Introduction to Stereo Rendering

Herbert Grasberger*

Institute of Computer Graphics
Vienna University of Technology
Vienna / Austria

Abstract

In this paper we describe different techniques of rendering stereo scenes in real-time. We discuss the physical properties of the given stereo setup and revise some basics knowledge needed for generating the stereo pairs. After that we discuss several techniques for rendering stereo pairs according to correctness. We explain certain effects and their appearance in a stereo rendered scene. Finally we explain how stereo rendering works with OpenGL.

1 Introduction

Every computer game or general computer graphics application encounters the problem, that it produces twodimensional images out of previously modeled threedimensional scenes. In the past techniques were already developed to percieve these threedimensional scenes in a threedimensional way.
The general approach to threedimensional vision is called stereo vision, where different images are rendered for both eyes.
Stereo rendering is all about rendering different images for the two eyes. As a consequence the program has to make sure, each eye of the observer sees only the image it is supposed to see.
One of the oldest way of stereo vision is ”color coded stereo” where each eye is covered with a filter for a certain color and the image to look at is colored according to the color filters. Each eye recieves a color filtered image as a result.
The newest technique is called ”spectrum multiplex” where the spectal peaks of colors are shifted in order to achieve stereo vision.
The approach used in our environment differs from the above approach in the way light is filtered. Filters for polarized light are used instead of color filters, which does not allow any differing colors between both images.
Apart from these passive techniques, where no active viewing device has to be used for seeing the propper images, there are also active stereo techniques like stereo vision by shutterglasses or similar approaches.

*herbert.grasberger@gmail.com

2 Properties of the Stereo Setup

The stereo setup consists of two JVC D-ILA projectors with integrated polarization filters and a special retroreflective screen. Each projector shows the image for one eye, runs at 60fps and is synchronized with the other.
Polarization refers to a property of electromagnetic waves. It describes the orientation of the oscillation in the plane orthogonal to the direction of the wave's travel. When it comes to light it specifies the direction of it's electric field. Light has two different oscillation properties that are useful for stereo rendering.

- linear polarization
  Light travels along a single plane. Filters are rotation dependant which means that if the filter is turned by 90° light with the other polarisation can be seen.
- circular polarization
  Light consists of two perpendicular electromagnetic plane waves of equal amplitude that are orthogonal in phase and which travel in a circular fashion. See Figure 1 for an example. Filters are rotation independant which means that the same light is filtered even when the filter itself is rotated.

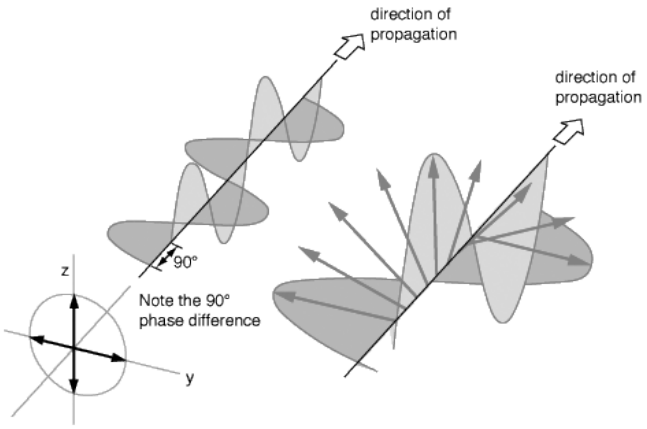

Figure 1: Illustration for circular polarization

A retroreflective surface only reflects light in the direction of the source in the same polarization that it had before.

This is completely different to an ordinary screen where light polarization is scattered in many ways.
The stereo technique used in this setup is called polarization stereo where each projector emits light of a certain polarization.
The projectors are equipped with filters for circular polarized light and there are glasses with the corresponding polarization. These parts in the conjunction with the retroreflective screen allows control of which image of the projector reaches which eye.
This effect can be used to render different images for the left and the right eye without any major difference in color for each of them. Also the so called "ghosting", which means when the user sees the light for the left eye with the right eye and vice versa is minimal with this system. Actually it depends on the colors displayed and is mostly noticeable when there are high frequency color changes on the image.

