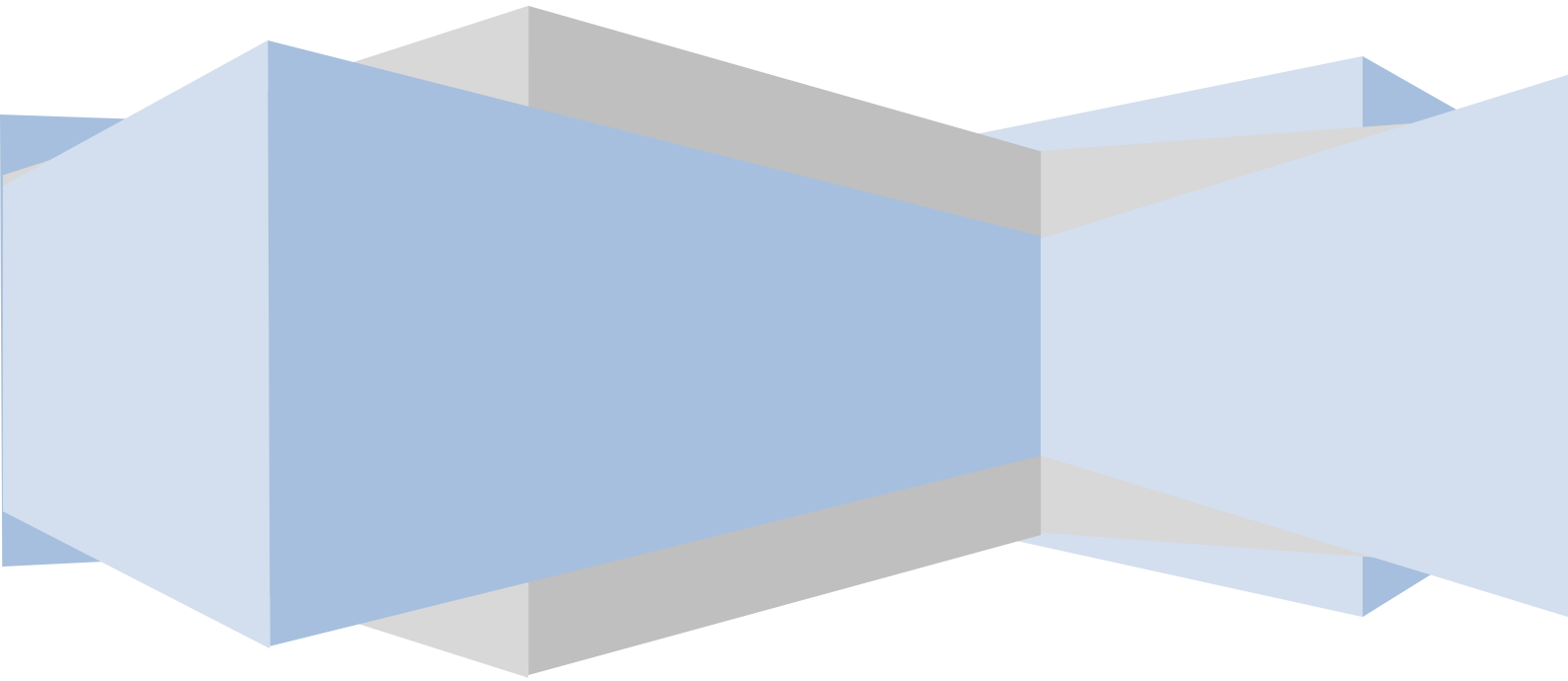
Volumetric Rendering of Terrain with caves

Karl Mitson (u0410589)

Supervisor: Zhijie Xu

Computer Games Programming



Abstract

Terrain is a commonly used feature in modern games with external environments. Currently terrain is represented by a 3D surface mesh; there are also several different algorithms to provide level of detail controls to the terrain. This allows for highly detailed and large terrain with very good performance. However, using a surface mesh only provides a representation of the surface of the terrain and not the details within the terrain such as caves.

The purpose of this project is to research Volumetric Rendering and implement a prototype application which demonstrates its use to render terrain. The application will also demonstrate the ability to represent caves in terrain by providing simple volume deformation tools.

This document provides an overview of the different volume rendering algorithms, other possible uses for volume rendering, current uses of volume rendering and also implementation details for the product that has been created as part of this project.

This project will demonstrate a new technique that could provide a much more detailed and flexible approach to rendering terrain in games. This project will also identify possible performance issues and future improvements.

Contents

Table of Figures

Table of Tables

1. Introduction

1.1. Overview

Terrain is a key element to any game with an external environment. Currently most games represent terrain using a 3D surface mesh. This approach has been continuously improved and is very flexible. However, certain features such as caves and terrain deformation are still very restricted when using a surface mesh.

The main aim of this project is to research and evaluate the possibility of using Volumetric Rendering to represent terrain in a game; primarily this will be the ability to represent caves within an object and also deformation of a volume.

This project will appeal to games programmers who are interested in terrain rendering techniques and also programmers looking for alternative uses for volumetric rendering.

1.2. Objectives

- Research into volumetric rendering and choose the best method for rendering and lighting
- Create a prototype application that demonstrates the use of volumetric rendering to create caves
    - The volume object should be coloured and illuminated correctly
    - The volume object must demonstrate the advantages of using volumetric rendering rather than a 3D mesh
    - The volume object must allow for user modification to demonstrate deformation
    - The volume data format must compress the volume data to improve loading times and file sizes
- Research volume rendering optimisation techniques and data compression to provide extra flexibility and performance
- Optimise the application to allow it to be used along with other games technologies
- Evaluate the result and state whether volumetric terrain is feasible with current generation graphics technology

1.3. Gantt Chart

A Gantt chart has been used to manage project deadlines and time allocations.

ID	Task Name	Duration	Start	Finish
1	Report Work in progress	135 days?	Mon 27/10/08	Fri 01/05/09
2	Interim Report	61 days?	Mon 27/10/08	Mon 19/01/09
3	Research Volume Rendering	15 days?	Mon 27/10/08	Fri 14/11/08
4	Develop basic game framework	13 days?	Mon 27/10/08	Wed 12/11/08
5	Design and implement user interface	8 days?	Mon 17/11/08	Wed 26/11/08
6	Develop Volume File Load/ Save	13 days?	Mon 01/12/08	Wed 17/12/08
7	Research Volume optimisations	30 days?	Mon 17/11/08	Fri 26/12/08
8	Create Poster	17 days?	Fri 26/12/08	Mon 19/01/09
9	Create Volume Renderer	13 days?	Mon 22/12/08	Wed 07/01/09
10	Create Volume Modification Tools	33 days?	Mon 12/01/09	Wed 25/02/09
11	Optimise Volume Modification	13 days?	Mon 02/03/09	Wed 18/03/09
12	Mouse Picking With Volume	13 days?	Mon 23/03/09	Wed 08/04/09
13	Heightmap Loading and Apply to volume	8 days?	Mon 13/04/09	Wed 22/04/09
14	Application Polish and Bug Fixing	18 days?	Mon 06/04/09	Wed 29/04/09
15	OpenQVis Save	3 days?	Thu 09/04/09	Mon 13/04/09

Figure 1- Gantt chart Tasks

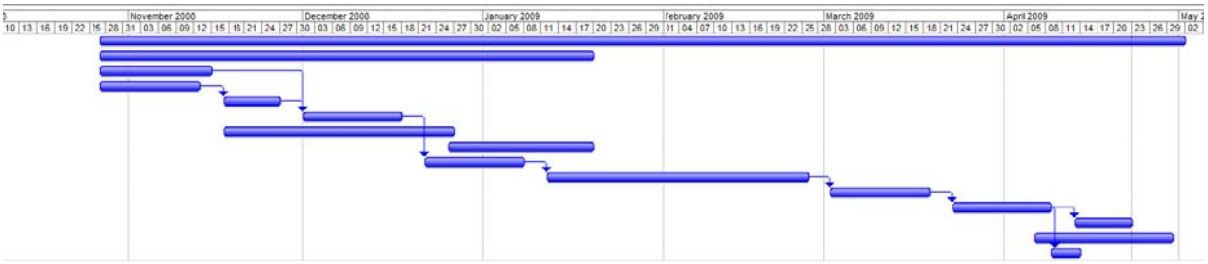


Figure 2 - Gantt chart

2. Research and Investigation

2.1. What is a Volume?

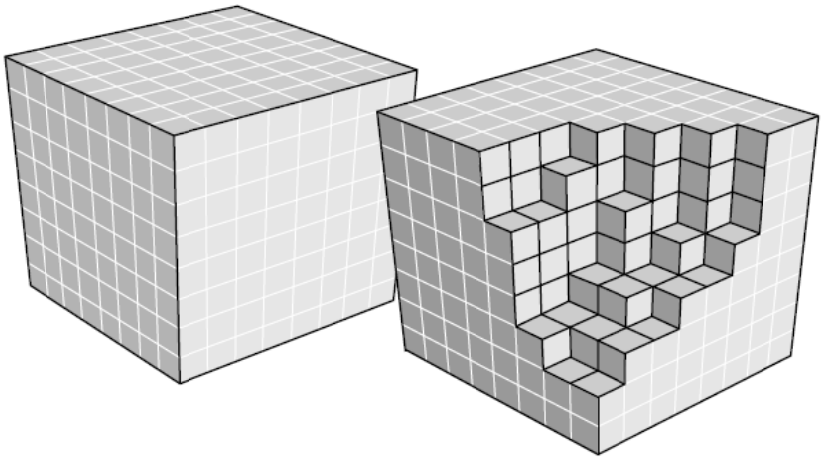


Figure 3 - Volume object (ENGEL, Klaus et al., 2004)

A volume is a 3- dimensional data format where each element in the dataset represents a unit of space. Each element is known as a voxel and can be represented using any value, such as light intensity, density, gradient etc. The value of a voxel is usually converted by a shader program from its default value to an RGBA colour. This technique is known as a transfer function and is usually a 1-Dimension texture that is used as a lookup table.

