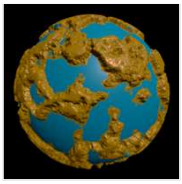
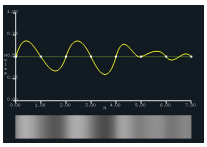
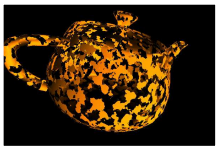
1

Noise !





Oregon State University
Mike Bailey
mjb@cs.oregonstate.edu



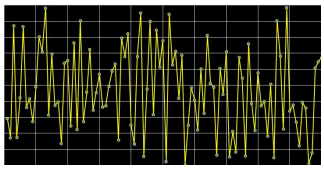
1

2

A Problem

One of the early criticisms of Computer Graphics is that it was too good, that is, everything was too perfect. Spheres were too perfectly round. And so on.

Computer Graphics needed a way to add imperfections. It seemed like random numbers could be used here. But *pure* random numbers are rather jarring:

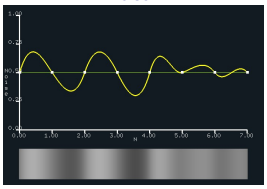


and that's not what we want. What we want is randomness, but *controlled randomness*. In Computer Graphics, this became known as **Noise**.

2

3

Noise:



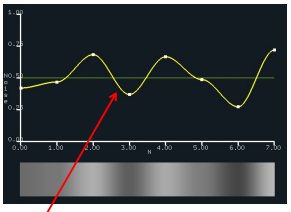
- Noise can be 1D, 2D, or 3D
- Noise output is a function of input value(s)
- Typically, those input values are where you are on the object, but they don't have to be
- Noise ranges from -1. to +1. or from 0. to 1.
- Noise might look random, but it really isn't
- Noise has ***Coherency*** (i.e., if you change the input value to the noise function a little, the output value will only change a little)
- Noise has ***Repeatability*** (i.e., if you supply the same inputs, the noise function will always give you back the same output)
- Noise is ***Continuous*** (i.e., it's smooth with no jarring jumps)

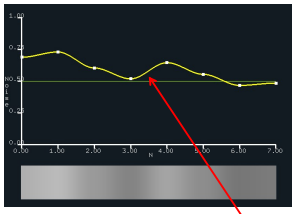
3

4

Positional Noise

Idea: Pick a random number at the whole-number input values and then fit a piecewise smooth curve through those points.





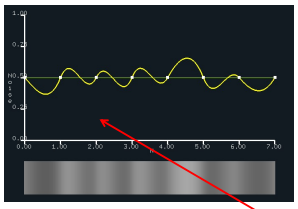
The problem is that, due to the uncertainty of random numbers, you might get a very good plus-or-minus distribution, or a not-so-good plus-or-minus distribution.

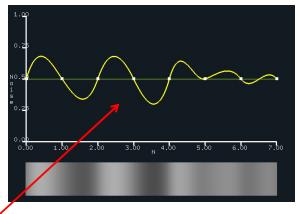
4

5

Gradient Noise

Idea: Place points at the mid-line at the whole-number input values and use random numbers to pick gradients (slopes) there, and then fit a piecewise smooth curve through those points with those slopes.



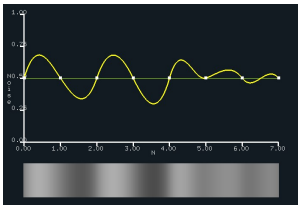


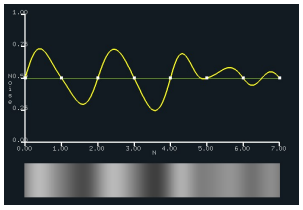
No matter what, you will get a very good plus-or-minus distribution.

