

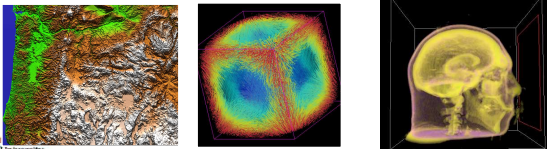


1

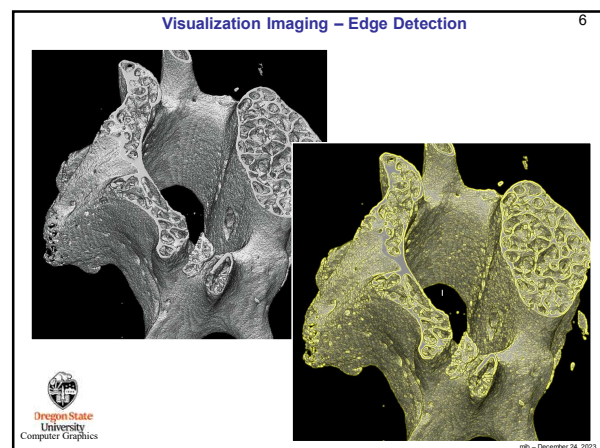
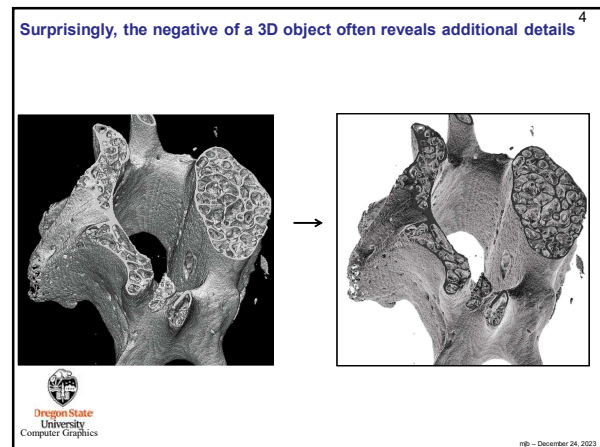
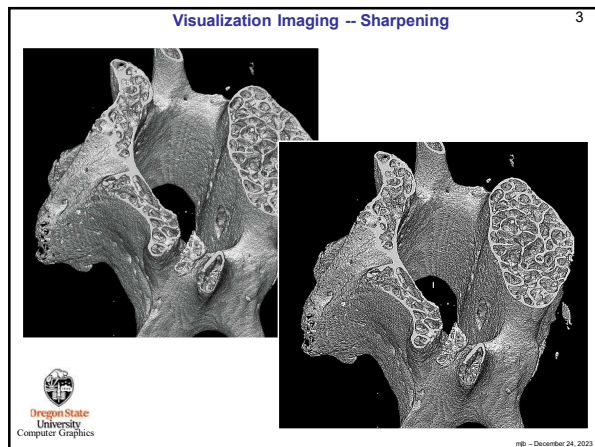
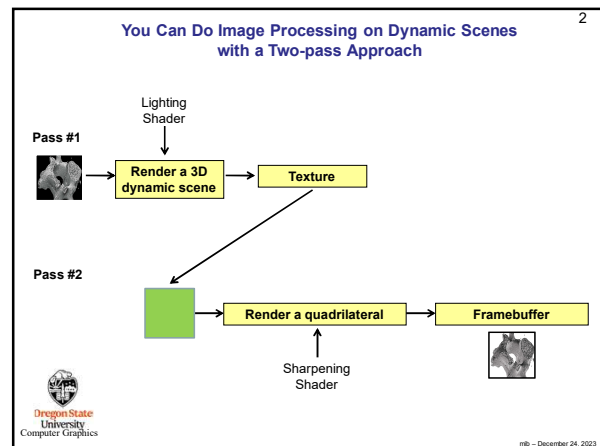
Using Shaders to Enhance Scientific Visualizations

