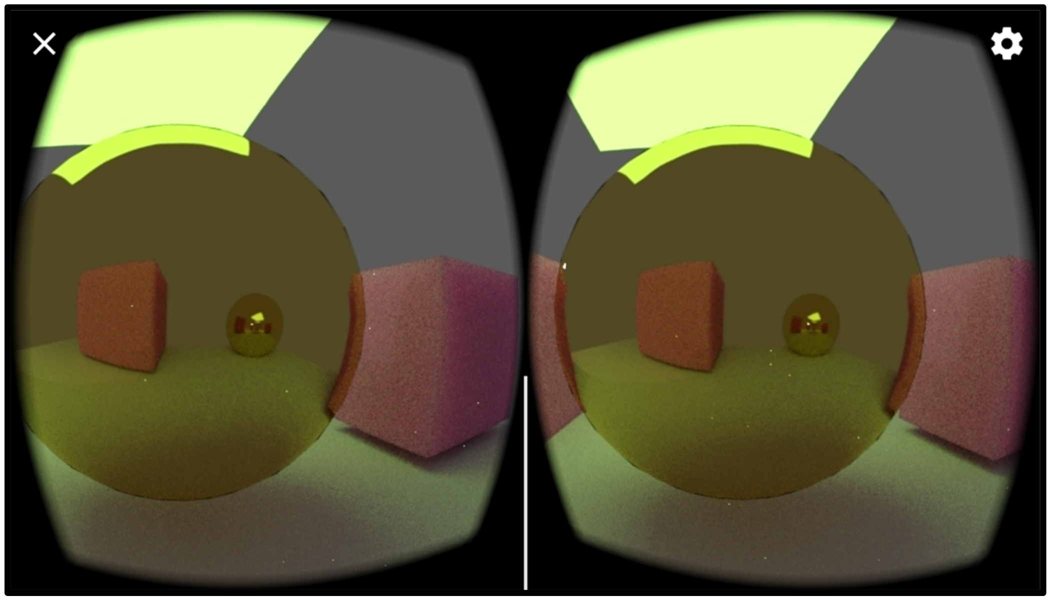
Spherical Stereographics: Inexpensive VR for your SmartPhone

Oregon State University

Mike Bailey

mjb@cs.oregonstate.edu

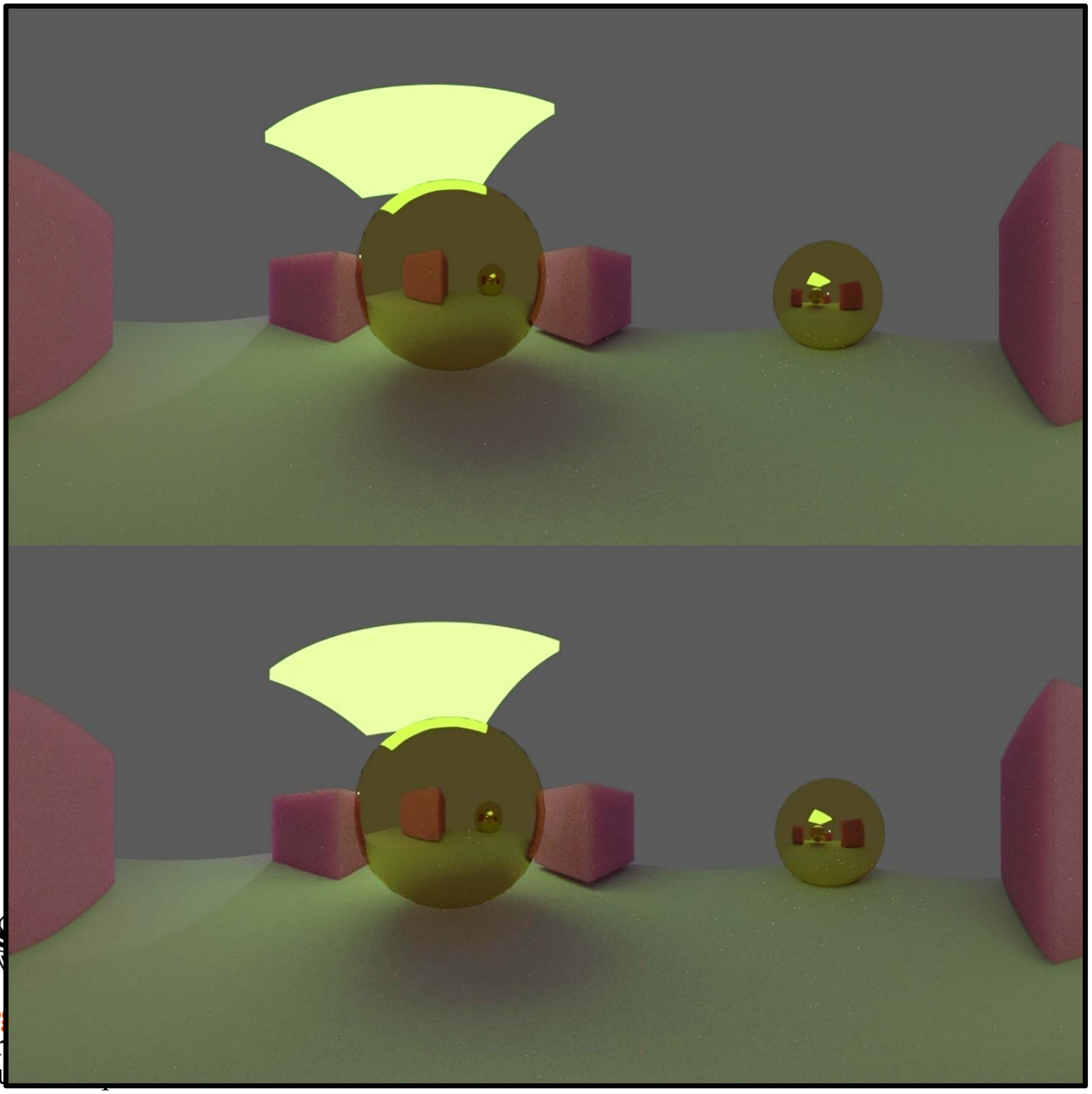
Spherical Stereographics



In these notes, you will see how to generate dynamic stereographics for your phone to view inside a VR headset.

