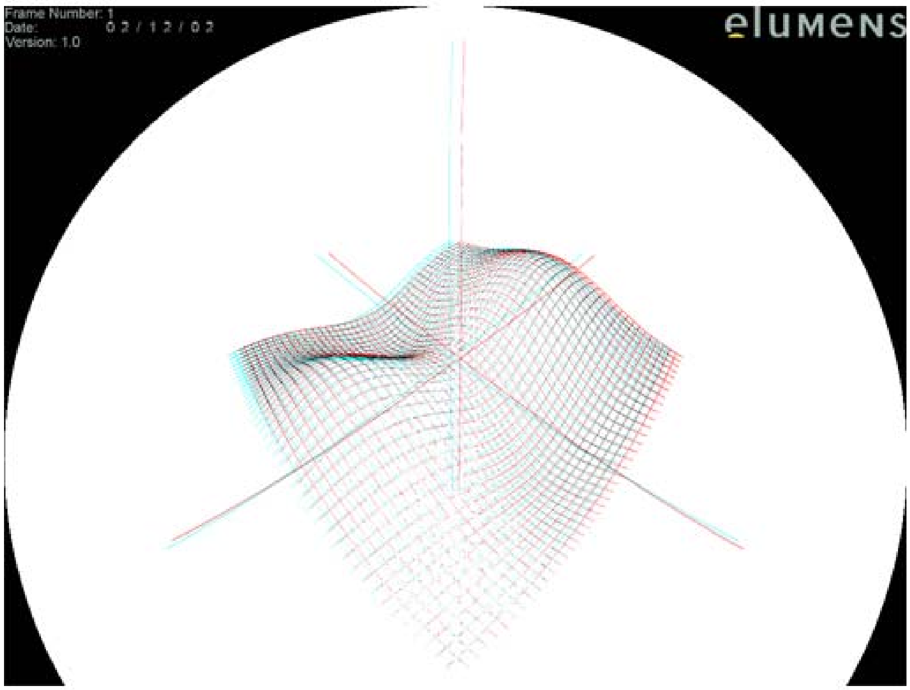
Projecting Anaglyph Stereo onto a Dome



$$z = \left(x^2 - y^2\right)e^{\frac{-x^2-y^2}{2}}$$

Catherine A. Daubert
March 25, 2002

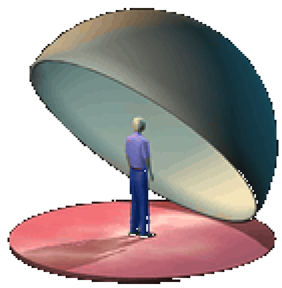
David McAllister,
Project Advisor

Introduction

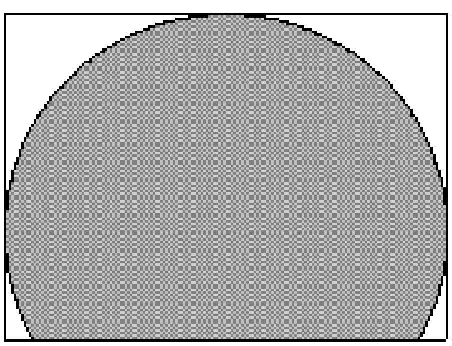
As computer graphics becomes more important in business, academia, and gaming, the need for believable displays becomes more and more prevalent.

Surrounding the viewer with an image, giving the viewer the feeling that he or she has actually been transported to another location, creates a sensation of immersion that makes a virtual world take on a true sense of realism.

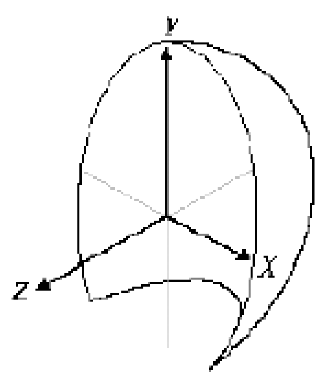
To this end, RTP-based Elumens Corporation has created the VisionDome, which supports a 180 x 180 degree field of view.



The VisionDome4 in the NC State Design School supports a 180x135 degree field of view:



Frontal view of truncated VisionDome



VisionDome4 Coordinate System
(all values normalized, ∈ [–1,1])

This large screen gets the viewer into the presentation, but ultimately the viewer is still surrounded by a flat image, which is not very realistic.

The solution to this problem is to introduce stereopsis, or a sense of depth, into the display.

Applications of dome stereo:

- Business Presentations
- Training/Simulations
- Architectural Design
- Entertainment (gaming, art)
- Education

Elumens currently offers frame-sequential stereo using LCD shutter glasses.

CrystalEyes3 System

Problem:

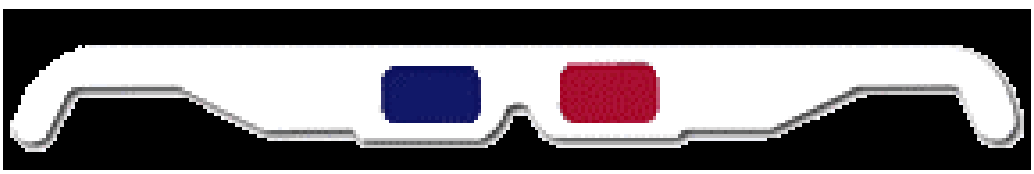
- LCD shutter glasses are expensive, ranging from around \$60 to \$800 per pair.

