RealVol: Immersive volume rendering in Unity, with interactive transfer functions

Palash Bansal*
2014072, IIIT-Delhi

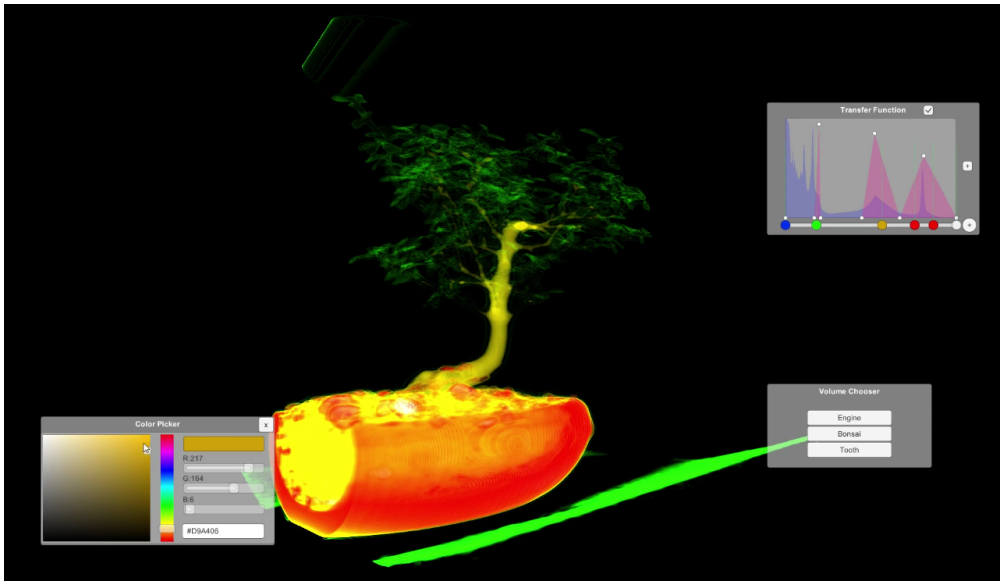

Figure 1: *Bonsai with a self defined transfer function*

Abstract

RealVol is a Unity 5 application that implements rendering of volume and medical datasets on GPU using shaders. It also supports interactive visual transfer functions that can be edited and saved on runtime and loaded later on. Since the application is written with Unity 5 engine, it is easily possible to implement HTC Vive VR support into it.
As a part of this project I have also done literature survey of global illumination on volume rendering with path tracing and photon mapping.

Architecture

The architecture is as follows

- Unity3D base
- Ray marching with 3D textures in HLSL shader
- Scripting with C#
- User interaction with Unity API in C#
- Plugging of transfer functions using 2D textures

Final Product

In the final product we can:

- Load a volume in Unity
- Interact with volume and rotate with trackball camera
- Apply and edit transfer function
- View volume with phong shading

Unity

Unity is a closed source game enigne which is used in various fields like game development, architecture, research etc.
Unity uses Mono for all its scripting and code execution. Mono, the open source development platform based on the .NET Framework, allows developers to build cross-platform applications with improved developer productivity. Monos .NET implementation is based on the ECMA standards for C# and the Common Language Infrastructure. It is very much similar to Java platform for code execution.
Unity runs code in a mono runtime which is like a JavaVM for C#. The runtime implements the ECMA Common Language Infrastructure (CLI). The runtime provides a Just-in-Time (JIT) compiler, an Ahead-of-Time compiler (AOT), a library loader, the garbage collector, a threading system and interoperability functionality.
Mono also provides class library with a lot of important classes that can be used while developing.
As it uses Mono Runtime, Unity can build and run on any platform available that the runtime supports like Windows, Mac, Linux, XBox, Playstation etc.