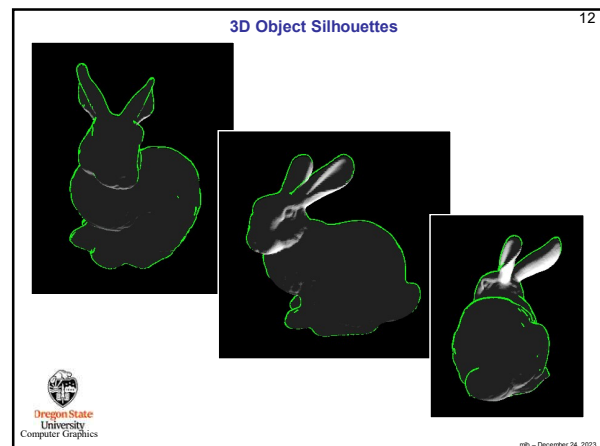
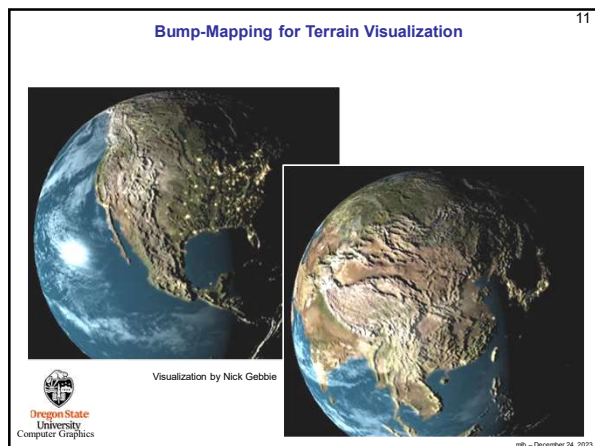
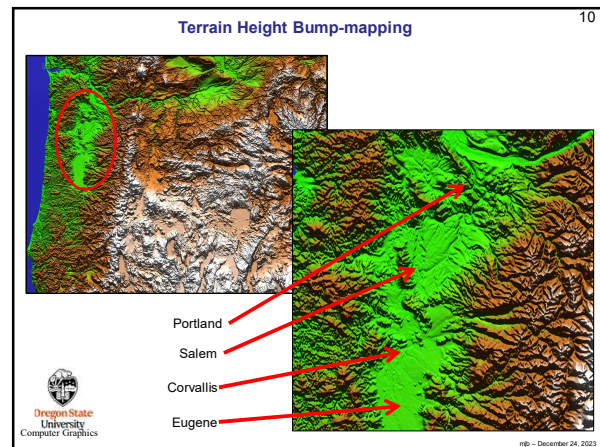
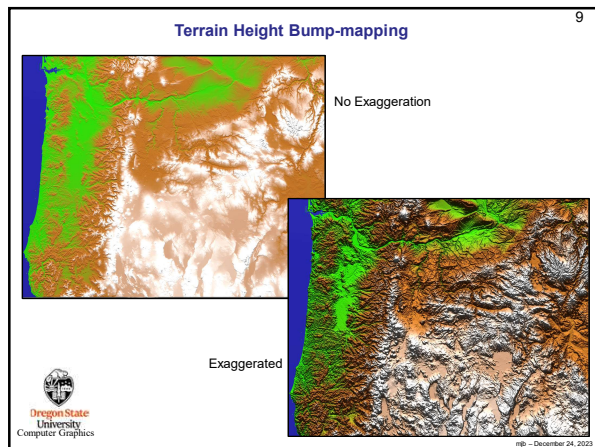
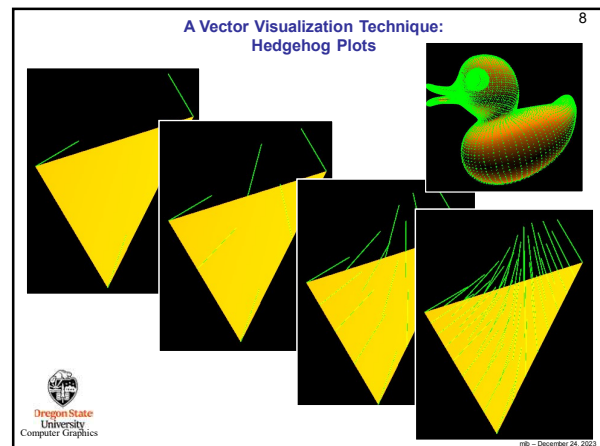
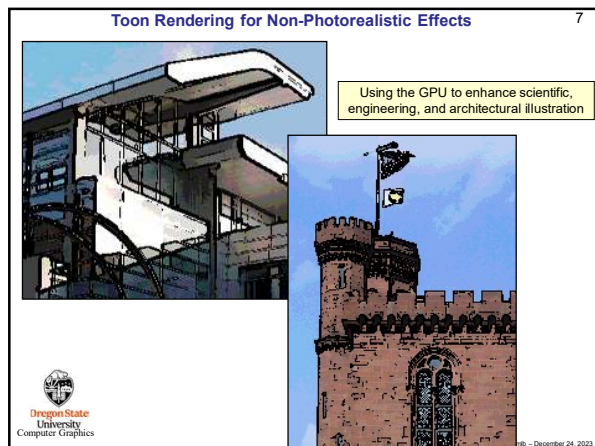

