Developing a Technique for the Real Time Visualisation and Manipulation of Volumetric Data in Unity and Virtual Reality

Abstract

There is a constant need for software technologies capable of visualizing, manipulating and managing complex volumetric data. This project's aim is to develop a system for viewing and manipulating Volumetric data in a Virtual Reality (VR) environment, using other work within the field a starting point and developing a new application in the VR medium. Utilising the HTC Vive VR toolset and the graphics capabilities of Unity this project will aim to create an intuitive and robust system for the visualisation of volumetric data.

This project aims to provide a tool for both researchers and novices that will allow immersive, straight forward exploration and manipulation of volumetric data while providing further evidence that VR is promising area for the field of data visualisation. It is intended that the user should be able to navigate and explore volumetric data, while also being able to transform and manipulate it in a natural fashion, allowing them to enhance their understanding and comprehension of the information within the data.

Within this document the current progress of the project will be explained, as well as providing, in detail, an explanation of the motivation, background, and overall goals of this project.

Table of Contents

1. Introduction

This project intends to utilise VR as a novel method for visualising volumetric data, aiding in the comprehension and understanding of said data.

The projects key aims, and objectives will be explained in this section.

1.1. Aims

- Provide a system for representing volumetric scientific data within unity
- Provide a method for converting volumetric data into a mesh that minimises the effect of distortion and artefacts within the entered data
- Provide a VR based method for interacting and working with the processed 3D information
    - Including: scaling and transforming data, measuring data, splitting or "tearing" data.
- Improve upon current visualisation techniques for volumetric data
- Provide a useful tool for researchers at a high standard of software quality

1.2. Objectives

- Construct a robust and efficient system for rendering volumetric data within unity
    - Handle large volumetric data sets of scales up to and exceeding 512 * 512 * 1024 voxels
    - Read and manipulate data stored in several "industry standard" file formats
- Develop a method for converting volumetric data into a mesh
    - Using image processing techniques to manipulate the volumetric data in order to generate a "surface" that skins the volumetric data in an accurate and complete fashion
- Implement different methods for interacting and working with volumetric data
    - Scale and transform data freely with intuitive controls
    - Measure elements of data in terms of scale and density etc.
    - Manipulate volumes, with the ability to split data by "tearing" it in a fashion that uses intelligent algorithmic processing of the data to split the data in a logical and expected fashion
- Work on improvements to techniques at the "state of the art" of rendering volumetric data
    - Investigate computational shaders that can render volumetric data at very high fidelity with a focus on real time performance
    - Provide efficient and computationally inexpensive methods of rendering very large data sets
- Bring together a suite of current tools and approaches to data visualisation under the VR medium

2. Motivation

Data visualisation and information representation is crucial to the scientific process. The quantity of data produced by experimentation and research is ever increasing, and there is a corelating increase in demand for intuitive and effective data visualisation methods that can reduce the high cognitive load required to understand the important components of complex information, *"The advantage of visual data exploration is that the user is directly involved in the data mining process"* [1] – the user gains better insight into the information by being involved as a key participant in the extraction of and interaction with said information.

Traditional 3D computer rendering systems rely on surfaces or "meshes" and rendering this data using surface visualisation techniques [2] without consideration of the "interior" of this information – the volume contained *within* the object. There are, however, several research fields that are concerned with the interior of three-dimensional data and work frequently with volumes that describe the full contents of a 3D object. For this reason, there have in the past three decades been many approaches developed that attempt to render volumetric data in real time [3][4][5][6]. Which aim to allow a user to see not only the surface details of data but also to investigate the interior of the objects and understand their internal structures.

As indicated by Günther: "*Recent innovations in biology ~ are now making large, spatiotemporally complex volumetric data available"* [7]. The data generated by such medical and scientific imaging (e.g. CT Scans, Microscopes etc.) represents a good example of volumetric data in the real world. It is frequently represented as a series of 2D "slices" that are stacked together to produce a three-dimensional image (which can themselves be stored as a vector showing one object in a series of discrete time steps), this can then be rendered using compliant software to allow the user to explore and interpret the data as a 3D volume. The difficulty presented by this problem is twofold. Firstly, current visualisation techniques for this data leave much to be desired in terms of intuitiveness and flexibility – interacting with a three-dimensional object on a two-dimensional screen is inherently challenging. Secondly there is a performance issue, for larger and more complex images real time rendering can become very challenging – in these instances a mesh representation of the object would be desirable – i.e. data which is surface based (composed of triangles) not volumetric. It should be considered, however, that *"Surface reconstruction is an important and established problem in computer graphics and computer vision"*[8] – there is no single approach that is best and this project must aim to evaluate and make use of the most applicable options available.

It is this project's aim to utilise unity and a virtual reality system to enable the user to more easily interact with and understand such higher dimensional data, giving, for example, researchers the ability to easily measure, dissect and inspect three-dimensional data. This is prompted in part by Donalek et al, who explain that *"VR has been shown to lead to better discovery in domains that whose primary dimensions are spatial"* [9]. It is my intention to maximise the ability of the user to benefit from this improved information discovery and comprehension, while giving them access to a suite of tools for manipulation and dissection of the data.

There is a good degree of work available in this field (see subsection 3.1 "Related Work), which provide both building points and inspiration for the work to be undertaken in this project. There are numerous motivators leading to the decision to utilise unity, and VR for data visualisation tasks such as this one, it is the sentiment of many that the field has a high degree of potential: "*Virtual- and augmented-reality tools allow researchers to view and share data as never before"* according to Matthews [10].