This has a very high Coolness-Factor. Your friends will *really* like it!

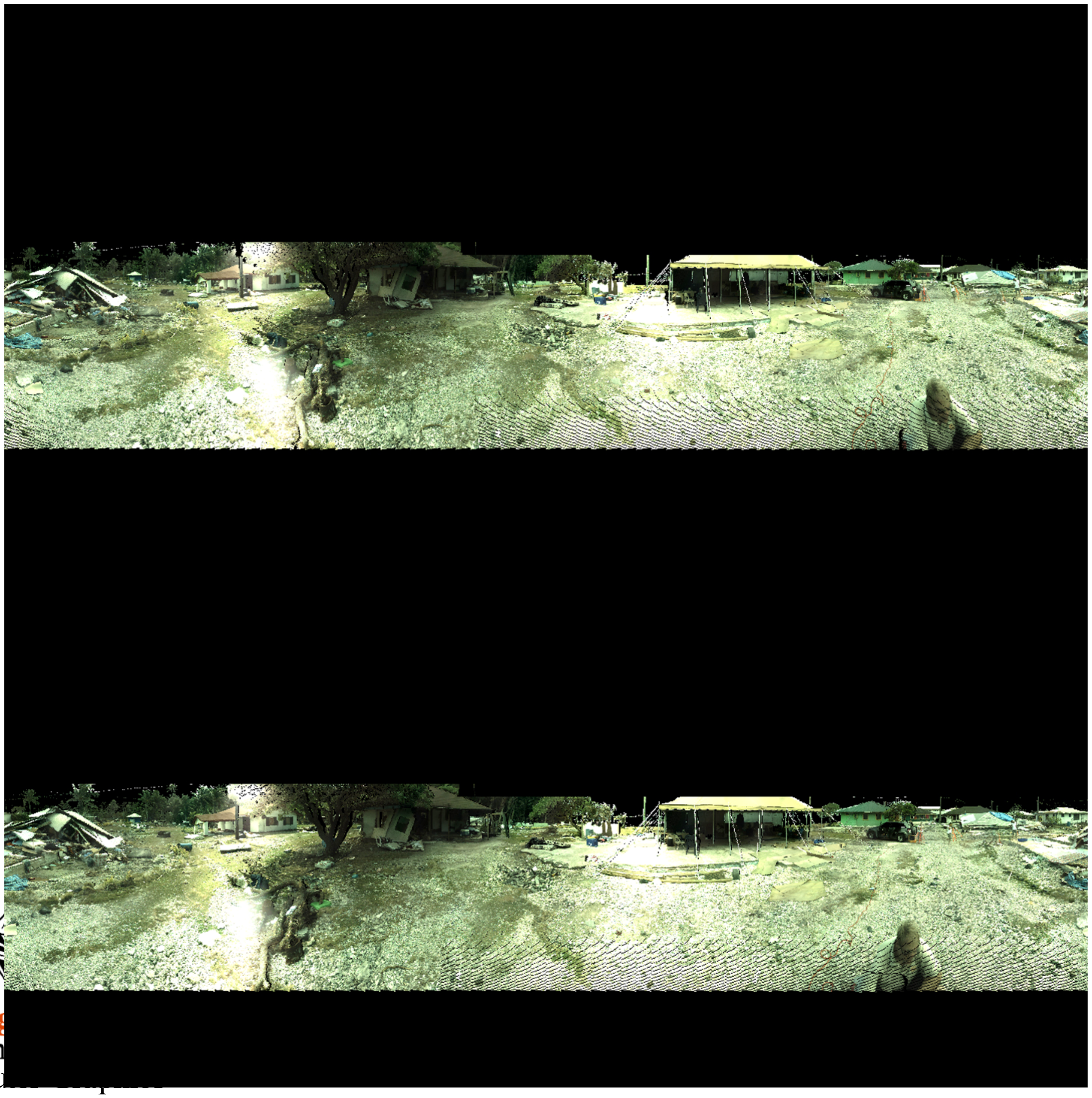
A Scene from Blender



Left Eye View

Right Eye View

Create your own scene

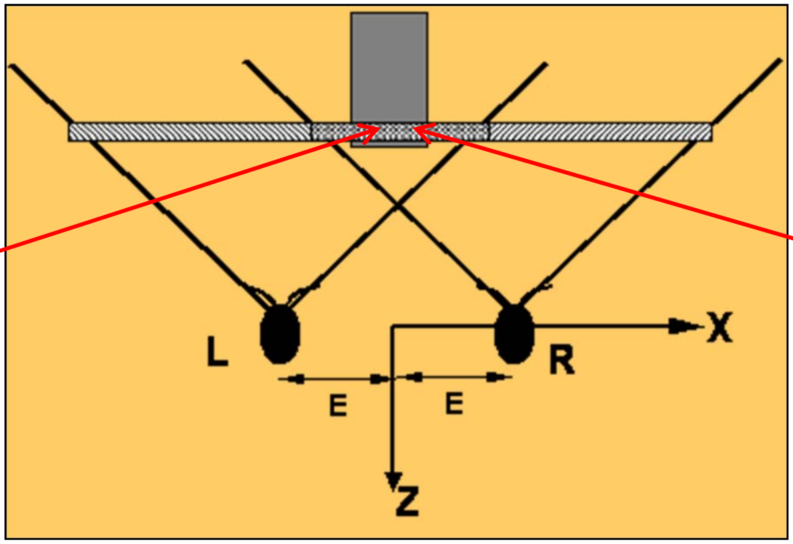


Left Eye View

Right Eye View

Two Side-by-side Perspective Viewing Volumes

The left eye sees the box towards the far right side of its display



The right eye sees the box towards the far left side of its display

The left eye view is obtained by translating the eye by -E in the X direction, which is actually accomplished by translating the scene by +E instead. Similarly, the right eye view is obtained by translating the scene by -E in the X direction. We now have a *horizontal parallax* situation, where the same point projects to a different horizontal position in the left and right eye views.

