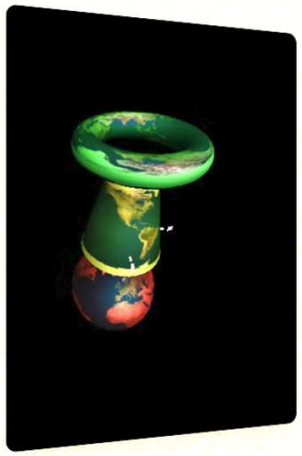
# Looking Glass Quilts for a Web Page- Embedded 3D Display, Using both OpenGL and Blender





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

# Here is How to Generate a Quilt from OpenGL



Here are some pre-defined constants you will need:

```
const int QUILTWIDTH  = 3360;
const int QUILTHEIGHT = 3360;
const int QUILTNUMWIDTH  = 8;
const int QUILTNUMHEIGHT = 6;
const int QUILTTOTALBLOCKS = QUILTNUMWIDTH * QUILTNUMHEIGHT;
const int BLOCKWIDTH  = QUILTWIDTH  / QUILTNUMWIDTH;                    // 420
const int BLOCKHEIGHT = QUILTHEIGHT / QUILTNUMHEIGHT;                   // 560
const float QUILTZ0P = 5.f;
const float QUILTDELTAEYE = 0.10f;
const float QUILT_ASPECT_Y_OVER_X = (float)BLOCKHEIGHT / (float)BLOCKWIDTH; // 4:3 = 1.3333

const char * QUILTFILENAME = "Quilt_qs8x6a0.75.bmp";        // must be in the format "*_qs8x6a0.75.*"

unsigned char QuiltArray[QUILTHEIGHT][QUILTWIDTH][3];                   // holds the entire quilt
unsigned char BlockArray[BLOCKHEIGHT][BLOCKWIDTH][3];                   // holds one block
```

# Triggering the File Output



```
void
Keyboard( unsigned char c, int x, int y )
{
        switch( c )
        {
                case 'w':
                case 'W':
                        CreateAndWriteQuilt( );
                        break;

                 .  .  .

                case 'q':
                case 'Q':
                case ESCAPE:
                        DoMainMenu( QUIT );     // will not return here
                        break;                          // happy compiler

                default:
                        fprintf( stderr, "Don't know what to do with keyboard hit: '%c' (0x%0x)\n", c, c );
        }

        // force a call to Display( ):

        glutSetWindow( MainWindow );
        glutPostRedisplay( );
}
```

Hit the 'w' key to trigger the 48 renderings and file-writing

# Generating the 48 Individual Images for the Quilt, I



```
void
CreateAndWriteQuilt( )
{
	glutSetWindow( MainWindow );

	float eyex = - QUILTDELTAEYE*(float)(QUILTTOTALBLOCKS)/2.f;
	for (int y = 0; y < QUILTNUMHEIGHT; y++ )
	{
		for( int x = 0; x < QUILTNUMWIDTH; x++ )
		{
			glDrawBuffer(GL_FRONT);
			glEnable(GL_DEPTH_TEST);
			glShadeModel(GL_FLAT);
			int vx = glutGet(GLUT_WINDOW_WIDTH);
			int vy = glutGet(GLUT_WINDOW_HEIGHT);
			int xl = (vx - BLOCKWIDTH) / 2;
			int yb = (vy - BLOCKHEIGHT) / 2;
			glViewport(xl, yb, BLOCKWIDTH, BLOCKHEIGHT);
			glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

			glMatrixMode( GL_PROJECTION );
			glLoadIdentity( );
			OffAxisPersp( 70.f, QUILT_ASPECT_Y_OVER_X, 0.1f, 1000.f, QUILTZ0P, eyex );

			glMatrixMode(GL_MODELVIEW);
			glLoadIdentity( );
			gluLookAt(0.f, 0.f, 8.f, 0.f, 0.f, 0.f, 0.f, 1.f, 0.f);

			DisplayScene( );

			glFlush();
			glFinish();

			. . .
```

Oregon State University
Computer Graphics

# Generating the 48 Individual Images for the Quilt, II



```
                        . . .

                        // done rendering -- upload the pixels:

                        glReadBuffer(GL_FRONT);
                        glPixelStorei(GL_PACK_ALIGNMENT, 1);
                        glReadPixels(xl, yb, BLOCKWIDTH, BLOCKHEIGHT, GL_RGB, GL_UNSIGNED_BYTE, BlockArray);

                        //  location for this block in the large QuiltArray:

                        for( int yb = 0; yb < BLOCKHEIGHT; yb++)
                        {
                                for( int xb = 0; xb < BLOCKWIDTH; xb++)
                                {
                                        QuiltArray[y*BLOCKHEIGHT+yb][x*BLOCKWIDTH+xb][0] = BlockArray[yb][xb][0];
                                        QuiltArray[y*BLOCKHEIGHT+yb][x*BLOCKWIDTH+xb][1] = BlockArray[yb][xb][1];
                                        QuiltArray[y*BLOCKHEIGHT+yb][x*BLOCKWIDTH+xb][2] = BlockArray[yb][xb][2];
                                }
                        }

                        Sleep(250);

                        eyex += QUILTDELTAEYE;
                }  // x
        }  // y

        WriteArray( QUILTFILENAME );
}
```

# DisplayScene( ) – just do the scene drawing, I



```
void
DisplayScene()
{
	// rotate the scene:

	glRotatef((GLfloat)Yrot, 0.f, 1.f, 0.f);
	glRotatef((GLfloat)Xrot, 1.f, 0.f, 0.f);

	// uniformly scale the scene:

	if (Scale < MINSCALE)
		Scale = MINSCALE;
	glScalef((GLfloat)Scale, (GLfloat)Scale, (GLfloat)Scale);

	// possibly draw the axes:

	if (AxesOn != 0)
	{
		glColor3fv(&Colors[WhichColor][0]);
		glCallList(AxesList);
	}

	// since we are using glScalef( ), be sure the normals get unitized:

	glEnable(GL_NORMALIZE);
	. . .
```

# DisplayScene( ) – just do the scene drawing, II



```
  . . .

glShadeModel(GL_FLAT);
if (SmoothOn)
{
        glShadeModel(GL_SMOOTH);
}

glDisable(GL_LIGHTING);
glTexEnvf(GL_TEXTURE_ENV, GL_TEXTURE_ENV_MODE, GL_REPLACE);
if (LightingOn)
{
        glEnable(GL_LIGHTING);
        glEnable(GL_LIGHT0);
        glTexEnvf(GL_TEXTURE_ENV, GL_TEXTURE_ENV_MODE, GL_MODULATE);
}

glBindTexture(GL_TEXTURE_2D, Tex);
glDisable(GL_TEXTURE_2D);
if (TextureOn)
{
        glEnable(GL_TEXTURE_2D);
}

  . . .
```