3. Related Work

3.1. Immersive and Collaborative Data Visualisation Using Virtual Reality Platforms

In this paper Donalek et al. [9] highlight the benefits of using virtual reality for the visualisation of big data, and explains that data visualisation forms the "*bridge between the quantitative content of the data and human intuition*" [9]. The paper itself focuses on the overall benefit of virtual reality for high dimensional data visualisation and not specifically on using the medium for visualising volumetric data, but the benefits outlined in the work form key motivators for the work of this project.

Donalek et al. outline both data visualisation solutions using "off the shelf" software, as well as the development of a new visualisation system based in Unity titled "*iViz*" (see figure 1). The paper notes that VR visualisation tools *"offer us an opportunity where any scientist can, with a minimal or no cost, have visual data exploration capabilities that are now provided by multimillion- dollar cave-type installations"* [9]. The benefits that Donalek et al outline help to explain why visualisation tools in VR are both desirable and functional, with a high potential for further development in the future.

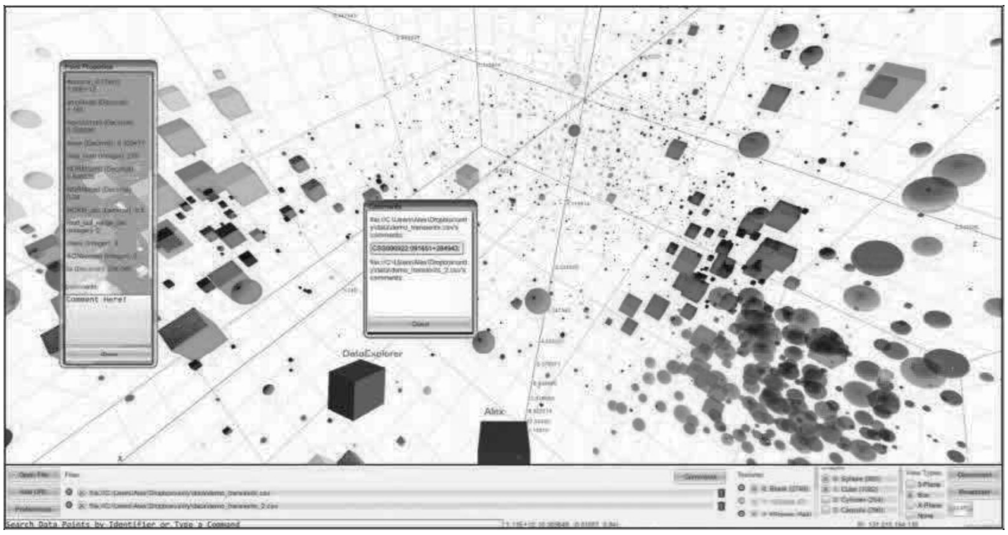


Figure 1: The user interface for iViz [8]

3.2. StudyDesk: Semi-immersive Volumetric Data Analysis

In this paper Stephenson et al. [11] document the functionality of the "StudyDesk" system, an early (2003) hardware and software solution that allows the visualisation of volumetric data from both medical imaging (under a software titled MediDesk) and SONAR. It is the first application of this solution that will be focused on here.

Stephenson et al.'s work predates modern virtual reality hardware by nearly a decade (the Oculus Rift – regarded as the first true consumer VR headset – didn't ship to backers until 2013, with the first 6 degrees of freedom system not being released until 2016). For this reason, the hardware behind the solution is limited, both in terms of performance and ergonomics, but the research still highlights that this field has been in development for some time.

The core of the MediDesk system is based around manipulating the data with a "Pen and pad" and allowing "*the user to drag, scale and cut the volume in a very natural way using the pen and pad metaphor*" [11]. The ergonomics of this system are interesting, though perhaps questionable, the decision to utilise a pen and paper metaphor ultimately results in the user having an experience that doesn't truly reflect the three-dimensional nature of the data they are interacting with.

Within the software developed there are some useful visualisation tools, for example see figure 2, showing the software's ability to simultaneously visualize MRA and CT data.

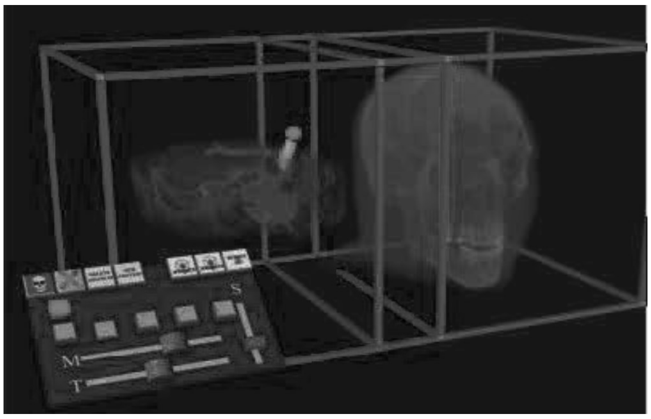


Figure 2: MediDesk simultaneous visualisation of MRA and CT data

The overall adoption of this system was minimal and to the author's knowledge was never made commercially available. Likely it was hampered by the high cost of computational and hardware resources at the time. Now however, with modern, commercially available, virtual reality systems this limitation is not an issue.

3.3. Scenery: Flexible Virtual Reality Visualisation on the Java VM

In this paper Günther et al. [7] introduce scenery, an open source, *"flexible VR/AR visualization framework for the Java VM that can handle mesh and large volumetric data, containing multiple views, timepoints, and color channels"* [7]. Scenery is presented as a robust, platform agnostic visualisation framework that is capable of visualizing volumetric data of very large sizes with high efficiency, all while being user and developer friendly.