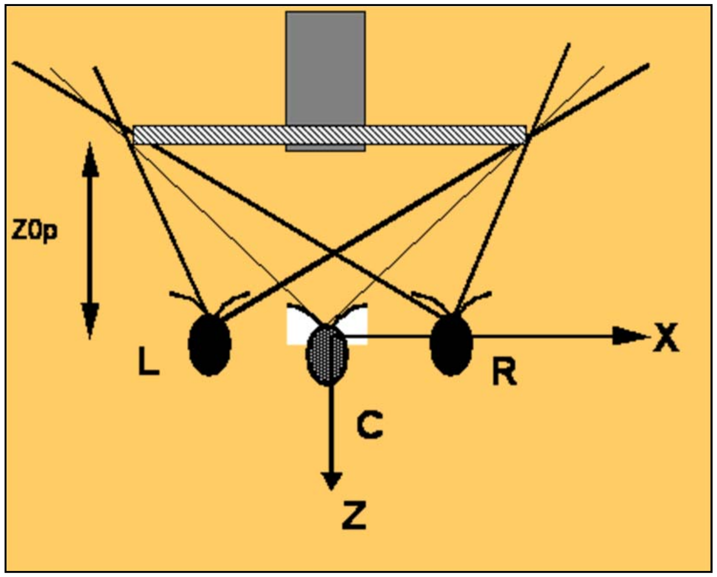
Note that this is a *situation*, not a *problem*. The difference in the left and right eye views requires at least *some* horizontal parallax to work. You can convince yourself of this by alternately opening and closing your left and right eyes. We just need a good way to *control* the horizontal parallax.

Two Side-by-side Perspective Viewing Volumes

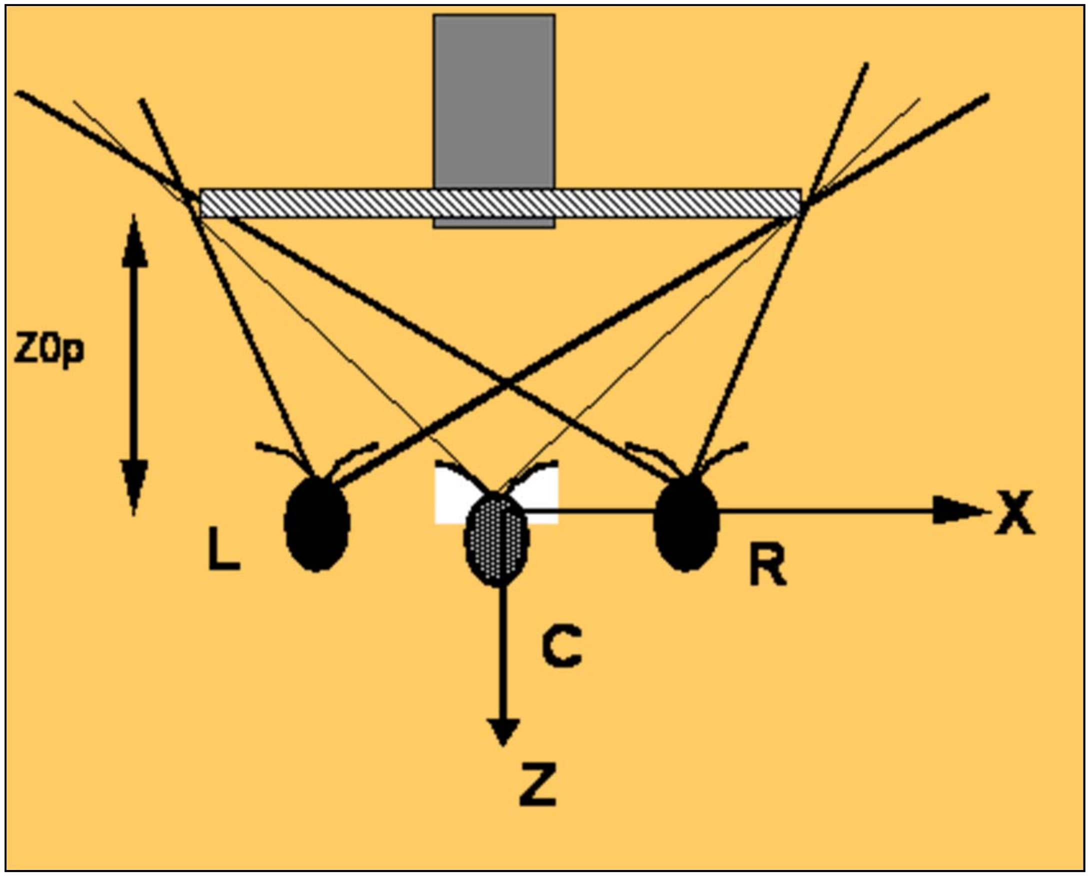
We do this by defining a distance in front of the eye, z0p, to the *plane of zero parallax*, where a 3D point projects to the same window location for each eye. To the viewer, the plane of zero parallax will be the glass screen and objects in front of it will appear to live in the air in front of the glass screen and objects behind this plane will appear to live inside the monitor. The plane of zero parallax is handled by:

1. Set the distance from the eyes to the plane of zero parallax based on the location of the geometry and the look you are trying to achieve.

