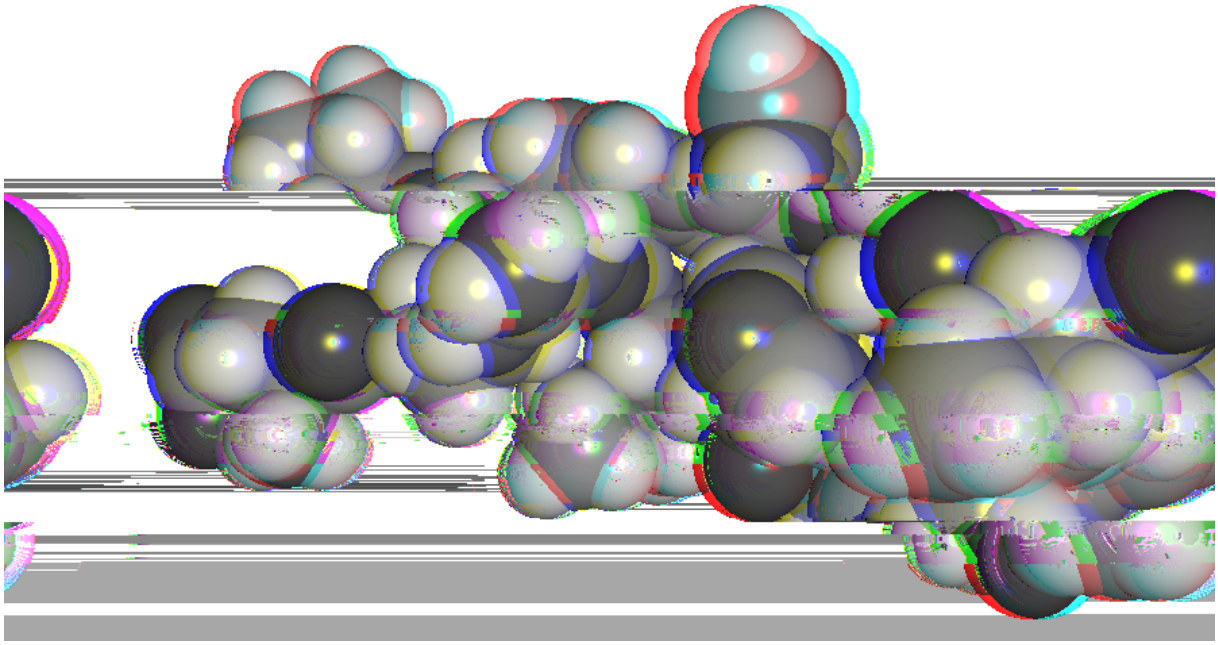
Universität Bielefeld
Technische Fakult"at

Stereoscopy in OpenGL by the example of Moleculevisualization



Thomas Waltemate

Datum: March 5, 2012

Bielefeld University
Universitätsstraße 25
33615 Bielefeld
Germany

Abstract

In this short elaboration is explained how to generate (realtime) stereoscopic images with OpenGl. First there will be a short introduction to stereoscopic rendering. Then there is an explanation of the formulas given, that are needed to calculate the frustums for the stereoimages. And nally there is shown the example of Moleculevisualization.

Contents

1 Introduction **1**
1.1 Toe-In 1
1.2 O -axis 2

2 Mathematic background **3**
2.1 Translation 3
2.2 Frustum 3
2.3 Rendering 4
2.4 Molecule Visualization 6

3 Conclusion **9**

Sources **11**

1 Introduction

To generate a stereoscopic image on the screen, there are needed two images: One for left eye and one for the right eye. There are two general approaches to make these images. The rst one is called "toe-in\ and the second one is called "o -axis\.

1.1 Toe-In

In the Toe-In approach there are two symmetric frustums and therefore two crossed projection planes as you can see in gure 1.1. While this approach does produce a working stereo image, it also has the issue that it makes the viewer feel sick or at least gives them some sort of headache.
This is it's main disadvantage, but it is easier to implement as well.