# DisplayScene( ) – just do the scene drawing, III



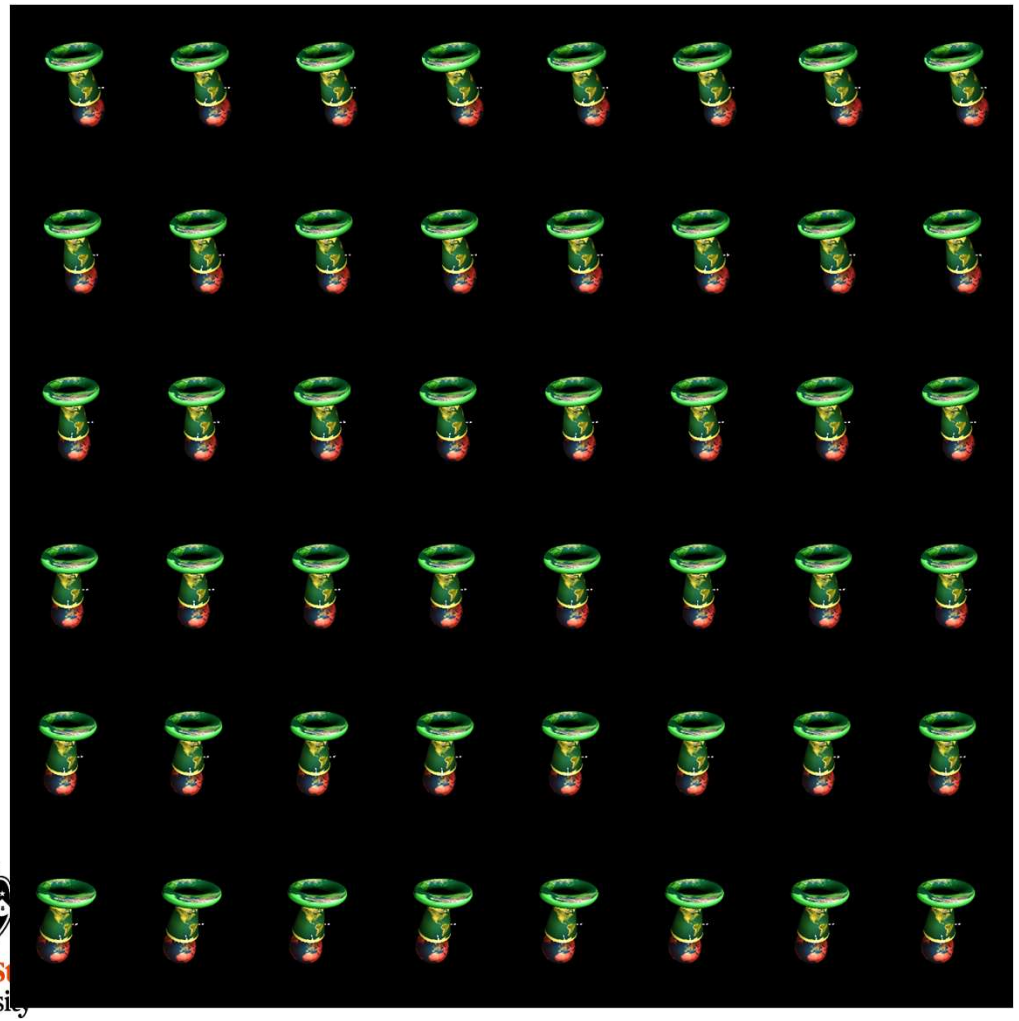
```
        . . .

        // draw the objects by calling up their display lists:

        glTranslatef(0.f, -2.25f, 0.);
        glColor3f(0.8f, 0.2f, 0.2f);
        SetMaterial(0.8f, 0.2f, 0.2f, 10.f);
        glCallList(SphereDL);

        glTranslatef(0.f, 1.75f, 0.);
        glColor3f(0.8f, 0.8f, 0.2f);
        SetMaterial(0.8f, 0.8f, 0.2f, 8.f);
        glCallList(ConeDL);

        glTranslatef(0.f, 2.5f, 0.);
        glColor3f(0.2f, 0.8f, 0.2f);
        SetMaterial(0.2f, 0.8f, 0.2f, 6.f);
        glCallList(TorusDL);

        glDisable(GL_LIGHTING);
        glDisable(GL_TEXTURE_2D);
        glShadeModel(GL_FLAT);
}
```

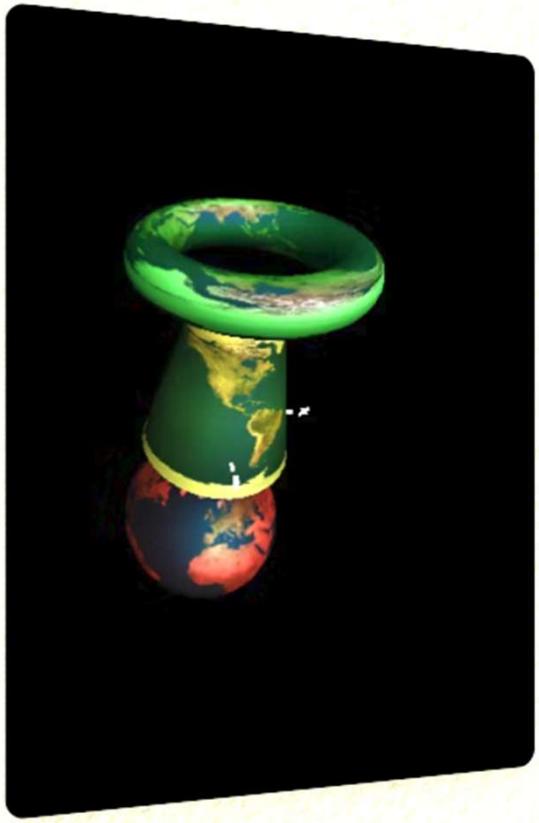
# Doing the Off-Axis Projection



```
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 );
}

void
OffAxisPersp( float fovxdeg, float aspect_y_over_x, float znear, float zfar, float z0p, float eyex )
{
            float tanfovx = tanf( (float)(M_PI/180.) * 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  -= eyex;
            right -= eyex;

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

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

# Quilt_qs8x6a0.75.bmp





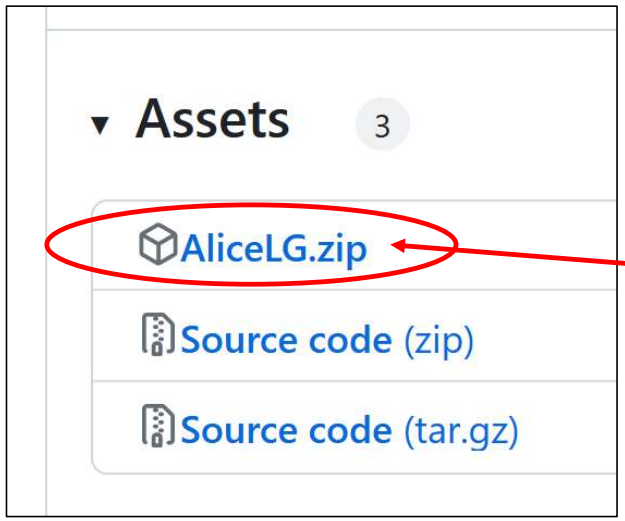
## How the usual Display( ) function calls DisplayScene( )

```
void
Display( )
{
	glutSetWindow( MainWindow );
	glDrawBuffer( GL_BACK );
	glClear( GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT );
	glEnable( GL_DEPTH_TEST );
	glShadeModel( GL_FLAT );

	// set the viewport to a square centered in the window:
	GLsizei vx = glutGet( GLUT_WINDOW_WIDTH );
	GLsizei vy = glutGet( GLUT_WINDOW_HEIGHT );
	GLsizei v = vx < vy ? vx : vy;                  // minimum dimension
	GLint xl = ( vx - v ) / 2;
	GLint yb = ( vy - v ) / 2;
	glViewport( xl, yb,  v, v );

	// set the viewing volume:
	glMatrixMode( GL_PROJECTION );
	glLoadIdentity( );
	if( WhichProjection == ORTHO )
		glOrtho( -2.f, 2.f,     -2.f, 2.f,     0.1f, 1000.f );
	else
		gluPerspective( 70.f, 1.f,      0.1f, 1000.f );

	// place the objects into the scene:
	glMatrixMode( GL_MODELVIEW );
	glLoadIdentity( );
	gluLookAt( 0.f, 0.f, 7.f,     0.f, 0.f, 0.f,     0.f, 1.f, 0.f );