Volume data can be rendered using many different techniques because of the flexibility of the voxel data and also based on the applications needs or system requirements. Each volume rendering technique has advantages and disadvantages and provides different levels of quality and performance.

2.1.1. Texture Based Volume Rendering

Texture based volume rendering is a technique that uses voxel data to generate 2D texture stacks or a 3D volume texture. The textures are then applied to proxy geometry (commonly a 3D plane) which is oriented based on each of the primary world axis, the object orientation or the camera orientation. Proxy geometry is a primitive that is used to represent a volume object. The proxy geometry that is used for the volume rendering is based on what data type is being used for the volume texture(s). For example, camera aligned planes would not suit a volume renderer that uses 2D texture stacks because a camera aligned plane would need to read from several separate textures to be able to retrieve all the data required.

For 2D texture based volume rendering the application will usually generate 3 stacks of texture maps. Each of the 3 stacks represents a major world axis (X, Y, and Z). When rendering the volume, the application will choose which axis to use based on the camera viewing direction. This is because if the volume was rendered along only one axis then there would be a point where the camera viewing angle is such that none of the rendered slices can be seen at all (e.g. a slice is not visible from its sides). The slices are rendered from back to front to allow for the correct blending effect.

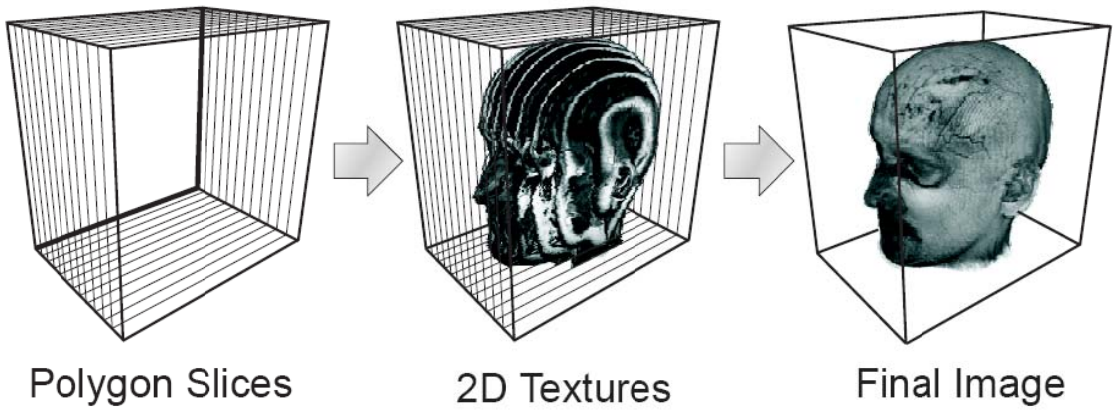


Figure 4 - Volume rendering using 2D textures and object aligned quads (ENGEL, Klaus et al., 2004)

The downfall to this technique is that there is a noticeable transition when the application changes the stack that is currently being rendered. Also the 2D texture stack approach does not provide a flexible sample rate by default. This is because the amount of texture slices is fixed based on the amount of voxels in the data set. A solution to this problem is to use linear interpolation to generate a texture that can be used between 2 neighbouring slices. This provides a flexible sample rate but also requires that a separate shader or technique is used to render the additional slices.

This technique essentially allows for tri-linear interpolation of the data from the volume since bi-linear interpolation is used for the 2D texture itself and the addition of the linear interpolation between the slices. This technique provides a flexible sample rate but still has the noticeable transitions when the active stack is changed.

For 3D texture based volume rendering the application will usually generate 3D planes that are sized to fit within a bounding box that represents the size of the volume. The amount of planes that are used is based on a set sampling rate which is highly flexible because the 3D texture uses tri-linear interpolation automatically when retrieving texture data (using UVW texture co-ordinates). The planes are oriented to always align facing towards the camera because this is similar to the sampling method used by the ray casting algorithm. The main advantage of this technique is that the application can store all the volume data in a single 3D texture. This requires less memory than the

2D texture stacks and also provides the ability to read the data from the texture by simply using a 3D texture co-ordinate.

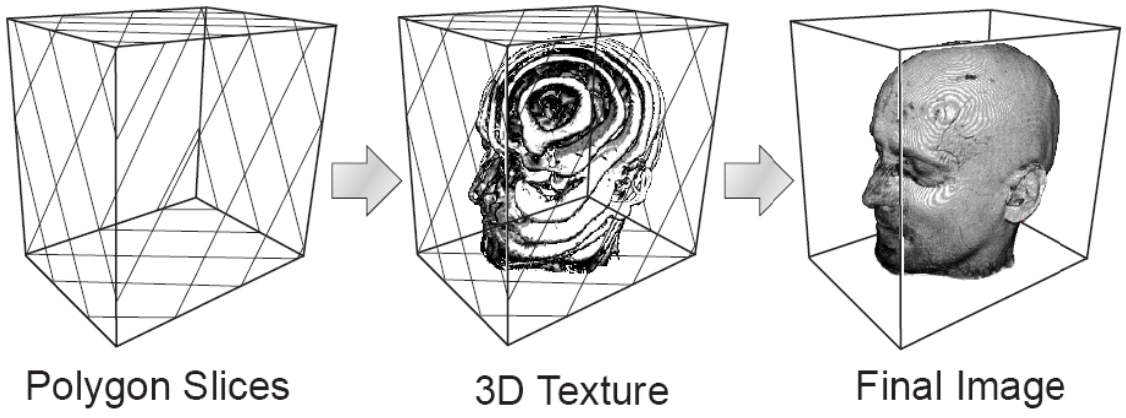


Figure 5 - Volume rendering using 3D texture and screen aligned quads (ENGEL, Klaus et al., 2004)

2.1.2. Ray Casting

The ray casting approach to volume rendering involves casting a ray from each pixel of the screen until it collides with the volume bounding box. Once a collision is found the shader program samples forwards through the volume at a set step size (sampling rate). The volume data is stored in a 3D texture and is uploaded to the GPU using a shader program. The value that is stored inside this texture for each voxel is retrieved using the tex3D function; this function uses the UVW co-ordinate of the plane to determine where to retrieve the texture data. The value retrieved is then transformed using a transfer function which converts the voxel data into an RGBA colour.

This process is repeated until the ray leaves the volume bounding box or until a terminate condition is met. An example of an early ray termination technique is to terminate the ray once the alpha value has reached a certain value (e.g. 0). There are also techniques such as empty space skipping which provide for extra performance. Empty space skipping is a technique where if the alpha value is below a certain value then the voxel is classed as empty and the colour calculation is skipped. Ray casting is the most detailed volume rendering technique, however it is also expensive to calculate. The sample rate for ray casting can also be different for each ray that is cast which provides extra performance flexibility.

2.1.3. Technique Comparison

Technique	Pros	Cons
Axis Aligned Planes (2D Texture Stacks)	Fast, Easy To Code	High Memory Usage, Visual Artifacts
Screen Aligned Planes (3D Texture Stacks)	Fast, Flexible Sample Rate	Visual Artifacts
Ray casting	High Quality, Flexible Sample Rate	High GPU Bandwidth Usage

Table 1 - Volume Rendering Technique Comparison (ENGEL, Klaus et al., 2004)

2.2. How is Volume data generated?

Volume data is used in various different research areas. In most cases volume data is scanned using scanning equipment (e.g. a CT scan). The scanner for a CT scan uses X-rays to cross section the object.

However, for this project scanned data cannot be used because there is no scanned data available to represent terrain with caves. This means that the data will have to be created manually. To simplify this, the product will contain various volume modification tools to allow the user to create their own data set.

2.3. How is Volume data currently used?

As computer graphics cards become more powerful, volume data is becoming more frequently used to recreate natural phenomena in games.

2.3.1. Clouds

Volumetric data can be used to represent clouds. The main benefit of this is that the cloud can be represented as if it was a gas or fog style mass rather than a billboard sprite or particle. This means that when the camera moves through the volume cloud, the cloud maintains its foggy density throughout the entire cloud (a particle or billboard would rotate or scale and eventually not be on screen as the camera passes it).

