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

Published in the *Journal of Graphics Tools*, Volume 3, Number 3, August 1999, pp. 1-9.

Michael Bailey[1]
Dru Clark

Oregon State University

Abstract

Stereographics is an effective way to enhance insight in 3D scientific visualization. This is especially true for visualizations consisting of complex geometry, such as molecular studies, or where one dataset needs to be registered against another, such as in earth science. But, as effective as it is, stereoviewing sees only limited use in scientific visualization because of the difficulty and expense of creating images that *everyone* can see. This paper demonstrates how a low-end, inexpensive viewing technique can be used as a "quick trick" to produce many of the same effects as high-end stereoviewing. Not only is this technique easy to view and easy to publish, it is easy to create. This paper shows how standard OpenGL features can be used to create such images, both statically and interactively.

Stereoviewing for Interactive and Published Scientific Visualization

The benefits of using stereoviewing in entertainment and scientific visualization are well known. By simulating human binocular vision, stereo imagery can greatly enhance a user's understanding and enjoyment of a 3D scene. The methods for simulating the views seen by the left and right eyes are fairly straightforward. If this is a real scene, then two cameras are typically used, each one separated by a distance that approximates the eye separation distance. If this is a synthetic scene, then special off-axis viewing transformations need to be used ([LIPTON91, HODGES85]).

[1] Contact information:
Oregon State University
Computer Science
Corvallis, OR 97331-3211
mjb@eecs.oregonstate.edu
dclark@ucsd.edu

The real trick is figuring out the best way to present the left and right eye images to just the left and right eyes, respectively. Over the years, a number of stereo viewing methods have been introduced.

1. **Optical Separation Devices:** left and right eye images are presented side-by-side with some sort of optical device used to channel the proper image into the proper eye. Many systems work this way, from mirrors that mount to monitor faces to stereo slide viewers to View-Masters™ to sophisticated virtual reality display devices. Some people can freeview side-by-side stereo views without any special equipment (in either a parallel or cross-eye viewing mode), but this is not common in the general population.

2. **Red/Blue Anaglyph:** left and right eye images are combined into a single image consisting of blues for the left eye portion of the scene, reds for the right eye portion of the scene, and shades of magenta for portions of the scene occupied by both images. The viewer wears a pair of glasses with red over the left eye and blue over the right eye. Each eyepiece causes linework destined for the other eye to meld into the background and causes linework destined for its own eye to appear black. Many people's first experience with stereo vision was using this technique while watching the classic movie *Creature from the Black Lagoon*.

3. **Polarized Lenses:** left and right views are projected through orthogonal polarizing filters into a single image, which is the viewed through polarizing lenses. Highly informative stereo slide presentations can be done this way, as well as movie and video presentations. This is also the basis for Disney's stereo movies *Captain Eo* and *Honey, I Shrunk the Audience*.

4. **CrystalEyes™:** this variation of the polarizing lenses is the most common of the single-monitor computer graphics stereo display methods. CrystalEyes displays the left and right eye views of a synthetic scene in sequential refresh scans of a monitor and then uses synchronized polarized shutter glasses to channel the correct image into the correct eye. This is also the basis for the stereographics in the CAVE virtual reality display environment ([CRUZ-NEIRA93]).

Other techniques have also been used for stereoviewing, such as lenticular displays, random dot stereograms, and the Pulfrich effect. There are surely others. As these are less relevant to interactive and published scientific visualization, they were not covered here.

Limitations of Existing Methods

These methods all work reasonably well for limited uses. But, they all have problems when used for serious interactive and publication scientific visualization:

- Methods 3 and 4 cannot be reproduced in print because their stereo effect is tied to their display method. This means that they cannot be used for stereo presentation in papers, articles, or on web pages.

- Methods 2, 3, and 4 create an image that is unrecognizeable unless the viewer is wearing the proper glasses. This also limits the use of these methods in print or on web pages.

- Method 1 cannot be seen by most people without special glasses. A large number of people cannot get these images to converge, even *with* the special glasses.

For scientific visualization, we need a stereoviewing method that can create an image that can:

- be viewed clearly as a single image without glasses

- be viewed as a stereographics image with inexpensive glasses

- appear in print

- appear on a web page

- be used with fast display hardware for interactive use.