Oregon State University
Mike Bailey
mjb@cs.oregonstate.edu


This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License



University Computer Graphics
vis.gdb
mjb - December 24, 2023





Visualization -- Polar Hyperbolic Space

Use the GPU to perform nonlinear vertex transformations

$$\Theta' = \Theta$$

$$R' = \frac{R}{R + K}$$

Oregon State University
Computer Graphics

mjb - December 24, 2023

Dome Projection for Immersive Visualization

Use the GPU to perform nonlinear vertex transformations

Oregon State University
Computer Graphics

mjb - December 24, 2023

Image Manipulation Example -- Where is it Likely to Snow?

Visible Infrared Water vapor

```
if( have_clouds && have_a_low_temperature && have_water_vapor )
    color = green;
else
    color = from visible map
```

Oregon State University
Computer Graphics

mjb - December 24, 2023

Writing 3D Point Cloud Data into a Floating-Point Texture for glman

```
fwrite( &nums, 4, 1, fp );
fwrite( &numt, 4, 1, fp );
fwrite( &nump, 4, 1, fp );

for( int p = 0; p < num; p++ )
{
    for( int t = 0; t < num; t++ )
    {
        for( int s = 0; s < num; s++ )
        {
            float red, green, blue, alpha;
            << assign red, green, blue, alpha >>
            fwrite( &red, 4, 1, fp );
            fwrite( &green, 4, 1, fp );
            fwrite( &blue, 4, 1, fp );
            fwrite( &alpha, 4, 1, fp );
        }
    }
}
```

Oregon State University
Computer Graphics

mjb - December 24, 2023

Point Cloud from a 3D Texture Dataset

Full data Low values culled

Oregon State University
Computer Graphics

mjb - December 24, 2023

Where to Place the Geometry?

I personally like thinking of the data as living in a cube that ranges from -1. to 1. in X, Y, and Z. It is straightforward to position geometry in this space and easy to view and transform it. This means that any 3D object in that space, not just a point cloud, can map itself to the 3D texture data space.

So, because the *s* texture coordinate goes from 0. to 1., then the linear mapping from the physical *x* coordinate to the texture *s* coordinate is:

$$-1. \leq x \leq 1. \quad \rightarrow \quad s = \frac{x + 1.}{2.} \quad \rightarrow \quad 0. \leq s \leq 1.$$

The same mapping applies to *y* and *z* to create the *t* and *p* texture coordinates.

In GLSL, this conversion can be done in one line of code using the vec3:

```
vec3 xyz = ???
...
vec3 stp = ( xyz + 1. ) / 2.;
```

Or You can also go the other way: `vec3 xyz = -1. + (2. * stp);`

Oregon State University
Computer Graphics

mjb - December 24, 2023


The Vertex Shader

```

out vec3 vMC;

void
main( )
{
    vMC = gl_Vertex.xyz;
    gl_Position = gl_ModelViewProjectionMatrix * gl_Vertex;
}