- A system of two pairs of glasses and an emitter range from \$1000 to \$2000 at least.

Solution:

- Cheaper stereo using anaglyphs

- Glasses for anaglyph viewing cost under 50 cents!
  (\$10 for the plastic frames..)



Question:

- How do we correctly project anaglyphs on the dome?

Using anaglyphs is different from shutter glass technology because both eye views are on the screen at the same time.

Here we explore:

- How to create stereopsis in two dimensions (briefly)

- Various methods of stereo computation for a flat screen in OpenGL

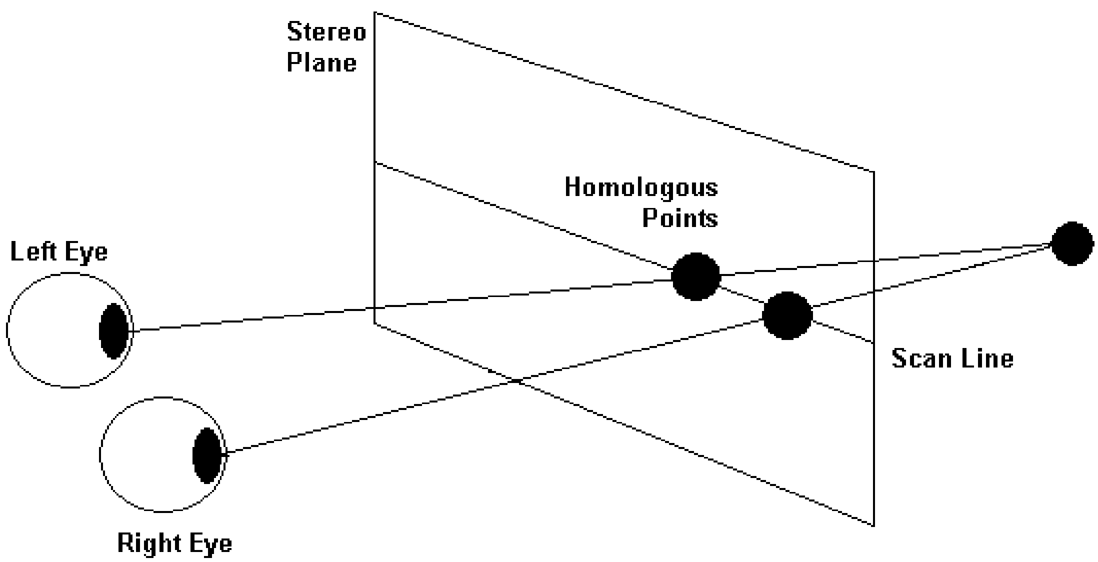
- How to combine methods of stereo computation with Elumens' TruFrame software

- My examples

Stereo Review

The right eye view and the left eye view are collectively called a stereo pair.

What makes stereo pairs effective is horizontal parallax between the two views, creating disparity between the two images on the retinas. This is the number one depth cue for people.



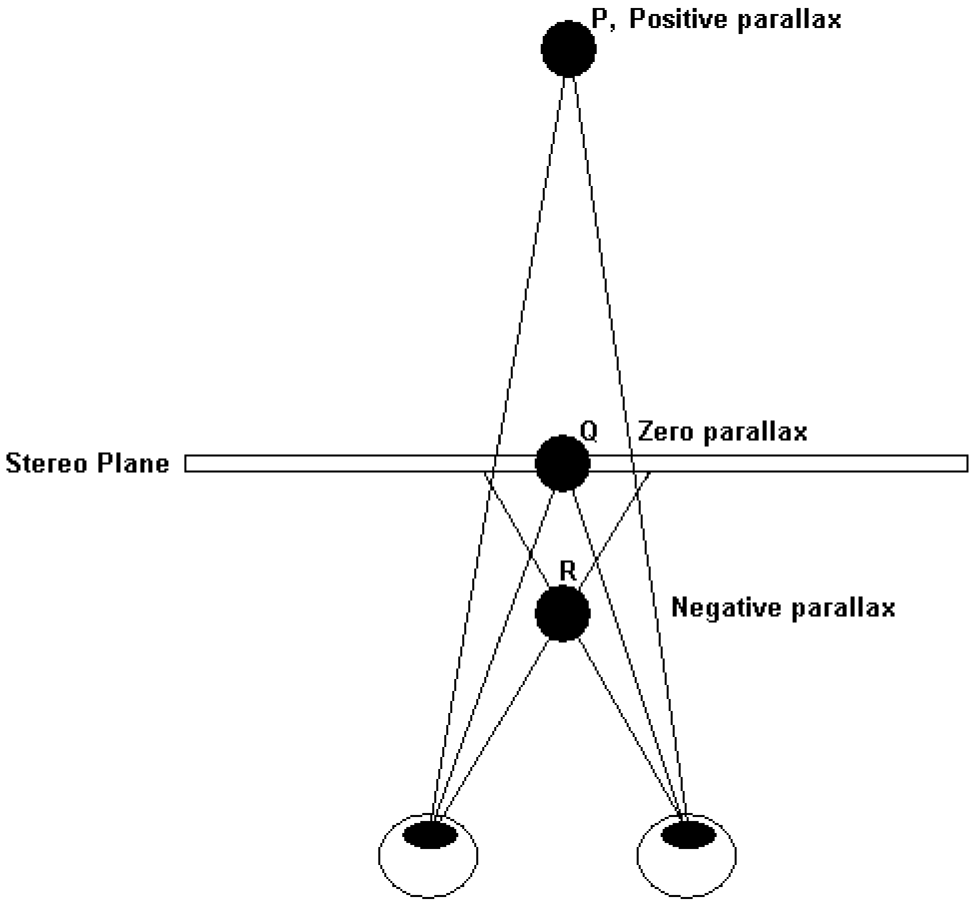
Homologous Points

Stereo pairs should not have vertical parallax since it causes discomfort for the viewer. This type of parallax also hinders the ability to detect depth to the point that the viewer may not be able to merge the two images at all.