Volumetric clouds are a growing area of the game industry as more games are using them to provide more realistic sky scenes. There are also several SDKs which provide the ability to easily implement volumetric clouds into a game or application.

Figure 6 - A cloud generated using volumetric rendering. (SCHPOK, Joshua et al., 2003)

(SCHPOK, Joshua et al., 2003) - Demonstrates a technique of cloud animation and rendering for interactive systems and offline renderers that uses volumetric rendering to render clouds. The clouds are generated based on a user defined configuration. This configuration defines the noise parameters and the external forces that are applied to the cloud object. The clouds are then rendered using several simple primitives and then applying the configured noise and forces to a volume contained within them.

Figure 7 - Sky generated by Weather SDK. (Simul » The Simul Weather SDK, 2009)

(Simul » The Simul Weather SDK, 2009) - Offer a Weather SDK which allows users to create volumetric clouds using a flexible API. The API is designed for use with games and is capable of rendering and animation of an entire sky scene at run time. This demonstrates that graphics

hardware has progressed enough to allow for a large amount of volume data to be rendered as part of a game engine.

(3D Cloud and Sky Visual Simulation: SilverLining by Sundog Software, 2009) – Also offer an SDK for games which allows for real time simulation and animation of volumetric clouds. The SDK also offers solutions for precipitation and natural scene lighting.

2.3.2. Lighting

Volume data can be used to simulate several different lighting phenomena. This allows for games to accurately simulate the effect given by Crepuscular rays (rays of light that pass between clouds). The volume data is used to represent the light intensity at a given location. The volume data may also be affected by the density of other volume objects.

If volumetric clouds are used then the lighting of these clouds is affected by light refraction within the volume itself. This allows the cloud to be lit from within the volume itself.

Figure 8 - An image demonstrating the Crepuscular rays effect (Crepuscular rays - Sunrays - Cambridge, 2009)

In games the simulation of Crepuscular rays has been simplified greatly by using an algorithm similar to the radial blur algorithm. This provides the ray effect of the Crepuscular rays but volumetric clouds are still required to provide the realistic lighting within clouds. The disadvantage of the radial

blur based Crepuscular rays algorithm is that rays are only cast if the camera is looking towards the light source.

2.3.3. Games

There are several games that have been created that take advantage of the features of volumetric rendering. Usually it is used for lighting or shadowing, however there are some games that use volumetric rendering as a much larger part of the game (e.g. Voxelstein 3D). Volume rendering is becoming more commonly used within games as graphics rendering technology progresses.

Voxelstein 3D is a first person shooter, based on Wolfenstein 3D by ID Software. The entire game world is created using Voxels using an engine called Voxlap. Voxlap allows for game worlds to be easily created using their editor tools. Voxlap also supports physics on voxels and various rendering options and other common engine features.

The use of volume rendering in Voxelstein 3D allows players to easily destroy any object in the game world. This is because volume data is simpler to deform than a mesh.

Figure 9 – Voxel based environment in Voxelstein 3D (Voxelstein 3D, 2009)

World in Conflict and Crysis use both volumetric clouds and volumetric lighting using their Direct X 10 rendering engines. The pictures below demonstrate how realistic a scene can look when using volumetric rendering.

Figure 10 - World in Conflict volume lighting comparison. (The State of DirectX 10 - Image Quality & Performance - HotHardware, 2009)

The image above shows realistic Crepuscular rays between the clouds and also realistic lighting of the clouds. The Direct X 10 version provides more accurate lighting.

Figure 11 - Crysis volume lighting (IGN: Crysis Screenshots (PC) 2131789, 2009)

In the image above from Crysis, the lighting is calculated using volumetric lighting which allows for the realistic visualisation of light rays without needing the light source in view. This produces an eerie environment and makes it seem as though the air has some kind of density (such as mist).

Doom 3 uses volume shadows (also known as stencil shadows) to provide realistic shadowing. This provides realism to the environment and also helps to set the horror style of the gameplay.

Figure 12 - Volume shadowing in Doom 3 (Doom 3 Screens for PC at GameSpot, 2009)

2.3.4. Medical applications

Medical applications such as the CT scan and MRI scan use volume data. This allows medical staff to view detailed information from within the patient without having to operate.

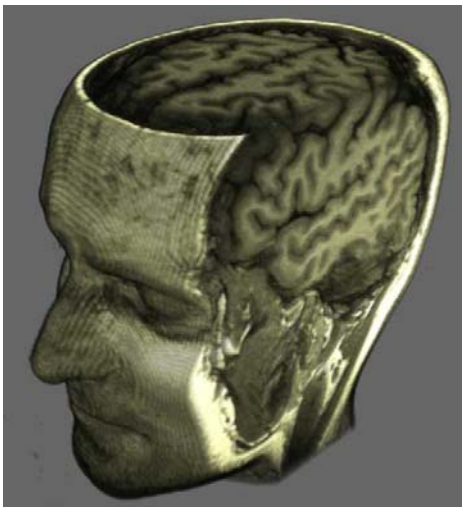


Figure 13 - CT scan of a human head (The OpenQVis Project at sourceforge.net, 2009)

In medical applications the volume rendering technology must produce high detailed images but it is also only required to provide interactive speeds or still images. This means that higher quality rendering algorithms, such as ray casting, can be used.

2.4. What benefits are there for using Volume data?

Volume data has many different benefits, below is a list of benefits for games:

- Flexible data format - Voxels have a flexible data format, this allows for volume data to represent almost anything in the game world. For example an RGBA colour could be stored into the volume data, the RGB values could represent the normal direction for this voxel and the Alpha channel could be used to process the transfer function colour.
- Easy modification/deformation - Volume data is easy to deform and modify compared to a 3D mesh because the data itself is a 3D array of values (RGBA, BYTE, etc). Also, no extra processing is required to close the volume when voxels are added or removed since the volume data represents the mass of the object rather than a surface mesh. Modifying a 3D mesh requires careful manipulation of vertices to ensure that the shape of the mesh is maintained, and that there are no holes in the mesh.
- Culling - Culling can be applied to a volume using spatial partitioning (e.g. Quad tree) by segmenting the volume data into smaller chunks, similar to how a 3D mesh is segmented by its vertices.
- Generation options - Volume data can be generated from 3D mesh data which allows for volumes to represent any shape. This can be used to create a volume object of a mesh that can then be used for deformation. Volume data can also be generated procedurally (e.g. a cloud or an explosion particle) using algorithms such as Perlin noise.
- Advanced techniques - Volume rendering can be used to simulate advanced illumination techniques such as atmospheric scattering (Atmospheric Scattering, 2009). This can be done by specifying each voxel as a density. When the volume is rendered using ray tracing, the ray is refracted based on each voxels index of refraction and a certain amount of light is emitted from each voxel. This allows for accurate lighting within transparent objects such as glass and also gaseous objects such as clouds.
- High Detail – A volume represents the entire objects mass, this means that if a clipping plane is applied then the volume object would display the inside detail of the object.
- Sampling Options – Volume rendering allows for the sampling rate to be modified at runtime which can help improve performance on higher resolution volume data.

- Rendering Options – Volume data can be rendered using several different techniques which allows developers to select the most appropriate rendering algorithm for their application, based on performance and quality factors.

2.5. Why Use Volume Rendering for terrain?

The main advantage of volume data is that it represents a mass rather than a surface. This provides the ability to represent highly complex terrain easily.

Figure 14 - Charyn Canyon, a complex terrain example (Flickr Photo Download: Charyn Canyon, 2009)

The terrain in the image above could be created using a single volume object. However, to create the same terrain using standard terrain methods would be difficult because standard terrain typically spaces each of its vertices evenly over the terrain. Standard terrain also only stores a single height for a given point on the terrain, so multiple vertices would need to be manually added to represent the steep edges and ridge detail of the cliffs. This means that a standard terrain mesh cannot easily create steep cliffs and also ridge detail.

Figure 15 - Volume Vs Terrain Surface Mesh

The image above shows a rough example of the difference in quality between a volume terrain and a standard terrain surface mesh. The terrain mesh is the red line and the black line is the volume data. As the image demonstrates, the terrain surface mesh does not maintain the ridge details that enter into the terrain mass.

Another advantage of volume terrain is that it is easy to deform because a volume object contains data for the mass within the terrain instead of just the surface data. This allows for caves to be easily added into the terrain without complicated mesh manipulation algorithms.

Future adaptations of volume terrain could allow for the simulation of earthquakes, landslides or avalanches, using volume based physics and animations which could be used for games and also weather warning systems.

2.6. Other possible uses for Volume Rendering

Volume data could be used in a 3D modelling application such as 3D Studio Max. The benefit would be that the user could easily create highly detailed models without needing to modify vertices. Volume data would also allow the user to easily create details within the object such as the seeds within an apple or the contents inside of a box without having to create several separate models.

Volume rendering could be used by resource mining industries by using terrain density scans to determine where there are resources.