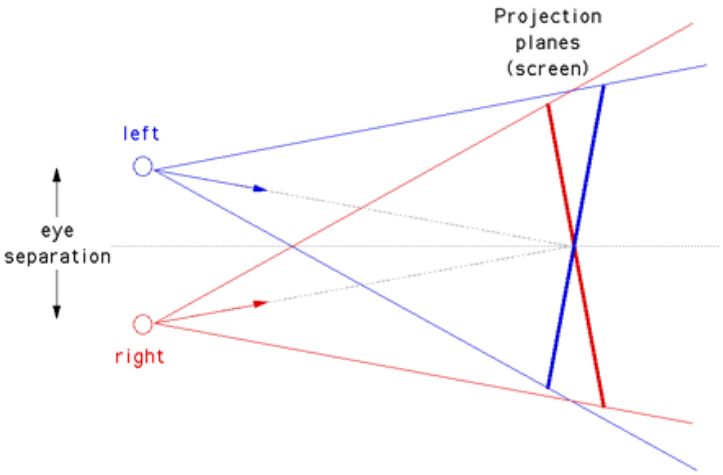


Figure 1.1: Toe-In approach.

1.2 Off-axis

The O -axis approach uses two asymmetric frustums. This leads to two parallel projection planes as one can see in gure 1.2. Of course it produces a working stereoscopic image, too. While it has the disadvantage that it is more di cult to implement | because it needs a more complicated frustum | it does not cause any problems like sickness or headaches as the Toe-In approach does.
This should be the approach to choose when you want to produce stereo images.

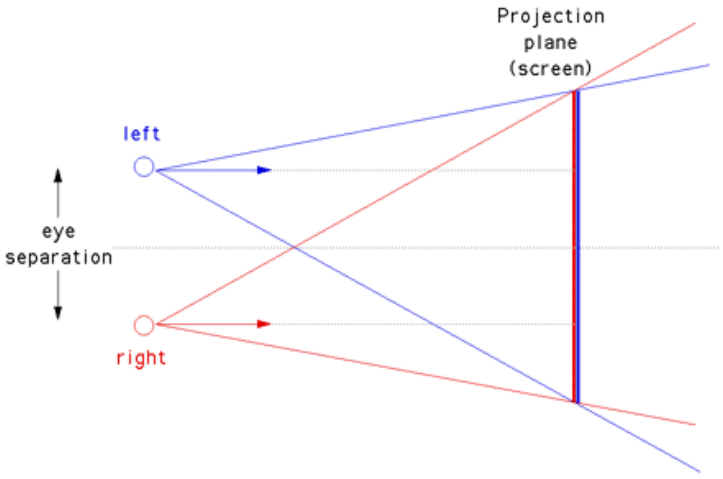


Figure 1.2: O -axis approach.

2 Mathematic background

Because the O -axis approach is the better one, only this approach is explained.
To get the two pictures for the left and right eye, there three steps needed. The rst is transforming the camera or the scene/object, the second is calculating the frustums and the third is the rendering of the scene.

2.1 Translation

The translation of the scene/object is easily done by the old OpenGL method glTranslate(+/-(0.5*eye_separation), 0.0, 0.0) or an equivalent function for generating a translation matrix. The translation of the camera can be achieved with the GLEW function gluLookat(...) or again with a equivalent function that generates a look-at and therefore view matrix.

2.2 Frustum

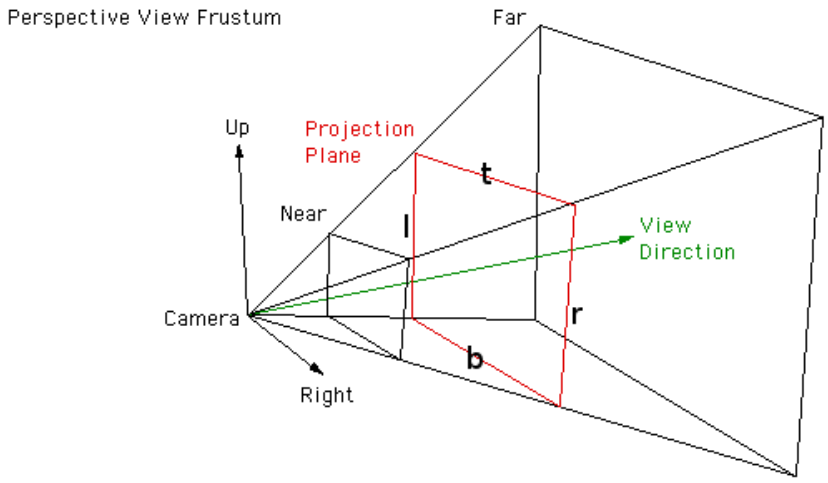


Figure 2.1: Frustum.