ChromaDepth

ChromaDepth™ was invented by Richard Steenblik ([STEENBLIK87]) 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 below in Figure 1, contain very thin diffractive optics that have the efficiency of refractive optics. While being very thin and inexpensive[2], 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.

Figure 1: ChromaDepth Glasses

[2] Prices for the “opera-style” ChromaDepth glasses shown in Figure 1 range from 10¢ to 90¢, depending on volume.

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 ([FOLEY90]).

The next step is to specify how to compute the texture coordinate based on scene depth. [WOO97] 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:

$$s = A*x + B*y + C*z + D \qquad (1)$$

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:

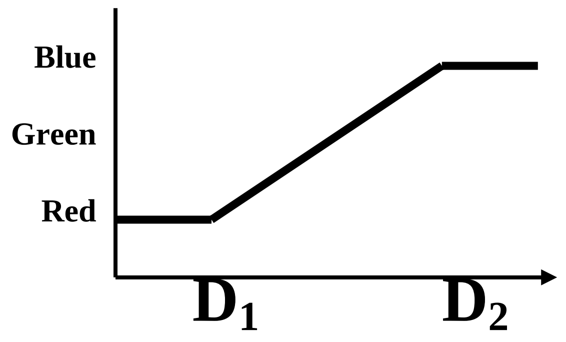

Figure 2: Ramp from Red to Blue as a Function of Distance in Front of the Eye

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:

$$s = \frac{-Z}{D_2 - D_1} - \frac{D_1}{D_2 - D_1} \tag{2}$$

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. In practice, this is not a restrictive limitation, as all scene and object transformations can be included