2. Looking from the Cyclops eye at the origin, determine the left, right, bottom, and top boundaries of the viewing window on the plane of zero parallax as would be used in a call to glFrustum( ). These can be determined by knowing Z0p and the field-of-view angle Φ:



Two Side-by-side *Non-symmetric* Perspective Viewing Volumes

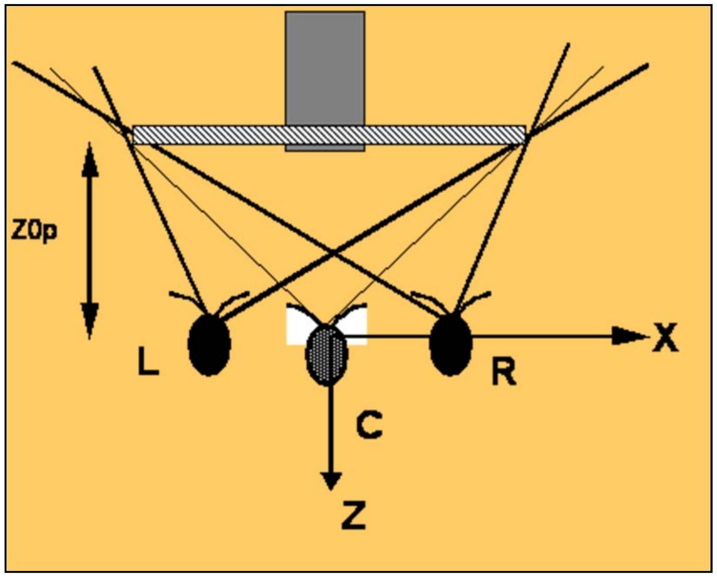


Cyclops eye:

*L0p = -Z0p * tan(ø/2)*
*R0p = Z0p * tan(ø/2)*
*B0p = -Z0p * tan(ø/2)*
*T0p = Z0p * tan(ø/2)*

Two Side-by-side *Non-symmetric* Perspective Viewing Volumes

Use the Cyclops's left and right boundaries as the left and right boundaries for each eye, even though the scene has been translated. In the left eye view, the boundaries must then be shifted by +E to match the +E shift in the scene. In the right eye view, the boundaries must be shifted by -E to match the -E shift in the scene.



Left eye:

*R0p = Z0p * tan(ø/2) + E*

*L0p = -Z0p * tan(ø/2) + E*

Right eye:

*R0p = Z0p * tan(ø/2) - E*

*L0p = -Z0p * tan(ø/2) - E*

Stereo Perspective

```
float
Tand( float deg )
{
	float rad = deg * (float)M_PI / 180.f;
	return (float)tan( rad );
}
```

```
void
FrustumZ( float left, float right, float bottom, float top, float znear, float zfar,   float zproj )
{
	if( zproj != 0.0 )
	{
		left	*= ( znear/zproj );
		right	*= ( znear/zproj );
		bottom *= ( znear/zproj );
		top	*= ( znear/zproj );
	}

	glFrustum( left, right, bottom, top, znear, zfar );
}
```

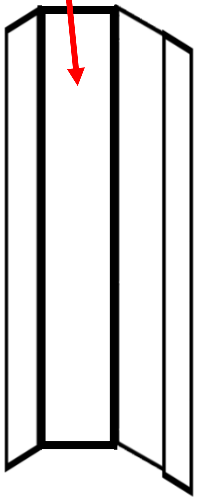
Stereo Perspective

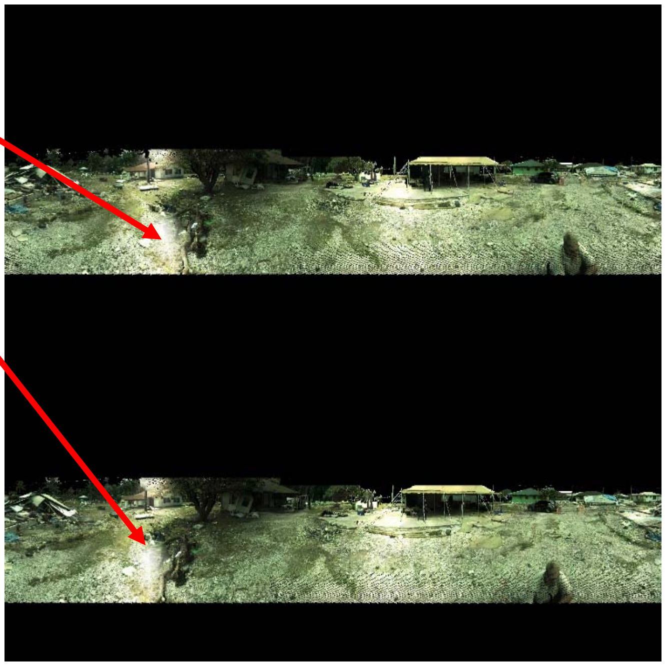
```
void
StereoPersp( float fovxdeg, float aspect_y_over_x, float znear, float zfar, float z0p, float eye )
{
        float tanfovx = Tand( fovxdeg / 2.f );

        float right = z0p * tanfovx;
        float left   = -right;

        float bottom = aspect_y_over_x * left;
        float top       = aspect_y_over_x * right;

        left   = left   - eye;
        right = right - eye;

        FrustumZ( left, right, bottom, top, znear, zfar, z0p );

        glTranslatef( -eye, 0.0, 0.0 );
}
```

Spherical Stereo Strategy (S^3?)

Strategy:

1. Leave the eye in the center of the 3D scene
2. Rotate the look-at position 360° in a series of discrete steps
3. For each look-at position, render two stereo views, each essentially a very narrow **vertical strip** of pixels
4. Upload the pixels from each render and collect them in a single image as left- and right-eye panoramic views





Program Setup

```
const int PHIDEG                    =    5;
const int NUMSEGS                   = 360 / PHIDEG;
const int PIXELS_PER_SEG            =   20;
const int WIDTH                     =   NUMSEGS * PIXELS_PER_SEG;
const int HEIGHT                    =  WIDTH / 2;
const float ASPECT_Y_OVER_X         =  (float)HEIGHT / (float)PIXELS_PER_SEG;
const float Z0P                     =  100.f;
const float ZNEAR                   =      1.0f;
const float ZFAR                    =  200.0f;
const float EYESEP                  =        0.25f;
const float EX                      =       0.f;
const float EY                      =       0.f;
const float EZ                      =       0.f;

unsigned char    LeftRight[  3*2*WIDTH*HEIGHT  ];        // 3 = color components, 2 = L+R images
```