Günther et al's goals are more generalised than the aims of this project, with more of a focus on efficient visualisation rather than specific data interactions but their implementations of specific features are exemplary, and their transparency and excellent development standards set a high bar for work in the field.

Scenery not only supports rendering volumetric data but provides complete support for the OpenVR and SteamVR libraries, as well as allowing visualisation of agent-based simulations and other complex data. Scenery has been used to develop multiple applications, including a plugin for the widely used image analysis tool Fiji [12] under the name sciview (see figure 3 for an example visualisation).

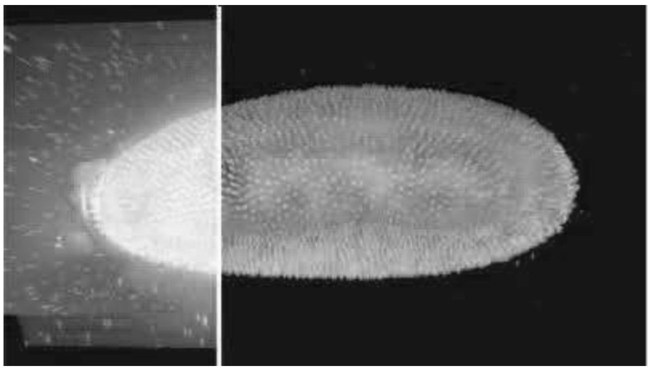


Figure 3: D. melanogaster embryo visualised with scenery / sciview. Günther et al. [7]

3.4. 3D Visualization of Astronomy Data Cubes using Immersive Displays

In this paper Ferrand et al. [13] report on *"an exploratory project aimed at performing immersive 3D visualisation of astronomical data"* [13]. This work is interesting especially as it looks at visualising the data in Unity, and the benefits that the Unity environment can provide for scientific visualisation problems, namely due to its ability to easily integrate different advanced displays including VR headsets and the CAVE environment [14].

As with most fields dealing with big data, astronomy faces an issue of attempting to allow researchers and experts to digest and comprehend "*the large amount of diverse data generated by modern instruments or simulations*" [13]. Understanding this complex, often multi-dimensional data requires moving away from the standard, two-dimensional visualisation techniques left over from the pen and paper era.

Ferrand et al.'s work is different in scope to the aim of this project, specifically in that it targets astronomy data and utilises the CAVE environment – an expensive, specialised visualisation tool. But their work still highlights how Unity can be utilised for a wide variety of industry applications outside of developing games and point out that it serves as an excellent foundation for visualisation tools because of its feature rich implementation of 3D rendering technologies.

3.5. Relation to this Project

As can be seen from the above sections there is a large amount of work and research being done in the field of visualisation in VR, visualising volumetric data sets and using Unity for data visualisation. The key areas in which this project will differentiate itself are:

- A focus on efficiency when dealing with large volumetric data sets within Unity
    - Being able to render large and complex data sets with low computational – focus on real time interaction for the user is critical for user immersion
    - Low load times and high responsiveness
- A focus on user interaction and HCI
    - Haptic feedback when interacting with volumetric data
    - Intuitive controls for transforming and managing data using hand-held controllers

4. Description of Work

This project aims to develop software providing the following key elements:

- A user-friendly method for loading volumetric data from system storage to Unity
    - With caching support to prevent redundant computation
    - The software should support as many industry standard volumetric data types as possible, with the ability to convert each into a standard 3DTexture (Unity's standard for three-dimensional data)
- Methods to manipulate Volumetric data both with a mouse and with VR controllers
    - Translation, scaling and rotation controls that are as intuitive as possible
    - Haptic feedback when working with hand-held controllers
    - Slicing controls for the mouse that utilise slicing planes and allow the user to dissect object along an axis
    - A "tearing" approach to splitting volumes, allowing the user to "grab" the object in VR with controllers and rip it, with the volume splitting in a natural fashion
- A method to convert volumetric data into surfaces using image processing techniques
    - Using thresholding or some similar method to enable the generation of surfaces that accurately skin volumes, with consideration of internal structure
- Methods to calibrate and tune rendering quality to allow real time rendering of volumes of varying complexities, by changing shader properties etc.
    - Potentially different variations on shader models that the user can choose at runtime that prioritise either performance or quality etc.

5. Design

5.1. Software Components

The software as it currently stands has several key components, the current prototype's data flow is explained by figure 4 below.