Horizontal parallax can then be positive or negative. Choose the stereo plane to be somewhere near the middle of the viewing area.

- Objects with positive parallax are seen to be behind the plane.
- Objects with negative parallax fall in front of the plane.
- Objects with zero parallax are at the stereo plane.

- Parallax values greater than the interocular distance are unacceptable, since they cause the eyes to roll outward from each other, an uncomfortable and unrealistic movement. When this happens, it is said that the viewer is wall-eyed.



Horizontal Parallax Diagram

Parallax can be measured in inches or degrees. The parallax distance between homologous points would be in inches; the parallax angle θ describes the angle made by the homologous points and the viewer.

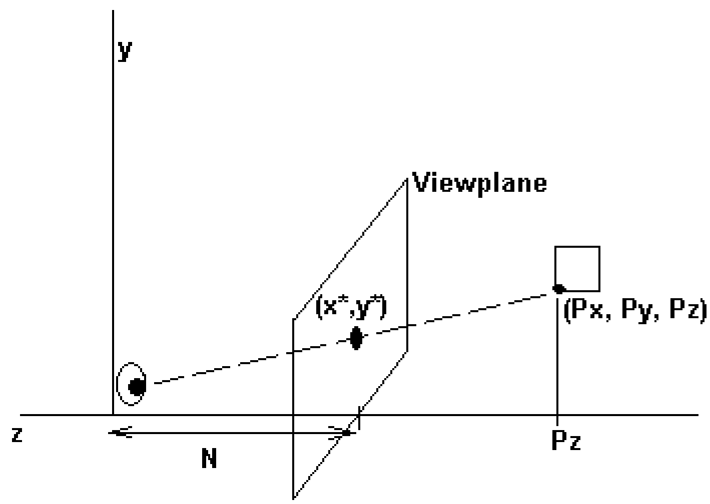
Finding the parallax p from θ:

$p = 2d\tan(\theta/2)$, where d is the distance from the viewer to the stereo plane.

- Maximum parallax ≈ 1.5°
  - Standard computer monitor: p ≈ 0.5 inches
  - Four meter dome: p ≈ 1.9 inches
  - Or 3% of screen width

Calculating Stereo Pairs

Perspective Projection

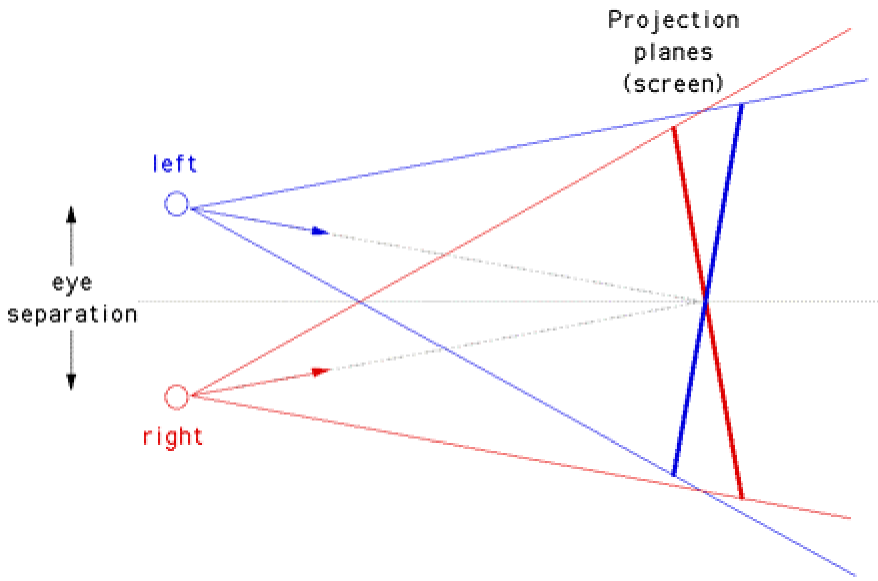


$$\frac{x^*}{P_x} = \frac{N}{-P_z}, \qquad \frac{y^*}{P_y} = \frac{N}{-P_z}$$

$$(P_x, P_y, P_z) \rightarrow \left(N\frac{P_x}{-P_z}, N\frac{P_y}{-P_z} \right) = (x^*, y^*)$$