Scripting in C#

C# is an elegant and type-safe object-oriented language that enables developers to build a variety of secure and robust applications that run on the .NET Framework. C# syntax is highly expressive, yet it is also simple and easy to learn. The curly-brace syntax of C# will be instantly recognizable to anyone familiar with C, C++ or Java. Developers who know any of these languages are typically able to begin to work productively in C# within a very short time. C# syn-

*e-mail:palash14072@iiitd.ac.in

tax simplifies many of the complexities of C++ and provides powerful features such as nullable value types, enumerations, delegates, lambda expressions and direct memory access, which are not found in Java. C# supports generic methods and types, which provide increased type safety and performance, and iterators, which enable implementers of collection classes to define custom iteration behaviors that are simple to use by client code. Language-Integrated Query (LINQ) expressions make the strongly-typed query a first-class language construct.

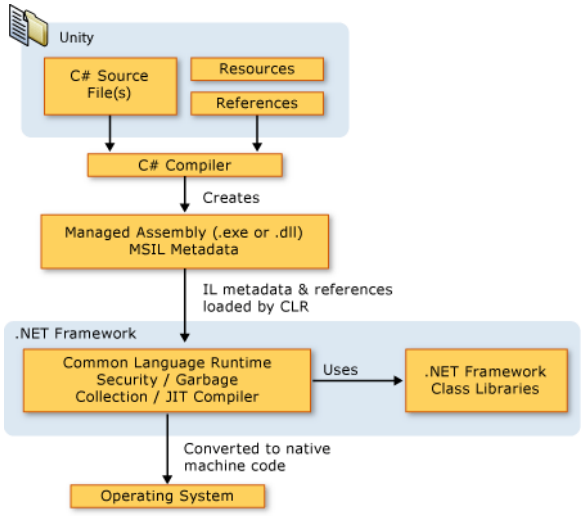

Figure 2: *Unity architecture of C# scripting in Unity*

Data formats

RealVol supports the following 2 types of data formats that are the most widely used ones for volumetric rendering and medical datasets(CT Scans and MRIs).

- RAW file format.
- DCM image set

RAW file format

This is the standard and most used format for volume data sets. In this, the volume data is stored byte by byte directly. Loading these datasets directly, without conversion, is possible in RealVol. In this format the data is arranged byte by byte. The data file contains the gridded data (raw voxels) as a binary array. Each voxel can be stored as 8 or 16bit signed/unsigned integer. The position, in this array, of any voxel with coordinates X, Y and Z is given by:

$$X + Y \times N_x + Z \times N_x \times N_y$$

Where:
N_x = number of nodes in X dimension and
N_y = number of nodes in Y dimension

DCM image set

DCM stands for DICOM(Digital Imaging and Communications in Medicine), it is the current standard in medical imaging and all CT/MRI scans produce images in this format. Loading these datasets in Unity is possible after converting these into jpeg/png/tiff images using external scripts.

