Using ChromaDepth to obtain Inexpensive Single-image Stereovision for Scientific Visualization

What is ChromaDepth ?

ChromaDepth™ was invented by Richard Steenblik as a way to amplify the common chromostereoscopy phenomenon into a useful display tool. ChromaDepth consists of two pieces: a simple pair of glasses and a display methodology.

The glasses, shown here, contain very thin diffractive optics that have the efficiency of refractive optics. While being very thin and inexpensive, they behave like thicker glass prisms. The optics are designed so that red light is bent more than green and green more than blue. The lenses are oriented sideways, so the overall bending effect looks like parts of the scene have been shifted horizontally inwards (ie, towards the center of your nose). The red hues are shifted more than the greens and the greens are shifted more than the blues. Thus, red elements in the 3D scene appear to converge closest to the viewer and the blue elements appear to converge the farthest away.

The corresponding display methodology is then quite simple: color code the scene in linear a rainbow spectrum based on depth so that those elements that are close to the eye are displayed as red and those farthest away are displayed as blue.

Creating ChromaDepth Scenes with OpenGL

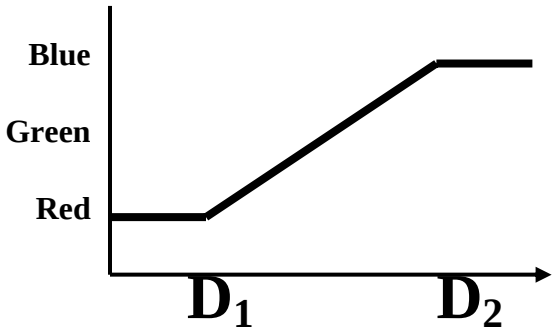
In order to create a ChromaDepth scene in OpenGL, a 1D texture representing a color ramp from red to green to blue needs to be created. This can be done in a number of ways. We did ours by interpolating in hue-saturation-value (HSV) space.

The next step is to specify how to compute the texture coordinate based on scene depth. The OpenGL Programming Guide provides a good description of the OpenGL *automatic texture-coordinate generation* capability. This can be used to generate contours on a 3D model in world or eye space, or can be used for dramatic environment mapping effects. It can also be used to automatically generate coloring for ChromaDepth.

The OpenGL *texgen* capability requires the program to supply four coefficients, A, B, C, and D for the texture coordinate equation:

$$s = A*x + B*y + C*z + D$$

Because we want to apply the coloring based on depth with respect to the eye coordinate system, we must choose the coefficients based on how close to the eye we want objects to become solid red and how far from the eye we want them to become solid blue. The situation looks like this:



Ramp from Red to Blue as a Function of Distance in Front of the Eye

The texture coordinate, *s*, needs to be 0. for objects that are D_1 units in front of the eye and 1. for objects that are at D_2 units. The equation to do this is:

$$s = \frac{-Z}{D_2 - D_1} - \frac{D_1}{D_2 - D_1}$$

giving us the coefficients for the *texgen* texture coordinate equation:

```
A = 0.
B = 0.
C = -1. /( D2- D1 )
D = -D1 /( D2- D1 )
```

The choices for D_1 and D_2 are purely up to the viewer's taste. It is tempting to make them the same as the near and far variables that are specified for the viewing volume. This works well, however the full dynamic ChromaDepth range will only be achieved when objects fill the entire depth of the viewing volume. In other words, this requires the programmer to place the near and far clipping planes very tightly around the scene.

It works even better to set D_1 and D_2 to be somewhere between the near and far clipping planes. Typically we try to fit a spherical bounding volume around the center of the scene, and set D_1 and D_2 to the limits of the sphere.

But, the final choice for D_1 and D_2 rests with the nature of the scene and the viewing tastes of the programmer. In practice it is nice to make D_1 and D_2 setable from sliders, although this is more work.

Sample OpenGL code to set and use these texture generation parameters is:

```
#define D1           5.0
#define D2          15.0

float  TexGenParams[] =
{
       0.,
       0.,
       -1./(D2-D1),
       -D1/(D2-D1)
};
```

• • •

```
glMatrixMode( GL_MODELVIEW );
glLoadIdentity();

glTexGeni( GL_S, GL_TEXTURE_GEN_MODE, GL_EYE_LINEAR );

glTexGenfv( GL_S, GL_EYE_PLANE, TexGenParams );

glEnable( GL_TEXTURE_GEN_S );
glEnable( GL_TEXTURE_1D );


gluLookAt( 0., 0., 10.,  0., 0., 0., 0., 1., 0. );


       << Object Transformations >>

glCallList( ObjectList );
```

The `glTexGeni()` call sets the texture generation mode to be in the eye coordinate space. The `glTexGenfv()` call supplies the texture coordinate equation coefficients.