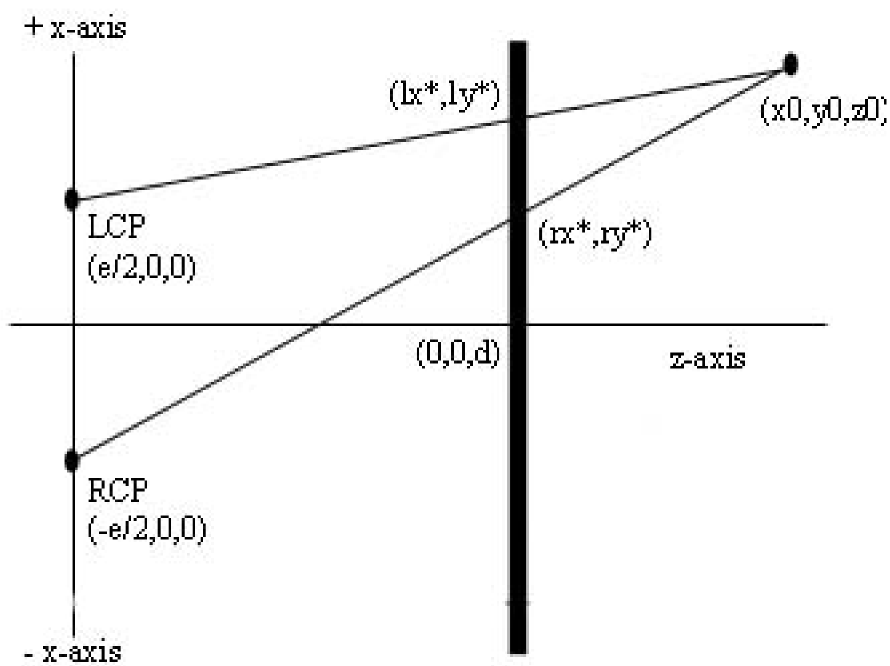
Ways to calculate stereo pairs (using a perspective projection):

- Rotation, or the toe-in method



Problem: Causes vertical parallax and distorts the stereo plane. Not a friendly way to generate stereo.

- Two Centers of Projection
  Choose a value e for the camera offset. Translate the scene e/2 to the left and right to generate the stereo pairs.



Two centers of projection model; viewplane at z=d.

No vertical parallax:
Determine placement on viewing plane using parametric equations of line between (x0,y0,z0) and LCP.

$$x = x_o + t(e/2 - x_0) \qquad y = y_o + t(0 - y_0) \qquad z = z_o + t(0 - z_0)$$

At viewing plane, z = d. Solve for t.

$$t = (z_0 - d)/z_0$$

Substitute into parametric equations to get:

$$(lx^*, ly^*) = \left(\frac{dx_0}{z_0} - \frac{ed}{2z_0} + \frac{e}{2}, \frac{dy_0}{z_0} \right)$$

Same for RCP.

$$(rx^*, ry^*) = \left(\frac{dx_0}{z_0} - \frac{ed}{2z_0} - \frac{e}{2}, \frac{dy_0}{z_0} \right)$$

Equivalent y-coordinates $\Rightarrow$ no vertical parallax.

Perspective transformation:

$$\left(P_x, P_y, P_z\right) \rightarrow \left(N\frac{(P_x + e/2)}{-P_z} - \frac{e}{2}, N\frac{P_y}{-P_z} \right) = \left(x^*, y^*\right) \quad \textbf{(Left Eye)}$$

$$\left(P_x, P_y, P_z\right) \rightarrow \left(N\frac{(P_x - e/2)}{-P_z} + \frac{e}{2}, N\frac{P_y}{-P_z} \right) = \left(x^*, y^*\right) \quad \textbf{(Right Eye)}$$

In OpenGL, this shift is implemented by translation in the modelview matrix. The e term was recalculated for each frame in the animation based on a set parallax angle.

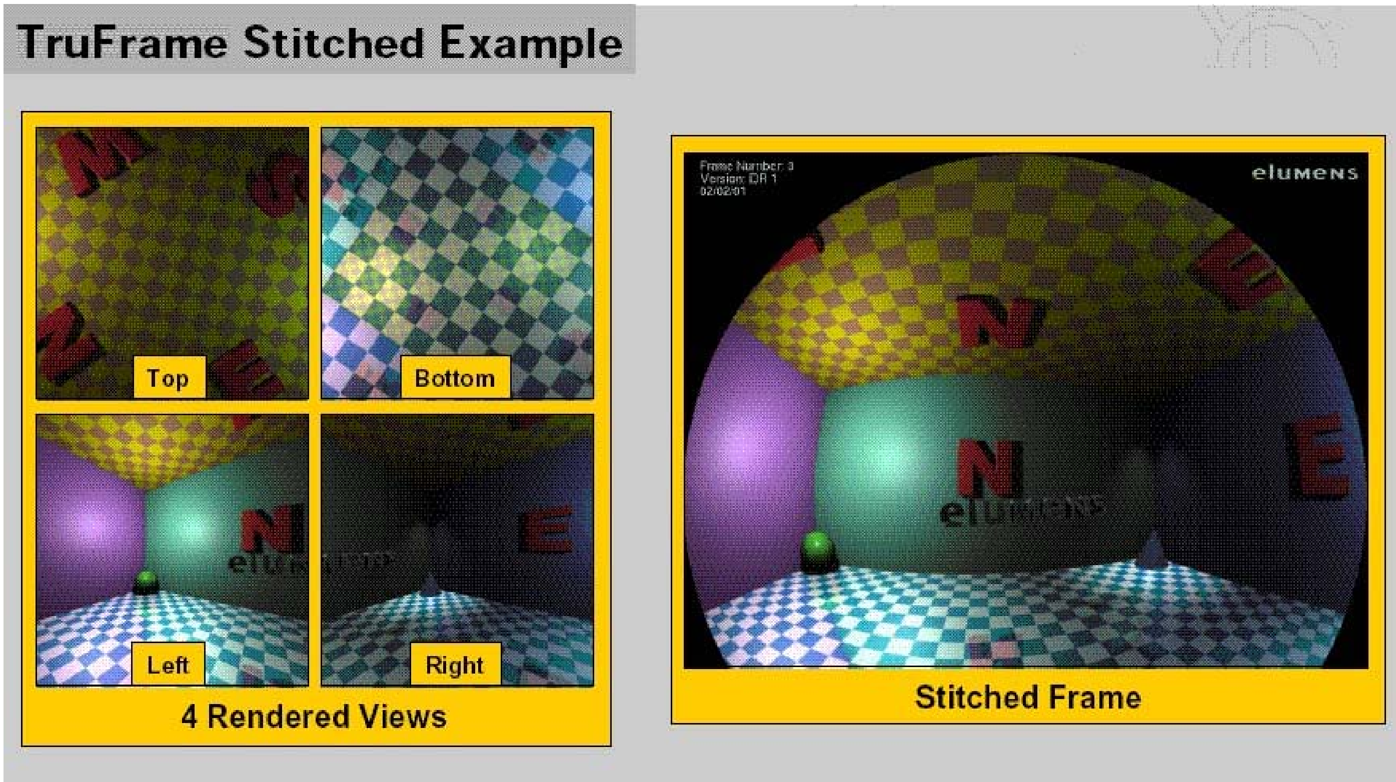
An additional shift (yielding both positive and negative parallax) is created by altering the projection matrix so that the perspective view volume is asymmetrical.