	// draw just the scene:
	DisplayScene( );

	glutSwapBuffers( );
	glFlush( );
}
```

# Here is How to Generate a Quilt from Blender

You can also generate these quilts automatically from Blender.

The first thing you need to do is retrieve a Blender Add-on.
The Add-on's name is **AliceLG.zip** and it can be found at https://github.com/regcs/AliceLG/releases

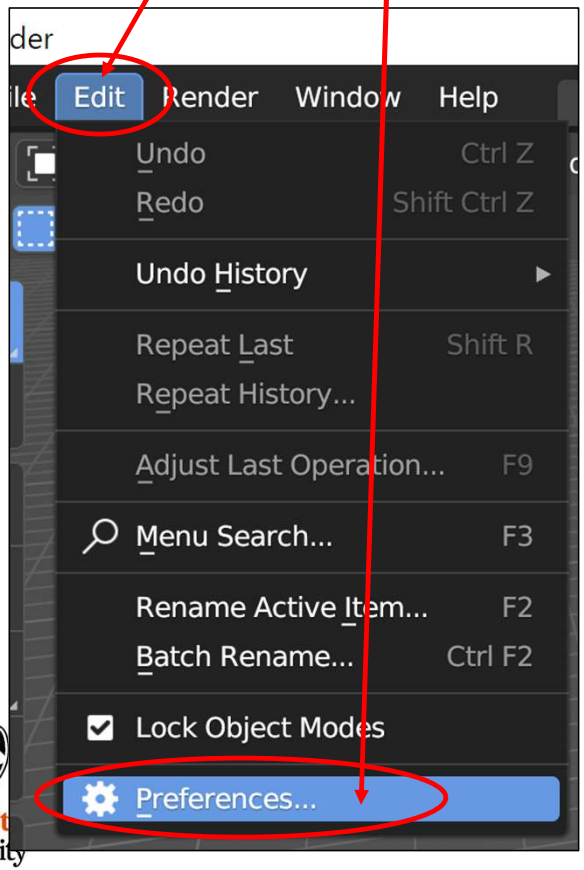


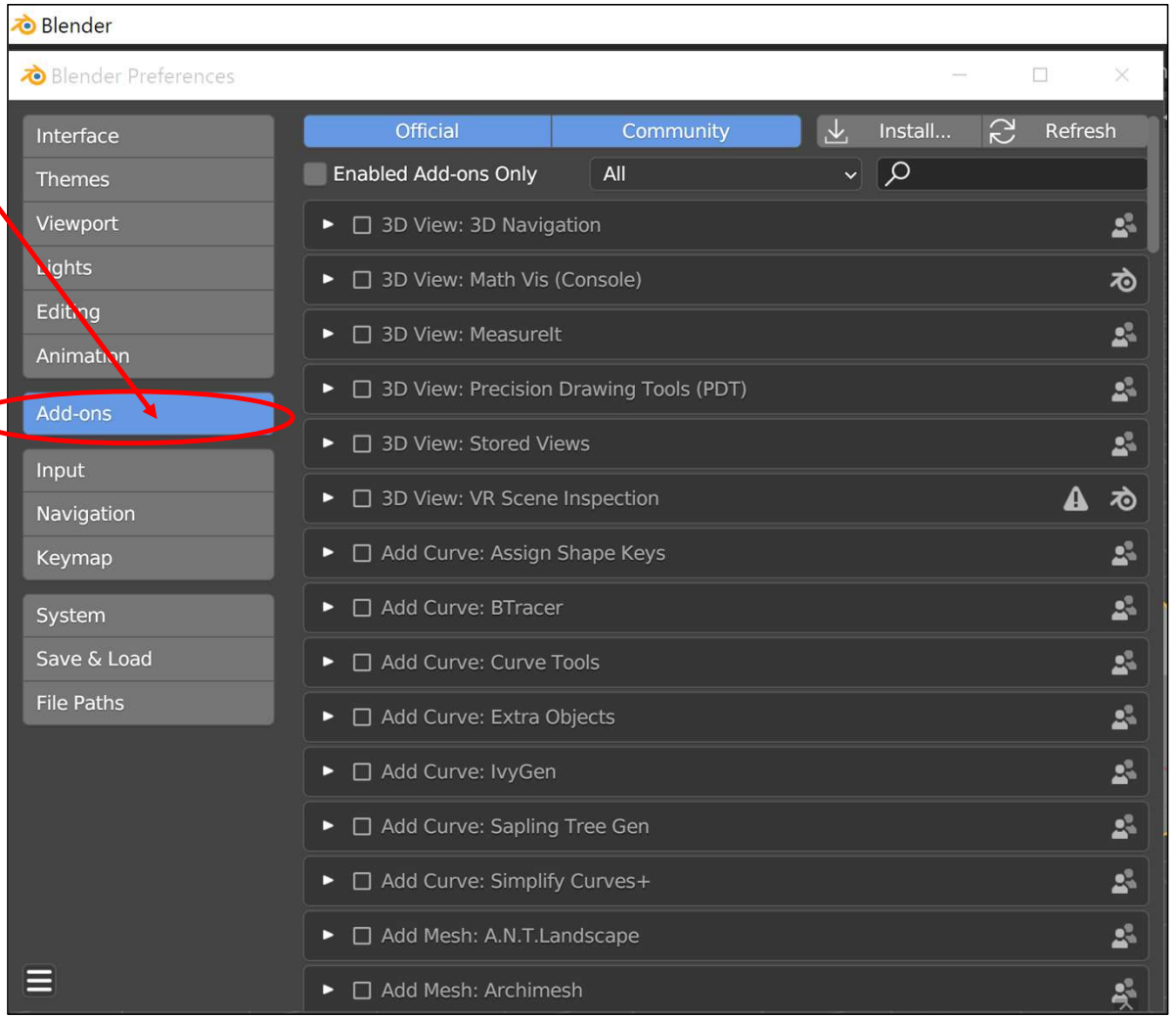

Click here and save this file anywhere.
Do not un-zip it!

# Getting the AliceLG.zip Blender Add-on, I

Run Blender.

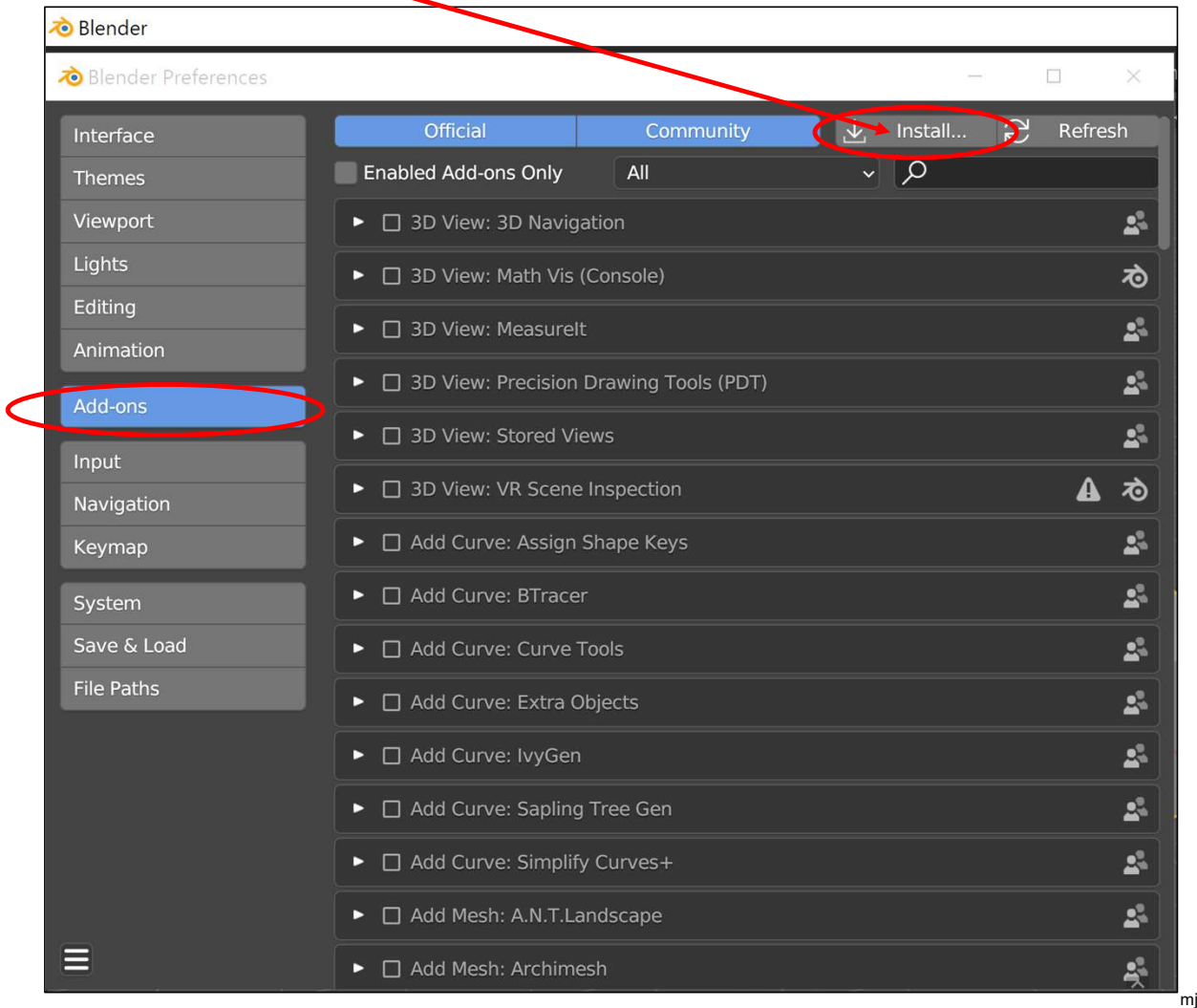
Click on **Edit→Preferences →Add-ons.**



# Getting the AliceLG.zip Blender Add-on, II

Click on **Install**