For calculating the frustum the following values are needed: Left l, Right r, Bottom b, Top t, Near near and Far far. The values are used like in gure 2.1 and they are calculated as follows:

$$wd2 = near \cdot tan(\frac{\pi}{180} \cdot \frac{FOVy}{2})$$

$$b = -wd2$$

$$t = wd2$$

Frustum for the left eye:

$$l = b \cdot \frac{width}{height}$$

$$r = b \cdot \frac{width}{height} + 0.5 \cdot eye\_seperation \cdot \frac{near}{focaldistance}$$

Frustum for the right eye:

$$l = b \cdot \frac{width}{height} + 0.5 \cdot eye\_seperation \cdot \frac{near}{focaldistance}$$

$$r = b \cdot \frac{width}{height}$$

The values width, height, FOVy, eye_seperation, near and focaldistance are constants given by the environment.
The OpenGL function glFrustum(GLdouble l, GLdouble r, GLdouble b, GLdouble t, GLdouble near, GLdouble far) can be used with the just calculated values to get the needed projection matrix and therefore the two frustums that are needed.

2.3 Rendering

To render the stereo images one has to follow these steps:

1. Calculate the Frustum for the left eye
2. Transform scene/camera to the right/left
3. Render the scene
4. Calculate the Frustum for the right eye
5. Transform scene/camera to the left/right
6. Render the scene

One can use di erent types of stereorendering, like: anaglyph, quadbu ering, Side-by-side or Top-bottom.

Anaglyph

Generating anaglyph images is easy. The only thing to do is set the right colormasks before rendering the speci c image. For the left eye the colormask has to be: glColorMask(GL_TRUE, GL_FALSE, GL_FALSE, GL_FALSE) and for the right eye: glColorMask(GL_FALSE, GL_TRUE, GL_TRUE, GL_FALSE).

Quadbuffering

To generate the images for quadbu ering it is neccessary to render the two pictures into the correct bu ers.
First of all you have to activate quadbu ering. In GLUT for example this is done with the GLUT_STEREO ag in the glutInitDisplayMode(...) function (This does only work if you have a Nvidia Quadro Card or an AMD equivalent). Then you just have to activate the correct bu er before rendering the two images: glDrawBu er(GL_BACK_LEFT) for the left eye and glDrawBu er(GL_BACK_RIGHT) for the right eye. After that you have to swap bu ers as usual.

Side-by-side and Top-bottom

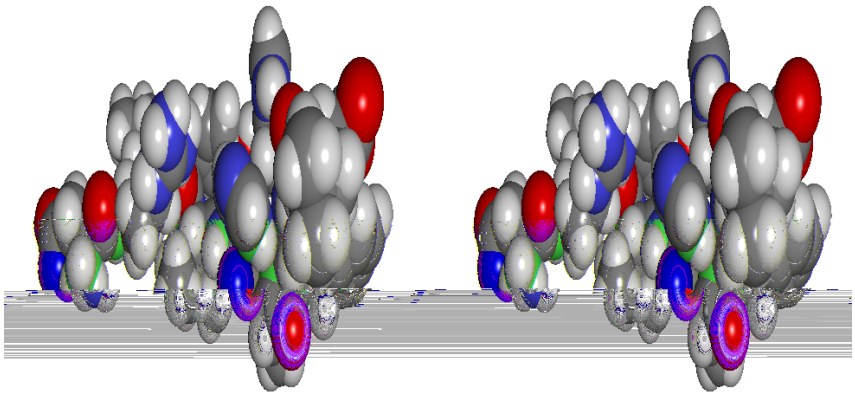


Figure 2.2: A side-by-side stereo image.