3 Viewing Pipeline

When generating an image using polygon-based computer graphic techniques points have to be transformed. In order to compute which position a vertex of a model has on the screen it has to undergo different transformations. See figure 2 for the complete transformation illustration.

First, all vertices of a single model have to be transformed from object space, which is where the coordinates are given in relation to the origin of the object (called modeling coordinates in Figure 2), into the world space. Then the vertices have coordinates according to the origin of the world. In OpenGL the transformation matrix responsible for this transformation is called the *world matrix*.

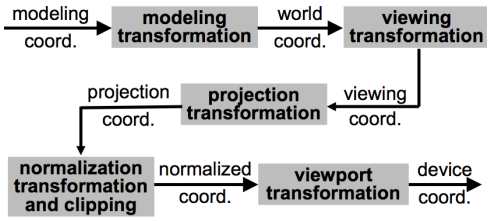

Figure 2: The full viewing pipeline [6]

Each vertex has to be transformed into the viewing coordinates. After this transformation, done by the *view matrix*, the vertices have coordinates relative to the position of the camera. Then comes the projection transformation. It is dependant on the camera parameters, for example field of view, perspective, etc., and maps only the vertices that should be projected onto the screen into the coordinate range of $[-1,1]$.
The last two transformations are defined by the camera properties, its position, direction, and projection parameters expressed in the form of the view and projection matrix. For stereo rendering, each eye has to be addressed seperately.
Several techniques for generating the stereo pairs exist. Each technique handles modifications of these two matrices differently. Therefore these two matrices are described in detail in the following two sections.

3.1 View Matrix revisited

As stated before, the view matrix transforms coordinates from world space to the view space. This is done by an affine transformation which is composed of a rotation and a translation. The translation translates the origin into the position of a camera.

$$\mathbf{T} = \begin{pmatrix} 1 & 0 & 0 & -p_x \\ 0 & 1 & 0 & -p_y \\ 0 & 0 & 1 & -p_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{1}$$

In the translation matrix called **T** found in equation 1 the vector p is the position of the camera.
The rotation matrix rotates the points from the normalized axes of the world coordinate system into the local view, side and up axis of the camera frame.

$$\mathbf{R} = \begin{pmatrix} s_x & s_y & s_z & 0 \\ u_x & u_y & u_z & 0 \\ -v_x & -v_y & -v_z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{2}$$

The vectors in the rotation matrix called **R** found in equation 2 are called s for the side vector, u for the up vector and v for the view vector of the camera.
To create the viewing matrix **V** **T** and **R** have to be composed with the multiplication found in equation 3.

$$\mathbf{V} = \mathbf{R} * \mathbf{T} \tag{3}$$

In order to transform a vertex into the viewspace it has to be right multiplicated with the matrix **V**.

3.2 Projection Matrix revisited

The projection matrix contains all information on the internal camera parameters, like its field of view or the near and far plane. When applied to vertices in viewing coordinates it maps the position values to the range of $[-1,1]$ as mentioned before. Actually the range [-1,1] is reached after the division with the homogene coordinate and not directly after the projection transformation. Figure 3 demonstrates the mapping of real values to the range. If a vertex has coordinates out of this range after the projection transformation, it is not visible in the rendered scene.

Actually, there are some different forms of the projection matrix, which differ in the features of the matrix and in complexity of computation. The most common form is the *symmetrical projection matrix*, which can be used in

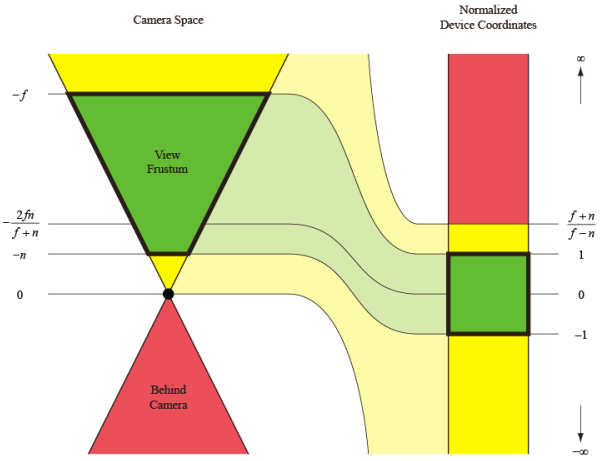

Figure 3: Illustration of the mapping from world coordinates to normalized coordinates [7]. In the green section there are all vertices that are actually visible.