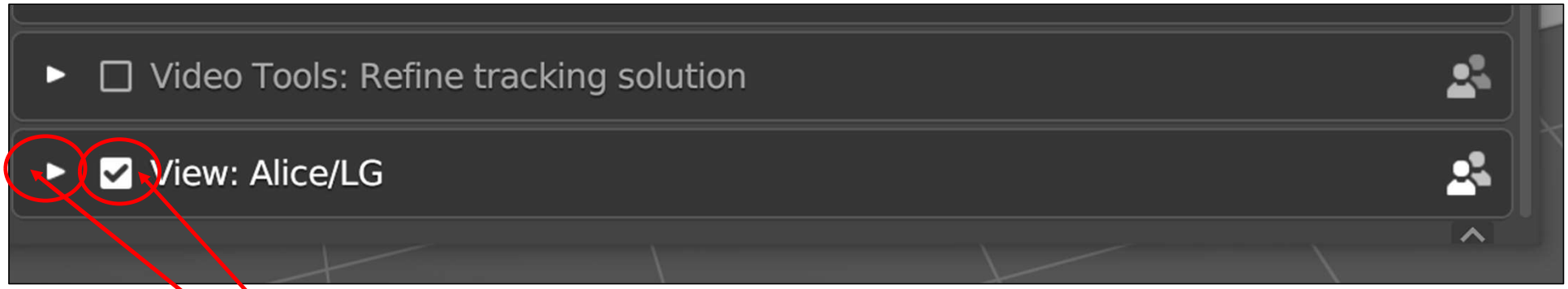

Then navigate to where you installed **AliceLG.zip** and select it.

Don't un-zip it!

# Getting the AliceLG.zip Blender Add-on, III



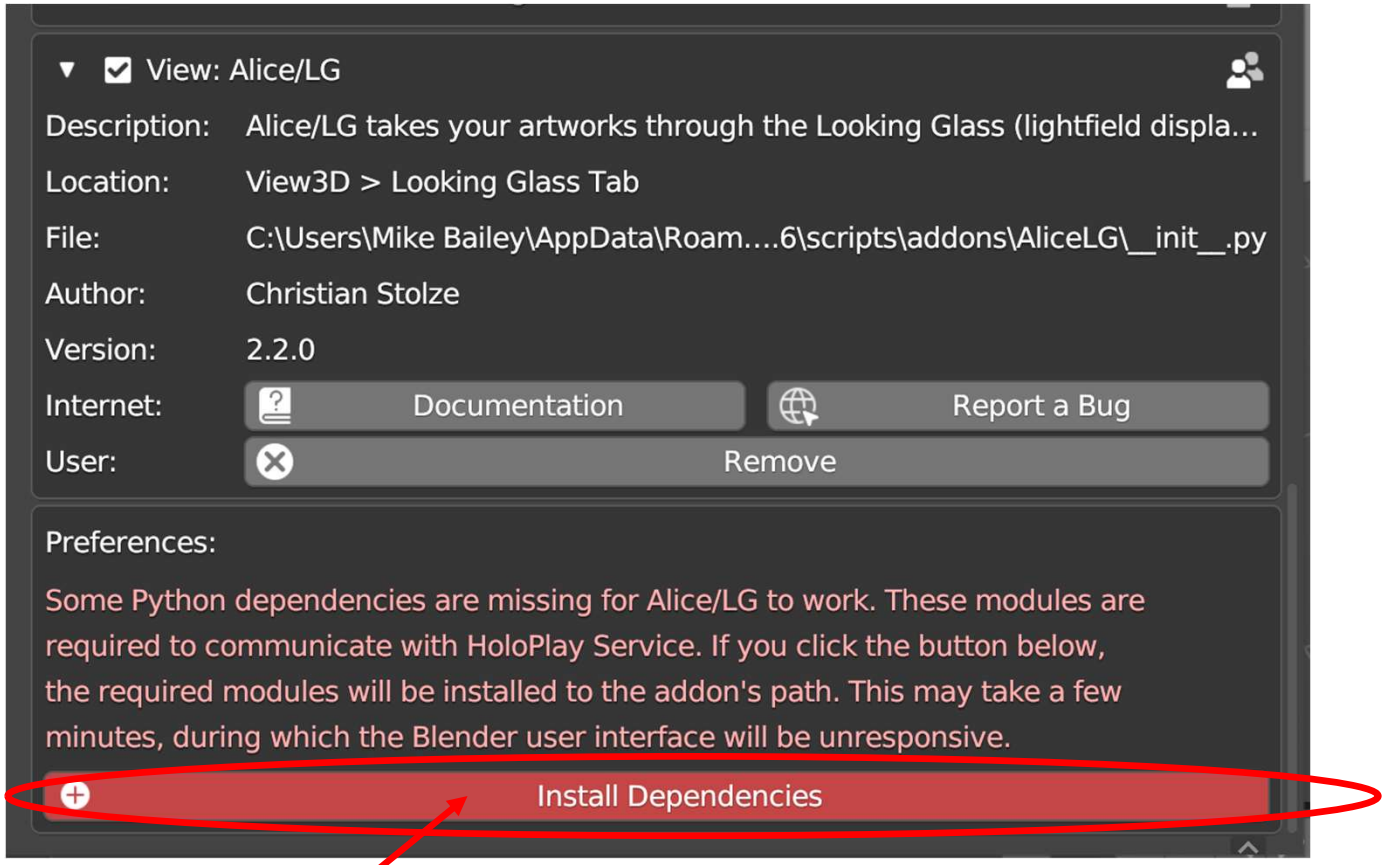
Scroll down the list of Add-ons until you see **View: Alice/LG**




Click the **checkbox** to enable it.
Then click the **arrow** and scroll down so you can see all the information.

# Getting the AliceLG.zip Blender Add-on, IV

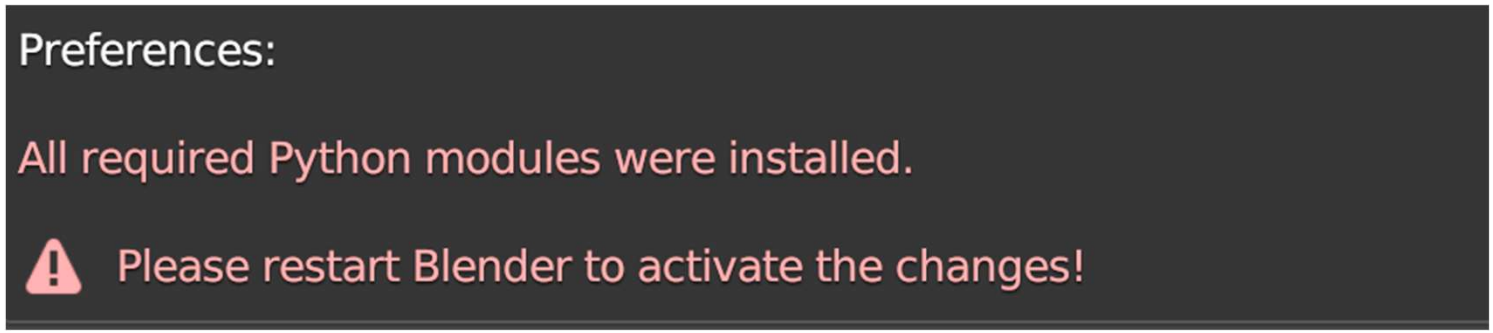
**Scroll** down so you can see all the information.




Then click on **Install Dependencies**. This might take a while. Be patient.

# Getting the AliceLG.zip Blender Add-on, V

When it's done, you will see this:

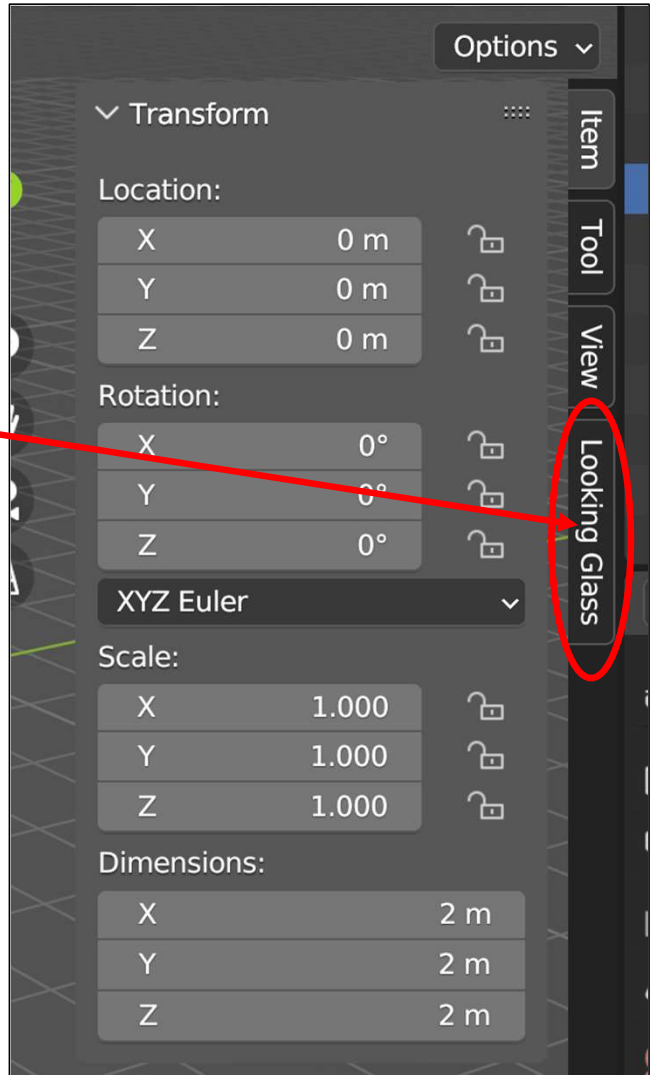