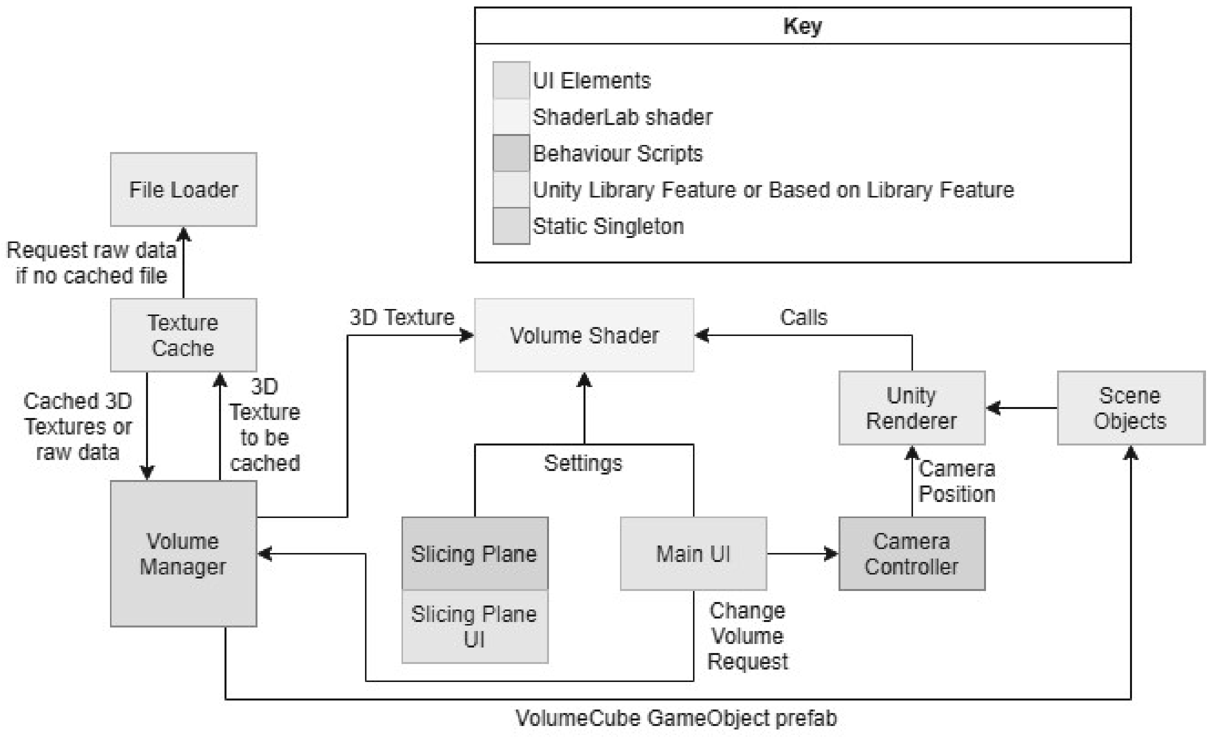


Figure 4: The current prototype and data flow within it

Each component's task is as follows:

- **Volume Manager**
  - This is a static singleton behaviour script that is responsible for managing the behaviour of volumes at runtime. It makes the necessary calls to load the volume as requested and is also responsible for acting as an arbiter for the application of settings to the shader.
- **Texture Cache and File Loader**
  - These are sub-components of the volume manager that make use of Unity's excellent Asset database system to manage volumes and their equivalent 3DTextures at runtime. Each time a new volume is requested the system checks to see if a matching 3DTexture exists within the cache folder, if one is found then this is loaded immediately, otherwise the raw image data is loaded by the volume manager and compiled into a 3DTexture object which is then cached in the cache folder.
- **Slicing Plane and Slicing Plane UI**
  - The slicing plane in the above diagram represents the script that is attached to the slicing plane scene object. This is responsible for maintaining information about the slicing plane (orientation, position), and then passing this information to the volume shader via the Volume Manager's arbitration methods.
  - The slicing plane's UI is another component attached to the scene object, this is responsible for showing the user the rotation and translation controls at run time, as well as managing mouse events to move them

- **Main UI**
    - The main UI in this case are the screen space UI elements – including controls for the shader like density and sampling quality, as well as colour channel toggles, volume selection menu etc.
- **Camera Controller**
    - The camera controller behaviour is a simple behaviour script that allows the user to navigate around the scene using a "fly cam" controlled by the mouse and keyboard
- **Unity Renderer and Scene Objects**
    - The Unity renderer is Unity's core rendering system – their Scriptable Render Pipeline (SRP) [15], a high performance, highly customisable 3D graphics renderer
    - The scene objects are a dynamic tree hierarchy of *GameObjects*, the focus of Unity's component-based system, a *GameObject* is a world object that can have various components attached to it, e.g. a mesh renderer or a behaviour script. Unity uses these objects to create and render scenes.
- **Volume Shader**
    - The shader responsible for rendering volumes at runtime, designed to run on the GPU in order to provide massively parallel speedup to the problem of rendering highly complex volumes. A task that would be impossible with the low core count and slow floating-point arithmetic of a conventional CPU. (The concrete details of this are explained further in sub section 7.2.1, Volume Shader)

6. Methodology

6.1 Software Development Approach

The project is being developed with a vague focus on applying an agile methodology. It is loosely structured into week-long "sprints" concluding in meetings with the project supervisor to discuss progress and set goals for the next week. This approach was chosen as it works best for a one-person development team, enabling a focus on dynamic application development and the ability to integrate features that become feasible and interesting on the fly.

Rapid Application Development (RAD) has been applied to this project, i.e. there has been a focus on developing functional prototypes through the project lifecycle, with a series of iterative prototypes being built on top of one another to eventually lead to the final product. This is outlined in the project plan (shown in sub section 8, progress update).

6.2. Versioning

In order to maintain a complete version history the Unity application has been version controlled using GitHub and the GitKraken [16] desktop client since the start of the project. This aids the RAD approach significantly, as development is shown in clear steps, with each commit forming a series of stepping-stones between stable versions. Similarly, if a change causes undesirable or dangerous side effects said change can be immediately rolled back without endangering the project.

The use of GIT for version control is an industry standard: "*Git is the most commonly used version control system"* [17], and it was chosen for this project for that reason.

7. Implementation

7.1. Why Unity?

Unity is a full featured game engine and development tool. Although this project is not interested in gaming, the rich options for 3D graphics, cross platform support and visual processing given by modern game engines makes using one seem like an obvious choice.

Unity was chosen for this project as:

- It has excellent graphics capabilities
- It can support building for numerous platforms, including VR headsets, Windows, Mac and Linux
- It has high usability for developers
- It has an excellent suite of options for developing shaders that are capable of handling 3DTextures
- It has a native representation for volumetric data in its 3DTexture data type (though no native method of rendering these)

7.2. Current Prototype

The project is still very much in the development phase, but several novel and successful features have been developed, these will be explained within this section. The current version of the prototype at the time of writing is entirely PC based, with no VR support yet included (although implementation is planned imminently).