Elumens' Truframe Software

TruFrame is an OpenGL based software package that stitches flat images together so that they can be viewed correctly on the surface of the dome.

Input:

- Off-axis positions of the viewer (Eye X,Y, and Z) and the projector (Lens X,Y, and Z) in dome coordinates
- Four views of a scene taken with a camera with a 90° horizontal and vertical field of view.



To create the four necessary views, the camera is oriented directly in front of the scene. Each view is then captured by employing a certain rotation:

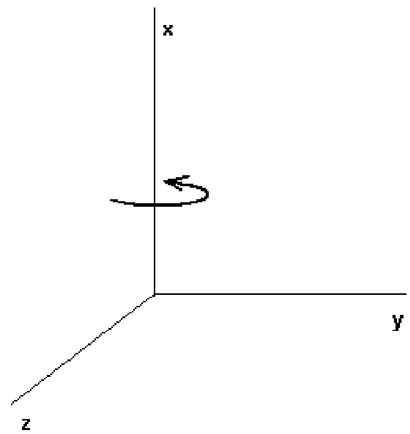
1. Right – rotate camera 45° to the right
2. Left – rotate camera 45° to the left
3. Top – rotate camera 45° to the right, 90° upward
4. Bottom – rotate camera 45° to the right, 90° downward

- Rotating the camera to the right (or left) is equivalent to rotating the scene to the left (or right).

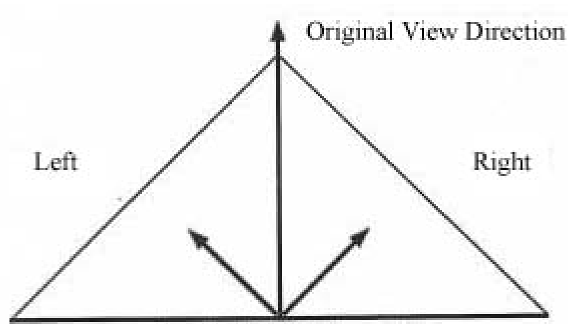
In OpenGL these rotations are implemented using the *glRotate*()* method, which basically multiplies the current matrix in the viewing matrix stack by the appropriate rotation matrix. Then each pixel is multiplied by this matrix.

> Example: Creation of the Right eye's modeling matrix begins by multiplying this matrix by the identity:

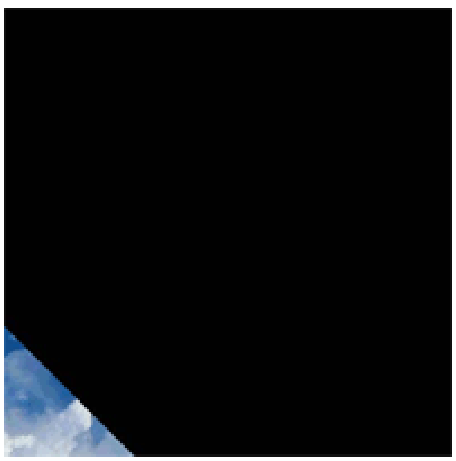
$$\begin{pmatrix} \cos 45 & 0 & \sin 45 & 0 \\ 0 & 1 & 0 & 0 \\ -\sin 45 & 0 & \cos 45 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$



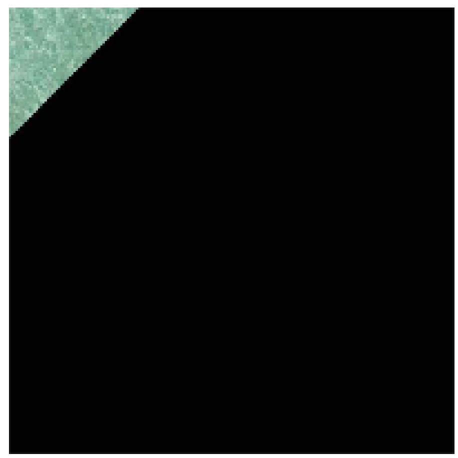
Note that this method must be called directly after the modelview matrix is set to the identity, and before any other modelview matrix operations are completed, since rotating the camera is equivalent to rotating the coordinate system.