some stereo projection techniques. For the most advanced stereo technique we need a projection matrix that also supports an *asymmetric projection*.
In the matrix itself the variable n is used for the near plane, f for the far plane, r,l are the values for the left and right maxima, whereas t,b stand for top and bottom. The matrix for asymmetrical projection has the following form:

$$\mathbf{P} = \begin{pmatrix} \frac{2n}{r-l} & 0 & \frac{r+l}{r-l} & 0 \\ 0 & \frac{2n}{t-b} & \frac{t+b}{t-b} & 0 \\ 0 & 0 & \frac{-(f+n)}{(f-n)} & \frac{-2nf}{f-n} \\ 0 & 0 & -1 & 0 \end{pmatrix} \tag{4}$$

4 Stereo Rendering Techniques

4.1 Basics

The basic principle of stereo rendering is rendering each image twice, once for each eye with a certain eye offset in the direction of the side vector of the camera. Typically the distance between the left and the right eye is about 6.3cm so the matrices are shifted from the neutral position along half the eyeoffset. This modification has to be done for every stereo technique.
For one technique it is necessary to modify the projection matrix too.

These operations result in a parallax effect on objects, which means that an object has different positions in the left and the right image. An example of a positive parallax can be seen in Figure 4. The distance between these two objects corresponds to the distance of the object to the projection plane. Objects on the projection plane have no parallax, whereas the position of the parallax points is related to the distance of the point to the projectionplane.
Because human vision is able to use this parallax effect to create a threedimensional impression in the human brain

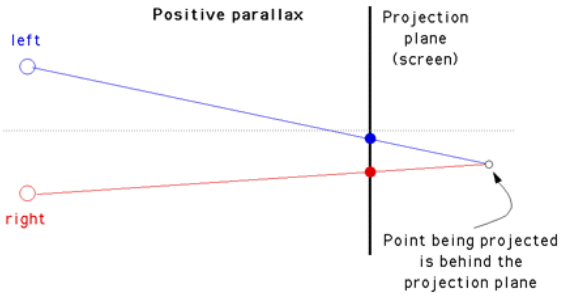

Figure 4: Illustration of a positive parallax when looking at a point from two observers [2]

from looking at objects in the real world, the stereoscopic vision works. Paul Bourke describes the basics of these effects in his webpage [2].

4.2 Offset

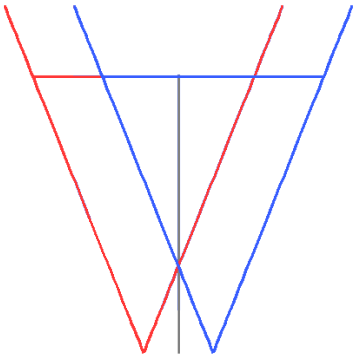

Figure 5: Illustration of the offset technique. Notice the areas that are only seen by one camera.

The so called offset technique is the most simple technique in stereo rendering, because it only shifts the view matrix for each eye according to the proper distance. The projection matrix stays the same and as a result there are parts in the viewing volume that are only visible in one of the two cameras (Figure 5). This results in discomfort because the some objects are visible only in one eye.

The company nVidia [1] offers a stereo driver for its high-end graphics cards with stereo support that enables stereo rendering with applications that are not programmed for beeing used in a stereoscopic environment. There is not much information about the technique that is used in this driver, but the resulting image looks like it is done with this approach.

