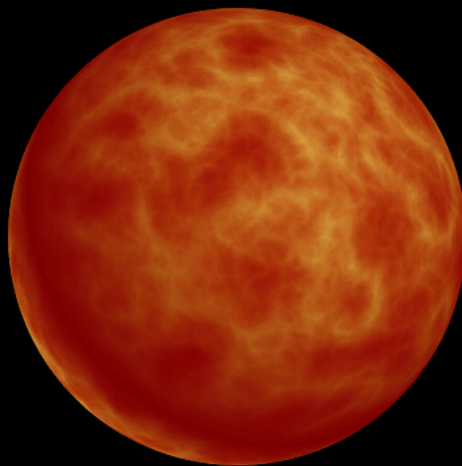
Demo of assignment executable



Perlin Turbulence

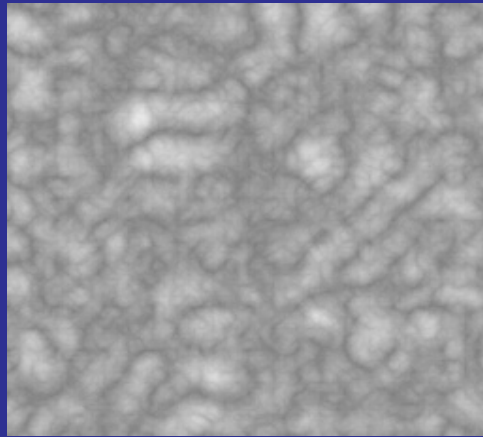
```
double turbulence(double x, double y, double z)
{
    double value = 0;
    double f;

    for(f = MINFREQ; f < MAXFREQ; f *= 2)
        value +=
            fabs(gnoise(x * f, y * f, z * f) / f);
    return(value);
}
```



(www.noisemachine.com – Perlin)