TruFrame Views from Above

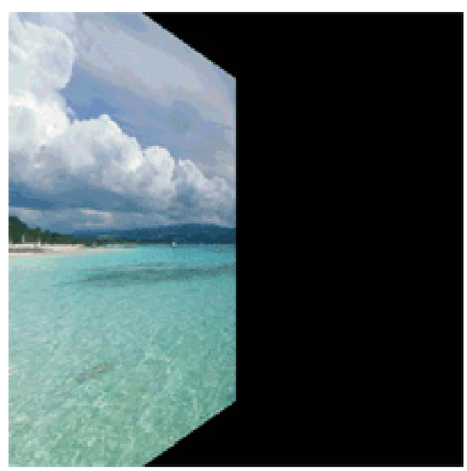


Top



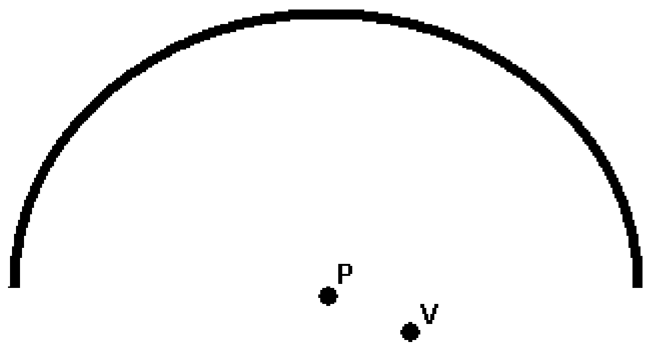
Bottom

Left



Right

TruFrame textures a portion of a sphere shaped like the VisionDome with the four views using the off-axis positions to get the correct orientation.



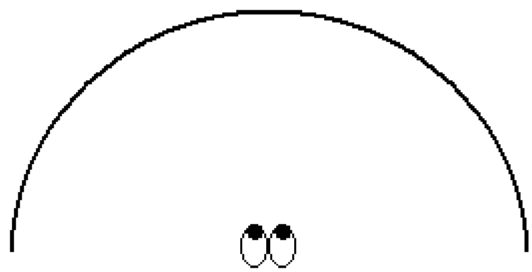
Dome from Above

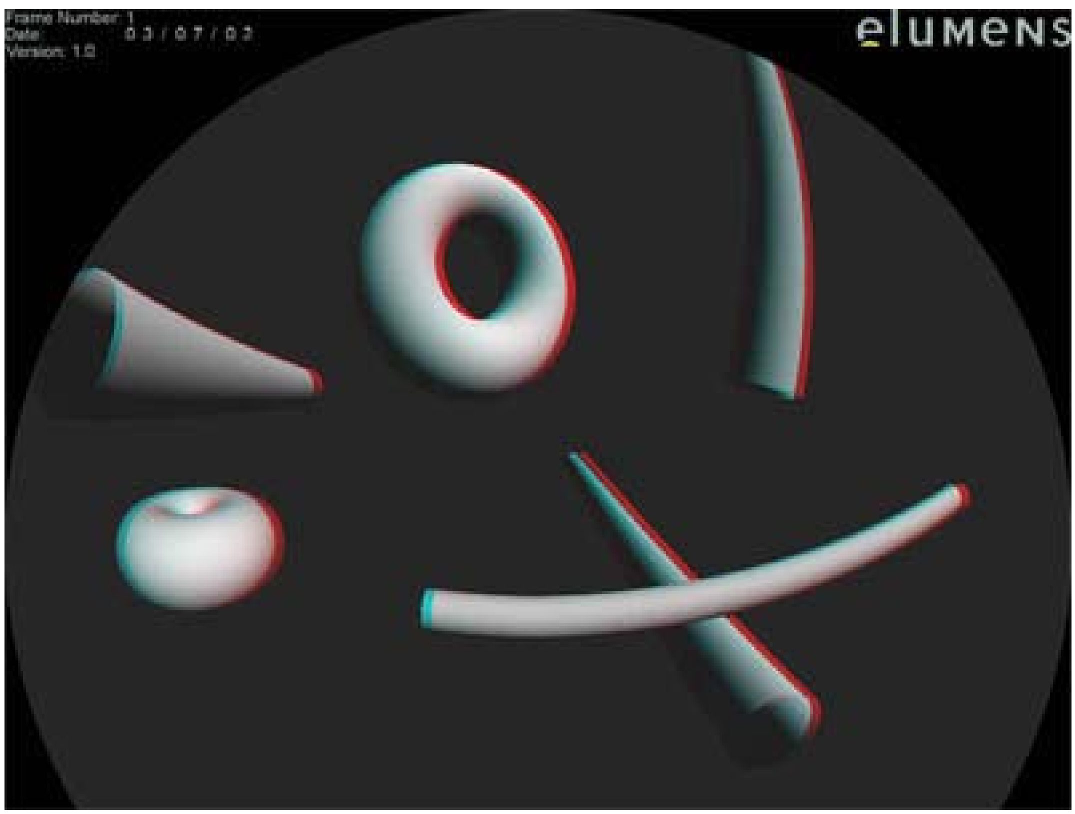
Creating Dome Anaglyphs

- Without stereo, dome is for monocular viewing.
- With stereo, two viewing positions must be considered.
- Possible approaches:
  - ➢ Render eight views, four for each eye.
    - Stitch each eye separately, and then combine the two dome-ready images in Adobe Photoshop.
    - Create an anaglyph of each eye view, condensing eight images to four, and then stitch. WRONG
  - ➢ Create the anaglyph in OpenGL using color masking; render the four views and stitch.