Program Setup

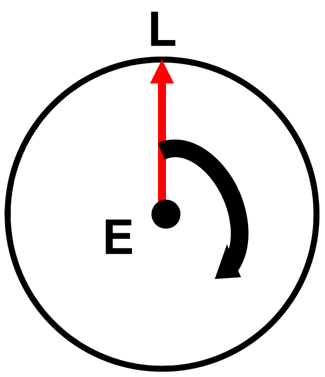
```
void
InitGraphics( )
{
        glutInitDisplayMode( GLUT_RGBA | GLUT_SINGLE | GLUT_DEPTH );
        glutInitWindowPosition( 0, 0 );
        glutInitWindowSize( PIXELS_PER_SEG, HEIGHT );

        MainWindow = glutCreateWindow( WINDOWTITLE );
        glutSetWindowTitle( WINDOWTITLE );

        glClearColor( BACKCOLOR[0], BACKCOLOR[1], BACKCOLOR[2], BACKCOLOR[3] );
        glutDisplayFunc( Display );

#ifdef WIN32
        GLenum err = glewInit( );
        if( err != GLEW_OK )
        {
                fprintf( stderr, "glewInit Error\n" );
        }
        else
        {
                fprintf( stderr, "GLEW initialized OK\n" );
        }
        fprintf( stderr, "Status: Using GLEW %s\n", glewGetString(GLEW_VERSION));
#endif
}
```

Drawing the Scene in Strips, I

```
void
DrawAndWriteSegments( )
{
        unsigned char array[3*PIXELS_PER_SEG*HEIGHT];

        glutSetWindow( MainWindow );
        glDrawBuffer( GL_FRONT );
        glEnable( GL_DEPTH_TEST );
        glShadeModel( GL_FLAT );
        glViewport( 0, 0, PIXELS_PER_SEG, HEIGHT );

        for( int eye = 0; eye <= 1; eye++ )
        {
```

Drawing the Scene in Strips, II

```
for( int eye = 0; eye <= 1; eye++ )
{
        glMatrixMode( GL_PROJECTION );
        glLoadIdentity( );
        StereoPersp( (float)PHIDEG, ASPECT_Y_OVER_X, ZNEAR, ZFAR, Z0P,
                    eye == 0 ? -EYESEP : EYESEP );

         // left goes on the top -- right goes on the bottom
         unsigned char *FullArray = ( eye == 1 ? &LeftRight[0] : &LeftRight[3*WIDTH*HEIGHT] );

         int col = 0;            // column in the full array
         for( int lookDeg = 90; lookDeg > -270; lookDeg -= PHIDEG )
         {
                glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
                glMatrixMode( GL_MODELVIEW );
                glLoadIdentity( );

                float lx = Cosd( (float)lookDeg )  +  EX;
                float ly = Sind(  (float)lookDeg )  +  EY;
                float lz = EZ;
                gluLookAt( EX, EY, EZ,  lx, ly, lz,  0., 0., 1. );

                glCallList( LidarList );
                glFlush( );
                glFinish( );
```