```

 mjb - December 24, 2023

The Fragment Shader

```

uniform float uMin, uMax;
uniform sampler3D uTexUnit;
in vec3 vMC;
const float SMIN = 0.;
const float SMAX = 120.;

void
main( )
{
    vec3 stp = ( vMC + 1. ) / 2.; // maps [-1.,1.] to [0.,1.]

    if( any( lessThan( stp, vec3(0.,0.,0.) ) ) )
        discard;


    if( any( greaterThan( stp, vec3(1.,1.,1.) ) ) )
        discard;

    float scalar = texture( uTexUnit, stp ).r; // data is hiding in the red component
    if( scalar < uMin || scalar > uMax )
        discard;

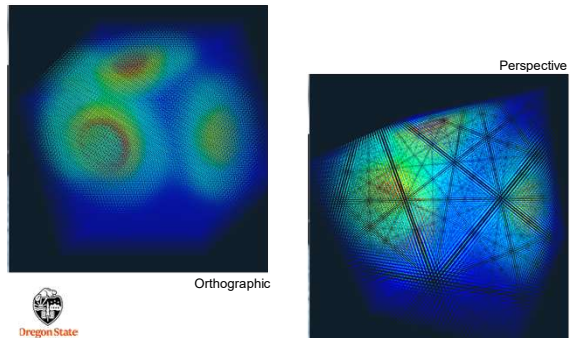
    float t = ( scalar - SMIN ) / ( SMAX - SMIN );
    vec3 rgb = Rainbow( t );
    gl_FragColor = vec4( rgb, 1. );
}


```

SIMD functions to help GLSL if-tests

 mjb - December 24, 2023

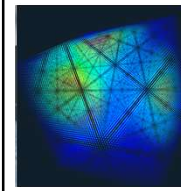
A Problem with Uniform Pointclouds: Row-of-Corn and Moire Patterns



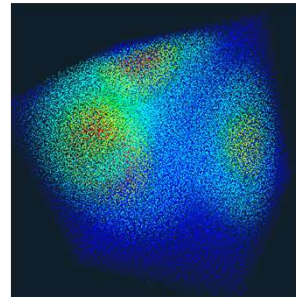
 mjb - December 24, 2023

Uniform Points vs. Jittered Points


"Pointcloud"



"Jittercloud"



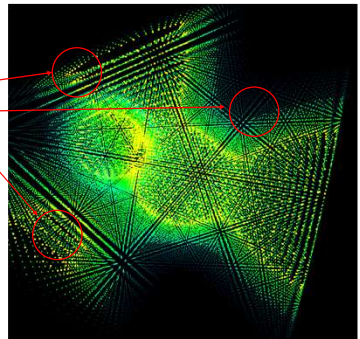
"Jittering" moves each point a small random amount in $\pm x$, $\pm y$, and $\pm z$. Because our data value lookup comes from (s,t,p) which comes from (x,y,z), the lookup will be correct at the jittered points.


 mjb - December 24, 2023

Enhanced Point Clouds

The shaders can potentially change:

- Color
- Alpha
- Pointsize



 mjb - December 24, 2023

Color Cutting Planes

Now, change the Point Cloud geometry to a quadrilateral geometry. If we keep the coordinate range from -1. to 1., then the same shader code will work, **except that we now want to base the color assignment on Eye Coordinates instead of Model Coordinates**:

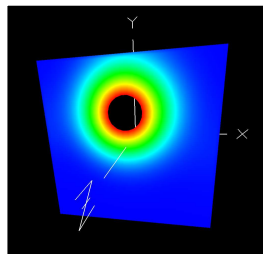
```

in vec3 vEC;


void
main( )
{
    vec3 stp = ( vEC + 1. ) / 2.;
    // maps [-1.,1.] to [0.,1.]
    ...
}

```

Eye (transformed) coordinates are being used here because the cutting plane is moving through the data.



Note that the plane can be oriented at any angle because the s-t-p data lookup comes from the transformed x-y-z coordinates of the cutting plane

 mjb - December 24, 2023

Color Cutting Planes

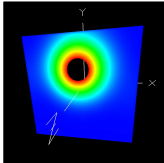
The cutting plane is actually just being used as a **fragment-generator**. Each fragment is then being asked "what data value lives at the same place you live"?

```

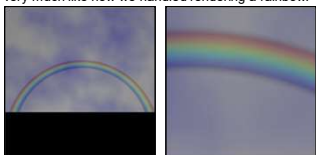
in vec3 vEC;

void main( )
{
    vec3 stp = ( vEC + 1. ) / 2.;
    // maps [-1.,1.] to [0.,1.]
    ...