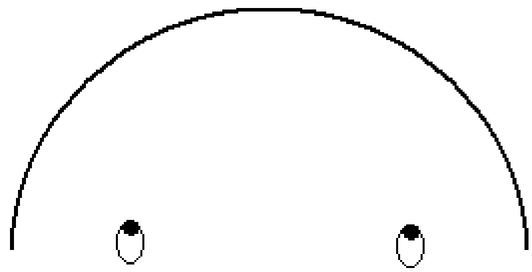
The approach depends on the radius of the dome.

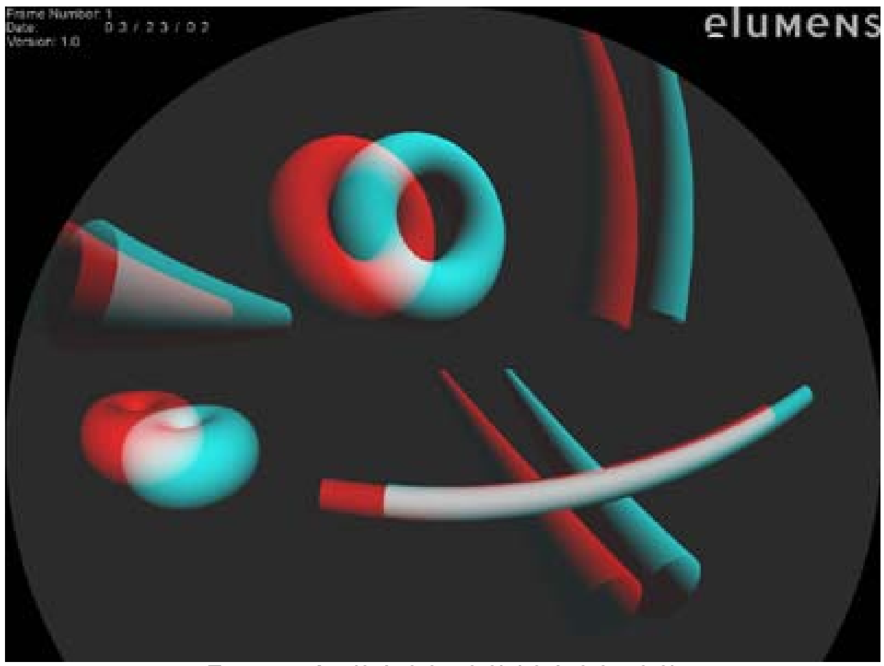
With the VisionStation4, the radius is very large with respect to the interocular distance, and so creating the anaglyph in OpenGL first does not skew the projection.





One image stitched with eye coord (0, 0.37, 0.30)



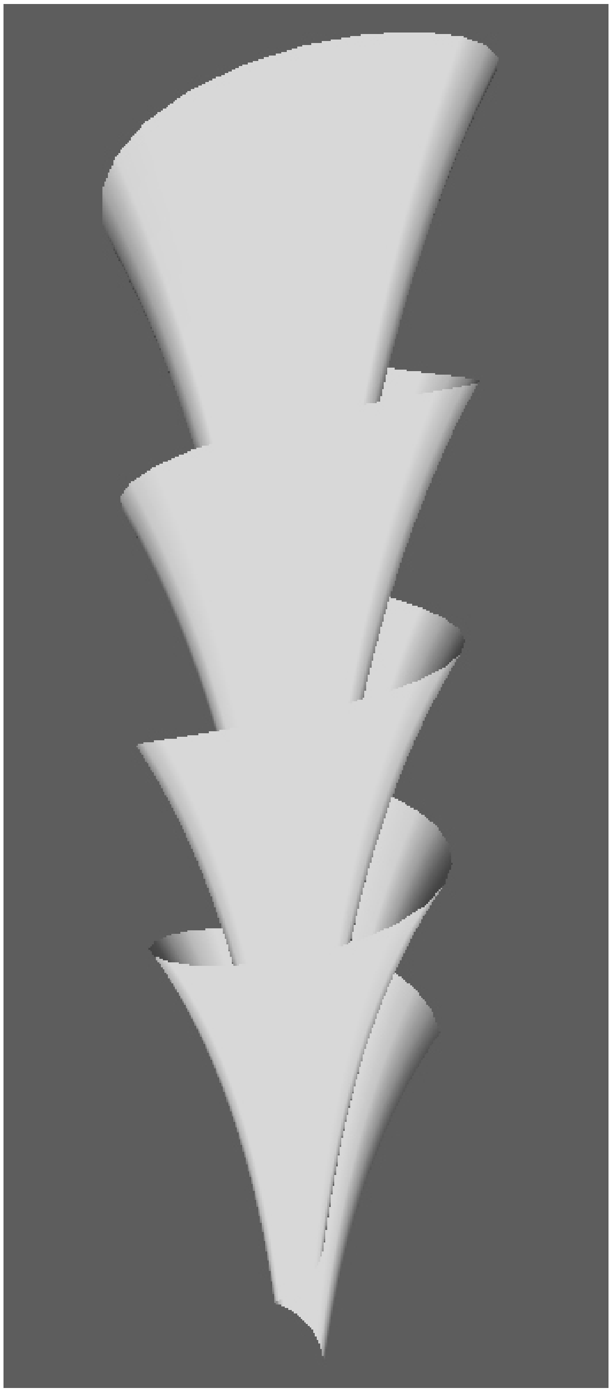


Eye coords: (0.1, 0.37, 0.3),(-0.1, 0.37, 0.3)