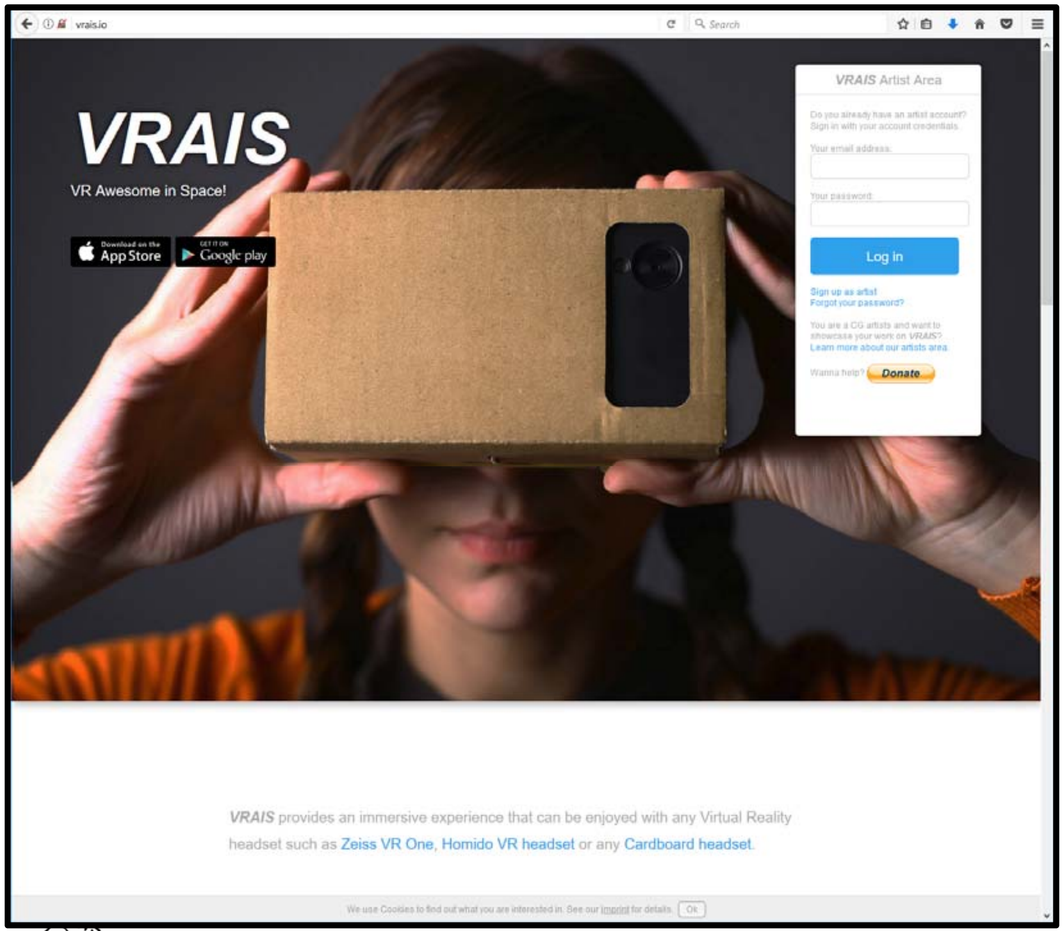


Drawing the Scene in Strips, III

```
                glPixelStorei( GL_PACK_ALIGNMENT, 1 );
                glReadPixels( 0, 0, PIXELS_PER_SEG, HEIGHT, GL_RGB, GL_UNSIGNED_BYTE, array );

                for( int y = 0; y < HEIGHT; y++ )
                {
                        memcpy( &FullArray[3*y*WIDTH + 3*col],
                                &array[3*y*PIXELS_PER_SEG + 0], 3*PIXELS_PER_SEG );

                        //for (int x = 0; x < PIXELS_PER_SEG; x++)
                        //{
                              //FullArray[3*y*WIDTH + 3*(col+x) + 0] = array[3*y*PIXELS_PER_SEG + 3*x + 0];
                              //FullArray[3*y*WIDTH + 3*(col+x) + 1] = array[3*y*PIXELS_PER_SEG + 3*x + 1];
                              //FullArray[3*y*WIDTH + 3*(col+x) + 2] = array[3*y*PIXELS_PER_SEG + 3*x + 2];
                        //}
                  }

                  col += PIXELS_PER_SEG;

            }   // lookDeg

      }   // eye

      WriteArray( (char *)"Lidar.bmp", LeftRight);
}
```

Displaying on your Phone

Go to: http://vrais.io



vrais.io

VRAIS

VR Awesome in Space!

VRAIS Artist Area

Do you already have an artist account? Sign in with your account credentials.

Your email address:

Your password:

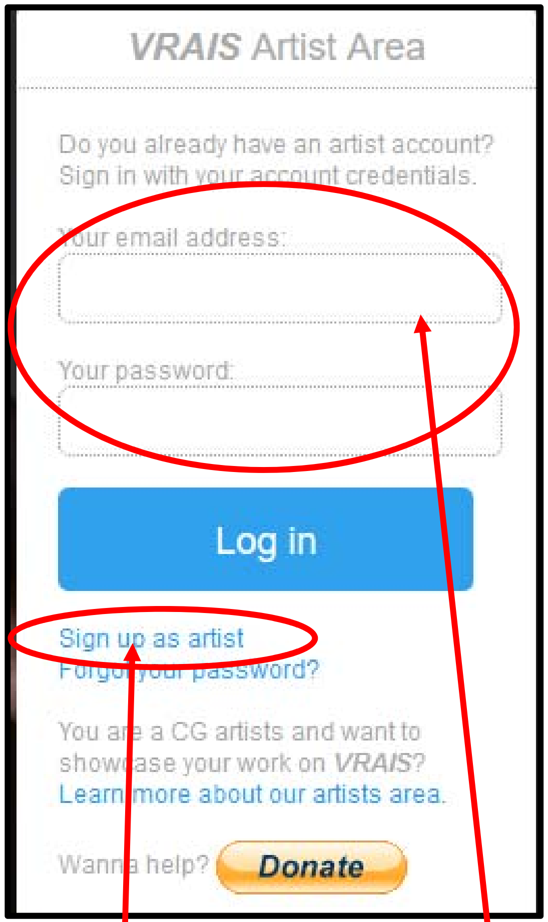
Sign up as artist
Forgot your password?

You are a CG artists and want to showcase your work on *VRAIS*? Learn more about our artists area.

Wanna help?

VRAIS provides an immersive experience that can be enjoyed with any Virtual Reality headset such as Zeiss VR One, Homido VR headset or any Cardboard headset.

We use Cookies to find out what you are interested in. See our imprint for details.



VRAIS Artist Area

Do you already have an artist account? Sign in with your account credentials.

Your email address:

Your password:

Sign up as artist
Forgot your password?

You are a CG artists and want to showcase your work on *VRAIS*? Learn more about our artists area.

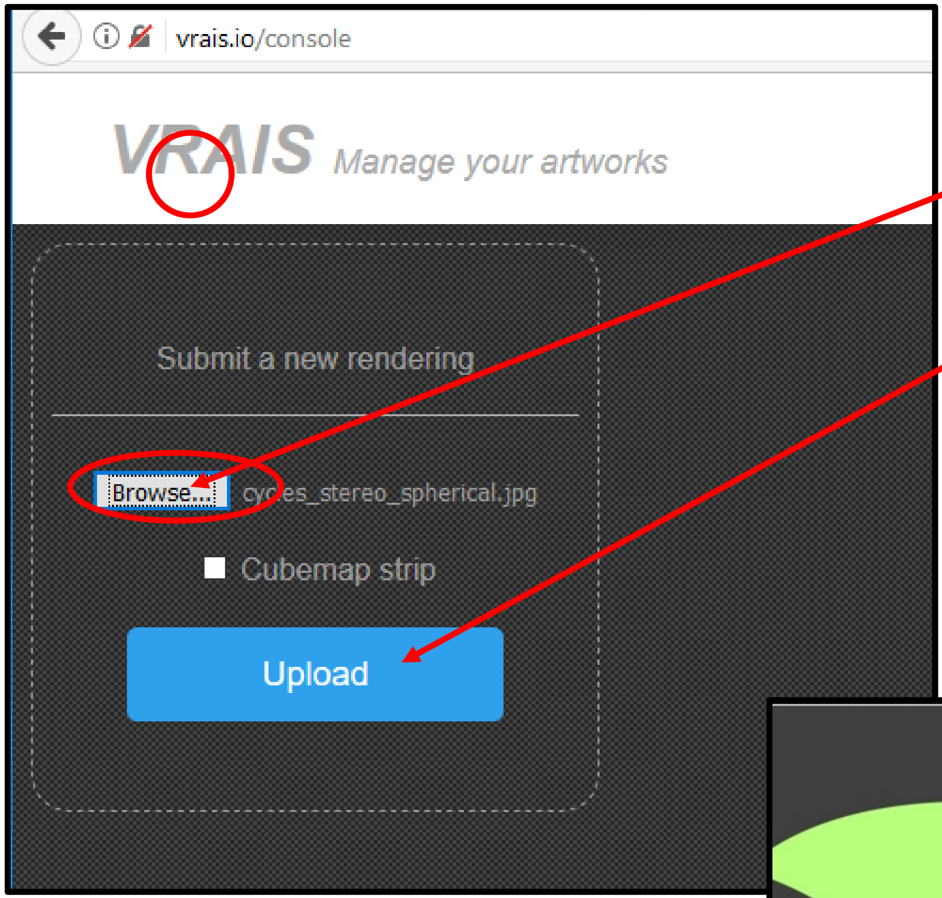
Wanna help?