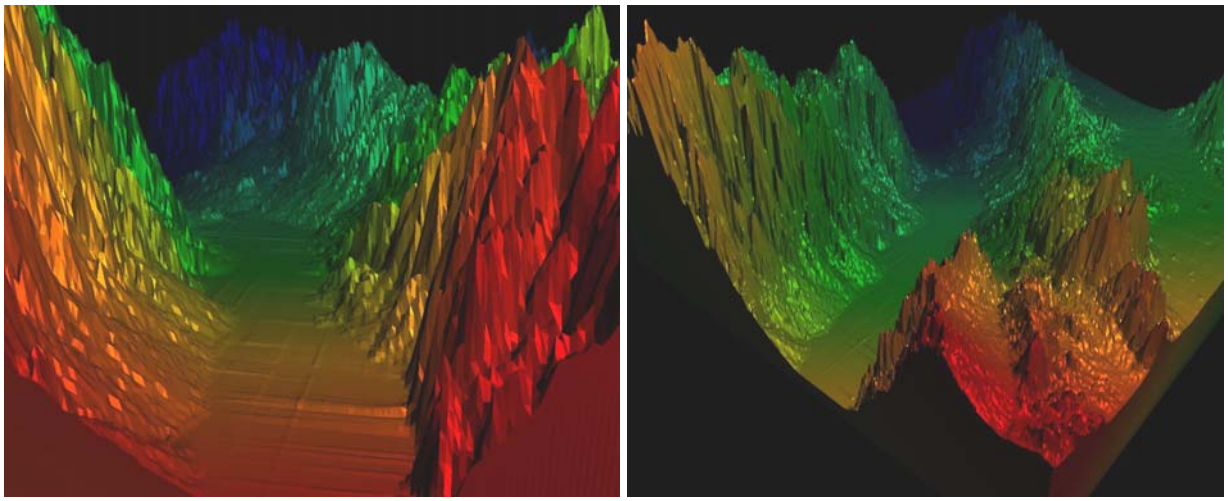
The scene and its objects can be dynamically transformed, but all of those transformations must come *after* the calls to `glTexGeni()` and `glTexGenfv()`. This is because the texture coordinate equation parameters apply to the model-view coordinate system as it exists at the moment `glTexGenfv()` is called. Because we want the coloring to vary strictly by depth in the eye-viewing direction, the model-view coordinate system must be untransformed when the texture coordinate equation is specified.

Web Page

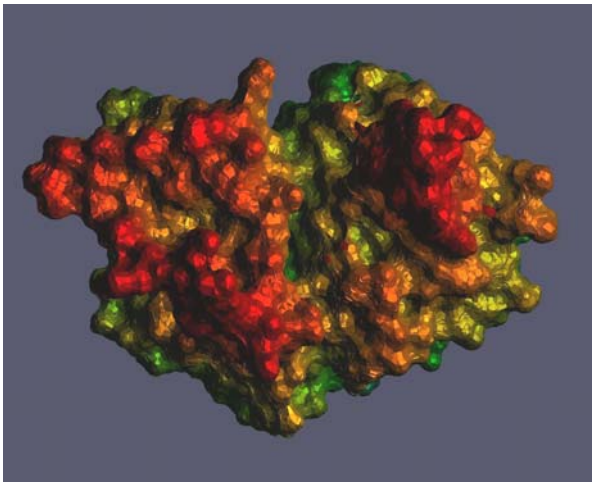
Links to a gallery of scientific visualization ChromaDepth images and to more information about the ChromaDepth process can be found at:

`http://web.engr.oregonstate.edu/~mjb/chromadepth`

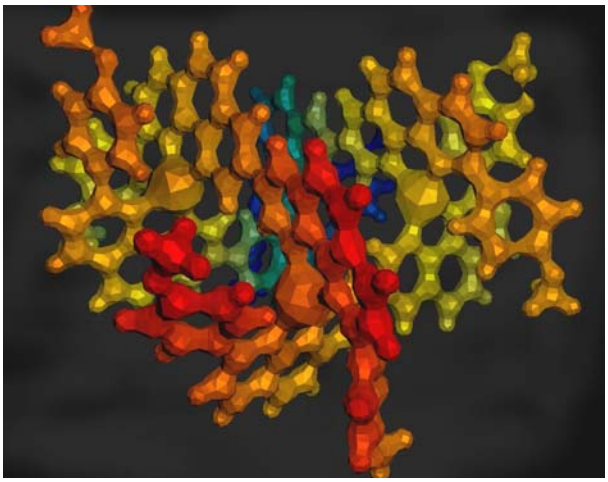
Examples



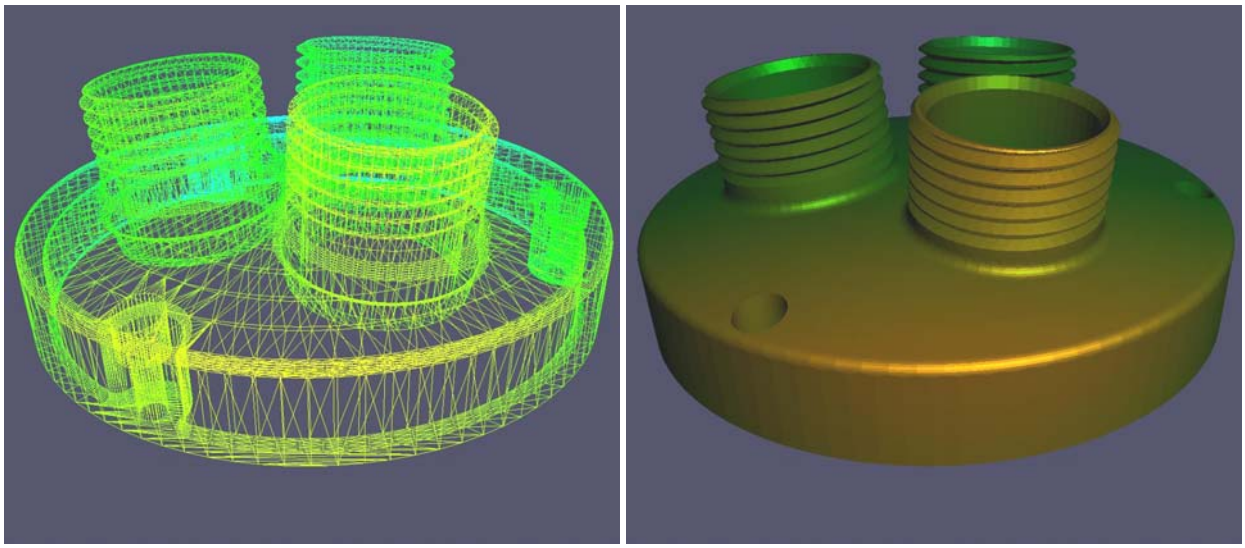
Two views of an earthquake fault valley



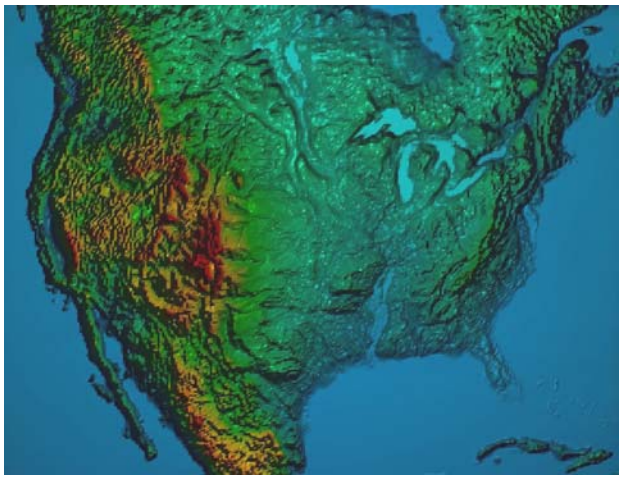
Protein Kinase



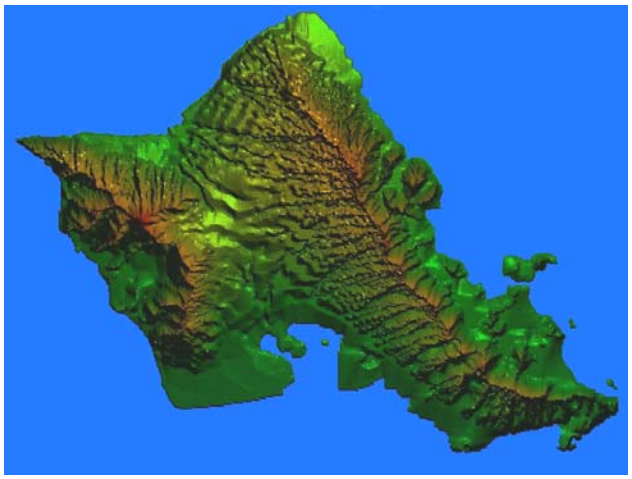
Grid of Phenanthroline



Mechanical CAD



United States Map



Oahu Map