Volume rendering could also be used to simulate tunnel construction scenarios. The application could display a volumetric terrain representation of the area that they wish to build the tunnel. The user could then use simple tools to create the tunnel, and the application could assess which parts of the tunnel are likely to collapse and also the best areas to place supports. The application could also identify areas of dense terrain that may slow down the construction of the tunnel.

Volume rendering could also be used to simulate coastal erosion scenarios. The application could be provided with current terrain density data and erosion data which would be used to provide a visual simulation of the erosion and the resulting effects on the terrain. This could provide an early warning system for areas where housing is built close to a coastline.

2.7. Similar Technologies

2.7.1. Constructive Solid Geometry (CSG)

Constructive Solid Geometry is a technique which allows Boolean operations to be applied to sets of Solid Geometry. Solid Geometry is a list of polygons that form a closed mesh (Solid, Primitive or Object (3.4 Constructive Solid Geometry with the Stencil Buffer, 2009)). CSG allows for users to use simple primitives to create complex objects.

There are 3 basic types of Boolean operation that can be applied to a set of solid objects.

- Union – The union of 2 primitives gives the vertices that appear in either primitive.
- Intersection – The intersection of 2 primitives gives the vertices that appear in both primitives.
- Difference – The difference of 2 primitives gives the vertices that are in the first primitive but not the vertices from the second primitive.

These Boolean operations can be applied to the same set of solids by using a binary tree to organise the operations.

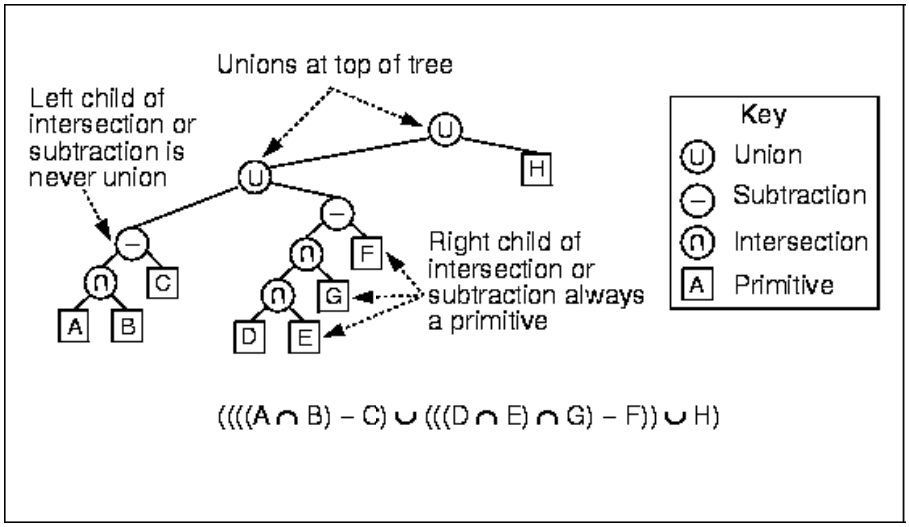


Figure 16 - Example CSG Tree (3.4 Constructive Solid Geometry with the Stencil Buffer, 2009)

Once the operations have been defined, the application will triangulate the polygons into the resulting 3D mesh ready for rendering by the graphics API.

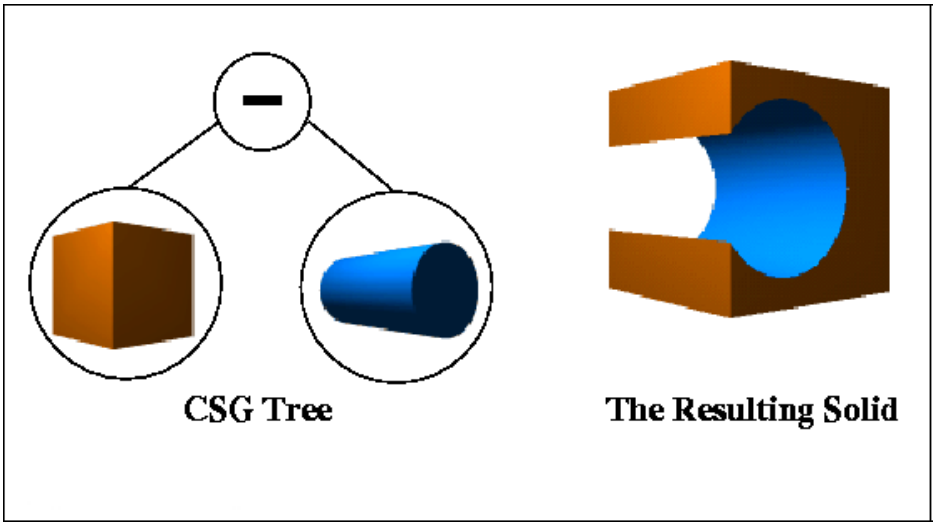


Figure 17 - A simple CSG example (3.4 Constructive Solid Geometry with the Stencil Buffer, 2009)

The advantage of CSG is that the user can easily create complex objects using simple primitives. CSG also has good support for spatial partitioning using portal occlusion systems. A portal occlusion system is used to define 'portal zones' which are used to decide whether or not to display a group of primitives. However, CSG is usually only generated once and then used as a static mesh because CSG is a slow operation.

3. Product Specification

The product for this project is a 3D application that will demonstrate Volumetric Rendering of terrain. The volume data will be loaded from a custom file format. The first time the application is run, the user will be presented with a 3D cube. This cube represents the volume data that has been loaded from the volume data file. The user can rotate the camera around the volume and also use deformation tools to add or subtract from the volume. The user will be able to save their modifications to a file so that they can be restored at a later date.

The application will export the volume to an OpenQVis file format so that the result can be viewed using several different volume rendering techniques from within the OpenQVis application. The product should also be able to load a heightmap file to allow the user to use common terrain creation features.

4. Product Development

The product for this project was created using Microsoft Visual Studio using C++ and the DirectX API. C++ and DirectX were chosen because it is commonly used for creating PC games and it is also a

platform that is covered during the games programming course. The product also uses High Level Shading Language to render the 3D volume.

The product uses 2D texture stacks to store the volume data and uses axis aligned quads as the proxy geometry. This provides fast rendering and was simple to create.

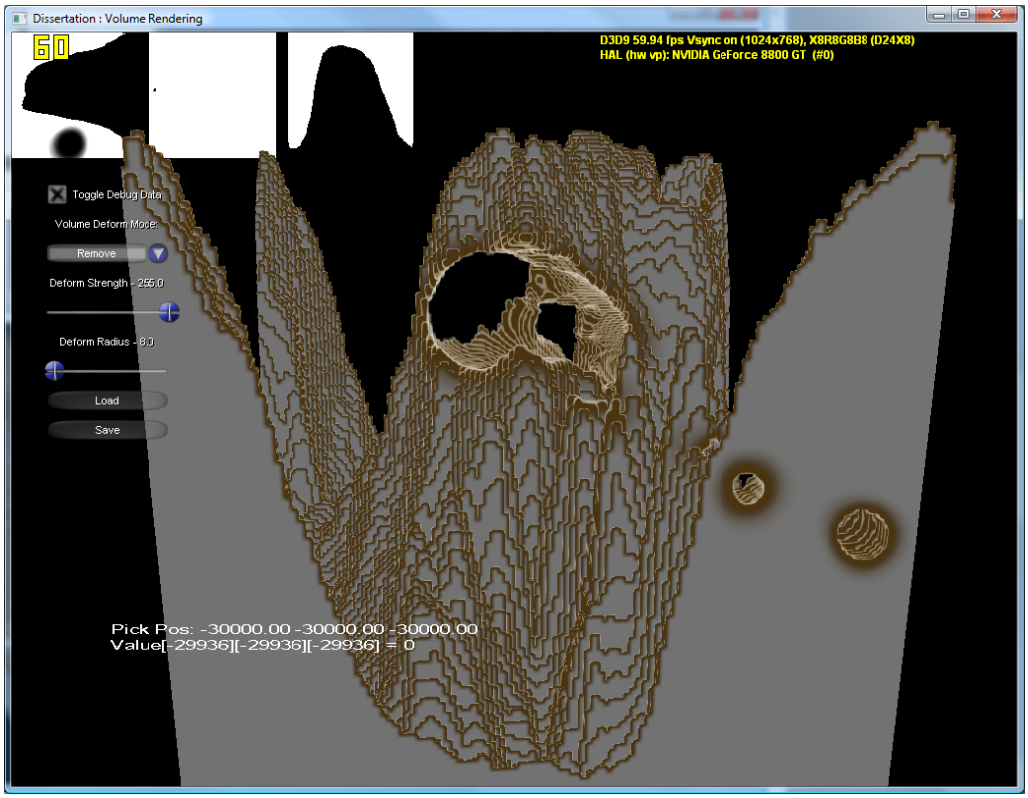


Figure 18 - Volume Terrain with a cave rendered using created product

4.1. Product Features

4.1.1. Camera Controls

The user can rotate the camera around the volume and also zoom the camera in and out. Rotation is controlled using the arrow keys and the user can zoom in using PGUP(Num9) on the number pad and zoom out using the PGDN(Num3) number pad key (Num Lock must be turned on to use these keys).