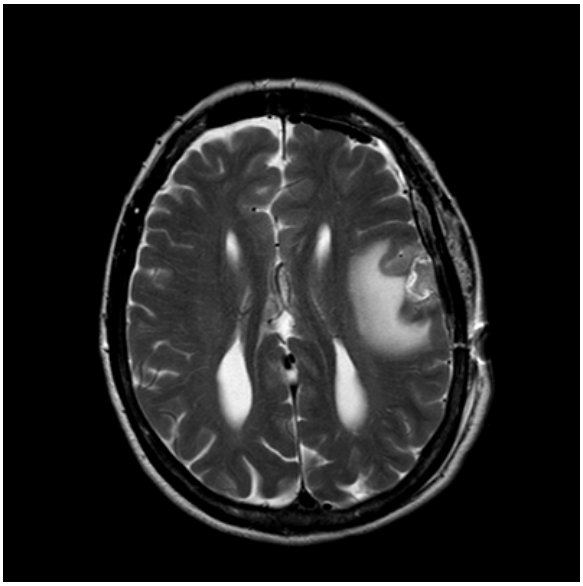

Figure 3: *Sample DCM image, from Osirix Viewer*

Ray Marching

Volume ray casting, sometimes called volumetric ray casting, volumetric ray tracing, or volume ray marching, is an image-based volume rendering technique. It computes 2D images from 3D volumetric data sets (3D scalar fields). Volume ray casting, which processes volume data, must not be mistaken with ray casting in the sense used in ray tracing, which processes surface data. In the volumetric variant, the computation doesn't stop at the surface but "pushes through" the object, sampling the object along the ray. Unlike ray tracing, volume ray casting does not spawn secondary rays. The technique of volume ray casting can be derived directly from the rendering equation. It provides results of very high quality rendering. Volume ray casting is classified as an image-based volume rendering technique, as the computation emanates from the output image and not the input volume data, as is the case with object-based techniques. The traditional algorithm comprises of the following steps:

- Ray casting. For each pixel of the final image, a ray of sight is shot ("cast") through the volume. At this stage it is useful to consider the volume being touched and enclosed within a bounding primitive, a simple geometric object usually a cuboid that is used to intersect the ray of sight and the volume.
- Shading. For each sampling point, a gradient of illumination values is computed. These represent the orientation of local surfaces within the volume. The samples are then shaded (i.e. coloured and lit) according to their surface orientation and the location of the light source in the scene.
- Compositing.
- Global Illumination.

Compositing

After all sampling points have been shaded, they are composited along the ray of sight, resulting in the final colour value for the pixel that is currently being processed. The composition is derived directly from the rendering equation and is similar to blending ac-

etate sheets on an overhead projector. It may work back-to-front, i.e. computation starts with the sample farthest from the viewer and ends with the one nearest to him. This work flow direction ensures that masked parts of the volume do not affect the resulting pixel. The front-to-back order could be more computationally efficient since, the residual ray energy is getting down while ray travels away from camera; so, the contribution to the rendering integral is diminishing therefore more aggressive speed/quality compromise may be applied (increasing of distances between samples along ray is one of such speed/quality trade-offs).

Figure 4: *High quality Ray Marching, Source: Wikimedia Commons*

GI in ray marching

Global illumination (shortened as GI), or indirect illumination, is a general name for a group of algorithms used in 3D computer graphics that are meant to add more realistic lighting to 3D scenes. Such algorithms take into account not only the light that comes directly from a light source (direct illumination), but also subsequent cases in which light rays from the same source are reflected by other surfaces in the scene, whether reflective or not (indirect illumination). Theoretically, reflections, refractions, and shadows are all examples of global illumination, because when simulating them, one object affects the rendering of another (as opposed to an object being affected only by a direct light). In practice, however, only the simulation of diffuse inter-reflection or caustics is called global illumination. Images rendered using global illumination algorithms often appear more photorealistic than those using only direct illumination algorithms. However, such images are computationally more expensive and consequently much slower to generate. One common approach is to compute the global illumination of a scene and store that information with the geometry (e.g., radiosity). The stored data can then be used to generate images from different viewpoints for generating walkthroughs of a scene without having to go through expensive lighting calculations repeatedly. This is done with the use of photon map data structures. GI will be implemented in RealVol at a later stage.

Figure 5: *Volume rendering of a foot with and without global illumination, by Leonhard Rabel*

Transfer functions

In engineering, a transfer function (also known as system function or network function and, when plotted as a graph, transfer curve) is a mathematical representation for fit or to describe inputs and outputs of black box models. In volume rendering transfer functions map volume intensities to color and alpha values which can be linearly interpolated to generate a gradient.
Two type of 1D transfer functions are implemented in RealVol

- Color transfer function. This transfer function maps RGB color values to intensity values in the volume. The color at the other intensities are interpolated between the defined one .
- Alpha transfer function. This transfer function is used to control the opacity of certain parts in the volume. It is able to specify alpha values to certain intensities and alpha values for other intensities is linearly interpolated from the existing ones provided.