Turbulence 10 x 10

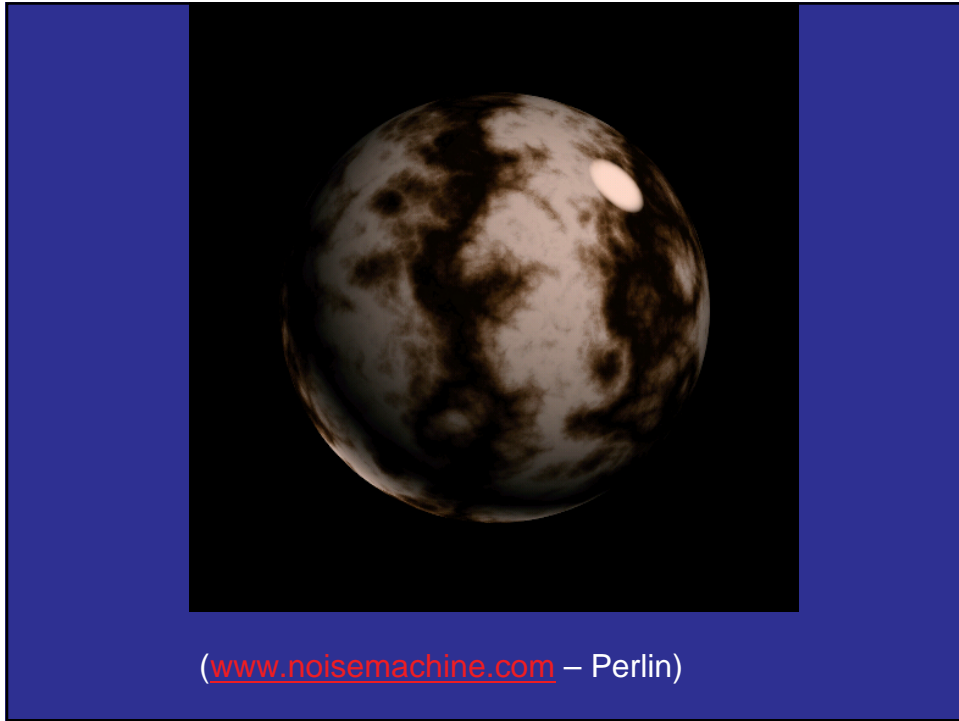


Spatial regularities

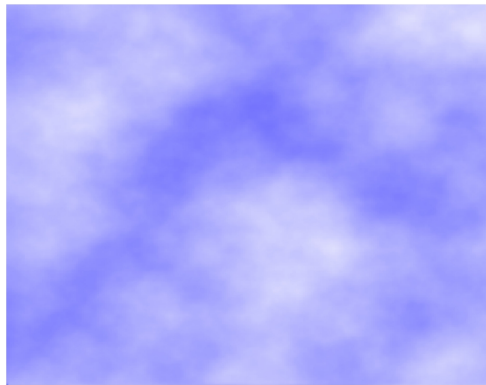
- Multiply intensities by sine of x coordinates.
- *Demo*

Map Intensities to Color

Demo



Nonlinear changes in intensity



3D Textures

- Can use to generate video
- Move through 2D slices
- *Example*

Summary

- Smooth randomness.
 - More low frequencies than high.
 - Only low frequencies are kind of boring.
 - World seems to have such frequencies.
- User Control
 - Spatial and color variations.
 - Controls somewhat intuitive.

OpenGL Texturing

- Create and specify a texture object
 - Create a texture object
 - Specify the texture image
 - Specify how texture has to be applied for each pixel
- Enable texture mapping
- Draw the textured polygons
 - Identify the active texture
 - Specify texture coordinates with vertices

Specify a 2D Texture Object

- ***glTexImage2D***(GLenum *target*, GLint *level*, GLint *internalformat*, GLsizei *width*, GLsizei *height*, GLint *border*, GLenum *format*, GLenum *type*, const GLVoid **texels*);
 - Eg: ***glTexImage2D***(GL_TEXTURE_2D, 0, GL_RGBA, 128, 128, 0, GL_RGBA, GL_UNSIGNED_BYTE, *image*);
 - ***format*** and ***type*** used to specify the way the texels are stored
 - ***internalFormat*** specifies how OpenGL should store the data internally
 - ***width*** and ***height*** have to be powers of 2; you can use ***gluScaleImage***() to scale

Specify how Texture is applied

- **glTexParameter{f}**(GLenum target, GLenum pname, TYPE param)
- **target** can be: GL_TEXTURE_1D, GL_TEXTURE_2D, ...

<i>pname</i>	<i>param</i>
GL_TEXTURE_WRAP_S	GL_CLAMP, GL_REPEAT
GL_TEXTURE_WRAP_T	GL_CLAMP, GL_REPEAT
GL_TEXTURE_MAG_FILTER	GL_NEAREST, GL_LINEAR
GL_TEXTURE_MIN_FILTER	GL_NEAREST, GL_LINEAR

Enable the Texture and Draw

- **glEnable**(GL_TEXTURE_2D)
 - Enable 2D texturing
- **glTexCoord2f**(GL_FLOAT u, GL_FLOAT v)
 - Specify texture coordinates per vertex (just as normals, color, etc).
 - *demo*

Create a Texture Object

- **`glGenTextures(GLsizei n, GLuint* textureIDs);`**
 - Returns *n* currently unused texture ID in *textureIDs*
 - Each texture ID is an integer greater than 0
- **`glBindTexture(GLenum target, GLuint textureID);`**
 - *target* is `GL_TEXTURE_1D`, `GL_TEXTURE_2D`, or `GL_TEXTURE_3D`
 - if *textureID* is being used for the first time a new texture object is created and assigned the ID = *textureID*
 - if *textureID* has been used before, the texture object with ID = *textureID* becomes active

Putting it all together

In initialization:

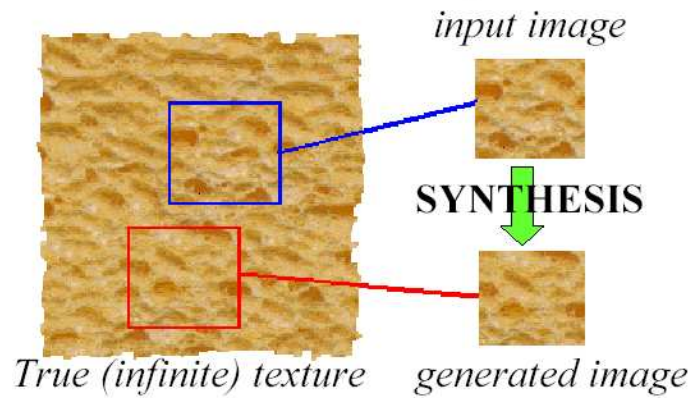
```
glGenTextures(...);  
glBindTexture( ... );  
glTexParameteri(...); glTexParameteri(...); ...  
glTexImage2D(...);  
glEnable(GL_TEXTURE_2D);
```

In display:

```
glBindTexture( ... ); // Activate the texture defined in  
initialization  
glBegin(GL_TRIANGLES);  
glTexCoord2f(...); glVertex3f(...);  
glTexCoord2f(...); glVertex3f(...);  
glTexCoord2f(...); glVertex3f(...);  
glEnd( );
```

Texture Synthesis

The Goal of Texture Synthesis



Synthesis as probabilistic modeling

- Infer probability distribution for pixels
- Fill in pixel based on probability distribution

Simple Statistics

- Assume every pixel is independent
- Make new texture with same intensity probabilities as old.
 - Example, if sample has 1/20 of pixels with intensity 117, then a pixel in new sample has intensity 117 with probability 1/20.
 - *Demo*

Markov Model

- Pixels independent leads to white noise.
 - Markov Model Captures local dependencies.
 - Each pixel depends on neighborhood.
 - Example, 1D first order model
- $$\begin{aligned} P(p_1, p_2, \dots, p_n) &= \\ &P(p_1) * P(p_2|p_1) * P(p_3|p_2, p_1) * \dots \\ &= P(p_1) * P(p_2|p_1) * P(p_3|p_2) * P(p_4|p_3) * \dots \end{aligned}$$

Markov model of Printed English

- From Shannon: “A mathematical theory of communication.”
- Think of text as a 1D texture
- Choose next letter at random, based on previous letters.

•Zero'th order:

XFOML RXKHJFFJUJ ZLPWCFWKCYJ
FFJEYVKCQSGHYD
QPAAMKBZAACIBZIHJQD

- Zero'th order:

XFOML RXKHJFFJUJ ZLPWCFWKCYJ
FFJEYVKCQSGHYD
QPAAMKBZAACIBZIHJQD

- First order:

OCRO HLI RGWR NMIELWIS EU LL
NBNESEBYA TH EEI ALHENHTTPA
OOBTTVA NAH BRI

- First order:

OCRO HLI RGWR NMIELWIS EU LL
NBNESEBYA TH EEI ALHENHTTPA
OOBTTVA NAH BRI

- Second order

ON IE ANTSOUTINYS ARE T
INCTORE T BE S DEAMY ACHIN D
ILONASIVE TUCOOWE AT
TEASONARE FUSO TIZIN ANDY
TOBE SEACE CTISBE


•Second order

ON IE ANTSOUTINYS ARE T
INCTORE T BE S DEAMY ACHIN D
ILONASIVE TUCOOWE AT
TEASONARE FUSO TIZIN ANDY
TOBE SEACE CTISBE

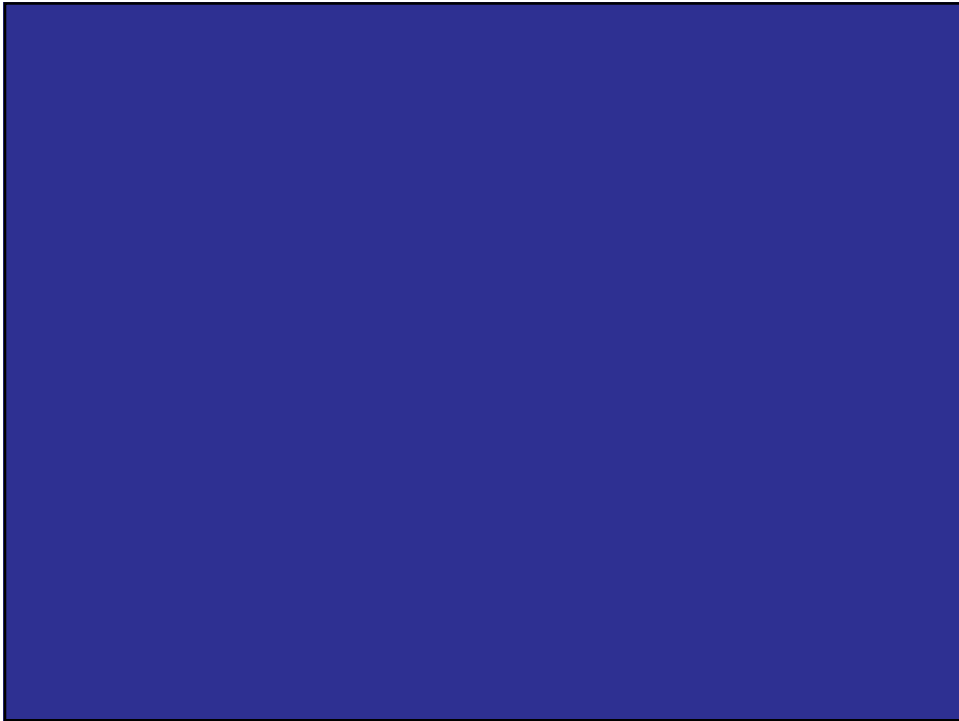
Third order:

IN NO IST LAT WHEY CRATICT FROURE
BIRS GROCID PONDENOME OF
DEMONSTURES OF THE REPTAGIN IS
REGOACTIONA OF CRE.

- Zero'th order: XFOML RXKHJFFJUJ
ZLPWCFWKCYJ FFJEYVKCQSGHYD
QPAAMKBZAACIBZIHJQD
- First order: OCRO HLI RGWR NMIELWIS EU
LL NBNESEBYA TH EEI ALHENHTTPA
OOBTTVA NAH BRI
- Second order ON IE ANTSOUTINYS ARE T
INCTORE T BE S DEAMY ACHIN D
ILONASIVE TUCOOWE AT TEASONARE
FUSO TIZIN ANDY TOBE SEACE CTISBE
- Third order: IN NO IST LAT WHEY CRATICT
FROURE BIRS GROCID PONDENOME OF
DEMONSTURES OF THE REPTAGIN IS
REGOACTIONA OF CRE.



I know that I shall meet my fate
Somewhere among the clouds above;
Those that I fight I do not hate,
Those that I guard I do not love;
My countrymen Kiltartan Cross,
My countrymen Kiltartan's poor,
No likely end could bring them loss
Or leave them happier than before.



Nor law, nor duty bade me fight,
Nor public men, nor cheering crowds,
A lonely impulse of delight
Drove to rocid this tumult in the clouds;
I balanced all, brought all to mind,
The years to come seemed waste of breath,
A waste of breath the years behind
In balance with this life, this death.

Markov models of words

- First order:

REPRESENTING AND SPEEDILY IS AN GOOD APT
OR COME CAN DIFFERENT NATURAL HERE HE
THE A IN CAME THE TO OF TO EXPERT GRAY
COME TO FURNISHES THE LINE MESSAGE HAD
BE THESE.