5

6

Quintic (5th order) Interpolation Creates More Continuity Than Cubic





Cubic: C^1 continuity at the whole-number values

Quintic: C^2 continuity at the whole-number values

6

Coefficients for Cubic and Quintic Forms

7

$$N(t) = C_{N0}N_0 + C_{N1}N_1 + C_{G0}G_0 + C_{G1}G_1 + C_{C0}C_0 + C_{C1}C_1$$

Noise values — Gradients — Curvatures

Cubic

$$C_{N0} = 1 - 3t^2 + 2t^3$$
$$C_{N1} = 3t^2 - 2t^3 = 1 - C_{N0}$$
$$C_{G0} = t - 2t^2 + t^3$$
$$C_{G1} = -t^2 + t^3$$
$$C_{C0} = 0$$
$$C_{C1} = 0$$

Quintic

$$C_{N0} = 1 - 10t^3 + 15t^4 - 6t^5$$
$$C_{N1} = 10t^3 - 15t^4 + 6t^5 = 1 - C_{N0}$$
$$C_{G0} = t - 6t^3 + 8t^4 - 3t^5$$
$$C_{G1} = -4t^3 + 7t^4 - 3t^5$$
$$C_{C0} = \frac{1}{2}t^2 - \frac{3}{2}t^3 + \frac{3}{2}t^4 - \frac{1}{2}t^5$$
$$C_{C1} = \frac{1}{2}t^3 - t^4 + \frac{1}{2}t^5$$

Oregon State University Computer Graphics

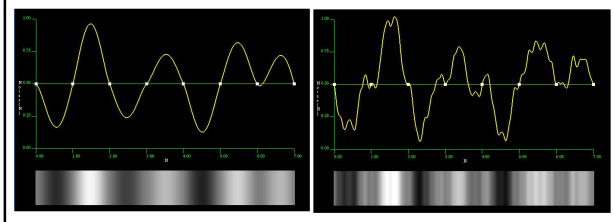
mjb – January 24, 2024

7

Noise Octaves

8

Idea: Add multiple noise waves, each one twice the frequency and half the amplitude of the previous one



1 Octave **4 Octaves**

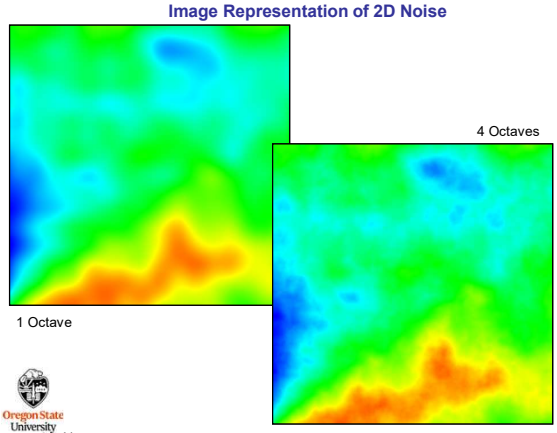
Oregon State University Computer Graphics

mjb – January 24, 2024

8

Image Representation of 2D Noise

9



4 Octaves

1 Octave

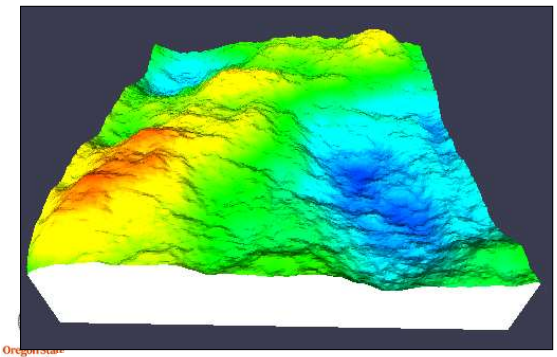
Oregon State University Computer Graphics

mjb – January 24, 2024

9

3D Surface Representation of 2D Noise

10



4 Octaves

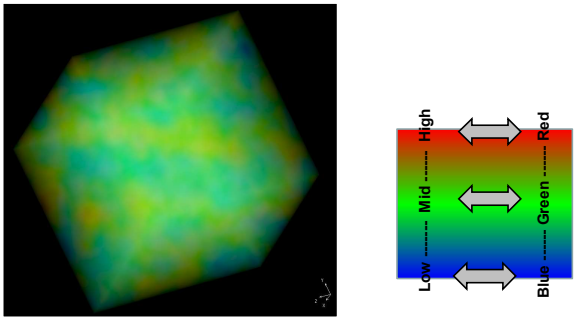
Oregon State University Computer Graphics