4.1.2. Volume Deforming / Building

The user can add or remove voxels from the volume object using simple point and click. The user can also adjust the radius of voxels affected and also the strength (which adjusts how much density is removed from the voxels). The volume Deform Mode drop down box (Figure 19) allows the user to choose between remove and add. This allows for the user to build volumes as well as deform them.

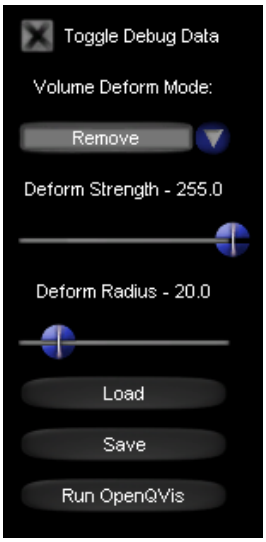


Figure 19 - Application User Controls

4.1.3. Transfer Function

The volume object density affects the final colour of the volume. Very dense areas of the volume are represented by a grey colour (to represent rock) and low density areas are represented by brown (to represent mud or dirt). The transfer function can easily be replaced by a different 1 Dimensional image.

Figure 20 - Transfer function used in the product

4.1.4. OpenQVis Save

The volume data can be saved to OpenQVis format which allows the user to view the volume inside of the OpenQVis application. This provides the user with the ability to view the volume using a variety of volume rendering techniques. The OpenQVis files are saved to the 'OpenQVis' subdirectory. When OpenQVis is launched from within the application the user is required to navigate to this subdirectory to find the OpenQVis compatible volume data in .DAT format.

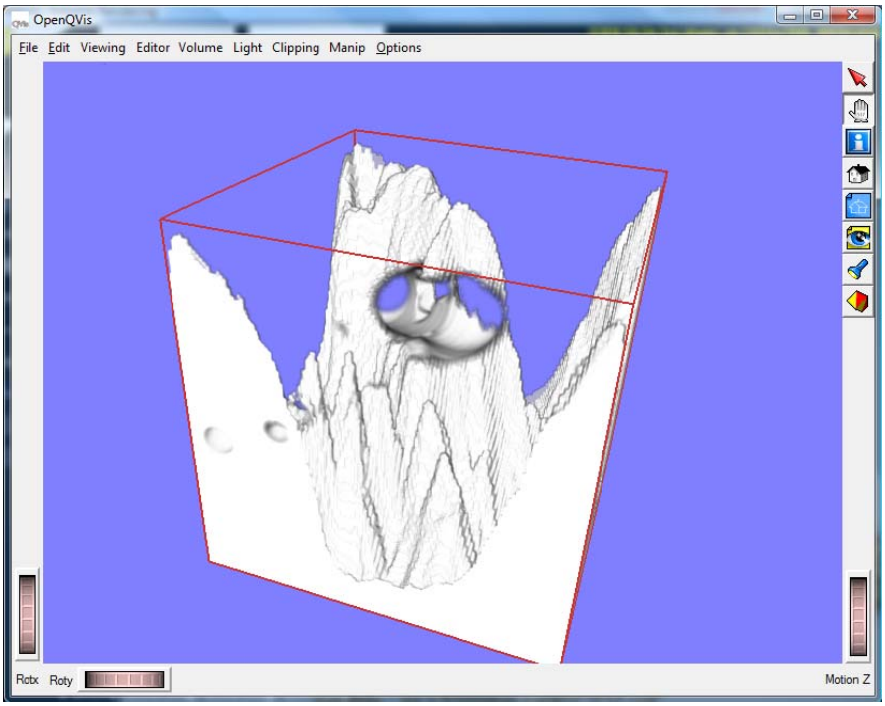


Figure 21 - Terrain Volume rendered in OpenQVis

4.1.5. Heightmap Support

Heightmaps can be loaded into the application to be applied to the volume. This allows the user to use existing terrain heightmaps to create the basic volume model before adding detail such as caves.

4.1.6. Volume Picking

The application uses mouse picking to allow the user to point and click to edit the volume.

4.2. Product Implementation

4.2.1. Implementation Considerations

Before implementing the product, there are several basic requirements that the application must implement to allow for future improvements, and a modular design.

- Volume Data structure – The application should store a volume data format that can be used to modify the volume data efficiently and also load and save the data to a file.

- Rendering Structure – The application should use the volume data structure when creating a render data structure. This means that each renderer that is implemented will gain access to any changes that are made to the volume data format. The rendering structure should also manage any textures and proxy geometry required for rendering.
- Modular camera – The camera data structure should not be a dependency for any other data formats so that it can be easily modified or replaced based on gameplay requirements.

The prototype product uses a custom volume file format. This is mainly because there is currently no standard format for volume files; therefore most applications define a volume data structure that best fits their application requirements and data format.

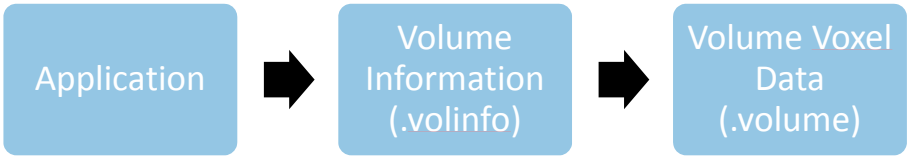


Figure 22 - Application file format

The custom data format is based around the data format for OpenQVis. There are 2 files required when loading volume data.

1. The volume information file (.volinfo) contains the volume dimensions.
2. The volume data file (.volume) contains the raw voxel information.

The volume information file is used because in future versions of the application the user may want to store additional information about a volume object such as the volume data format.

4.2.2. Application Flow

Below is a flow chart representing the core functionality of the product.

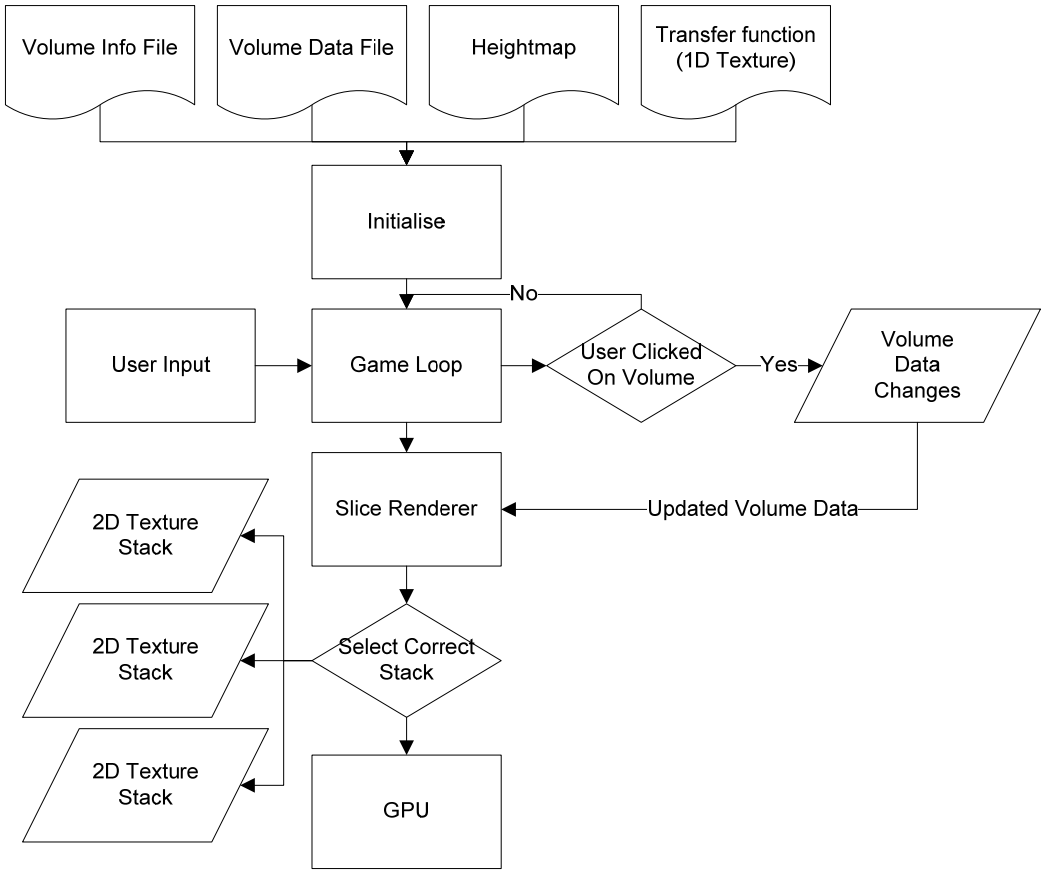


Figure 23 - Basic Application Flowchart

The first step that the application must do is load the volume data. This volume data is stored in the volume data structure. Once this data is loaded it then loads a heightmap (if specified). The heightmap is used to make the volume look more like a terrain object.