Features currently available include:

- Loading of volumetric data stored as sequences of "slices" – image files stored sequentially in a folder
    - Most standard image types are supported: TIFF, JPEG, PNG, etc.
- Selecting which data to display at run time via an on screen drop down
- Novel caching of volumetric data to both secondary storage (long term caching between software usage) and to RAM (short term caching at run time) to improve load times drastically over handling raw image data
    - This is a feature that is lacking in many other visualisers – with very long loading times being the norm in most cases
- Visualisation of volumetric data in real time using a volumetric shader
    - Based on several examples and extended to form a novel implementation that is of high efficiency and supports slicing planes and toggling colour channels at run time
- Manipulation of the volumetric data through:
    - Changing the apparent density of the data using a slider
    - Changing the "spacing" (stretching) of the data
    - Toggling different colour channels within the data (see figure 5.4)
    - Manipulating a "slicing plane" that can bisect volumes to show internal data (see figures 5.2 and 5.3) – using controls that are novel and represent better HCI than many currently available visualizers
- Manipulation of sampling quality used within the volume shader to control the quality versus performance trade off

See figures 5.1-5.4 overleaf for demos of the current software.

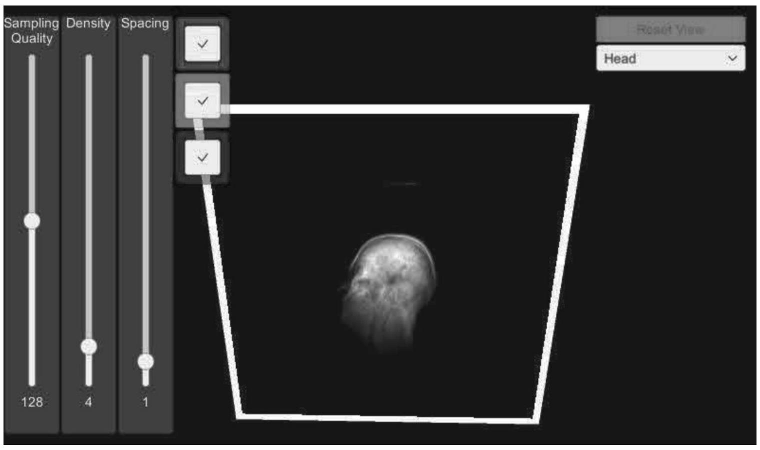


Figure 5.1: The main screen on start, the cutting plane is initialised at the centre of the volume. (Only the right side of the head, in front of the volume, is rendered)

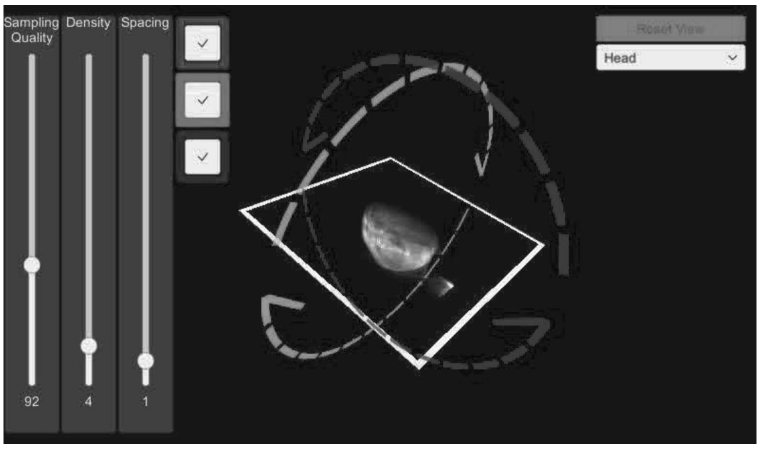


Figure 5.2 The cutting plane has been rotated using the in-world rotation controls, manipulated with the mouse

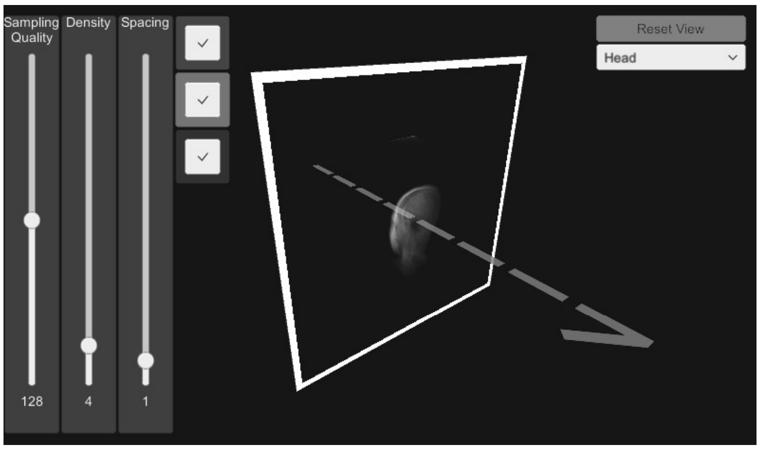


Figure 5.3 The cutting plane has been translated using the in-world translation control, manipulated with the mouse

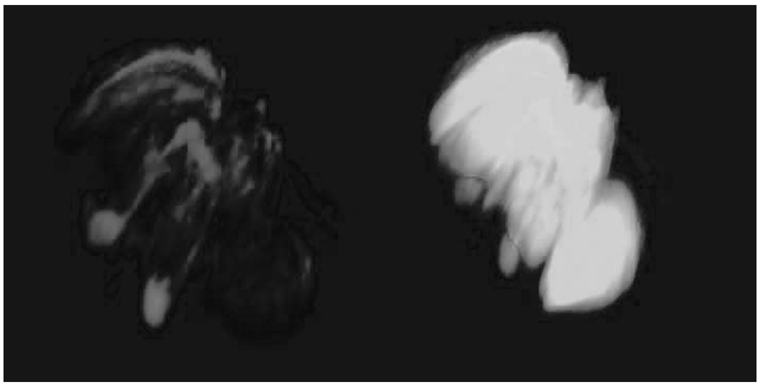


Figure 5.4 A volume with separate information in the red and green channels. Left: Just red shown, Right: red and green shown.