mjb – January 24, 2024

10

3D Volume Rendering of 3D Noise

11



1 Octave

Has continuity in X, Y, and Z

Oregon State University Computer Graphics

mjb – January 24, 2024

11

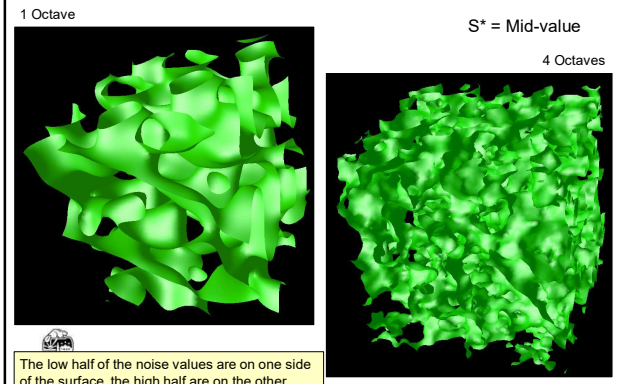
Volume Isosurfaces of 3D Noise

12

1 Octave

S* = Mid-value

4 Octaves



The low half of the noise values are on one side of the surface, the high half are on the other

Computer Graphics

mjb – January 24, 2024

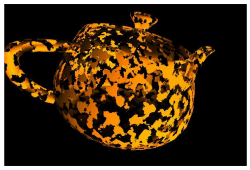
12

Examples

13

Color Blending for Marble

Color Blending for Clouds



Deciding when to Discard for Erosion

Oregon State University Computer Graphics

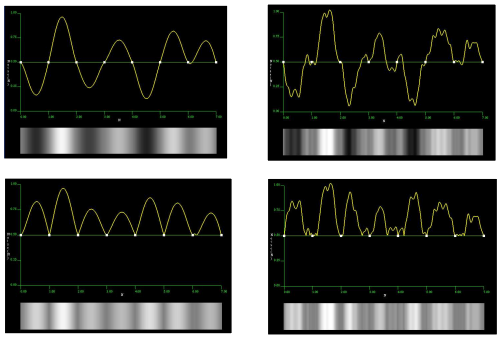
Turbulence

14

Idea: Take the absolute value of the noise about the centerline, giving the noise a "sharper" appearance and creating "creases". ***Warning: this is not the same use of the term as fluid "turbulence".***

Normal

Turbulent



1 Octave **4 Octaves**

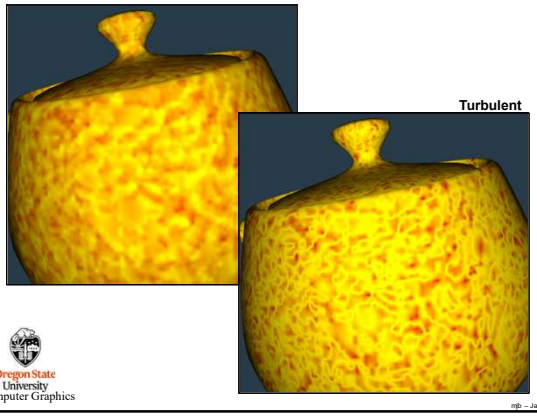
Oregon State University Computer Graphics

Turbulence Example

15

Normal

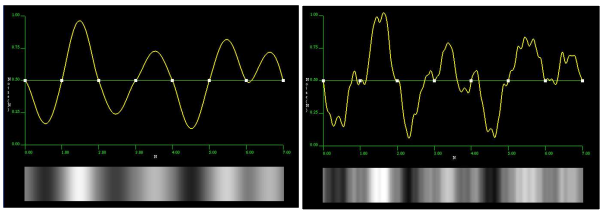
Turbulent



Oregon State University Computer Graphics

Remember Noise Octaves? What if we create a lookup table of noise octaves and hide it in a texture?

16



1 Octave **4 Octaves**

Oregon State University Computer Graphics

A Noise Texture in Glman

17

The *glman* tool automatically creates a 3D noise texture and places it into Texture Unit **3**. Your shaders can access it through the pre-created uniform variable called **Noise3**. You just declare it in your shader as:

```
uniform sampler3D Noise3;
. . .
vec4 nv  = texture( Noise3, uNoiseFreq * vMCposition );
```

The "noise vector" texture *nv* is a vec4 whose components have separate meanings. The .r component is the low frequency noise. The .g component is twice the frequency and half the amplitude of the .r component, and so on for the .b and .a components. Each component is centered around the middle value of .5