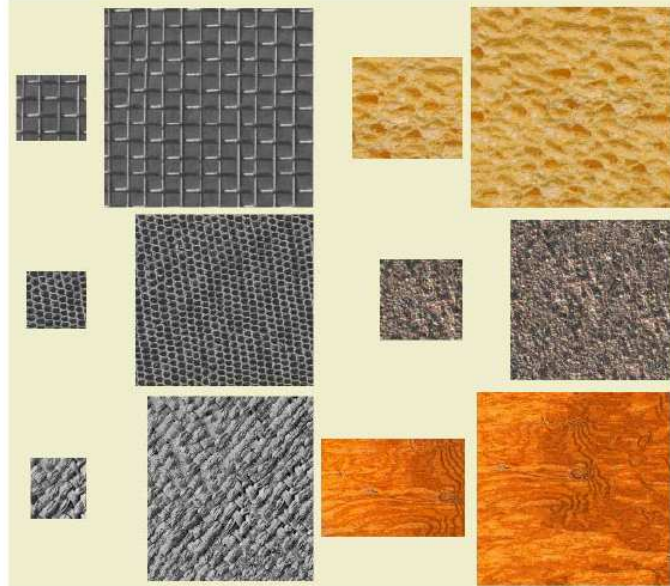
- Second order:

THE HEAD AND IN FRONTAL ATTACK ON AN
ENGLISH WRITER THAT THE CHARACTER OF
THIS POINT IS THEREFORE ANOTHER METHOD
FOR THE LETTERS THAT THE TIME OF WHO
EVER TOLD THE PROBLEM FOR AN
UNEXPECTED.

Same thing for pixels

- Each pixel depends on neighbors.
 1. As you synthesize, look at neighbors.
 2. Look for similar neighborhoods in sample texture.
 3. Randomly choose one
 4. Copy pixel from that neighborhood.
 5. Continue.

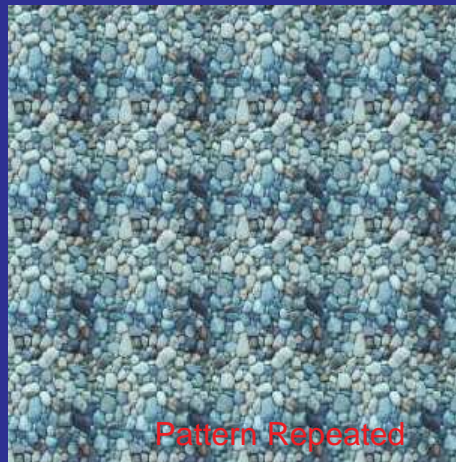
Efros and Leung



This is like copying, but not just repetition

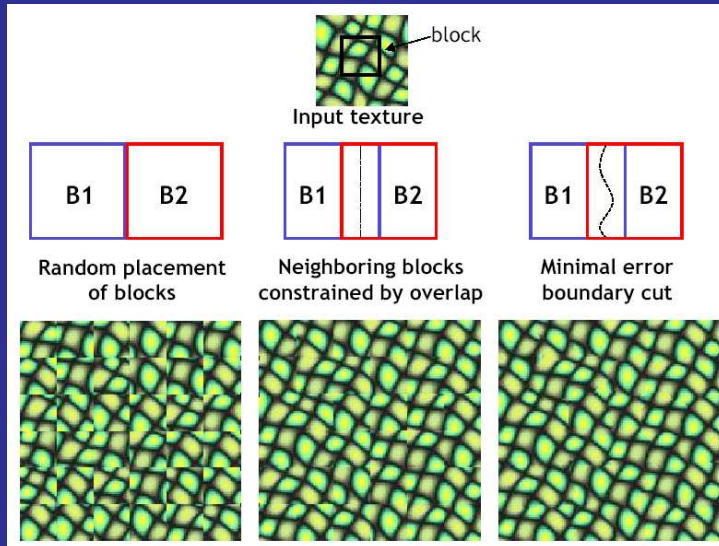


Photo



Pattern Repeated

With Blocks





Failures (Chernobyl Harvest)



Summary

- Texture is made of structure and randomness.
- Can get structure by providing knobs to user.
- Or from samples.