Visual Rendering & Oculus Rift

Marco Godi & Francesco Giuliari

April 8, 2015

Contents

1 Introduction

The scope of this project is to make a Volumetric viewer compatible with the Oculus Rift headmount display.

1.1 Volumetric rendering

Volumetric rendering is the name used to describe techniques used to display a 2D projection of a 3D data set. It differs from "normal" rendering Techniques because while more costly it offers more the opportunity to interact with the 3D model so we can split the model, see his internal components and overall get more information at run time.

1.2 The Oculus Rift

The Oculus Rift is a virtual reality headset by Oculus VR that offer the user a fully immersive experience so they can see and interact with the application as if he was seeing with his own eyes instead of looking at a monitor. As of today the most advanced medel is the DK2, for this project we're using the DK1 but it should be compatible with the DK2 since they share the same libraries.

2 Requirements

2.1 Hardware

- It needs a fairly powerful computer to run smoothly because the Volumetric rendering is a resources hungry technique and, since the oculus needs to have a different texture for both of the eyes, the rendering cost doubled.
- The graphic card needs to be able to use OpenGL 3.1 .
- Oculus Rift DK1 or DK2

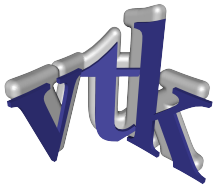
2.2 Software

- The application at the moments only supports Windows operationg system since it uses OS dependant libraries, it was developed and tested on Windows 8.1.
- To succesfully use the Oculus Rift headest the needs to have the Oculus Runtime software available at the Oculus VR official site.

3 External Libraries

Since this project needed to be completeted in a fairly reasonable amount of time we decided to use some external libraries for an easier menagement, for the volumetric rendering we're using the VTK libraries, and, for the Oculus Rift menagement we're using the Oculus SDK libraries

3.1 VTK



The Visualization Toolkit (VTK) is an open-source, freely available software system for 3D computer graphics, modeling, image processing, volume rendering, scientific visualization, and information visualization. VTK also includes ancillary support for 3D interaction widgets, two- and three-dimensional annotation, and parallel computing. At its core, VTK is implemented as a C++ toolkit, requiring users to build applications by combining various objects into an application. The system also supports automated wrapping of the C++ core into Python, Java, and Tcl, so VTK applications may also be written using these interpreted programming languages.

3.2 Oculus SDK

The Oculus SDK is the libraries offered by Oculus VR to menage Oculus Rift headset implementation. It offer methods to keep tracking of the headpose, obtain hardware specifics form the Oculus headset and it allow for creation of virtual heaset useful for debugging the application.

4 Project

4.1 Approach

The application use the VTK libraries to load the Dicom files, create two viewport with the appropriate distance determined by the eye spacing and then applies a barrel distortion shader so that the image is fixed to display properly on the Oculus Rift.

4.2 Classes

Main
: This class import the DICOM files and create the volumetric viewport.

Rift_Pass
: This class apply the barrel distortion shader.

Split_window
: This class create two viewports from a single one and keep them synchronized.

Oculus_middleware
: This class is used as a class used to initializate the Oculus and return hardware related information such as the head pose and textures size.

5 Development journal

Over the course of this stage we had to go through many stages in which we were faced with different problems each one with many alternative solution. In this section will be posted which of these solutions were implemented in the project and the reasons for its choosing.

5.1 Volumetric rendering libraries

The first thing we had to decide was how to handle the volumetric rendering, the possible libraries are linked below

Voreen
: Voreen is an open source rapid application development framework for the interactive visualization and analysis of multi-modal volumetric data sets. It provides GPU-based volume rendering and data analysis techniques and offers high flexibility when developing new analysis workflows in collaboration with domain experts. The Voreen framework consists of a multi-platform C++ library, which can be easily integrated into existing applications, and a Qt-based stand-alone application.

VTK
: The Visualization Toolkit (VTK) is an open-source, freely available software system for 3D computer graphics, modeling, image processing, volume rendering, scientific visualization, and information visualization. VTK also includes ancillary support for 3D interaction widgets, two- and three-dimensional annotation, and parallel computing. At its core, VTK is implemented as a C++ toolkit, requiring users to build applications by combining various objects into an application.

Open Inventor
: Open Inventor® is an object-oriented, cross-platform 3D graphics toolkit for the development of industrial-strength, interactive applications using C++, .NET or Java. Its easy-to-use API, its extensible architecture, and its large set of advanced components provide developers with a high-level platform for rapid prototyping and development of 3D graphics applications.

After a long discussion among ourselves where we studied the pros and cons of each library we decided to use the VTK libraries mainly for these two reason :

- Vooren, while offering a whole framework and many already implemented function, has almost no documentation and we didn't have enough time to learn it from scratch.
- Open inventor is not open sourced.

5.2 Omegalib

Now that we decided to use the VTK as volumetric rendering library, we looked up projects similar to this one so we could use that as a base or as an inspiration.After a little search we found a project that seems to be the silver bullet for our problem, that being Omegalib by UIC Electronic Visualization Laboratory.

The main features that leaded us to try using Omegalib are:

- It implements two rendering libraries, OpenSceneGraph and VTK.
- It natively redirects the output on various devices, among them is the OculusRift.
- Support for a wide range of input peripherals (controllers, motion capture systems, touch surfaces, speech and brain interfaces), through the Omicron toolkit.
- It offers a python wrapper so that the program could be written using a more simple language.

We decided that learning to use the Omegalib was our best bet to make this project easier to extend for future development, so we rewrote our vtk application as a python script. On paper the last thing that needed to be done was assigning the oculus rift configuration to the application, but when we tried doing that it didn't work and the results weren't as expected. We attempted reviewing the whole library code that manage the output control, specifically how it interacted with the oculus rift configuration and then contacted the Omegalib developer on the project mailinglist to see if there was a way to see what the problem was and how to resolve it. Turns out that, while Omegalib implements the VTK, it doesn't support the volumetric rendering, at least it doesn't as of now, it may be implemented in the future.

Because of this any and all plans of using Omegalib for this project fell through so we were back at the beginning and had to go with pure VTK again.