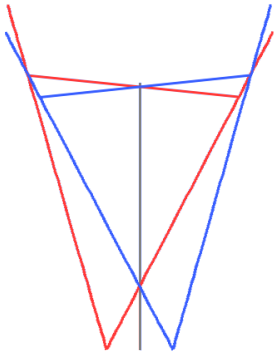

Figure 6: Illustration of the toe-in technique. notice the different far planes.

4.3 Toe-In

The toe-in approach is used in many feature films with stereo support, because there is no asymmetric projection used, which corresponds to the normal lenses of a film camera. In this approach the camera is not only shifted like in the approach before, but it is also rotated towards the focus point (Figure 6). This method might seem to produce a correct solution at first glance, however, it introduces a slight vertical parallax, that might cause discomfort if the user is looking at the image for a longer time. This is because the vertical parallax is not the same on the whole image but it increases with the distance from the image center.

4.4 Off-axis

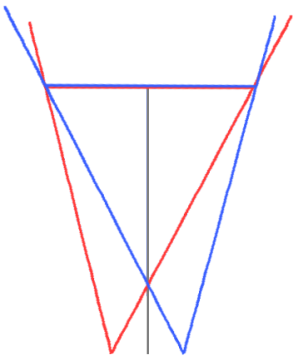

Figure 7: Illustration of the off-axis technique

The off-axis method is the most advanced method, because it not only uses the offset view matrix, but it also changes the projection matrix so that the projection gets asymmetric. This is done by modifying the projection matrix with different left and right values for each eye. Then the farplane of both eyes are identical, and there is a slight offset at the nearplane that is not noticable (Figure 7). This approach doesn't produce any vertical parallax and as a result it is the least stressfull method for creating the stereo pairs.

4.5 Optimization Technique - Reprojection

Figure 8: Illustration of the cache hits (green) and cache misses (red) during the reprojection technique [10]

The reprojection technique can be used with all of the previously discussed techniques, because it is more of an optimization than a single technique.
In reprojection, the image is rendered for the first eye as usual, and after that each pixel is reprojected to the other eye. If this pixel is visible in the other eye, then it is reused directly, if it is not visible the new pixel is computed on the fly. This results in a hit and miss approach (Figure 8).
This technique is described in more detail in [10]. At first, this approach sounds very straight forward, but if more complex surface shaders like parallax mapping are used in the scene, then this approach might be less performant, even unuseful, because most of the pixels will have to be recomputed.
In parallax mapping for example the color of the pixel depends on the viewing direction, which is different for each eye. As a result every pixel with a parallax mapped surface has to be recomputed anyway.
The nVidia [1] stereo driver is supposed to optimize in a similar fashion.

5 Real-Time Rendering Effects and Stereo Rendering

This section accounts for some common effects in today's games and demos according to their usefulness in a stereoscopic environment. The effects can be divided into two parts:

- scene effects
  Effects that are related to surfaces or that are assigned to certain objects in the scene.

- postprocessing effects
  Effects that operate on the final image of the scene.

5.1 Scene Effects

5.1.1 Billboards

Billboards have a long history in computer graphics environments.
They are textures mapped onto camera oriented faces, quads in most cases. Often billboards are used to suggest more geometric detail than there actually is. This can be seen in Figure 9. Apart from particle systems, where the billboards are usually very small, they have a very poor look even for standard rendering.
In a stereoscopic environment the use of billboards creates visual problems, because in this case, the billboards look flat due to the missing parallax and as a result billboards do not work in this case.

Billboards can be used for distant objects but they

Figure 9: Trees composited out of several billboards

will be noticed more easily than in the traditional non-stereoscopic threedimensional environment.
In [13] Risser describes an extension to billboards called *true impostors* that uses an additinal heightmap when rendering these billboards. This method might improve billboard rendering in a stereoscopic environment at a very low cost.