7.2.1 The Volume Shader

A key component of the software is the shader, responsible for taking the 3D texture and using GPU to render it in real time to the screen. The approach employed thus far is relatively simple but provides acceptable results and renders very rapidly (with data sizes of up to 512 * 512 * 1024 pixels being displayable even on integrated graphics on a laptop processor – Intel Core i3 at 2.20Ghz with Intel UHD Graphics 620 – at framerates of at least 30fps, with smaller data sizes giving framerates of greater than 60fps).

In order to render a volume, the shader takes in the 3D texture information as a "Sampler3D" – a buffer that is loaded onto the GPU's memory and made available for each instance of the fragment shader to analyse to determine each fragment's colour.

The approach of the current shader relies on a technique known as "ray marching" where each fragment (discrete pixel) of the rendered volume is calculated based on the results of passing a ray through the volume and taking samples at discrete intervals. For each fragment of the rendered cube as seen from the camera's position a simple algorithm is run, that can be understood intuitively as follows (see also figure 6 for visual reference):

- A virtual ray is sent from the camera out to the position of the fragment in world space
- Where the virtual ray passes through the volume two values are calculated; *D*: the total length of the virtual ray *inside* the volume and *d*: where *d = D / SampleQuality*. Starting at the end of the real (internal) ray the algorithm "marches" (hence ray marching) backwards through the volume, moving a distance of *d* on the path of the ray each time, and taking a sample from the texture at each corresponding point.
- At each step the algorithm checks for the location of the slicing plane, and if it is found that the current point is "behind" the slicing plane, then its colour values are ignored and are not included in the final fragment
- The final colour for each pixel then becomes the sum of each sample taken from inside the volume, giving the impression (at least with high enough sample quality) of an entire 3D object being represented.

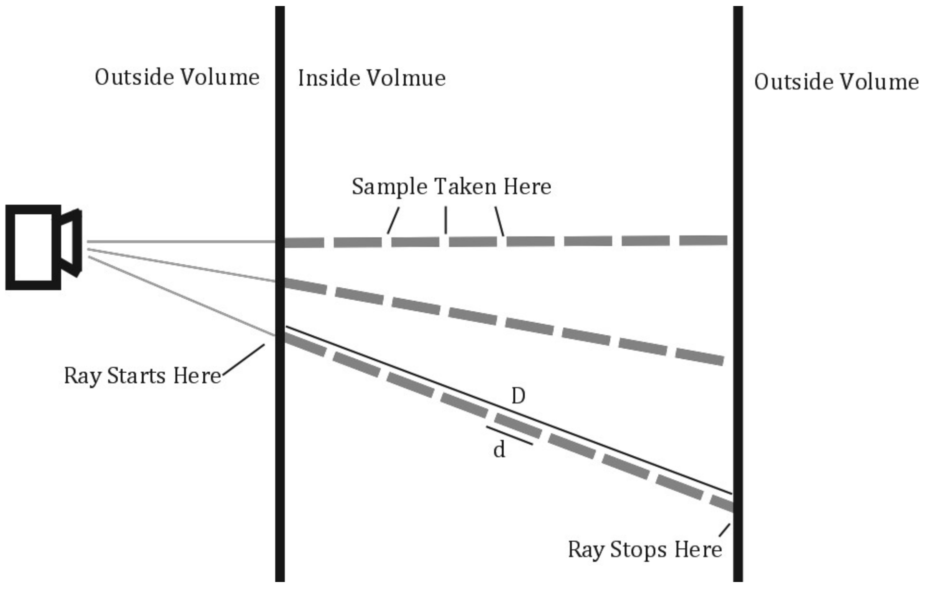


Figure 6: The Raymarching Algorithm for Rendering Volumes

This algorithm has several key benefits. Firstly, is speed, the technique is fairly simple and can be run on a GPU efficiently with a minimal (but present) performance hit as volume size increases. Secondly the algorithm has an inherently encoded quality control, which can be changed at runtime in order to tune performance for the size of data set and quality of machine running the software (see figure 7 for a comparison of two different sampling qualities). It should be noted though that at very low sampling qualities the result is unlikely to be understandable, nor useful. Similarly, it should be remembered that there is no point setting sampling quality so high as to exceed the granularity of the input volume (where ray marching steps are smaller than an individual pixel), as there will be no further quality benefit. Thirdly the algorithm allows efficient implementation of slicing planes that can arbitrarily occlude elements of the volume at any angle and position.