Component	Term	Term Range	Term Limits
0	nv.r	0.5 ± .5000	0.0000 → 1.0000
1	nv.g	0.5 ± .2500	0.2500 → 0.7500
2	nv.b	0.5 ± .1250	0.3750 → 0.6250
3	nv.a	0.5 ± .0625	0.4375→ 0.5625
	sum	2.0 ± ~ 1.0	~ 1.0 → 3.0
	sum – 1	1.0 ± ~ 1.0	~ 0.0 → 2.0
	(sum – 1) / 2	0.5 ± ~ 0.5	~ 0.0 → 1.0
	(sum – 2)	0.0 ± ~ 1.0	~ -1.0 → 1.0

Oregon State University Computer Graphics

A Noise Texture in Glman

18

So, if you would like to have a four-octave noise function that ranges from 0. to 1, then do this:

```
float n = nv.r + nv.g + nv.b + nv.a;        // range is 1. → 3.
n = ( n - 1. ) / 2.;                        // range is now 0. → 1.
```

If you would like to have a four-octave noise function that ranges from -1 to 1, then do this instead:

```
float  n = nv.r + nv.g + nv.b + nv.a;       // range is 1. → 3.
n = ( n - 2. );                             // range is now -1. → 1.
```

By default, the *glman* 3D noise texture has dimensions 64 × 64 × 64. You can change this by putting a command in your GLIB file of the form

Noise3D 128

to get dimension 128 × 128 × 128 , or choose whatever resolution you want (up to around 400 × 400 × 400).

Oregon State University Computer Graphics

A Noise Texture in Glman

19

The first time *glman* runs, it creates noise textures for you, it will take a few seconds. But *glman* then writes them to a local file, so that the next time this texture is needed, it is read from the file, which is a lot faster.

Getting a noise value from a 2D quantity (such as vST) works the same way as a 3D noise texture, except you get at it with:

```
uniform  sampler3D  Noise3;
...
vec4 nv = texture( Noise3, uNoiseFreq * vec3(vST,0.) );
float  n = nv.r + nv.g + nv.b + nv.a;          // range is 1. → 3.
n = ( n - 1. ) / 2.;                           // range is now 0. → 1.
```

Here we promote vST to be a vec3 so that it can use a 2D slice of the 3D noise texture.

Oregon State University
Computer Graphics

mjb – January 24, 2024

19

A 3D Noise Texture in Your C/C++ Program

20

The easiest way to read a noise texture into your C/C++ program is to get one of the noise textures from *glman* and know how to read it in. These pages will tell you how.

```
GLuint          Noise3;          // a global
GLSLProgram     Pattern;         // a global
. . .
// in InitGraphics:

glGenTextures(1, &Noise3);
int nums, numt, nump;
unsigned char * texture = ReadTexture3D( "noise3d.064.tex", &nums, &numt, &nump);
If( texture == NULL ) { … }

glBindTexture(GL_TEXTURE_3D, Noise3);
glTexParameterf(GL_TEXTURE_3D, GL_TEXTURE_WRAP_S, GL_REPEAT);
glTexParameterf(GL_TEXTURE_3D, GL_TEXTURE_WRAP_T, GL_REPEAT);
glTexParameteri(GL_TEXTURE_3D, GL_TEXTURE_WRAP_R, GL_REPEAT);
glTexParameterf(GL_TEXTURE_3D, GL_TEXTURE_MAG_FILTER, GL_LINEAR);
glTexParameterf(GL_TEXTURE_3D, GL_TEXTURE_MIN_FILTER, GL_LINEAR);
glTexImage3D(GL_TEXTURE_3D, 0, GL_RGBA, nums, numt, nump, 0, GL_RGBA,
                GL_UNSIGNED_BYTE, texture);

Pattern.Init( );
bool valid = Pattern.Create( "pattern.vert", "pattern.frag");
if (!valid)
. . .
```

3D

Dimensions

University
Computer Graphics

mjb – January 24, 2024

20

A 3D Noise Texture in Your C/C++ Program

21

```
unsigned char *
ReadTexture3D( char *filename, int *width, int *height, int *depth)
{
        FILE *fp = fopen(filename, "rb");
        if( fp == NULL )
                return NULL;

        int nums, numt, nump;
        fread(&nums, 4, 1, fp);
        fread(&numt,  4, 1, fp);
        fread(&nump, 4, 1, fp);
        fprintf( stderr, "Texture size = %d x %d x %d\n", nums, numt, nump );

        *width  = nums;
        *height = numt;
        *depth  = nump;

        unsigned char * texture = new unsigned char[ 4 * nums * numt * nump ];

        fread(texture, 4 * nums * numt * nump, 1, fp);
        fclose(fp);
        return texture;
}
```