5.1.2 Planar Mirroring

Mirroring means rendering the scene mirrored at an arbitrary axis into a texture and applying this texture onto an object. In the tested case mirroring was used to model the reflection on a water surface. This can be seen in Figure 10.
The first thoughts were that a mirror needs only rendered once out of the neutral eye position, and then it could be applied to the surface for both eyes without any problems. In the stereoscopic environment it turned out that it makes it very hard to grab the stereoeffect at the boundaries, and to the user the effect seems wrong in a certain way. When the reflection is rendered only once, it seems to the user, as if everything reflected had the same distance as the object on which the reflected scene is mapped onto.
This can lead as far as to break the whole stereo effect if

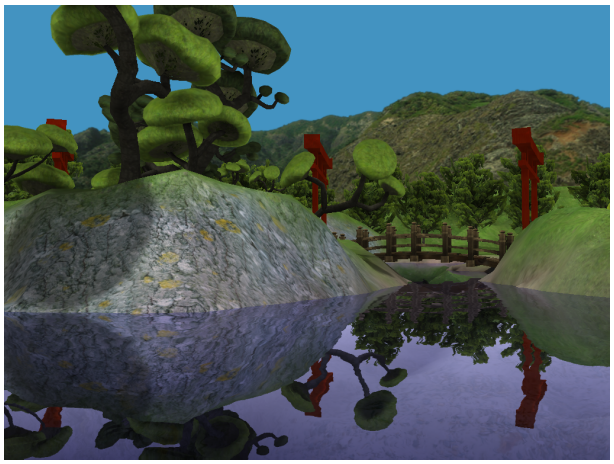

Figure 10: Mirroring on the water surface

the reflected scene occupies a certain amount of the screen. Fortunately, there is a very simple solution to this problem. If the reflection is rendered individually for each eye it works in the stereoscopic environment too.

5.1.3 Parallax Mapping

Figure 11: Example of parallax mapping

Parallax mapping [4] is an enhancement of normal mapping. Additionally to the information used in normal mapping a heightmap is used.
In parallax mapping the texture coordinate for the colormap lookup is displaced according to the heightmap as

a function of the view vector in tangent space.
Parallax mapping can create the impression of additional depth and structure on objects that are modeled with minor detail. This is used in today's videogames to enhance the visual impression of higher frequency details that would require too much geometry to be modeled in detail. In Figure 11 this can be seen at a wall that is modeled with a very low amount of vertices, but the surface detail is still well preserved.
This effect works very well in a stereoscopic environment beause when calculated separately for each eye, the texture lookups differ slightly, and as a result the structure of the parallax mapped surface becomes apparent in stereo.
As a result, enhancements of simple parallax mapping like steep parallax mapping [9], parallax occlusion mapping [16] or cone step mapping [3] will also work in stereoscopic environments. Relief mapping [11], which uses an analog approach for finer surface details is also supposed to work.

5.1.4 Fur Shading

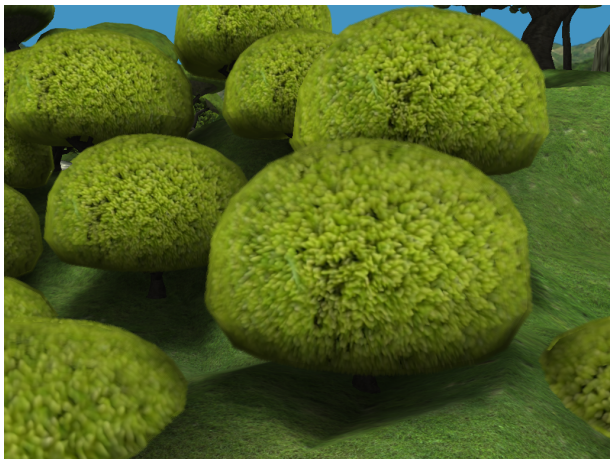

Figure 12: Fur shading on the plants

There are some approaches to fur shading, in this case multiple fur shells of increasing size of an object are rendered with different alpha values according to the current shell number.
This approach is simple and looks well when used in the traditional three dimensional environment which can be seen in Figure 12.
When the effect is used in a stereoscopic environment, the fur looks very well too. If the number of shells and the shell size is high enough, then it really creates the effect of extruded fur geometry.
A more advanced fur shading effect which extends the idea above to the idea of extruding fins can be found in [8].