Now exit Blender and start it up again.

# Finding the Looking Glass Tab in Blender

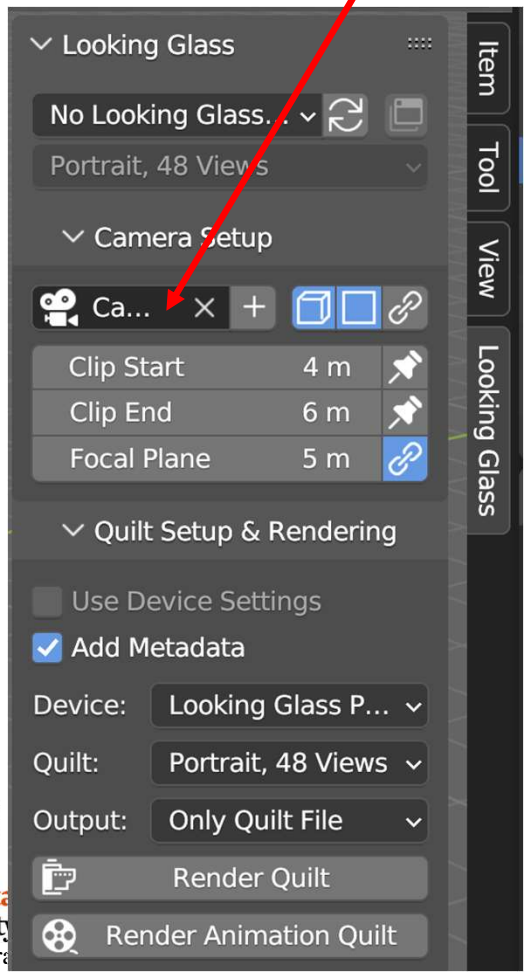
Hit the **'n'** key and you will see the usual Transform panel, but something new has been added.

Click on the **Looking Glass tab**.

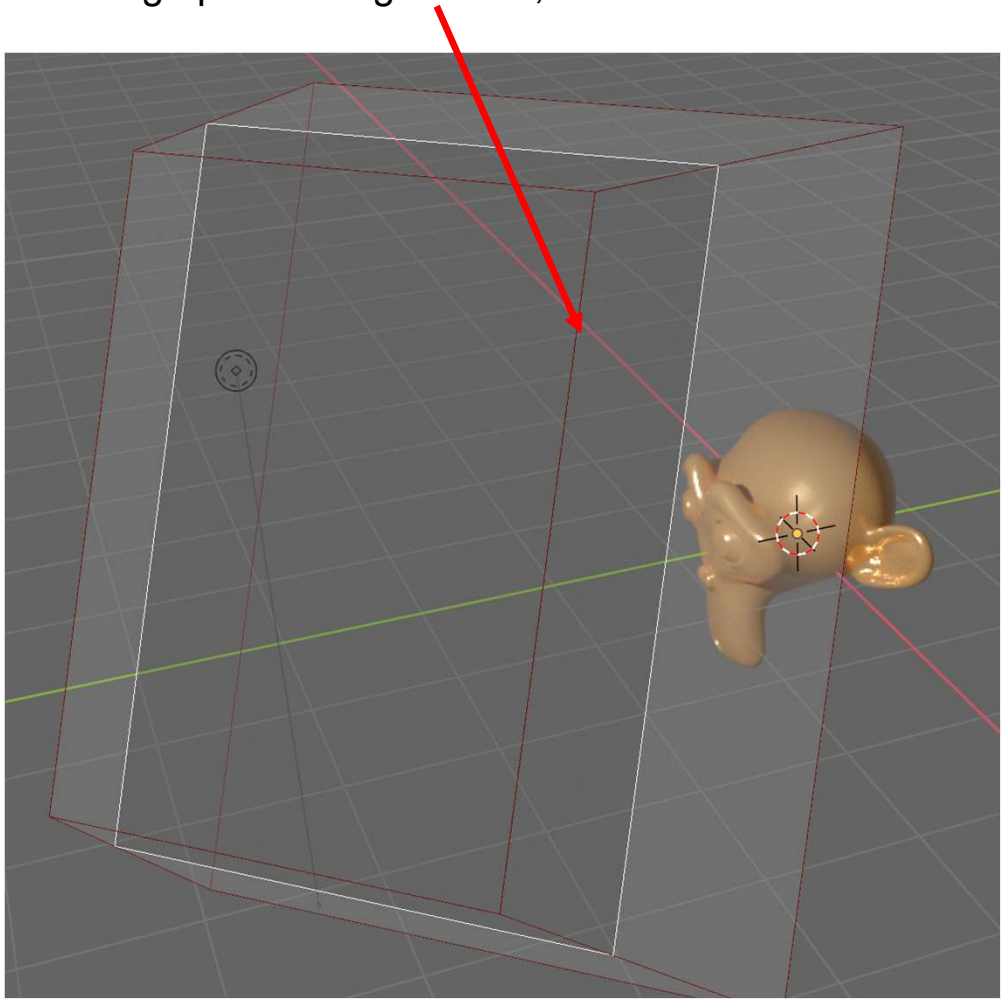


# Using the Looking Glass Tab, I

Click here and select **Camera**

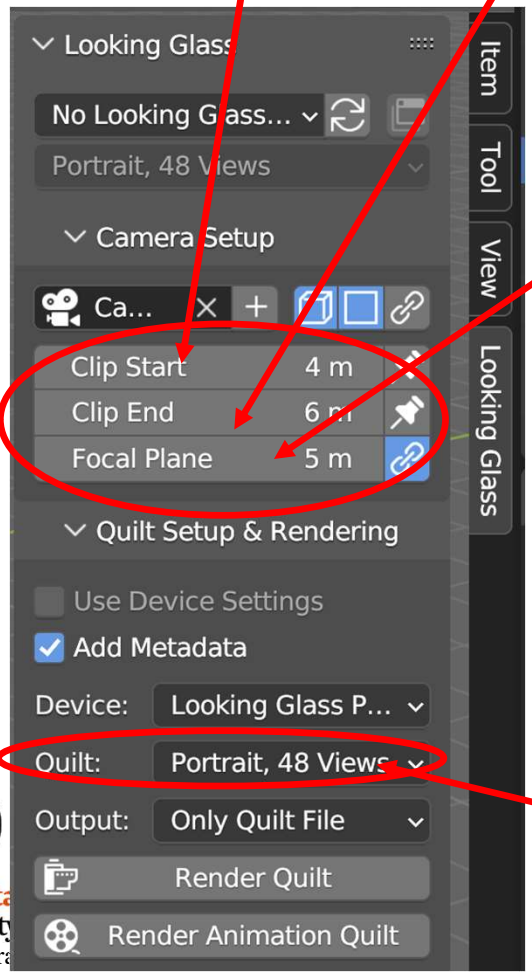



This will bring up a viewing volume, like this:

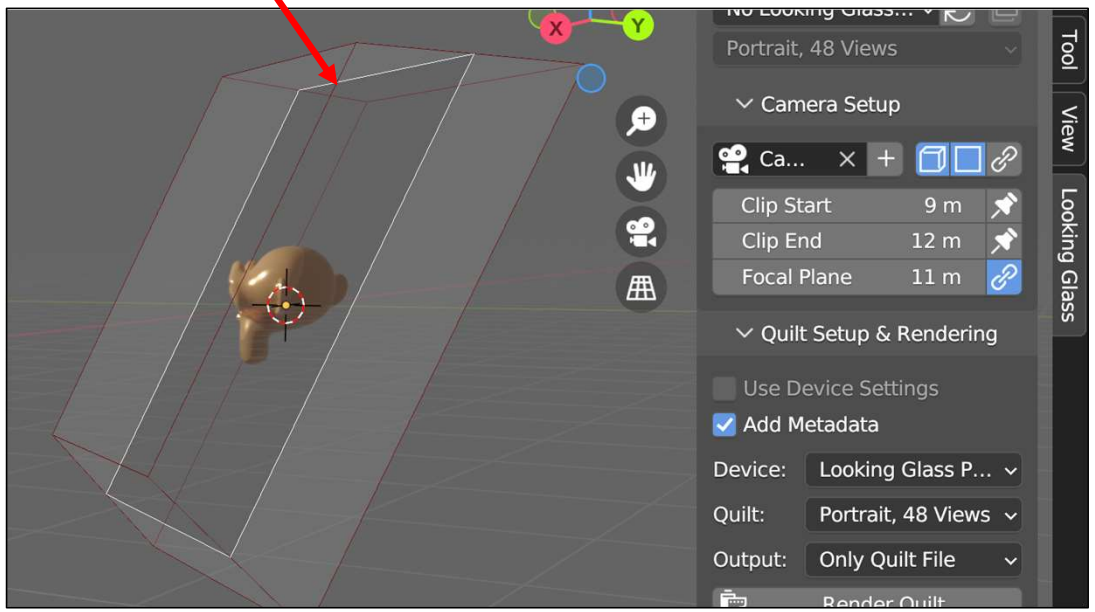


# Using the Looking Glass Tab, II

Now adjust the **Clip Start** and **Clip End** to completely surround your scene.




Then adjust the **Focal Plane** to be roughly down the middle of the scene, like this:

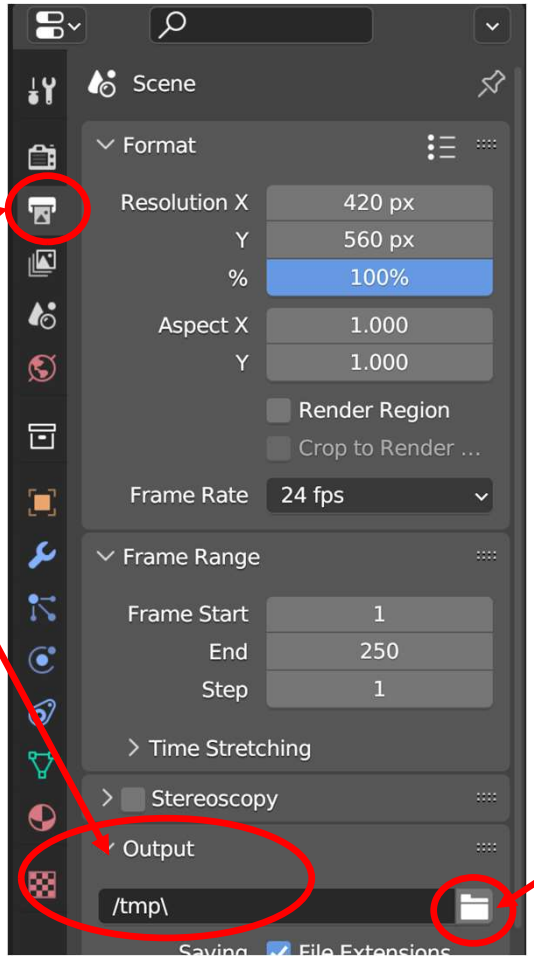


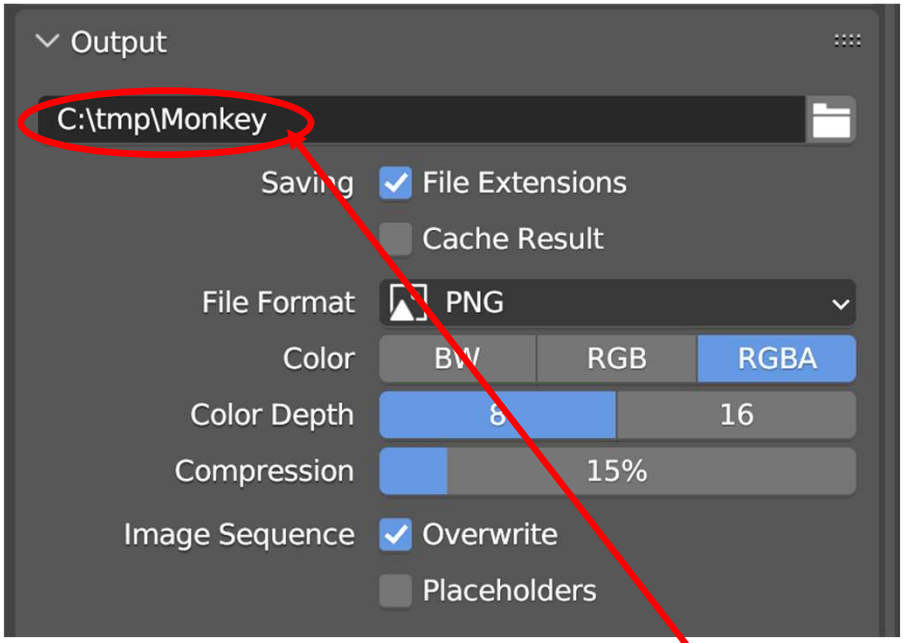

Select **Portrait, 48 Views**

# Using the Looking Glass Tab, III



Go select the **Output** icon and navigate down to **Output**.

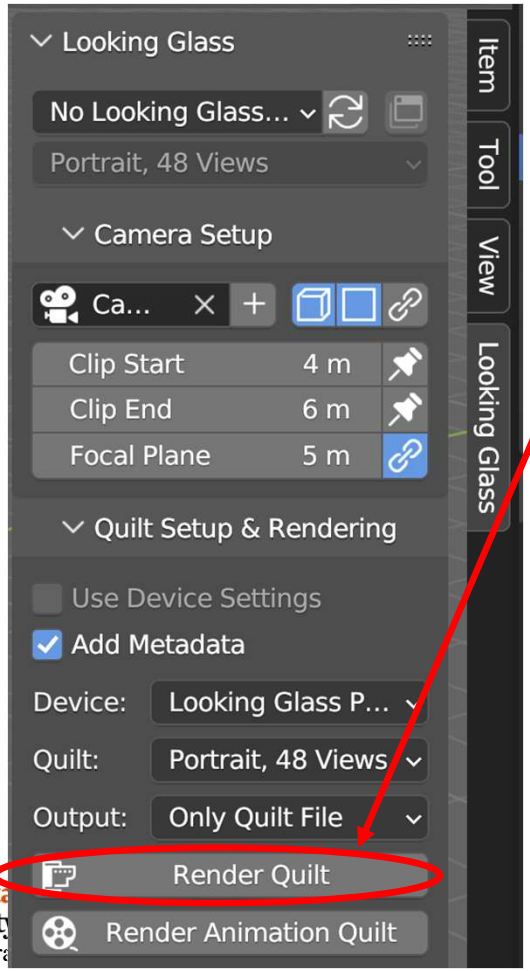



Click here and navigate to some folder for Blender to render your quilt into. Click Accept and then give a filename. On my system, it ended up looking like this.

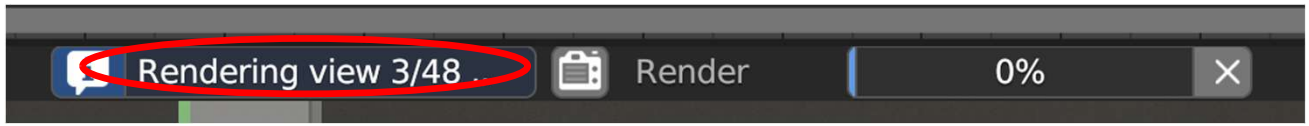
# Using the Looking Glass Tab, IV



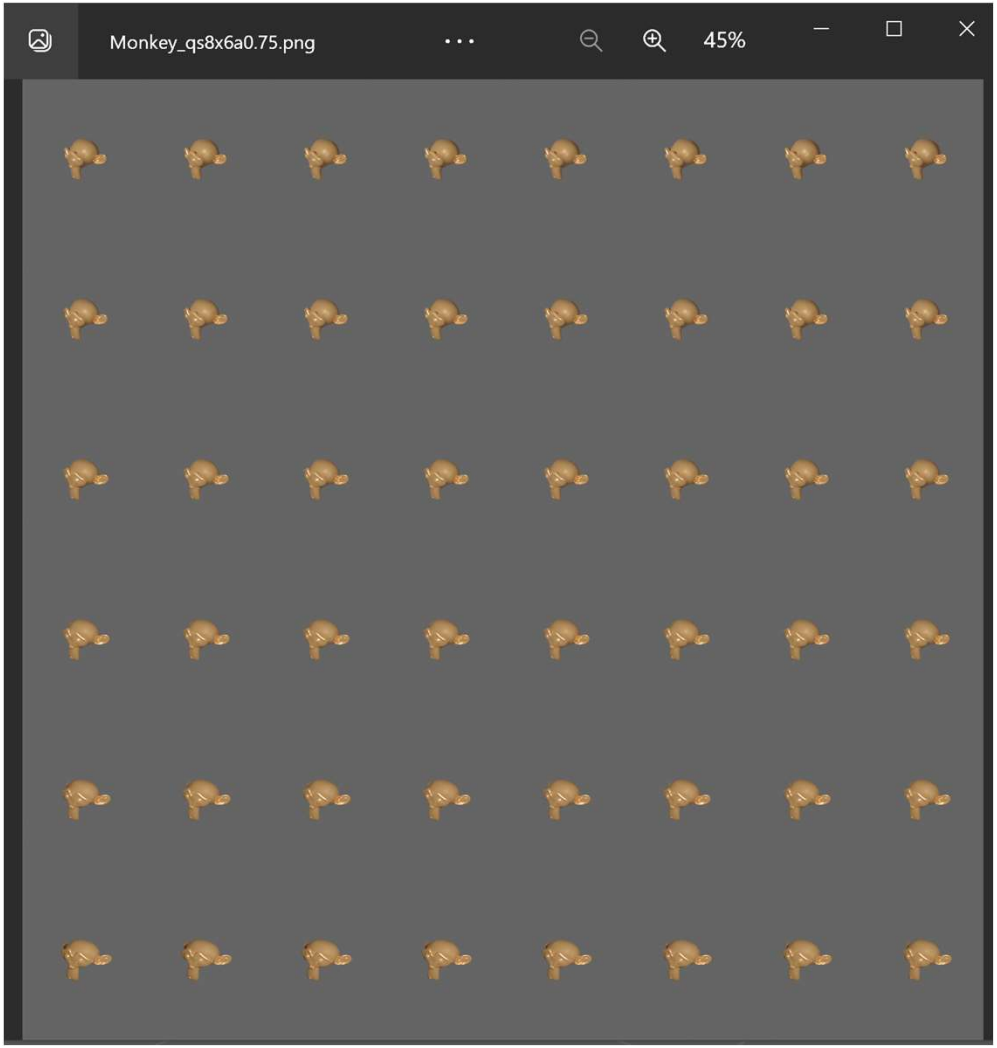



Going back to the Looking Glass tab, click on **Render Quilt**. This will trigger the rendering of 48 different scenes.

You can watch its progress at the bottom of the screen:



# Monkey_qs8x6a0.75.png

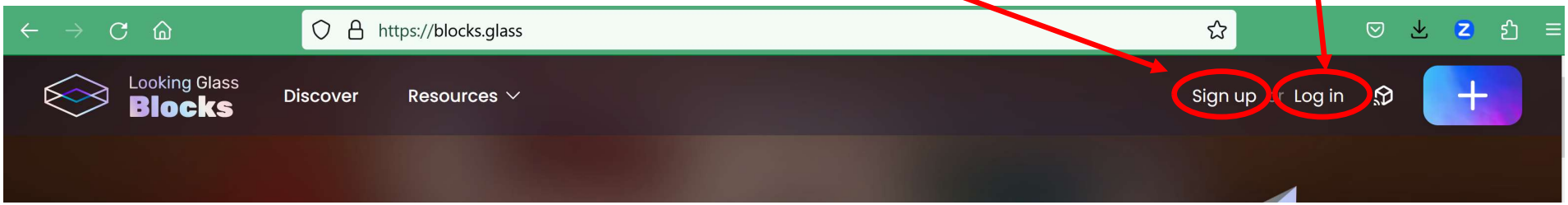


# How to Go From a Quilt to a Live "Hologram", I

Navigate to **https://blocks.glass**

If you don't have an account, **sign up** to get one. If you are under 18 years old, be sure to have your parents help you do this!

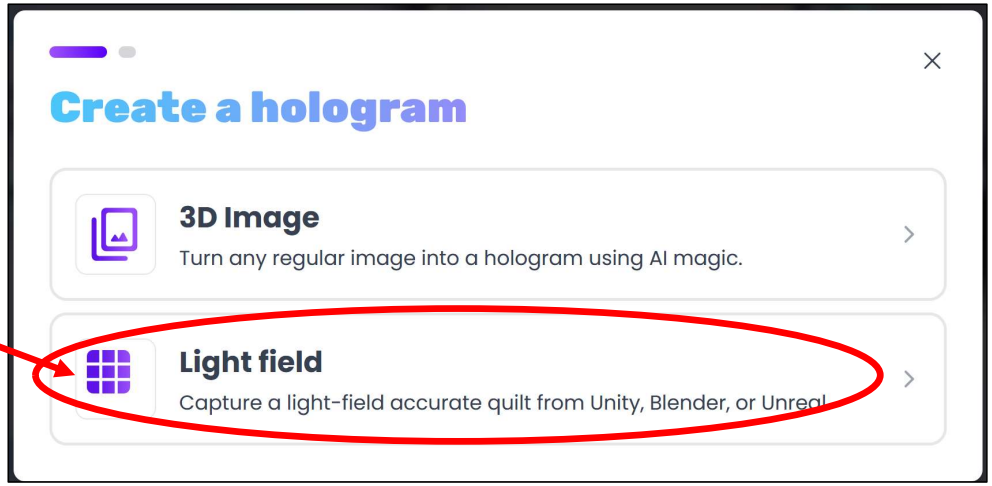
If you do have an account, **Log in** to it.



# How to Go From a Quilt to a Live "Hologram", II

Click on the big Plus Sign

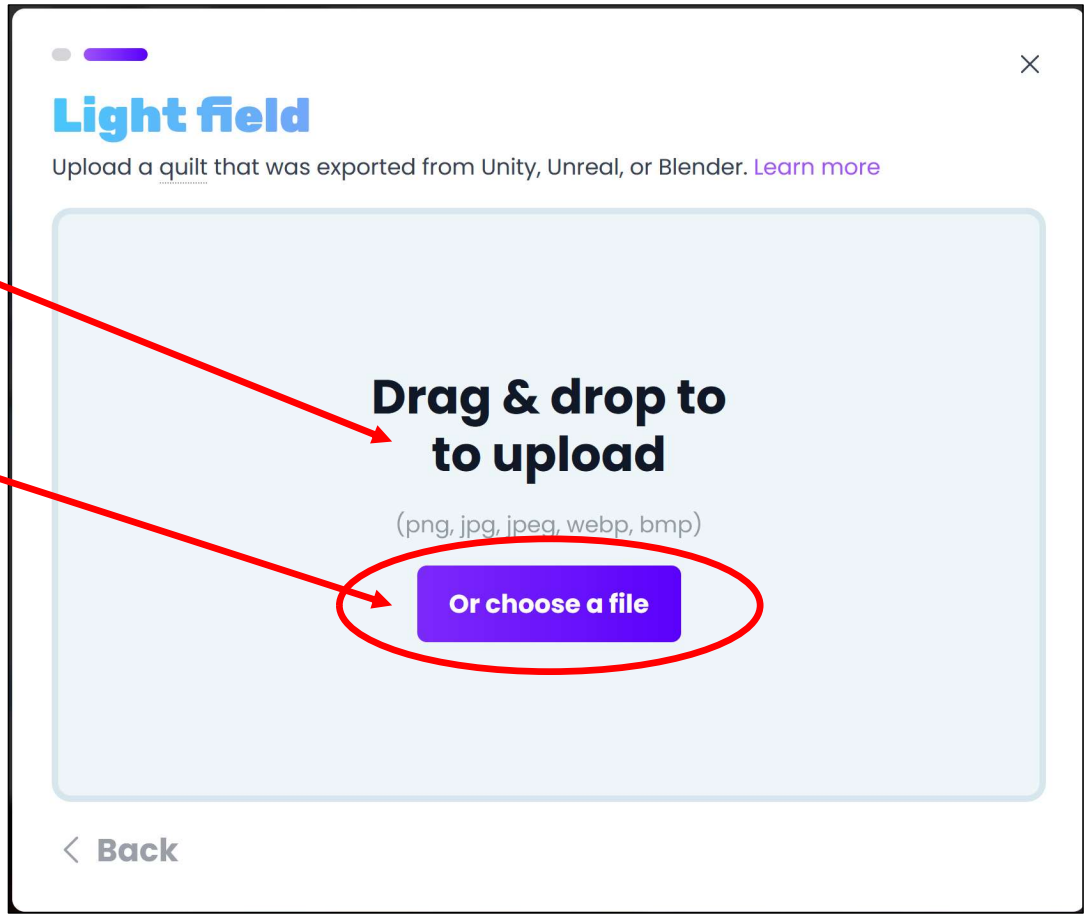

To download a Blender or OpenGL quilt, click here.



# How to Go From a Quilt to a Live "Hologram", III

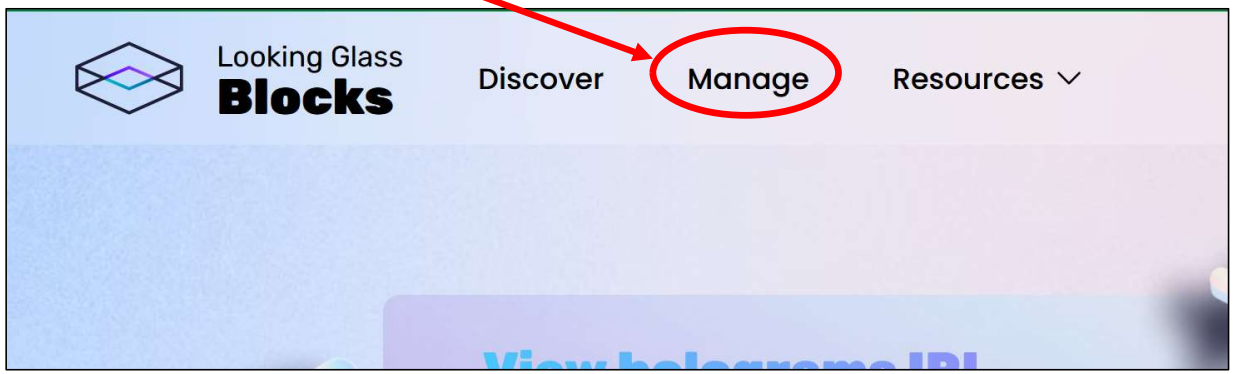


Drag-and-Drop your quilt file here,
or click here and navigate