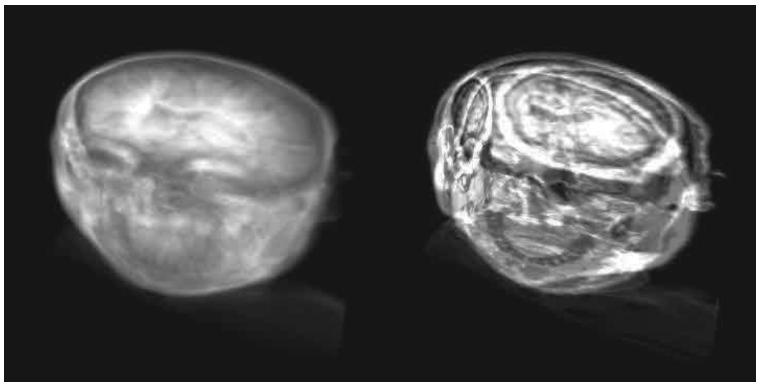


Figure 7: A comparison of the effect of sampling quality on rendering, with quality of 128 on the left and 7 on the right.

Generally, the results of this algorithm are acceptable, though it could be beneficial to investigate an improvement to this method in the future, aiming to perhaps reduce the "fuzziness" of rendered volumes.

7.3 Encountered Challenges

Currently the project has progressed without any major issues, though some elements have proved quite challenging.

Firstly, the development of controls to efficiently move the cutting plane ended up being a surprising challenge as there are no examples that the author could find on which to base these controls, and so the implementation is entire novel or "from scratch." The challenge of converting an on-screen mouse position into believable manipulation of an object in 3D space is entirely non-trivial, but the current prototype's implementation is both intuitive and responsive.

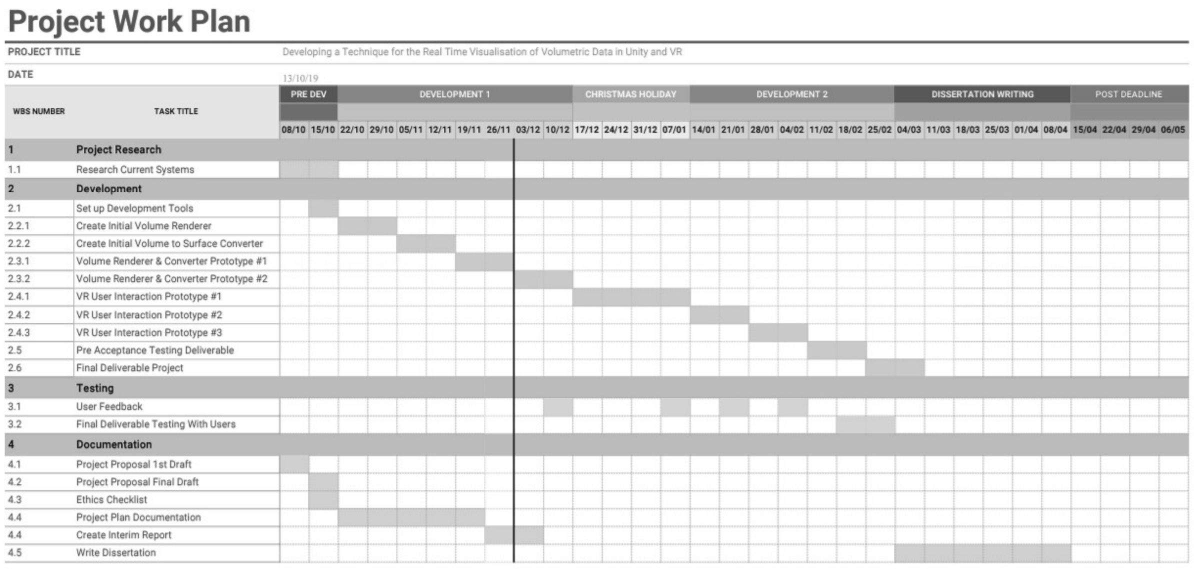
Secondly, developing a shader that was capable of rendering volumes with an appropriate level of detail, speed and desired features required bespoke code. The approach chosen was based on examples found online, but was adapted to increase efficiency, implement the slicing plane and allow colour channel toggling.

8. Progress Update

8.1 Project Management

The project has overall so far has had good time management, and good progress has so far been made. Thus far there have been no major setbacks, and the speed of development has matched with what is expected for the project at this time.

Time distribution hasn't gone entirely as expected, with some features (for example implementing the raymarching shader) taking less time than expected and some features (for example implementing the controls for the slicing plane) taking more. But the overall result has been appropriate and so there are no major issues. See figure 8 for an indication of progress based on the work plan (the black line indicates the rough date at time of submission).



Project Work Plan

PROJECT TITLE: Developing a Technique for the Real Time Visualisation of Volumetric Data in Unity and VR

DATE: 13/10/19

WBS NUMBER	TASK TITLE	PRE DEV		DEVELOPMENT 1								CHRISTMAS HOLIDAY				DEVELOPMENT 2							DISSERTATION WRITING						POST DEADLINE			
		08/10	15/10	22/10	29/10	05/11	12/11	19/11	26/11	03/12	10/12	17/12	24/12	31/12	07/01	14/01	21/01	28/01	04/02	11/02	18/02	25/02	04/03	11/03	18/03	25/03	01/04	08/04	15/04	22/04	29/04	06/05
1	**Project Research**																															
1.1	Research Current Systems	▨	▨																													
2	**Development**																															
2.1	Set up Development Tools		▨																													
2.2.1	Create Initial Volume Renderer			▨	▨																											
2.2.2	Create Initial Volume to Surface Converter					▨	▨																									
2.3.1	Volume Renderer & Converter Prototype #1							▨	▨																							
2.3.2	Volume Renderer & Converter Prototype #2									▨	▨																					
2.4.1	VR User Interaction Prototype #1											▨	▨	▨	▨																	
2.4.2	VR User Interaction Prototype #2															▨	▨															
2.4.3	VR User Interaction Prototype #3																	▨	▨													
2.5	Pre Acceptance Testing Deliverable																			▨	▨											
2.6	Final Deliverable Project																					▨	▨									
3	**Testing**																															
3.1	User Feedback										▨				▨		▨		▨													
3.2	Final Deliverable Testing With Users																				▨	▨										
4	**Documentation**																															
4.1	Project Proposal 1st Draft	▨																														
4.2	Project Proposal Final Draft		▨																													
4.3	Ethics Checklist		▨																													
4.4	Project Plan Documentation			▨	▨	▨	▨	▨																								
4.4	Create Interim Report								▨	▨																						
4.5	Write Dissertation																						▨	▨	▨	▨	▨	▨				