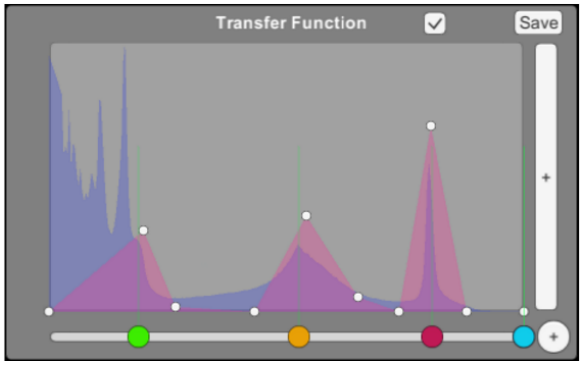

Figure 6: *Sample transfer function*

User Interface

RealVol features a great User Interface for rotating the volume, loading a new dataset, changing transfer functions and changing colors.

Controls

It is possible to move around in the volume with mouse and keyboard. Holding the left ctrl key on the keyboard enables the rotating and translation in the volume. Rotating in the volume is possible in a trackball fashion by holding and dragging the left mouse button.

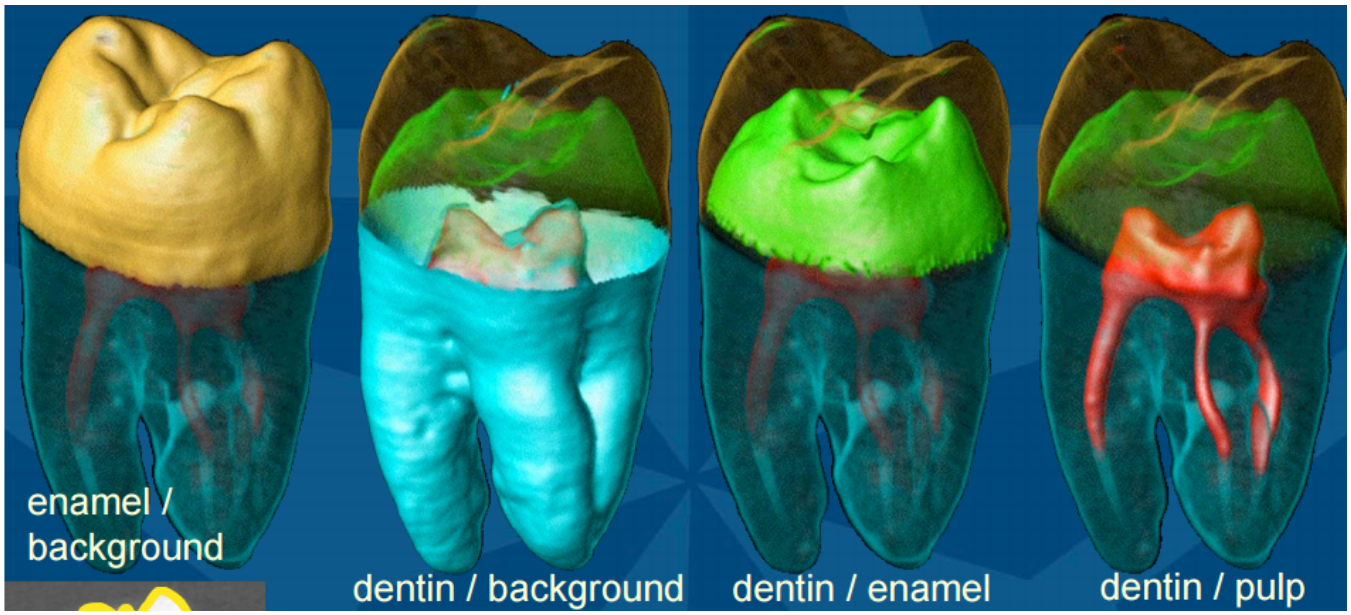

Figure 7: *The tooth dataset with different transfer funtions, by Gordon Kindlmann, University of Utah*

Volume Loading

The volume loading dialog box lists all the volumes that are possible to load into the scene at any moment of time. In figure 12 we can see the 4 available to load in the scene:

- Engine (RAW)
- Bonsai (RAW)
- Tooth (16 bit RAW)
- Dental Scan (DCM)