5.2 Postprocessing Effects

5.2.1 Bloom

The bloom effect [15] is achived by blurring the image of the rendered scene or parts of it by a filter. Often a simple separable gaussian blur is used with different sizes. The amount of bloom can be controlled by the size of the bloom kernel and the size of the source texture.
The resulting effect is a brighter image like in Figure 13. This effect does not use any depth or viewdependant information for the blurring, and as a result sharp edges in the original image might get blurred to a certain extent.
This results in a disturbance of the stereo effect, which gets larger when the filter kernel size increases or the size of the original texture decreases.
Because of that a simple bloom has to be used with care in a stereoscopic environment.

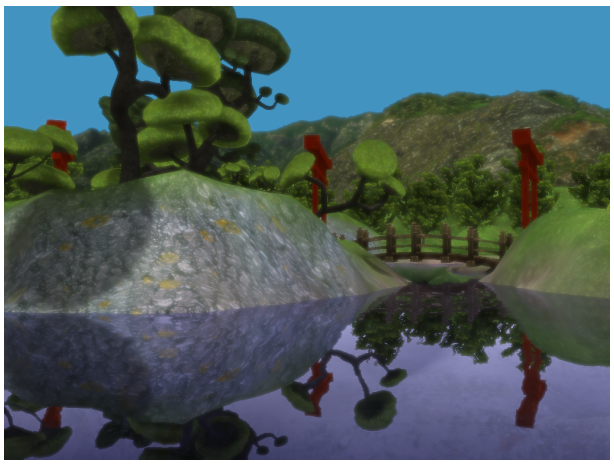

Figure 13: The image from Figure 10 with applied bloom

5.2.2 Depth of Field

When rendering an image without depth of field, we imitate the behavior of a pinhole camera, which is a camera with a indefinitely small aperture and a very long exposure time. This creates an image that is sharp at any distance. In film production or photographs this model is not used, and as a result there is a depth of field effect in many images.
An image with depth of field is only sharp at a certain distance and gets more and more blurred depending on the distance of objects to the focal distance (see Figure 14 for an example).
To recreate this effect we have to write out the depth of each pixel when rendering and then blur the image according to the pixel distances to the focused depth. The blurring itself is done similarly to the bloom effect above, but it is not done in the same intensity on the whole image. The blur size is scaled with the distance to the focal distance.

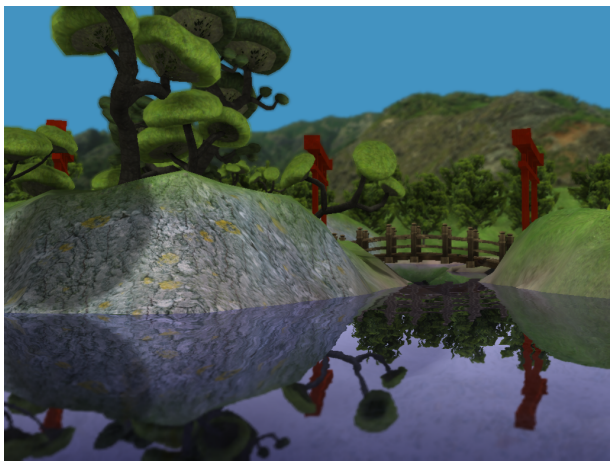

Figure 14: Depth of field with unsharp regions in the front and back, and sharp regions in the focused depth.

Depending on the general size of the blur, the depth of field effect can support stereo vision, but when the blurring is done too intensively it also starts do destroy parts of the stereo effect. The limit of filter size where it starts to destroy stereo vision is higher than with the ordinary bloom. Additional information to depth of field can be found in [5] and [14].