Examples

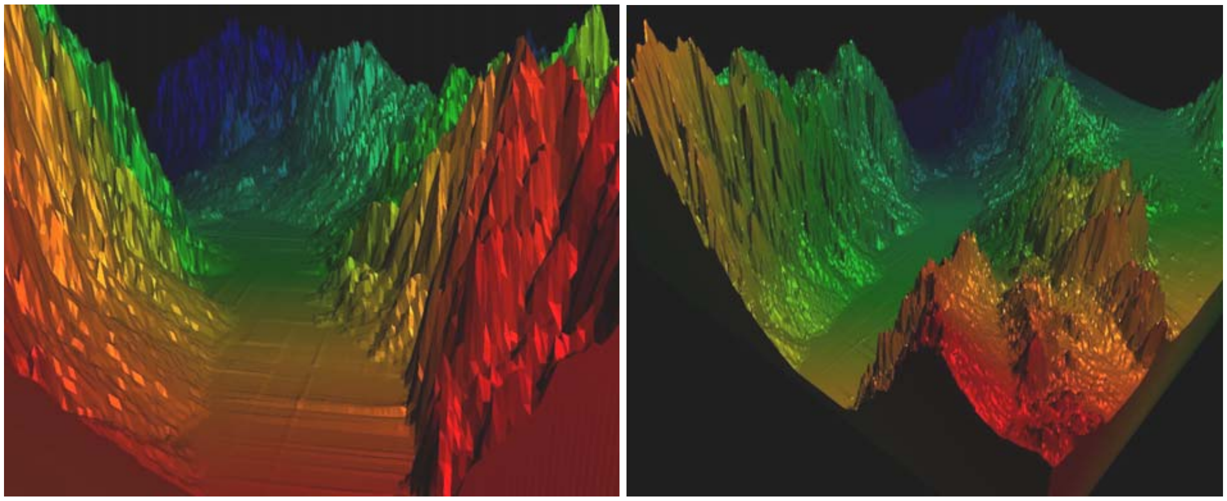

Figure 3: Two views of an earthquake fault valley

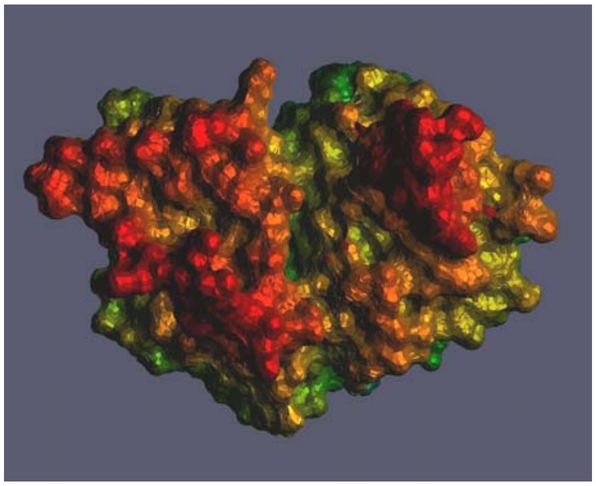

Figure 4: Protein Kinase

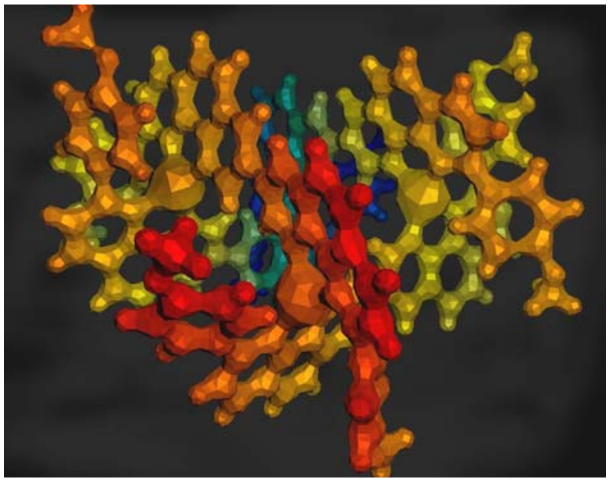

Figure 5: Grid of Phenanthroline

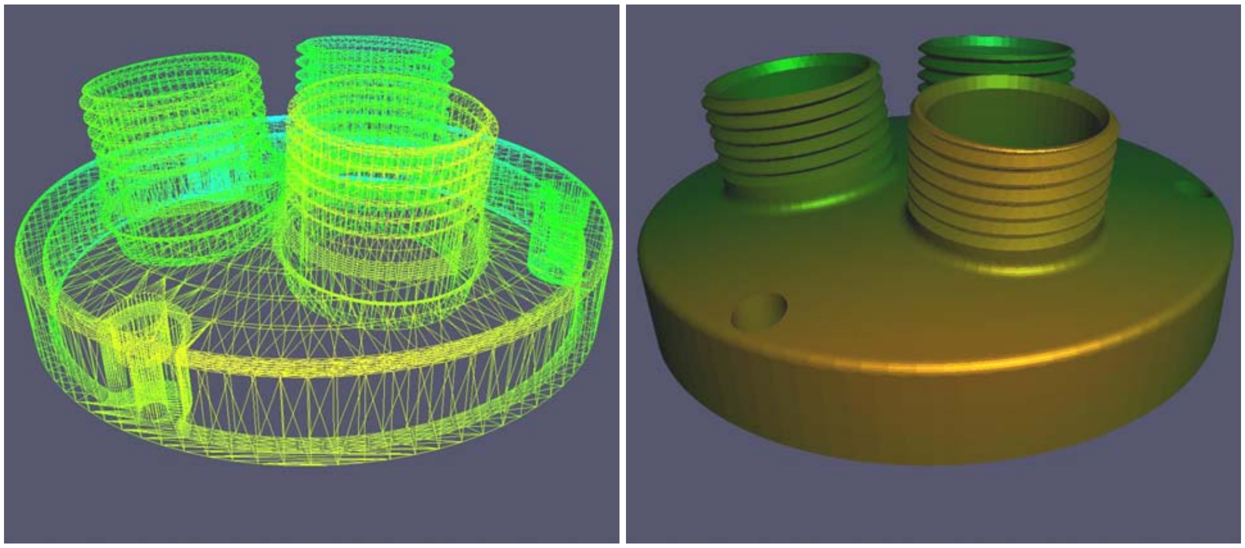

Figure 6: Mechanical CAD

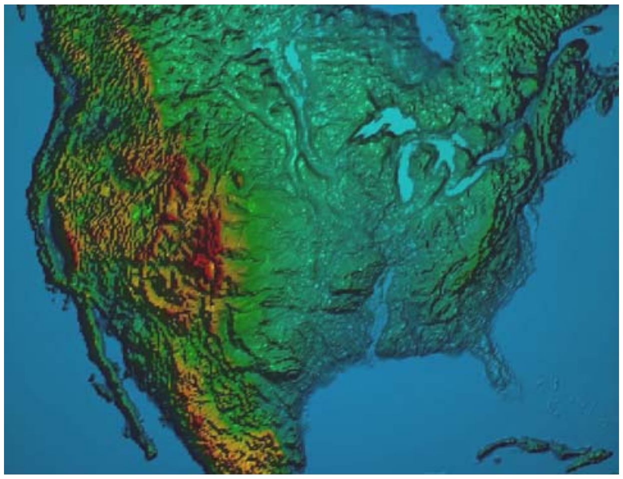

Figure 7: United States Map

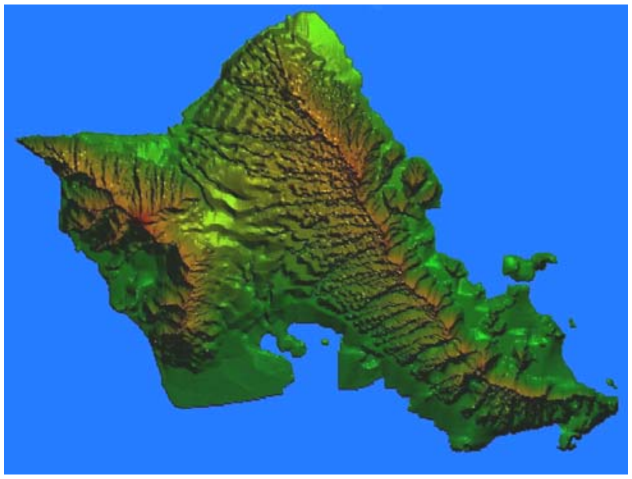

Figure 8: Oahu Map

Conclusions

ChromaDepth is a very useful “quick trick” to display 3D scenes in apparent stereovision. Scenes can be viewed highly interactively on an OpenGL-based graphics system or statically in a publication or a web page. There are no problems with having to separate and converge the two elements of a stereopair. ChromaDepth uses a single image that is valid with or without the glasses. Because the glasses are so simple and inexpensive, this method is practical for widespread use.

The major disadvantage of this method is that control of the color is given up to the depth display. Many scientific visualizations use color to represent a scalar variable overlaid on top of the geometry. In this case, that ability is lost.

It should also be noted that this technique usually looks better color-printed than on a monitor. The underlying technology for CRT monitors uses three unique wavelengths. Intermediate colors are obtained by displaying those unique wavelengths side-by-side. This tends to discretize the depth of the ChromaDepth scene somewhat. The underlying technology for color printing, however, can use more colors to get a wider range of wavelengths, so its range of depths can appear more continuous.

Acknowledgements

This work was partially supported by NSF grant MIP-9420099.