# How to Go From a Quilt to a Live "Hologram", IV

Click on **Manage**, then click on the name of the file you just uploaded



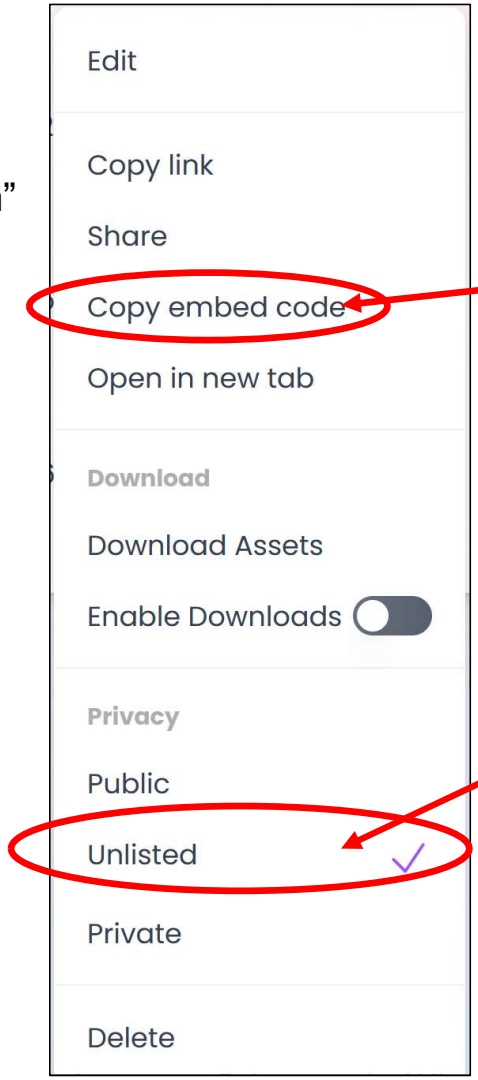

Wiggle the mouse left and right to see your hologram move in 3D

# How to Go From a Quilt to a Live "Hologram", V

Click on **Manage** again then click here

# How to Go From a Quilt to a Live "Hologram", VI

That will bring up a bunch of things you can do with your new "hologram"




Clicking here will copy the embedded HTML code, to get at this hologram from the web, to the clipboard. Paste that into one of your HTML web pages!

Be sure this permission is set to **Unlisted**.

# “Casting” a Quilt to a 3D Image on a Looking Glass Portrait Display



https://lookingglassfactory.com/looking-glass-portrait