5.2.3 High Dynamic Range Rendering

In this case high dynamic range rendering does not only mean that color values greater than 1 are created but it also includes global tone mapping of the produced image.
Tone mapping means that the values in the current image are mapped to values that display devices can recreate to get the best use of contrast on the display device. In real time graphics this approach is often combined with blurring parts of the image with high luminance with simple gaussian blur or more sophisticated star effects [12].
This is done to imitate human or camera vision. When a human looks into a very bright region, the user only sees white at a certain brightness, even if the colors in this region are distinct. These overbright regions also often smudge into the surrounding regions. This results in a lack of depth perception.
As a result high dynamic range rendering with tone mapping and some additional effects work in a stereoscopic environment, even if the blurring of overbright regions destroys the depth perception, because corresponds to human vision.
Of course also a combination of depth of field and high dynamic range rendering works if the parameters are chosen right. If the wrong parameters are chosen in both techniques the result will get unnatural and awkward.

6 Implementation

The implementation for testing the stereo effects was programmed with OpenGL and Glut. We use a nVidia Quadro FX 5600 graphics card for the quad buffered stereo output.
In order to use the quad buffered approach, which simply enables a second front and backbuffer in OpenGL, the OpenGL window has to be created with an additional flag: GLUT_STEREO.
Similar flags are supported in Qt (QGL::StereoBuffers) or in the WinAPI, where there is a PFD_STEREO pixelformat flag.
When this pixelformat is created the user simply has to set glDrawBuffers(GL_BACK_LEFT); or glDrawBuffers(GL_BACK_RIGHT); and call the appropriate drawing code there. Beware of setting glDrawBuffers(GL_BACK); when disabling offscreen render targets, because this draws in both buffers at once. This is a common mistake. When the buffer swapping command is called, both backbuffers are swapped at once. In order for this to operate, the nVidia display driver has to have certain options enabled:

- Stereo rendering must be enabled
- The display configuration for both outputs on the graphics card has to be set to clone
- The stereo mode has to be set to *clone*

7 Conclusion

This paper presents an overview of methods for rendering stereoscopic images. Three different ways to calculate the viewing and projection matrices are presented and their effect on the created image is discussed in detail.
Also the impact of real-time rendering effects on stereoscopic rendering is discussed.
In general it is very easy to port an existing application to handle stereo rendering. In most cases this can be done in less than 4 hours, provided that the certain camera and math classes are well written.

8 References

References

[1] nvidia.

[2] P. Bourke. Calculating stereo pairs, 1999.

[3] J. Dummer. Cone step mapping.

[4] T. K. et al. Detailed shape representation with parallax mapping. 2001.

[5] N. T. Guennadi Riguer and J. Isidoro. *Real-Time Depth of Field Simulation*, chapter 4.7. 2003.

[6] D. Hearn and M. P. Baker. *Computer Graphics with OpenGL*. Pearson Education International, 2004.

[7] E. Lengyel. Projection matrix tricks. Technical report, GDC.

[8] J. Lengyel, E. Praun, A. Finkelstein, and H. Hoppe. Real-time fur over arbitrary surfaces. In *I3D '01: Proceedings of the 2001 symposium on Interactive 3D graphics*, pages 227–232, New York, NY, USA, 2001. ACM.

[9] M. McGuire and M. McGuire. Steep parallax mapping. *I3D 2005 Poster*.

[10] D. Nehab, P. V. Sander, and J. R. Isidoro. The real-time reprojection cache. In *SIGGRAPH '06: ACM SIGGRAPH 2006 Sketches*, page 185, New York, NY, USA, 2006. ACM.

[11] F. Policarpo, M. M. Oliveira, and a. L. D. C. Jo˙ Real-time relief mapping on arbitrary polygonal surfaces. *ACM Trans. Graph.*, 24(3):935–935, 2005.

[12] E. Reinhard, M. Stark, P. Shirley, and J. Ferwerda. Photographic tone reproduction for digital images. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 267–276, New York, NY, USA, 2002. ACM.

[13] E. Risser. True impostors. In H. Nguyen, editor, *GPU Gems 3*. nVidia, 2007.

[14] T. Scheuermann. Advanced depth of field. Talk, ATI.

[15] G. Spencer, P. Shirley, K. Zimmerman, and D. P. Greenberg. Physically-based glare effects for digital images. In *SIGGRAPH '95: Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pages 325–334, New York, NY, USA, 1995. ACM.

[16] N. Tatarchuk. Parallax occlusion mapping. Talk, GDC, 2006.