Side-by-side and Top-bottom methods are useful for 3D-TVs and 3D-Projectors.
The general approach for both methods is the same. The viewport has to be adjusted so that it is divided in two halfs as you can see in gure 2.2. This can be achieved by using the glViewport() function.
For the Side-by-side method you have to adjust the viewport for the left eye like this: glViewport(0, 0, 0.5*width,height) and the right one like this: glViewport(0.5*width, 0, 0.5*width, height).
The viewports for the Top-bottom method are pretty similar. Left eye: glViewport(0, 0.5*height, width,0.5*height); Right eye: glViewport(0, 0, width, 0.5*height).

After each adjustment of the viewports you have to render the scene of course. It is possible that this is not correct for every TV/Projector, so you maybe have to invert pictures.

2.4 Molecule Visualization

Molecule Visualization is an important field for stereoscopic rendering, because it is pretty important that you can differentiate single atoms and parts of a molecule. This perception of depth could also be achieved by a shadow calculation, but this is sometimes pretty expensive and has not quite the same results as stereo rendering.
You can see a direct comparison of a molecule with and without stereorendering in figure 2.3 and figure 2.4. Of course you need Red-Cyan-3D-Glasses to view the stereo image correctly.

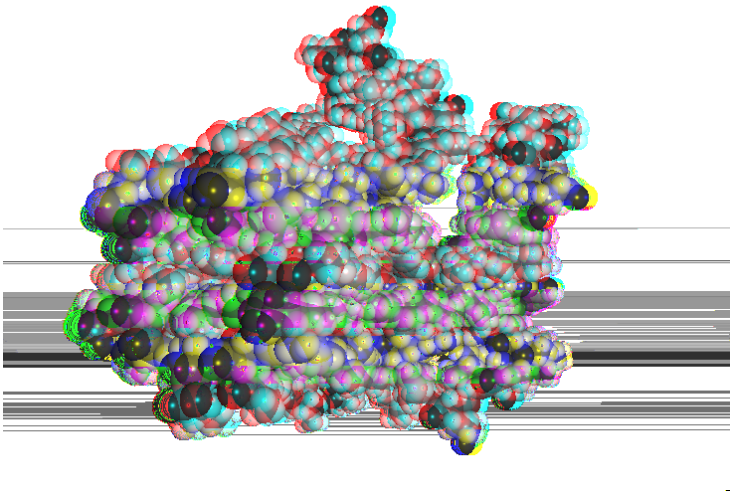


Figure 2.3: With stereo.

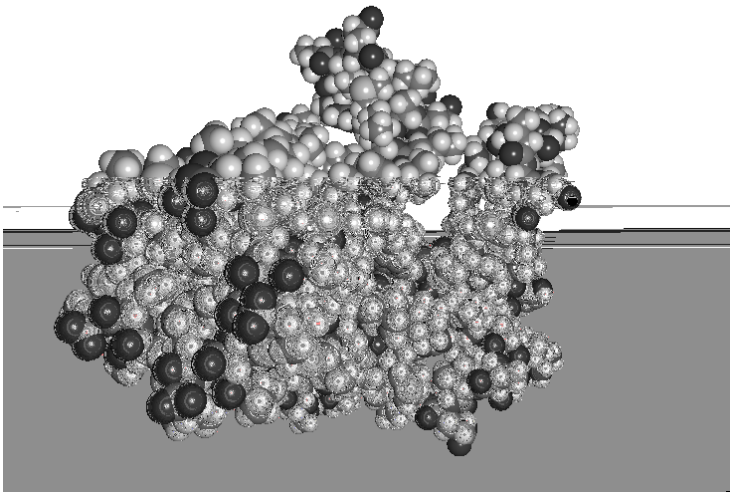


Figure 2.4: Without Stereo.

3 Conclusion

It is recommended that you use the O -axis approach for rendering stereo images | at least if you have the opportunity to calculate the frustum an easy way like in OpenGl.

Moleculevisualization is an important eld for stereo rendering.

Sources

- http://paulbourke.net/miscellaneous/stereographics/stereorender/
- http://www.opengl.org/
- Bachelorarbeit: GPU-Based Molecule Rendering