The application also loads the transfer function texture to be used by the volume renderer.

Once all initialisation is complete, the game enters the game loop. In the game loop state the user can edit the volume, edit parameters for the editing tool and also save the volume.

If the user clicks on the volume, the game sends the current mouse world position (calculated by picking against the volume object), and the current settings for the modification tool to the volume data structure. If the user has the build tool selected, then the volume data structure adds to the density of the voxels, otherwise it removes density from the voxels. Once the volume data structure has modified all the voxels affected by the user input, it sends back a 3D bounding box which specifies the space in the volume that has been modified. This 3D bounding box is then passed to the renderer so that it can update the texture(s). The bounding box is used to optimise the process by specifying a certain area of the volume to modify rather than updating the entire texture information.

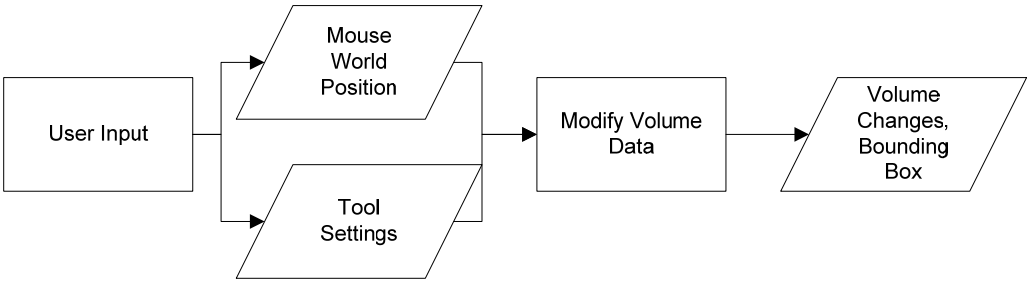


Figure 24 - Volume modification process

The renderer used in the application is the 2D texture renderer (Figure 4). The renderer manages the three 2D texture stacks and also the axis aligned quads used for rendering. Each of the 2D texture stacks represents a world axis (X, Y and Z). The renderer uses the camera direction to determine which of the stacks is active and also which direction to render the quads.

4.2.3. Class Design

There are 2 primary classes used within the application. They provide the core functionality of the application but have also been designed to be modular so that they can be upgraded or replaced easily.

- VolumeFile – The VolumeFile class handles loading and saving of the volume data. It also provides the functionality to modify the volume data and stores all the required information about the volume object.

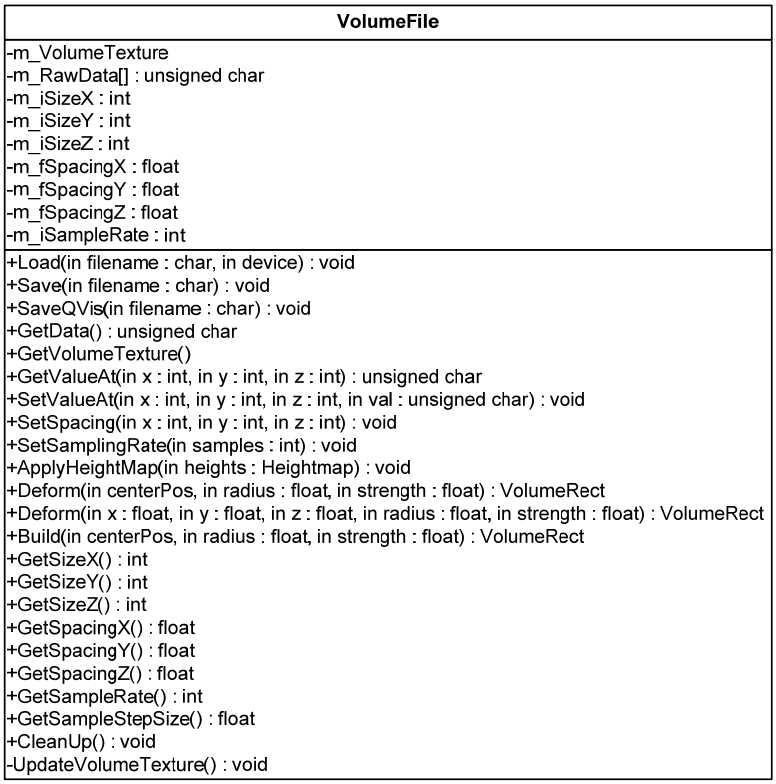


Figure 25 - VolumeFile class diagram

- SliceRenderer – The SliceRenderer class provides the functionality to render the volume using 2D texture stacks and quads. The SliceRenderer class uses the VolumeFile class to retrieve the volume data when it is created and also when it is modified.

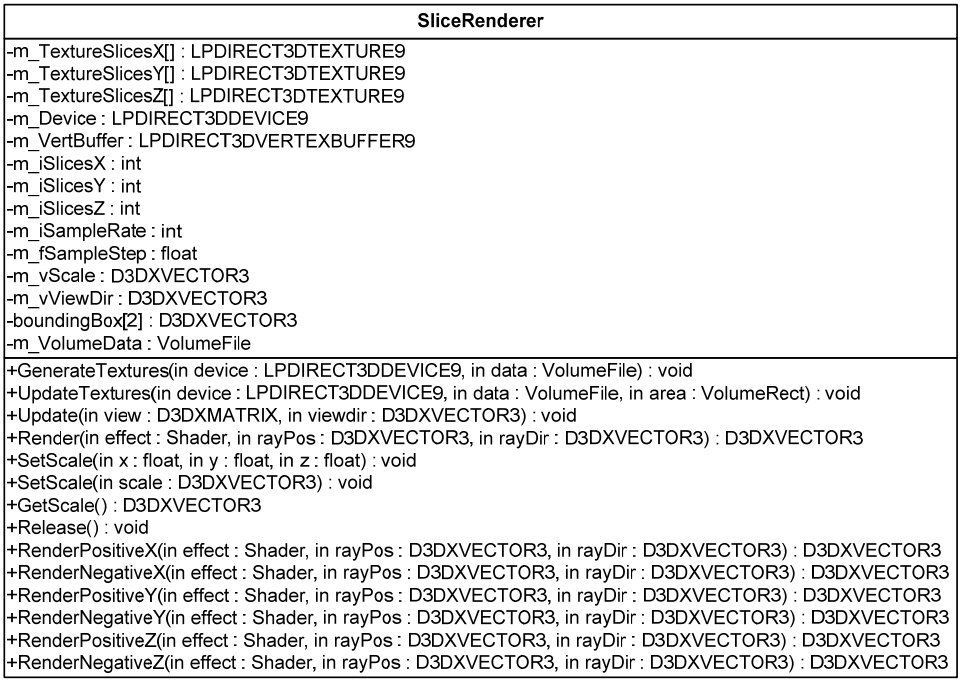


Figure 26 - SliceRenderer class diagram

The SliceRenderer class is designed to encapsulate all the functionality for rendering the volume data. This means that if another renderer was created (such as a ray casting renderer) the SliceRenderer class could simply be removed without leaving behind unused data.

The SliceRenderer class also provides collision detection by checking collisions with the quads and also by checking the current density value stored in the volume data. This is done by using the D3DXIntersectTri function which returns a distance from the ray position. The distance is added to the original ray position to give the position of the collision within the volume. The position within the volume is then used to retrieve the alpha value at this position. If the alpha value at the collision position is 0, the collision is ignored, otherwise the collision position is stored.

The process is repeated for all quads that are used to render the volume. Since they are rendered back to front, the collision position after all quads have been renderer would represent the closest collision point in the volume.

The VolumeFile class provides a common toolset for editing the volume data which can be used regardless of the volume rendering algorithm used.

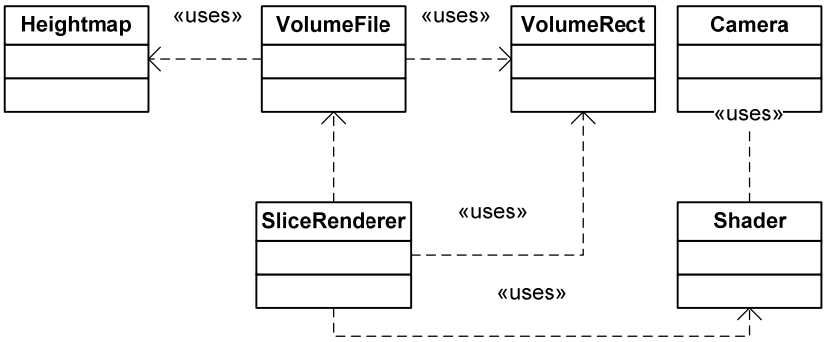


Figure 27 - Application class diagram

The class diagram above shows that the only dependency that the application has is that the renderer must make use of the VolumeFile class. This provides the ability to easily replace classes such as the camera so that they can be replaced or modified to provide the functionality required.