Figure 8: Work plan with rough progress indicator (black line)

In its current state the project has:

- A good initial prototype with promising features and good room for further development
- All early documentation finished
- A mostly completed research section

8.2 Contributions and Reflections

The project has progressed well thus far. Key goals have been met, and there exists a good plan for the project's continued development. The features that are so far implemented meet and sometimes exceed those that are available in a wide range of viewing systems that are available currently, and its computational performance and light weight nature is exceptional.

For the remainder of the project more attention must be paid to developing features that utilise VR and its benefits to increase data immersion and comprehension through haptic feedback when touching and transforming the data. With a key aim of providing the user with some novel method of data manipulation – this will likely take the form of a system for "splitting" data by tearing it with controllers, using some intelligent algorithm to split the data along a logical "seam".

Testing must be performed as soon as possible to validate the current work and ensure that it meets the standards of researchers and the general public alike, and to ensure that the project continues to develop features that help it to stand out and increase in functionality and usability.

9. Bibliography

[1] D. A. Keim, ‘Information visualization and visual data mining’, *IEEE Transactions on Visualization and Computer Graphics*, vol. 8, no. 1, pp. 1–8, Jan. 2002.

[2] J. Congote, A. Segura, L. Kabongo, A. Moreno, J. Posada, and O. Ruiz, ‘Interactive Visualization of Volumetric Data with WebGL in Real-time’, in *Proceedings of the 16th International Conference on 3D Web Technology*, New York, NY, USA, 2011, pp. 137–146.

[3] D. R. Ney, E. K. Fishman, D. Magid, and R. A. Drebin, ‘Volumetric rendering of computed tomography data: principles and techniques’, *IEEE Computer Graphics and Applications*, vol. 10, no. 2, pp. 24–32, Mar. 1990.

[4] E. K. Fishman *et al.*, ‘Volumetric rendering techniques: applications for three-dimensional imaging of the hip.’, *Radiology*, vol. 163, no. 3, pp. 737–738, Jun. 1987.

[5] H. Rusinek, M. R. Mourino, H. Firooznia, J. C. Weinreb, and N. E. Chase, ‘Volumetric rendering of MR images.’, *Radiology*, vol. 171, no. 1, pp. 269–272, Apr. 1989.

[6] Y. Jian, K. Wong, and M. V. Sarunic, ‘Graphics processing unit accelerated optical coherence tomography processing at megahertz axial scan rate and high resolution video rate volumetric rendering’, *JBO*, vol. 18, no. 2, p. 026002, Feb. 2013.

[7] U. Günther *et al.*, ‘scenery: Flexible Virtual Reality Visualization on the Java VM’, *arXiv:1906.06726 [cs]*, Aug. 2019.

[8] J. Chen, D. Bautembach, and S. Izadi, ‘Scalable Real-time Volumetric Surface Reconstruction’, *ACM Trans. Graph.*, vol. 32, no. 4, pp. 113:1–113:16, Jul. 2013.

[9] C. Donalek *et al.*, ‘Immersive and collaborative data visualization using virtual reality platforms’, in *2014 IEEE International Conference on Big Data (Big Data)*, 2014, pp. 609–614.

[10] D. Matthews, ‘Virtual-reality applications give science a new dimension’, *Nature*, vol. 557, pp. 127–128, Apr. 2018.

[11] P. Stephenson, L. M. Encarnação, P. Branco, J. Tesch, and D. Zeltzer, ‘Studydesk: Semi-immersive Volumetric Data Analysis’, in *Proceedings of the 1st International Conference on Computer Graphics and Interactive Techniques in Australasia and South East Asia*, New York, NY, USA, 2003, pp. 251–252.

[12] J. Schindelin, *Fiji: an open-source platform for biological-image analysis*, vol. 9(7). 2012.

[13] G. Ferrand, J. English, and P. Irani, ‘3D visualization of astronomy data cubes using immersive displays’, *arXiv:1607.08874 [astro-ph]*, Jul. 2016.

[14] ‘Visbox, Inc. | CAVE Systems’. [Online]. Available: http://www.visbox.com/products/cave/, http://www.visbox.com/products/cave/. [Accessed: 28-Nov-2019].

[15] Unity Technologies, ‘Scriptable Render Pipeline | Learn about Unity SRP’, *Unity*. [Online]. Available: https://unity.com/srp. [Accessed: 28-Nov-2019].

[16] ‘Free Git GUI Client - Windows, Mac & Linux | GitKraken’, *GitKraken.com*. [Online]. Available: https://www.gitkraken.com/. [Accessed: 28-Nov-2019].

[17] ‘What Is Git & Why Should You Use It? | Noble Desktop Blog | Tutorials, Resources, Tips & Tricks’. [Online]. Available: https://www.nobledesktop.com/blog/what-is-git-and-why-should-you-use-it. [Accessed: 28-Nov-2019].