Computer Graphics

mjb – January 24, 2024

21

A Noise Texture in Your C++ Program

22

```
void
Display( )
{
        . . .

        glActiveTexture( GL_TEXTURE3 );          // set to use texture unit 3
        glBindTexture(GL_TEXTURE_3D, Noise3 );

        Pattern.Use();
        Pattern.SetUniformVariable( "Noise3", 3);
        . . .
        << Draw something >>
        . . .
        Pattern.UnUse;
```

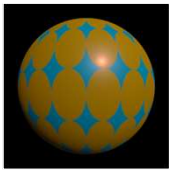
Oregon State University
Computer Graphics

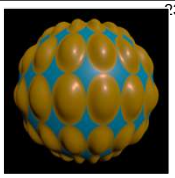
mjb – January 24, 2024

22

How to Use Noise

23



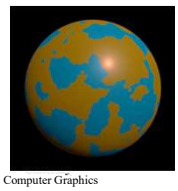


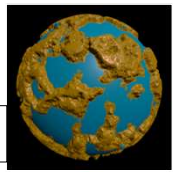
Have an equation that relates some input value (x,y,z or s,t) to output values (color, height)

Have actual input values of where we are right now

Add Noise to the actual input values to produce new "fake" input values

Use those new "fake" input values in the original equation





Idea: The graphics system will display "here", using display parameters as if you were "over there".

Computer Graphics

mjb – January 24, 2024

23

How to Index Noise from 3D Model Coordinates

24

In the vertex shader:

```
out vec3 vMCposition;
. . .
vMCposition = gl_Vertex.xyz;
```

How much to amplify the noise effect

How much to increase the sampling rate

Model coordinates where this fragment is

In the fragment shader:

```
uniform sampler3D Noise3;
uniform float uNoiseFreq, uNoiseAmp;
in vec3 vMCposition;
. . .
vec4 nv  = texture( Noise3, uNoiseFreq * vMCposition );
float  n = nv.r + nv.g + nv.b + nv.a;        // range is 1. -> 3.
n = n - 2.;                                  // range is now -1. -> 1.
n  *=  uNoiseAmp;
```

Now add the noise value, **n**, to the actual location. Compute the effect at that "fake" location but apply it at the actual location.

We typically do this in Model coordinates so that the pattern sticks to the object.

mjb – January 24, 2024

24

How to Index Noise from 2D Texture Coordinates

25

In the vertex shader:

```
out vec2 vST;
. . .
vST = gl_MultiTexCoord0.st;
```

How much to amplify the noise effect

How much to increase the sampling rate

Texture coordinates where this fragment is

In the fragment shader:

```
uniform sampler3D Noise3;
uniform float uNoiseFreq, uNoiseAmp;
in vec2 vST;
. . .
vec4 nv  = texture( Noise3, uNoiseFreq * vec3(vST,0.) );
float  n = nv.r + nv.g + nv.b + nv.a;      // range is 1. -> 3.
n = n - 2.;                                // range is now -1. -> 1.
n  *=  uNoiseAmp;
```

Now add the noise value, **n**, to the actual location. Compute the effect at that "fake" location but apply it at the actual location.

We typically do this in Model coordinates so that the pattern sticks to the object.

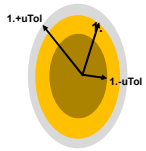
Oregon State University Computer Graphics

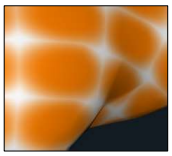
mjb – January 24, 2024

25

Elliptical Dots with Tolerance

26





$$1 - uTol \le \left(\frac{s-s_c}{A_r}\right)^2 + \left(\frac{t-t_c}{B_r}\right)^2 \le 1 + uTol$$

$$float\ d = \left(\frac{s-s_c}{A_r}\right)^2 + \left(\frac{t-t_c}{B_r}\right)^2$$

```
float t = smoothstep( 1.-uTol,  1.+uTol, d );
vec3 color = mix( ORANGE, WHITE, t );
```