5. Product Evaluation

5.1. What went well?

The product demonstrates that a volume object can be created and shaped to match a terrain heightmap. This volume object can be modified in realtime and maintain playable frame rates. The application also uses picking to allow the user to edit the volume which demonstrates basic collision detection with the volume object.

The modification tools demonstrate the ability to deform the volume. These features could be used in a first person shooter game where different weapon types have a different effect on the volume, based on its strength.

Although the transfer function does not provide the volume object with much detail to make it look more like terrain, this is simple to adjust in future improvements, and demonstrates the ability to colour the volume based on voxel density.

The application can save and load a custom data format and can also save to OpenQVis format.

Overall, the framerate for the application is consistently high (the test system was an Intel Core2 2.66Ghz, Nvidia Geforce 8800GT 512MB, 4Gb ram, Windows Vista x64 Ultimate). When the volume data is modified the frame rate is affected fairly heavily but still maintains playable frame rates and could also be optimised for extra performance. The volume file used was 128x128x128 in size using RGBA as the voxel format.

5.2. What went wrong?

The volume object did not have any form of lighting applied onto it. This was because to be able to light the terrain each voxel requires a normal. To generate the normal of a voxel usually requires access to neighbouring pixels. The application uses 2D texture stacks, which means that the application would need to reference several different textures to be able to sample neighbouring pixels. This would be a big performance hit because locking a texture for editing is a slow operation and normals would need to be recalculated every time the volume is edited. This could be resolved if the application used a single 3D volume texture because the HLSL shader would have access to the neighbouring pixels allowing the normals to be calculated on the GPU.

Another major issue was the memory usage because of having to store 3 stacks of 2D textures. Table 2 demonstrates the major issue with 2D texture stacks. The voxel size is 4 bytes because the RGBA format is used.

Voxel Size (Bytes)	Width	Height	Depth	Stacks	Raw Data Size (MB)	Total Size (MB)
4	128	128	128	3	2	26
4	256	256	256	3	16	208
4	512	512	512	3	128	1,664
4	1,024	1,024	1,024	3	1,024	13,312

Table 2 - Volume texture sizes

This means that volume data over 256x256x256 in size cannot be used on a system with less than 2GB of ram (maybe even greater because of other applications). If a 3D texture format was used, the texture sizes would be 4 times the raw data size (the data is stored as a single byte) or 1/3 of the size of the 3 2D texture stacks.

When the camera is close to the volume object there are visible gaps. This is because the 3D quads used to render the volume object are axis aligned which means that at certain camera angles a gap is visible between the quads. This could be fixed by using adjustable sampling rates (the amount of quads used would increase as the camera got closer) or by using view aligned quads with a 3D texture.

A similar issue is where the edges of the volume are coloured incorrectly (they fade from grey to brown even though the alpha for the voxel is 0). This is caused by the shader interpolating between voxels so the colour slowly fades to alpha rather than immediately going to 0. This could be fixed by

using point sampling instead of linear interpolation, however, this also affects the overall colour quality of the volume object.

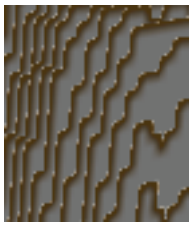


Figure 28 - Visible gaps and incorrect colouring of volume

There was also a small performance issue where the application performance would degrade as the camera enters the volume. This may be due to alpha blending since there are more transparent objects filling the screen as the camera enters into the volume.

To improve the performance of the volume rendering, the application could have used level of detail techniques. This would mean reducing the detail of the object when the camera is far away from the object and increasing the detail when this object is closer. This should provide consistent performance and also reduce the amount of visual artefacts when the camera is close to the volume.

One other issue is that the application only provided basic modification tools. The tool functionality could be made much more user friendly and also much more flexible. For example, if a user removed all the density from the volume there is currently no way of rebuilding it because the tools rely on collision detections with the volume.

6. Conclusions

The aim of this project was to research volume rendering and create a prototype application to demonstrate the use of volume rendering to represent caves in terrain. The product demonstrates well that volume rendering can allow for caves to be created in terrain. It also demonstrates the performance of volume rendering using the 2D texture rendering algorithm and also the possible performance implications.

The product also demonstrates possible flaws with using volume rendering for terrain such as the high memory usage and the colour and edge artefacts when using the 2D texture rendering algorithm. However, most of the issues found can be fixed either by using a different renderer or by simply spending more time on the project.

6.1. Possible Improvements

Below is a list of possible improvements that could be made to the product:

- Transfer Function - The transfer function was not flexible enough to represent the colour variety of a multi textured terrain algorithm. An additional volume texture or the RGB channel of the current volume texture could be used to represent the actual colour of each voxel allowing for a surface texture to be applied when the heightmap is loaded.
- Collision Detection - Collision detection was not implemented into the demo, with the exception of mouse picking. This could be implemented by only allowing the player to move where the volume had no density (0 alpha).
- Volume Physics - If the volume below an object is completely destroyed then the volume should drop down until it is colliding with the surface below (this has been done in Voxelstein 3D).
- Better rendering algorithm - At the edges of the volume the colour would bleed into the 0 alpha areas of the volume (due to interpolation). This could be fixed by using a better rendering algorithm or some kind of edge detection algorithm.
- 3D textures – By using the 3D volume texture format the memory usage and performance of the application would be a lot lower. This would also provide the application with a flexible sample rate because 3D textures provide tri-linear interpolation.
- Additional game objects – To test the overall performance of using volumetric terrain, additional game objects such as a skybox, buildings and characters could be added. This would help to assess the performance of a volume object when it is used in a more complex environment.
- Imposters – Imposters could be used to represent the volume object based on the distance from the camera. For example a surface mesh could be used to represent volume terrain from a long distance, because the user is unlikely to see the detail within the caves of the terrain from long distances.
- Dynamic colouring – Most terrain algorithms use multi-texturing to provide the terrain with realistic details. On a volume terrain this could be done by using the vertex position as a reference to decide which texture to apply.
- Data compression – Volume data uses a large amount of memory even when using a single 3D texture. By compressing the volume data and textures this would allow for more volume objects to be loaded at one time.
- Better tools – The tools currently in the application are not very flexible. To improve this, the user could be provided with a set of building primitives (e.g. cubes, spheres etc) that can be

positioned within the world and also scaled and rotated. This would provide similar functionality to CSG and also similar to a 3D modelling package.

7. Personal Critical Appraisal

7.1. Overall

Overall I think that this project has demonstrated the advantages of using volume rendering for terrain. The application created can create and render a volume terrain object and the user can easily modify the volume at runtime.

I have really enjoyed working on this project because I have a strong interest in terrain rendering techniques and this project has given me an insight into how the technique could be adapted further in the future.

7.2. Product Development

I think that the product provides evidence that volume rendering can be used to represent terrain with caves. Even by only providing simple tools the product demonstrates caves in terrain well and also demonstrates the current performance of the technique. However, in the project proposal I specified that I was going to research optimisation and compression techniques to further optimise the product, unfortunately due to bugs in the software I was unable to achieve this.

I also planned to create several different volume renderers to demonstrate the visual and performance difference between the 3 volume rendering techniques covered in this document. I did not get time to implement these features because volume rendering is a new subject area for me so the research took longer than I originally expected.

I think that this technique is a good example of what volume rendering can do for games and terrain. However, I think that without the support for directly rendering voxels using the GPU or high speed GPU based ray tracing, there are far too many quality and performance issues to address to be able to use this terrain technique within a large environment.

Close to the end of development I realised that the 2D texture stack renderer (Figure 4) was not the best choice. The renderer was easy to implement and had good performance but the steep memory requirements as volume sizes increase (Table 2) would cause a very large problem in a game environment. If I had used the 3D texture algorithm (Figure 5) then the application would have been faster and also the rendering quality would have been better.

7.3. Personal Skills Improved

This project has provided me with a solid understanding of volume rendering techniques and how they are applied to games and also how they could be used for future projects in a wide variety of industries.

I have also extended my knowledge of texture manipulation and optimisations. Also I have expanded my knowledge of using DXUT (DirectX Utilities) to create user interfaces for games.

This project has also enhanced my understanding of the requirements of managing a large scale research project along with other university deadlines.

References

2009. *3.4 Constructive Solid Geometry with the Stencil Buffer*. [online]. [Accessed 24 April 2009]. Available form World Wide Web: <http://www.opengl.org/resources/code/samples/advanced/advanced97/notes/node11.html>

2009. *3D Cloud and Sky Visual Simulation: SilverLining by Sundog Software*. [online]. [Accessed 19 Jan 2009]. Available form World Wide Web: <http://www.sundog-soft.com/>

2009. *Atmospheric Scattering*. [online]. [Accessed 22 April 2009]. Available form World Wide Web: <http://www.severewx.com/Radiation/scattering.html>

CARL, Granberg. 2007. *Programming An RTS Game With Direct3D*. Charles River Media.

