StereoScopy in Ogre3D
T-637-GEDE Game Engine Architectures
Reykjavik University

Helgi Leifsson, helgil08@ru.is
Instructor: Hannes Högni Vilhjálmsson

April 9, 2013

Abstract

In this project we explore what needs to be done for correct stereoscopic rendering and test using a custom configuration for a particular fresnel lens setup. Results are that stereoscopic rendering requires little code in Ogre3D and that rendering the scene twice is expensive.

1 Motivation

With displays, gyroscopes, accelerometers and magnetometers all becoming smaller and cheaper, along with modern computing hardware providing sufficient computing power, head-mounted displays are set to become common consumer instruments. For stereoscopy to work however, each scene has to be rendered twice, and therefore using more computing power. There must be some way to work around having to use twice as much rendering power for each scene in stereo.

2 Related Work

In [1] a computational model for stereoscopy is introduced and explained. In [2] the projection and view matrices can be found along with the required mathematical formulas for achieving stereoscopy.

3 Approach

Ogre3D requires only a few lines of code to set up two viewports and attaching a camera for each. They don't however take into account type and specifications of the lenses nor the interpupillary distance of the user. A new custom projection matrix has to be made for a field of view that fits the resolution and size of the viewports in millimeters and offsets the centers of the projection on the monitor. Towards the right side of the left viewport, and towards the left side of the right viewport.
After the two different projections have been created it is necessary to move the cameras apart to achieve the stereo effect. This is done by using view matrices that offset the views horizontally by half the interpupillary distance for each. For moving the camera around, it is possible in Ogre3D to disable the custom view matrices, move the cameras, and then re-offset the cameras from their center axis in each frame. This makes rotating the cameras around a center axis unecessary.
For this project the specifications of the Rend386 lenses, shown in Fig.1, were unavailable, so no corrective distortion was used. It is possible none is needed as lines seem straight when viewed through them because of how they're angled (45° optical axes as measured using a protractor).
The lenses were then placed on top of the monitor and users could manually set their IPD's and fly around in the world.

4 Results

Getting the stereoscopic effect proved challenging as fractions of millimeters can make a difference in the subjective experience.
There is some support for stereoscopy in Ogre3D but none was found fitting for this project where custom measurements of fresnel lenses were used.
Disabling one viewport increased the framerate from 47 frames per second to 80 on the test setup as shown by Fraps so the processing power required was about twice as much.

5 Future Work

In stereoscopy, culling is done using two culling frustums, one for each camera. It might be possible to use one frustum that covers what both cameras see and cull in one pass instead of two. This could improve performance.
Reprojection is another method where instead of rendering two scenes, one scene is rendered and the other is generated by using the information in the frame and Z-buffer to shift parts of the rendered scene in accordance with their depth. This could improve performance but might create unwanted visual artefacts.

Figure 1: The Rend386 fresnel lenses from 1991.

References

[1] W. Robinett, J. Rolland; *A Computational Model for the Stereoscopic Optics of a Head-Mounted Display*; University of North-Carolina at Chapel Hill, NC, USA, 1991

[2] Oculus VR Inc.; *Oculus SDK Software Documentation*; CA, USA, March 2013