```



This is very much like how we handled rendering a rainbow.



Oregon State University Computer Graphics

mjb - December 24, 2023

Gapped-Contour Cutting Planes

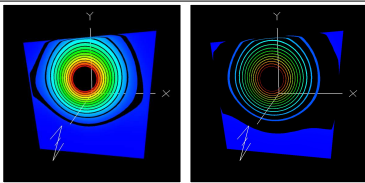
Let's say that we want "contour gaps" at each 10 degrees of temperature. Then the main change to the shader will be that we need to find how close each fragment's interpolated scalar data value is to an even multiple of 10. To do this, we add this discretization code to the fragment shader:

```

float scalar10 = float( 10*int( (scalar+5.)/10. ) );
if( abs( scalar - scalar10 ) < uTol )
    discard;

```

Notice that this uses a uniform variable called **uTol**, which is read from a slider and has a range of 0. to 5. **uTol** is used to determine how close to an even multiple of 10 degrees we will accept, and thus how thick we want the contour gaps to be.



Oregon State University Computer Graphics

mjb - December 24, 2023

Contour Cutting Planes are Also Color Cutting Planes

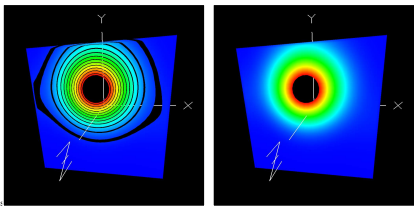
Note that when $uTol=5.$, the $uTol$ if-statement

```

float scalar10 = float( 10*int( (scalar+5.)/10. ) );
if( abs( scalar - scalar10 ) < uTol )
    discard;

```

always fails, and we end up with the same display as we had with the interpolated colors. Thus, we wouldn't actually need a separate *color cutting plane* shader at all. Shaders that can do double duty are always appreciated!

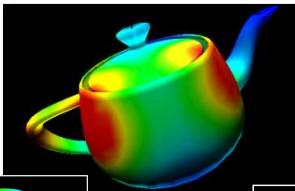


Oregon State University Computer Graphics

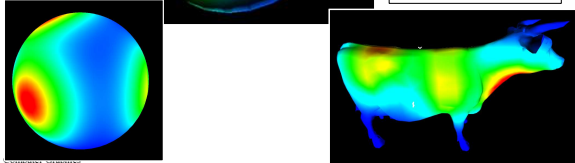
mjb - December 24, 2023

3D Data Probe - Mapping the Data to Arbitrary Geometry

The cutting plane is actually being used as a fragment-generator. Each fragment is then being asked "what data value lives at the same place you live"?



Some shapes make better probes than others do... ☺



Oregon State University Computer Graphics

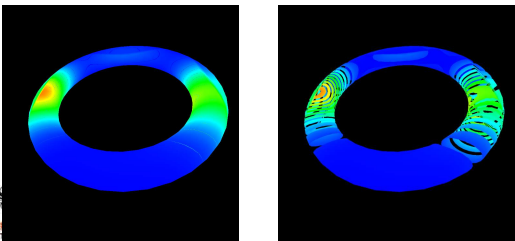
mjb - December 24, 2023

An Observation

Note that Point Clouds, Jitter Clouds, Colored Cutting Planes, Contour Cutting Planes, and 3D Data Probes are *really all the same technique*!

They just vary in what type of geometry the data is mapped to. They use the same shader code, possibly with a switch between model and eye coordinates.

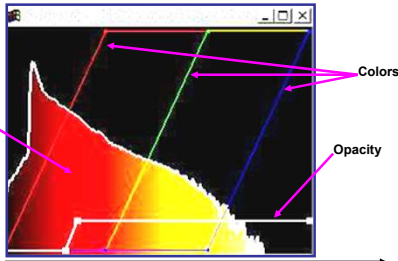
How about something less obvious like a torus?