2009. *Ceremonial Cave*. [online]. [Accessed 14 Jan 2009]. Available form World Wide Web: <http://www.texasbeyondhistory.net/ceremonial/images/ep19-1-sm2.jpg>

2009. *Clouds*. [online]. [Accessed 14 Jan 2009]. Available form World Wide Web: <http://www.vterrain.org/Atmosphere/Clouds/swell.jpg>

2009. *Constructive Solid Geometry*. [online]. [Accessed 24 April 2009]. Available form World Wide Web: <http://www.cs.mtu.edu/~shene/COURSES/cs3621/NOTES/model/csg.html>

2009. *Crepuscular rays - Sunrays - Cambridge*. [online]. [Accessed 22 April 2009]. Available form World Wide Web: <http://www.atoptics.co.uk/atoptics/rayim11.htm>

2009. *Crytek GmbH: Specifications*. [online]. [Accessed 22 April 2009]. Available form World Wide Web: <http://www.crytek.com/technology/cryengine-2/specifications/>

2009. *Doom 3 Screens for PC at GameSpot*. [online]. [Accessed 22 April 2009]. Available form World Wide Web: <http://uk.gamespot.com/pc/action/doom3/images/0/78/?tag=thumbs_below;thumb;78>

ENGEL, Klaus, Markus HADWIGER, Joe M KNISS et al. 2004. Real- Time Volume Graphics. *In*: *Siggraph 2004*.

2009. *Flickr Photo Download: Charyn Canyon*. [online]. [Accessed 25 April 2009]. Available form World Wide Web: <http://farm1.static.flickr.com/52/130031546_2e0d58d786_o_d.jpg>

2009. *Gallery: The OpenQVis Project at sourceforge.net*. [online]. [Accessed 14 Jan 2009]. Available form World Wide Web: <http://openqvis.sourceforge.net/gallery/TemporalBoneB.jpg>

GOTTLIEB, Eli Z. 2004. Rendering Volumes in a Vertex & Pixel Program by Ray Tracing. *In*: Wolfgang F ENGEL, (ed). *ShaderX 2 - Shader Programming Tips and Tricks with DirectX 9*, Wordware, pp.177-184.

2009. *How to. write an abstract Part: 1*. [online]. [Accessed 2 April 2009]. Available form World Wide Web: <http://info.emeraldinsight.com/authors/guides/abstracts.htm>

2009. *IGN: Crysis Screenshots (PC) 2131789*. [online]. [Accessed 22 April 2009]. Available form World Wide Web: <http://uk.pc.ign.com/dor/objects/694190/ea-crytek-title-untitled-project/images/crysis-20070920035952122.html>

IKITS, Milan, Joe KNISS, Charles HANSEN, and Aaron LEFOHN. 2003. Volume Rendering Techniques. *In*: Randima FERNANDO, (ed). *GPU Gems*, Addison Wesley Professional, pp.667-690.

2009. *Ken Silverman's Voxlap Page*. [online]. [Accessed 22 April 2009]. Available form World Wide Web: <http://advsys.net/ken/voxlap.htm>

KRUGER, J and R WESTERMANN. 2003. Acceleration Techniques for GPU-based Volume Rendering. *In*: *Visualization, 2003. VIS 2003. IEEE*., pp.287-292.

LEADWERKS. 2006. *http://www.leadwerks.com/files/csg.pdf*. [online].

2009. *The OpenQVis Project at sourceforge.net*. [online]. [Accessed 24 April 2009]. Available form World Wide Web: <http://openqvis.sourceforge.net/gallery/volren_MRHead.jpg>

REZK-SALAMA, C, K ENGEL, M BAUER et al. 2000. Interactive Volume Rendering on Standard PC Graphics Hardware Using Multi-Textures and Multi-Stage Rasterization. *In*: *2000 SIGGRAPH / Eurographics Workshop on Graphics Hardware*., pp.109-118.

2009. *Road Trip 2KX Homepage*. [online]. [Accessed 14 Jan 2009]. Available form World Wide Web: <http://www.roadtrip2000.com/images/TerrainMesh.gif>

SCHPOK, Joshua, Joseph SIMONS, David S EBERT, and Charles HANSEN. 2003. A real-time cloud modeling, rendering, and animation system. *In*: *2003 ACM SIGGRAPH/Eurographics symposium on Computer animation*., pp.160 - 166.

2009. *Simul » The Simul Weather SDK*. [online]. [Accessed 19 Jan 2009]. Available form World Wide Web: <http://www.simul.co.uk/weather>

2009. *Simul » The Simul Weather SDK*. [online]. [Accessed 19 Jan 2009]. Available form World Wide Web: <http://www.simul.co.uk/wp-content/uploads/20-9-08-3.jpg>

2009. *The State of DirectX 10 - Image Quality & Performance - HotHardware*. [online]. [Accessed 22 April 2009]. Available form World Wide Web: <http://hothardware.com/articles/The_State_of_DirectX_10__Image_Quality__Performance/Default.aspx?page=11>

STEGMAIER, Simon, Magnus STRENGERT, Thomas KLEIN, and Thomas ERTL. 2005. A simple and flexible volume rendering framework for graphics-hardware-based raycasting. *In*: *Volume Graphics, 2005*., pp.187- 241.

2009. *UDN - Two - BspBrushesTutorial*. [online]. [Accessed 24 April 2009]. Available form World Wide Web: <http://udn.epicgames.com/Two/BspBrushesTutorial.html>

2009. *UDN - Two - TechnologyFeatures*. [online]. [Accessed 24 April 2009]. Available form World Wide Web: <http://udn.epicgames.com/Two/TechnologyFeatures.html>

2009. *Voxelstein 3D*. [online]. [Accessed 22 April 2009]. Available form World Wide Web: <http://voxelstein3d.sourceforge.net/>

2009. *Writing an Abstract*. [online]. [Accessed 2 April 2009]. Available form World Wide Web: <http://research.berkeley.edu/ucday/abstract.html>

Appendices

Appendix 1 – Project Proposal

Digital Media

Final Year Project

08/09 Proposal

Student Name	Karl Mitson
Student Number	U0410589
Course	BSc Computer Games Programming
Project supervisor	Zhijie Xu

Project Title:

Can volumetric rendering be used to represent caves in terrain that can be used within a game?

Identified problem or issue that project will address – please indicate if there is a live client

The issue that this problem will address is creating a 3D volume that can represent caves in terrain. The application will use volumetric rendering to allow for caves and realistic ridge detail but must also be optimised to be able to perform in a game environment.

What are the key aims and objectives of your project?

- Research into volumetric rendering and choose the best method for rendering and lighting.
- Create a prototype application that demonstrates the use of volumetric rendering to create caves.
  - The volume object should be coloured and illuminated correctly
  - The volume object must demonstrate the advantages of using volumetric rendering rather than a 3D mesh.
  - The volume object must allow for user modification to demonstrate deformation.
  - The volume data format must compress the volume data to improve loading times and file sizes.
- Research volume rendering optimisation techniques and data compression to provide extra flexibility and performance.
- Optimise the application to allow it to be used along with other games technologies.
- Evaluate the result and state whether volumetric terrain is feasible with current generation graphics technology.

Briefly describe how you will provide evidence that you have achieved the knowledge and ability outcomes for this module – see below and comment on these as specifically as you can

Knowledge outcomes

1. Gather, examine and appraise research data to develop the project
2. Examine, appraise and select the technical tools and techniques relevant to the your project
3. Examine, appraise and select the design tools and methods relevant to your project

Ability outcomes

1. Employ an agreed range of design and development tools to design and produce a product that is a solution to the defined problem stated above.
1. Develop, execute and manage an agreed project development plan by using initiative and sound decision making
3. Employ an agreed range of evaluation methods to report on significant aspects of the product and project
4. Produce well-constructed documentation and presentation that critically assess, communicate and present project deliverables

Knowledge Outcomes

1. I will document my research findings and my reasons for choosing a specific technology.
2. I will document possible tools that can be used. I will choose the most compatible tools for my project and explain why it is the best choice.
3. I will design my project using various UML design methods such as Class diagrams and Flow charts.

Ability Outcomes

1. I will design my product using UML diagrams and I will develop my product using the tools specified in my research.
2. I will set myself deadlines for the project and I will use Gantt charts to demonstrate my progress and goals.
3. I will compare the prototype to alternative methods of terrain rendering and I will document the advantages and disadvantages of both methods and I will also document the performance difference.
4. I will research the different terrain techniques and compare them based on performance, visual quality and also feasibility for use in a game. I will also create a prototype to demonstrate volumetric terrain. This prototype will be evaluated based on how well it performed and whether the visual result is great enough for volumetric terrain to be used. I will also document possible further improvements to the prototype.

What do you intend your 'product' will be

The product will be a 3D application to demonstrate the use of volumetric rendering to display an object with caves. The product will aim to demonstrate how volume rendering can be used to represent caves. The user will be able to move the camera through the scene and will also be able to deform areas of the volume.

To assist in assessing your project – please tick what type of emphasis you are likely place on your project

- O **Equally weighted between research, software and design skills**
- O **Emphasis on research**
- O **Emphasis on software development**
- O **Emphasis on design skills**

Approved	
Date	