Wrapping Up

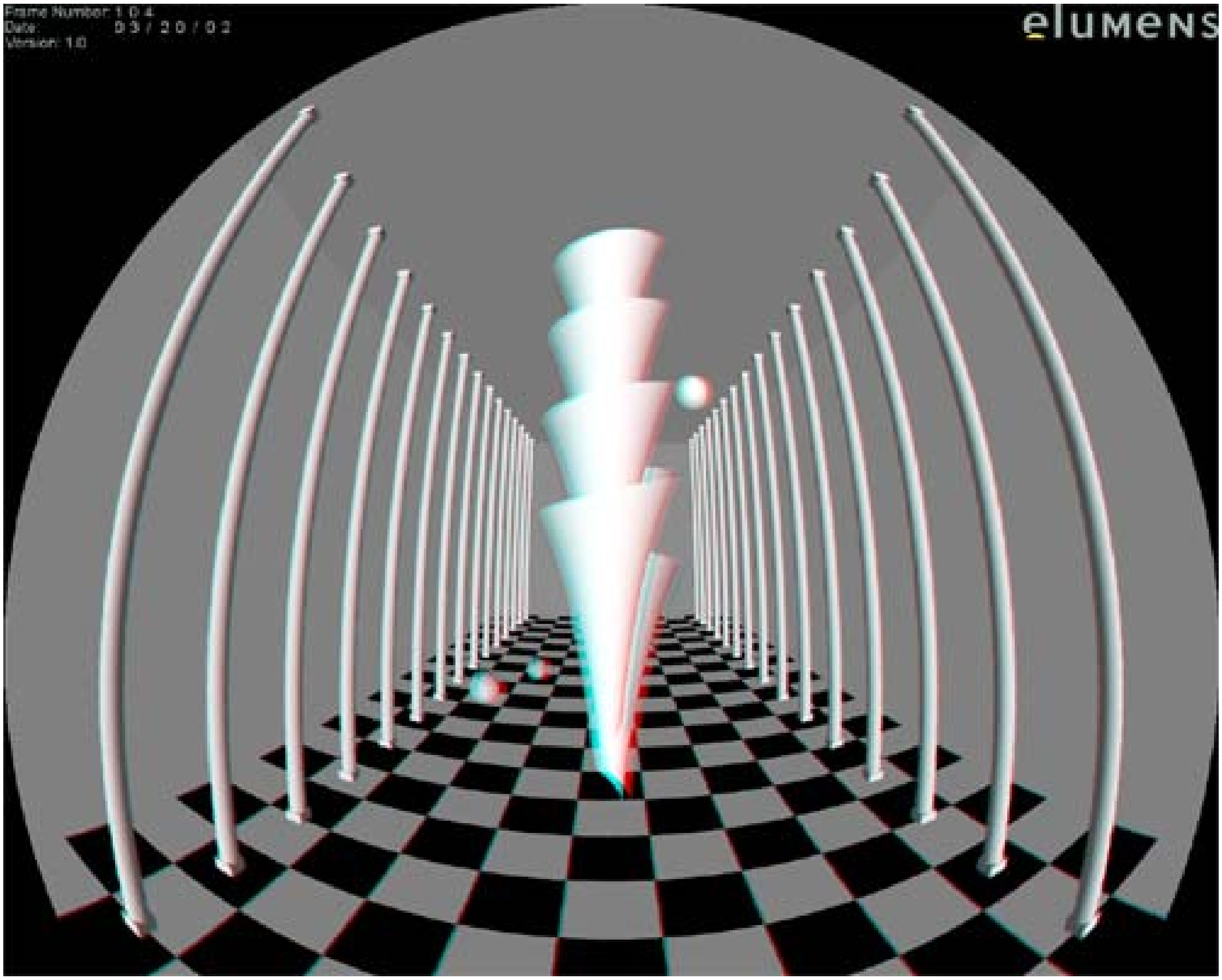
- Class audit
- All coding done in Visual C++ and OpenGL
- Texture Mapping a Rectangle
  - ➢ Read images into and out of OpenGL pixelwise… bitmaps

- Creating Anaglyphs
  - Animation
  - Mesh class… creates mesh from parametrically defined surfaces
  - Hardcoded to create Dini's Surface



$$x = a \cos u \sin v$$
$$y = a \sin u \sin v$$
$$z = a(\cos v + \log(\tan \frac{v}{2})) + bu$$

Dini's Surface



Acknowledgements

- Dr. McAllister
- NCSU Design School (Jay Tomlinson’s group)
- You, the committee
- NCSU Mathematics Department
- Hart