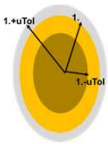
Oregon State University Computer Graphics

mjb – January 24, 2024

26

Elliptical Dots with Tolerance and Noise

27



Have an equation that relates some input value (x,y,z or s,t) to output values (color, height)

Have actual input values of where we are right now

Add Noise to the actual input values to produce new "fake" input values

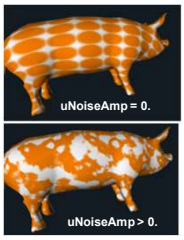
Use those new "fake" input values in the original equation

```
float n = nv.r + nv.g + nv.b + nv.a;   // 1. -> 3.
n = n - 2.;                            // -1. -> 1.
n  *=  uNoiseAmp;

. . .

float ds = st.s - sc;                  // wrt ellipse center
float dt =  st.t  - tc;                // wrt ellipse center
float oldDist   = sqrt( ds*ds + dt*dt );
float newDist = oldDist + n;
float scale = newDist / oldDist;       // this could be < 1., = 1., or > 1.

ds *= scale;                           // scale by noise factor
ds /= Ar;                              // ellipse equation
dt *= scale;                           // scale by noise factor
dt /= Br;                              // ellipse equation
float d = ds*ds + dt*dt;
float t = smoothstep( 1.-uTol, 1.+uTol, d );
vec3 theColor = mix( ORANGE, WHITE, t );
. . .
```

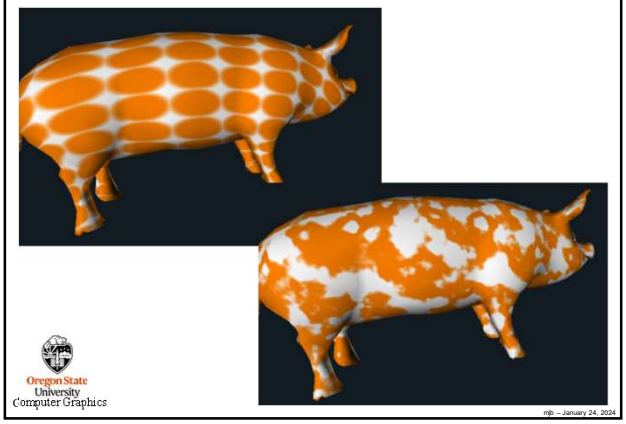


mjb – January 24, 2024

27

Elliptical Dots with Tolerance and Noise

28



Oregon State University Computer Graphics

mjb – January 24, 2024

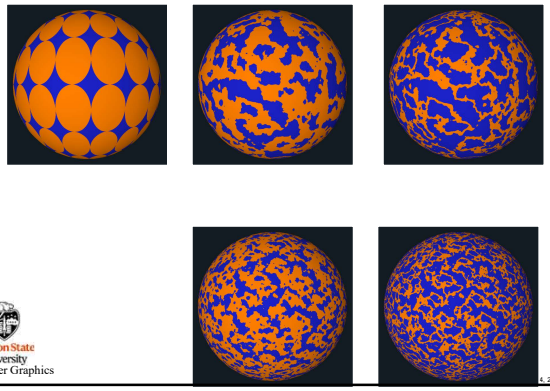
28

29

N = NoiseAmp * noise(NoiseFreq * PP);