VRAIS stands for:

- VR – Awesome In Space
- It's also the French word for "true"

If you've already registered, sign in here.
If not, sign up here.

Displaying on your Phone



vrais.io/console

VRAIS *Manage your artworks*

Submit a new rendering

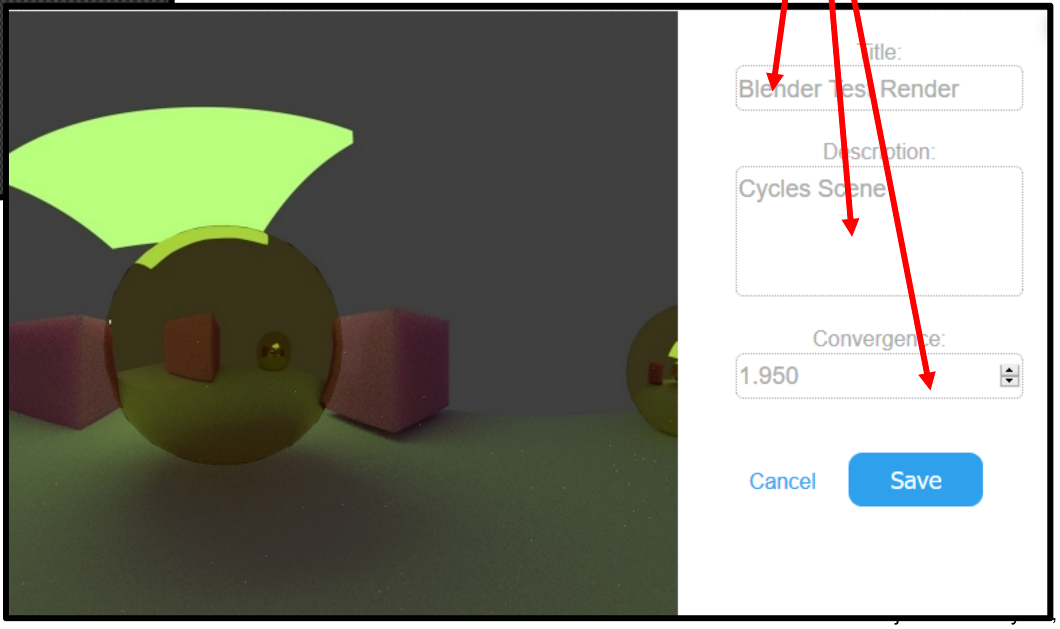
Browse... cycles_stereo_spherical.jpg

Cubemap strip

Upload

Click here and **Browse** to your JPEG file.
Click **Upload**.

Supply the **Title**, **Description**, and **Convergence** distance (Z0p).