Oregon State University Computer Graphics

mjb - December 24, 2023

Visualization Transfer Function - Relating Display Attributes to the Scalar Value



Frequency Histogram

Colors

Opacity

Scalar Value

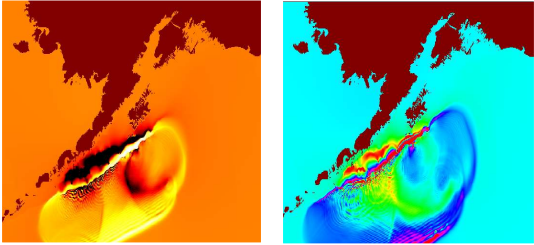
OSU vs Transfer Function Sculpting Window

Oregon State University Computer Graphics

mjb - December 24, 2023

31

Visualization -- Don't Send Colored Data to the GPU, Send the Raw Data and a Separate Transfer Function to the Fragment Shader



Use the GPU to turn the data into colored graphics on-the-fly


Oregon State University
Computer Graphics

Visualizations by Chris Janik
rjb - December 24, 2023


32

A Visualization Scenario

A thermal analysis reveals that a bar has a temperature of 0° at one end and 100° at the other end:



You want to color it with a rainbow scale as follows:



You also want to use smooth shading, so that you can render the bar as a single quadrilateral.

Should you assign colors first then interpolate, or interpolate first then assign colors?
Will it matter? If so, how?


Oregon State University
Computer Graphics

rjb - December 24, 2023

33


A Visualization Scenario

Assign colors from temperatures, then interpolate:



WRONG !

Interpolate temperatures first, then assign colors:



RIGHT !

Conclusion: let the rasterizer interpolate your scalar values and let your fragment shader assign colors and alphas to those values

Oregon State University
Computer Graphics

rjb - December 24, 2023

34

Point Clouds – Three Ways to Assign the Scalar Function

Without shaders:
Assigning colors first – problems with interpolation

```
glBegin( GL_POINTS );
< convert s0 to r0,g0,b0, a0 >
glColor4f( r0, g0, b0, a0 );
glPointSize( p0 );
glVertex3f( x0, y0, z0 );
...
glEnd( );
```

With shaders*
Put the data in attribute variables

```
Pattern.Use( );
glBegin( GL_POINTS );
Pattern.SetAttributeVariable( "Temperature", s0 );
glVertex3f( x0, y0, z0 );
...
glEnd( );
```

Oregon State University
Computer Graphics

rjb - December 24, 2023

35

Point Clouds – A Third Way – I really like this one

With shaders:
Hiding the scalar value in the w component

```
Pattern.Use( );
glBegin( GL_POINTS );
glVertex4f( x0, y0, z0, s0 );
...
glEnd( );
```

With shaders:
The hidden scalar value in the w component must be extracted and replaced with 1.0 in the vertex shader

```
out float vScalar;

void
main( )
{
    vScalar = gl_Vertex.w;
    gl_Position = gl_ModelViewProjectionMatrix * vec4(gl_Vertex.xyz, 1.);
}
```

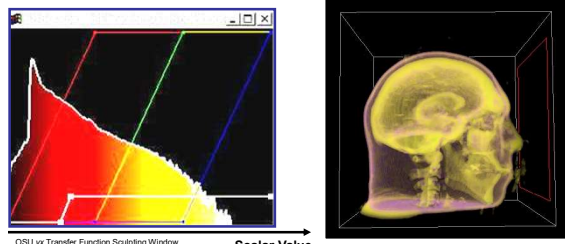
Oregon State University
Computer Graphics

rjb - December 24, 2023

36

Volume Rendering – a different way to think of visualizing 3D Scalar Data

Each voxel has a color and opacity depending on its scalar value



OSU vs Transfer Function Sculpting Window

Scalar Value

Visualization by Ankit Khare

Oregon State University
Computer Graphics

rjb - December 24, 2023

Volume Rendering – Compositing via Ray Casting

Thinking about it back-to-front:

$$color_{i_2} = \alpha_2 color_2 + (1 - \alpha_2) black,$$

$$color_{i_1} = \alpha_1 color_1 + (1 - \alpha_1) color_{i_2},$$

$$color^* = \alpha_0 color_0 + (1 - \alpha_0) color_{i_1}.$$

Gives the front-to-back equation:

$$color^* = \alpha_0 color_0 + (1 - \alpha_0) \alpha_1 color_1 + (1 - \alpha_0)(1 - \alpha_1) \alpha_2 color_2 + (1 - \alpha_0)(1 - \alpha_1)(1 - \alpha_2) black.$$