Noise Amplitude →

Noise Frequency ↓

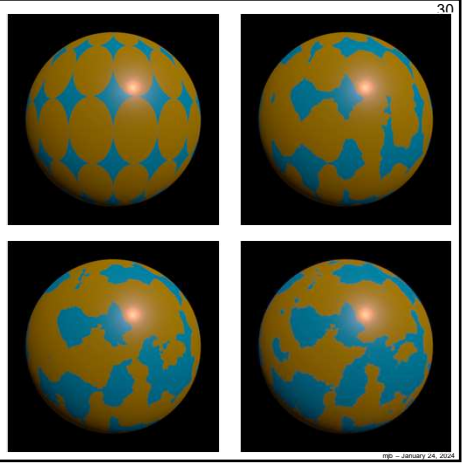


Oregon State University Computer Graphics

29

30

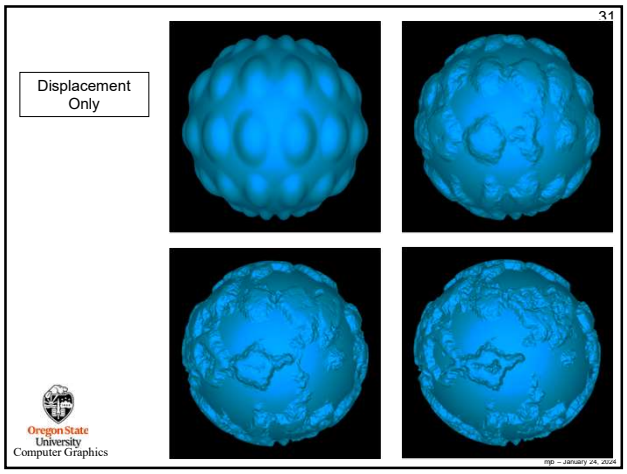
Color Only



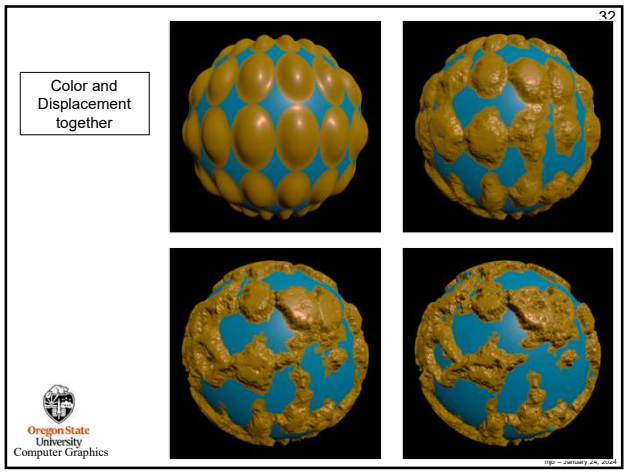
Oregon State University Computer Graphics

mjb – January 24, 2024

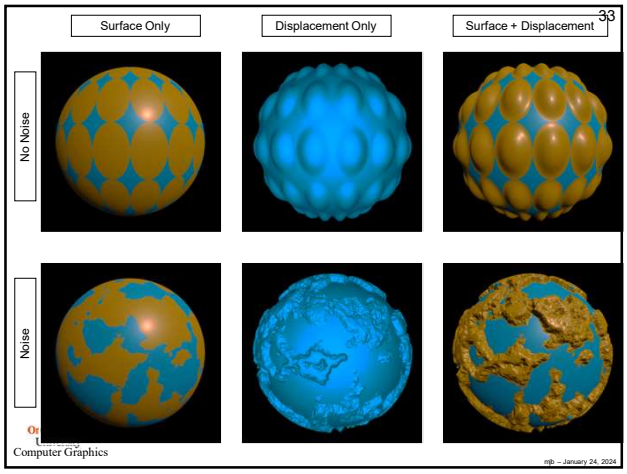
30


31
Displacement Only
Oregon State University
Computer Graphics
mjb – January 24, 2024

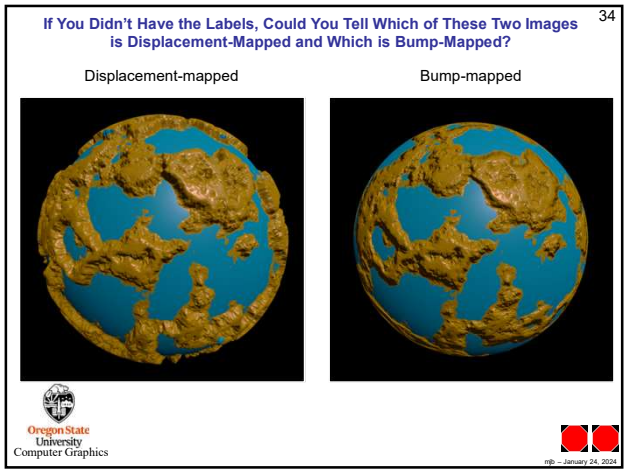
31


32
Color and Displacement together
Oregon State University
Computer Graphics
mjb – January 24, 2024

32


33
Surface Only
Displacement Only
Surface + Displacement
No Noise
Noise
Computer Graphics
mjb – January 24, 2024

33


34
If You Didn't Have the Labels, Could You Tell Which of These Two Images is Displacement-Mapped and Which is Bump-Mapped?
Displacement-mapped
Bump-mapped
Oregon State University
Computer Graphics
mjb – January 24, 2024

34