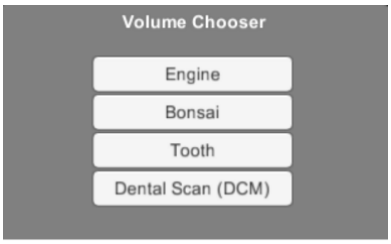

Figure 8: *The volume render dialog box*

Transfer Function Editor

The transfer function editor dialog box has multiple sections that allow us to create the best transfer function for our volume. The markers can be moved around to edit the AlphaTF, and the sliders in the bottom can be used to change the ColorTF. On Right clicking sliders, color picker box pops up, which can be used to change the color. It is also possible to save the TF. The parts in the dialog are:

- Intensity Histogram: This histogram shows the frequency of isovalues(normalized from 0 to 1) that occur in the volume. This is represented with a blue polygon that is triangulated and drawn over the dialog.
- AlphaTF: This is the pink polygon that we can see over the histogram, with editable drag drop markers.
- ColorTF: The sliders are position below the histogram and each knob inherits the UnitySlider which only supports 1 knob and has no color.
- The color picker box pops up when its right clicked on any slider, and its color can be changed. This color picker box is adapted and enhanced from UnityUIExtensions.

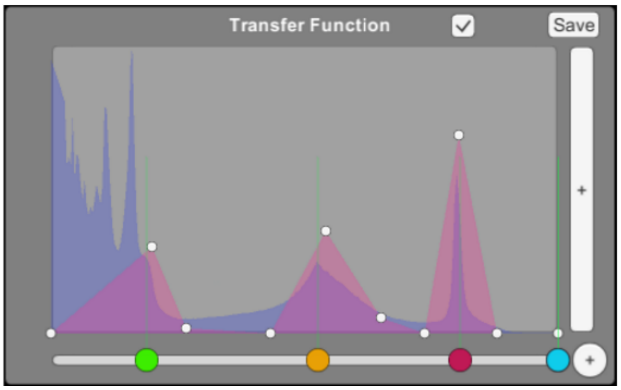

Figure 9: *The transfer function dialog box*

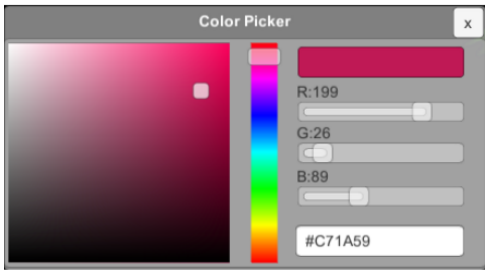

Figure 10: *The color picker dialog box*

Sample Renders

Figure 8, 9, 10 and 11 shows some sample volume renders in RealVol with multiple datasets and rendering techniques.

- Figure 8 shows standard ray marching on a CT scan rendered without transfer functions and normal compositing equations.This dataset was loaded from DICOM images by converting them to standard JPEG formats.
- Figure 9 shows standard ray marching on a CT scan rendered without transfer functions and rendered with first hit and Phong diffuse lighting.This dataset was loaded from DICOM images by converting them to standard JPEG formats.
- Figure 10 shows standard ray marching on a high quality volume dataset of a bonsai tree. This ray marching is done with transfer functions and compositing. The color and alpha points are chosen very carefully to highlight different parts of the image. This dataset was loaded from a RAW file.
- Figure 11 shows multiple renders from different angles of a tooth dataset that is loaded from a 16bit RAW file.

Future Work

This project is to be extended even beyond the Independant project, in December and coming semesters. The following is planned to be done:

- HTC Vive integration: To integrate RealVol with HTC Vive and create a walkthrough with medical datasets in VR.
- Light Scattering: Implement scaterring of light in the ray marching shader.
- Photon Mapping: Creation of Photon map data structure for storing of radiosity values.
- Global Illumination: Final goal of this project is to have real time GI in Virtual Reality with HTC Vive using photon mapping.