Oregon State University
Computer Graphics

mpb - December 24, 2023

Volume Rendering – Compositing via Ray Casting

uMin = minimum scalar value to display
uMax = maximum scalar value to display
uAmax = alpha value to use if this voxel is to be seen

```

float astar = 1.;
vec3 cstar = vec3( 0., 0., 0. );
for( int i = 0; i < uNumSteps; i++, STP += uDirSTP )
{
    if( any( lessThan( STP, vec3(0.,0.,0.) ) ) )
        continue;

    if( any( greaterThan( STP, vec3(1.,1.,1.) ) ) )
        continue;

    float scalar = texture3D( uTexUnit, STP ).r;
    if( scalar < uMin )
        continue;
    if( scalar > uMax )
        continue.;
    float alpha = uAmax;

    float t = ( scalar - SMIN ) / ( SMAX - SMIN );
    vec3 rgb = Rainbow( t );

    cstar += astar * alpha * rgb;
    astar *= ( 1. - alpha );

    // break out if the rest of the tracing won't matter:
    if( astar == 0. )
        break;
}
gl_FragColor = vec4( cstar, 1. );

```

Oregon State University
Computer Graphics

mpb - December 24, 2023

Volume Rendering – Compositing via Ray Casting

Oregon State University
Computer Graphics

mpb - December 24, 2023

Volume Filtering – Median Filter

Visualizations by Ankit Khare

Oregon State University
Computer Graphics

mpb - December 24, 2023

Volume Filtering – High Pass Filter Followed by Median Filter

Visualizations by Ankit Khare

Oregon State University
Computer Graphics

mpb - December 24, 2023

Volume Visualization for OSU'S College of Vet Medicine

Visualization by Chris Schultz

University
Computer Graphics

mpb - December 24, 2023

Vector Visualization: 2D Line Integral Convolution

43

Use a vector field equation, or "hide" the velocity field in another texture image: $(v_x, v_y, v_z) \equiv (r, g, b)$

At each fragment:

1. Find the flow field velocity vector there
2. Follow that vector in both directions
3. Blend in the colors at the other fragments along that vector

Circular Flow Field

Oregon State University
Computer Graphics

mjb - December 24, 2023

Vector Visualization: 2D Line Integral Convolution

44

```

lic2d.frag, 1
uniform int      uLength;
uniform sampler2D uImageUnit;
uniform sampler2D uFlowUnit;
uniform float    uTime;
in vec2          vST;

void
main( )
{
    ivec2 res = textureSize( uImageUnit, 0 );

    // flow field direction:
    vec2 st = vST;
    vec2 v = texture( uFlowUnit, st ).xy;
    v *= 1./vec2(res);

    st = vST;
    vec3 color = texture( uImageUnit, st ).rgb;
    int count = 1;

```

Oregon State University
Computer Graphics

mjb - December 24, 2023

Vector Visualization: 2D Line Integral Convolution

45

```

lic2d.frag, 11

st = vST;
for( int i = 0; i < uLength; i++ )
{
    st += uTime*v;
    vec3 new = texture( uImageUnit, st ).rgb;
    color += new;
    count++;
}

st = vST;
for( int i = 0; i < uLength; i++ )
{
    st -= uTime*v;
    vec3 new = texture( uImageUnit, st ).rgb;
    color += new;
    count++;
}

color /= float(count);
gl_FragColor = vec4( color, 1. );

```

Oregon State University
Computer Graphics

mjb - December 24, 2023

Vector Visualization: 2D Line Integral Convolution

46

Flow in a circle

Flow around a corner

Oregon State University
Computer Graphics

mjb - December 24, 2023

Vector Visualization: a Cool 2D Line Integral Convolution Example

47

<http://hint.fm/wind/>

Oregon State University
Computer Graphics

mjb - December 24, 2023

Vector Visualization: 3D Line Integral Convolution

48

Visualizations by Vasu Lakshmanan

Oregon State University
Computer Graphics

mjb - December 24, 2023