Thanks to Paula and Richard Steenblik at ChromaTek for their donation of the ChromaDepth glasses that are included with this issue of JGT.

Thanks to the following people and groups for the data used in various visualizations:

- Eric Frost, Andres Polit, and Mitra Fattahipour, San Diego State University, for the earthquake fault valley data.
- Lynn Ten Eyck and Susan Taylor, UCSD, for the protein kinase data.
- Jay Siegel, UCSD and Kim Baldridge, SDSC for the phenanthroline data.
- Mike Arnstein, UCSD, for the lighting lens base geometry.
- Stuart Gage, Michigan State University, for the data for the USA data.
- Todd Elvins and John Helly, SDSC, for the Oahu data.

Web Page

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

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

References

[CRUZ-NEIRA93]
Carolina Cruz-Neira, Daniel Sandin, Thomas DeFanti, “Surround-Screen Projection-Based Virtual Reality: The Design and Implementation of the CAVE”, *Proceedings of the SIGGRAPH ’93 Conference*, pp 135-142.

[FOLEY90]
Jim Foley, Andries van Dam, Steve Feiner, and John Hughes, *Computer Graphics Principles and Practices*, Addison-Wesley, 1990.

[HODGES85]
Larry Hodges and Dave McAllister, “Technology and Techniques for Stereoscopic Display of Computer Generated Images,” *Proceedings of Trends and Applications ’85: Utilizing Computer Graphics*, IEEE Computer Society Press, pp 107-115.

[LIPTON91]
Lenny Lipton, *The CrystalEyes Handbook*, Stereographics Corportation, 1991.

[STEENBLIK87]
Richard Steenblik, “The chromostereoscopic process: a novel single image stereoscopic process, “ *Proceedings of SPIE: True 3D Imaging Techniques and Display Technologies*, January 1987.

[WOO97]
Mason Woo, Jackie Neider, and Tom Davis, *OpenGL Programming Guide*, Addison-Wesley, 1997.