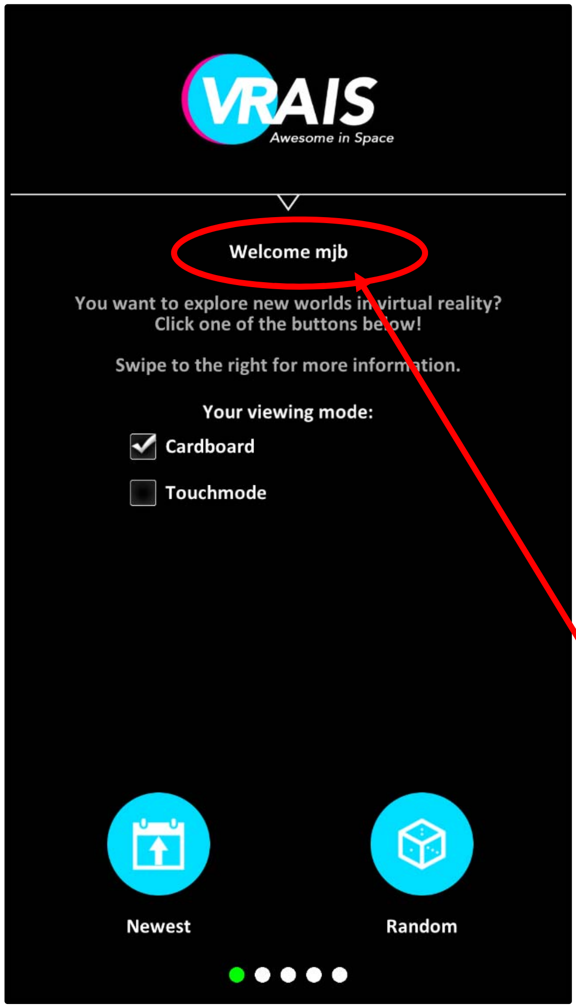
Title: Blender Test Render

Description: Cycles Scene

Convergence: 1.950

Oregon State University
Computer Graphics

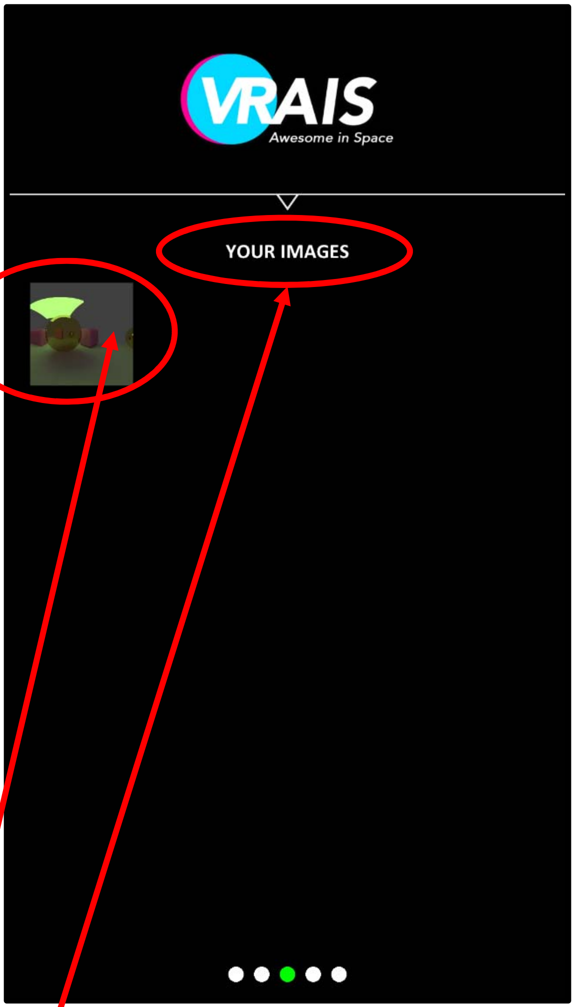
Displaying on your Phone


Welcome mjb

You want to explore new worlds in virtual reality?
Click one of the buttons below!

Swipe to the right for more information.

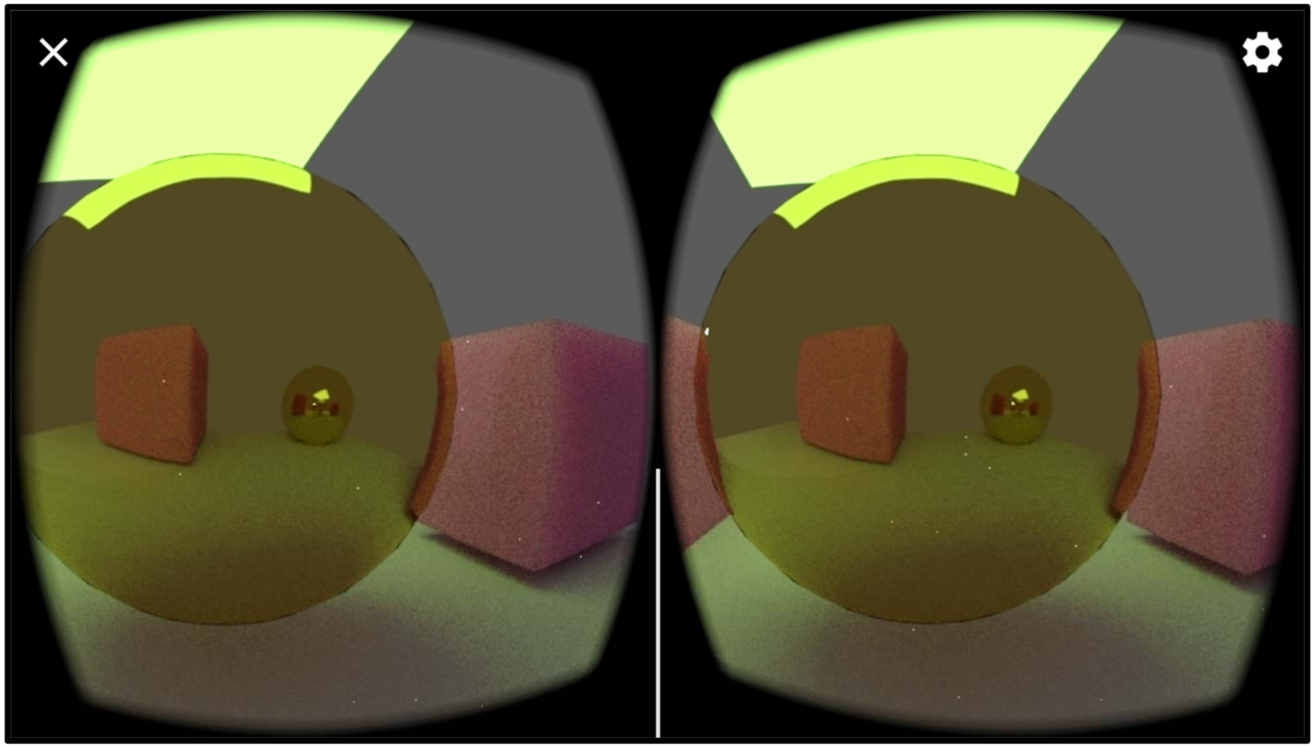
Your viewing mode:
- [✓] Cardboard
- [ ] Touchmode



The VRAIS app exists for both Android and iOS. Load it on your phone. Run it and login with the same information you registered on the VRAIS web site.

You see this screen.
Swipe left a couple of times until you see this screen.
Click on the image you want to load.

Displaying on your Phone



You get this stereopair. If you rotate your phone, you see that the view changes to look in that direction.

If you have a headset, plug your phone into it.

Voila! That's French for "voila". ☺

Getting your own Headset

Go to https://www.amazon.com/Mattel-DTH61-View-Master-Deluxe-Viewer/dp/B01CNSO79Q/ref=sr_1_6?ie=UTF8&qid=1519763407&sr=8-6&keywords=view-master

Or, go to http://amazon.com and enter: **View-Master**

This is the Mattel **View-Master Deluxe VR Viewer.**
It sells for under $25. Mine is an earlier model of this one, and I am very happy with it. I trust View-Master to get the mechanical design and the optics right. They've been doing this for years.

But, really, anything that claims to be compatible with **Google Cardboard** should work.