References

- HSV Slider, window base: https://bitbucket.org/ddreaper/unity-ui-extensions
- Ray marching reference: http://graphicsrunner.blogspot.in/2009/01/volume-rendering-101.html
- Transfer Function design: http://graphicsrunner.blogspot.in/2009/01/volume-rendering-102-transfer-functions.html
- Unity volume structuring: http://www.alanzucconi.com/2016/07/01/volumetric-rendering/
- Transfer Functions: http://www.ics.uci.edu/~gopi/CS211B/TransferFunctions.pdf
- Gi for interactive volume visualization https://wwwcg.in.tum.de/fileadmin/user_upload/Lehrstuehle/Lehrstuhl_XV/Teaching/SS13/BaMaSeminar/Rabel_Slides.pdf

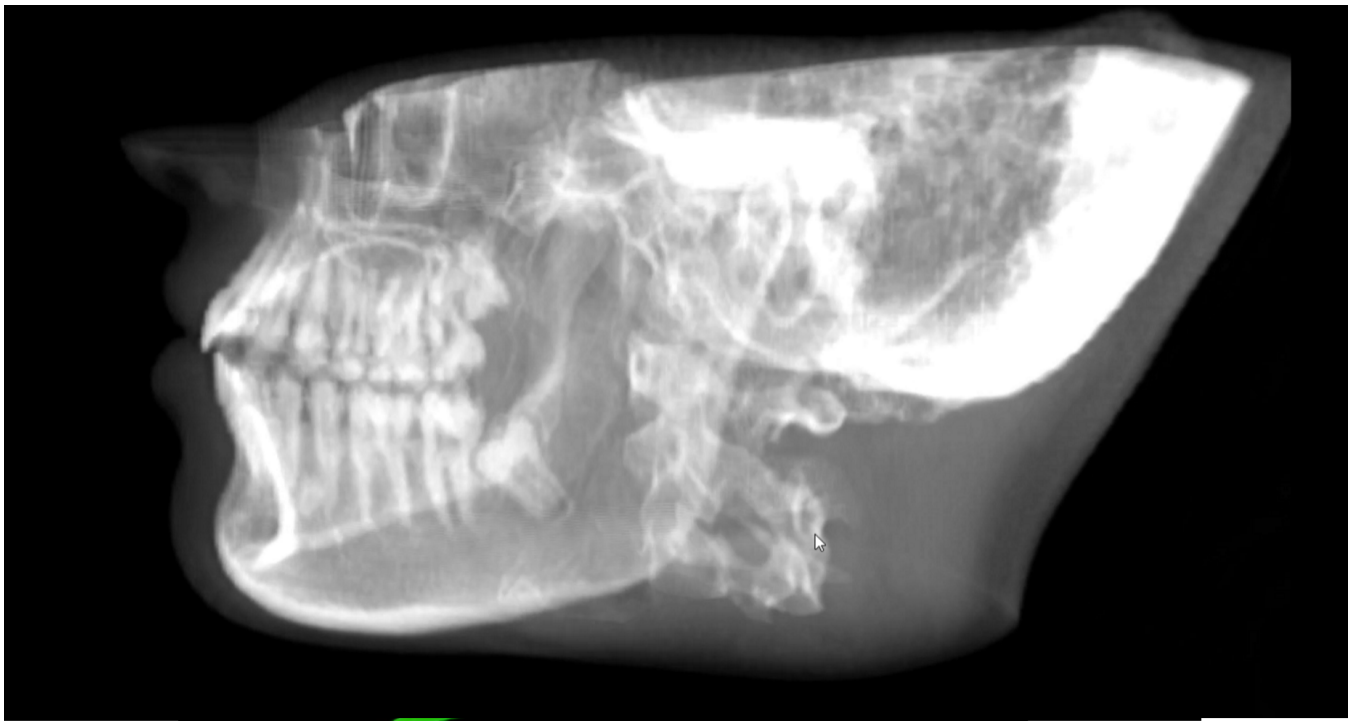

Figure 11: *CT scan rendered without transfer functions and normal compositing equations.*

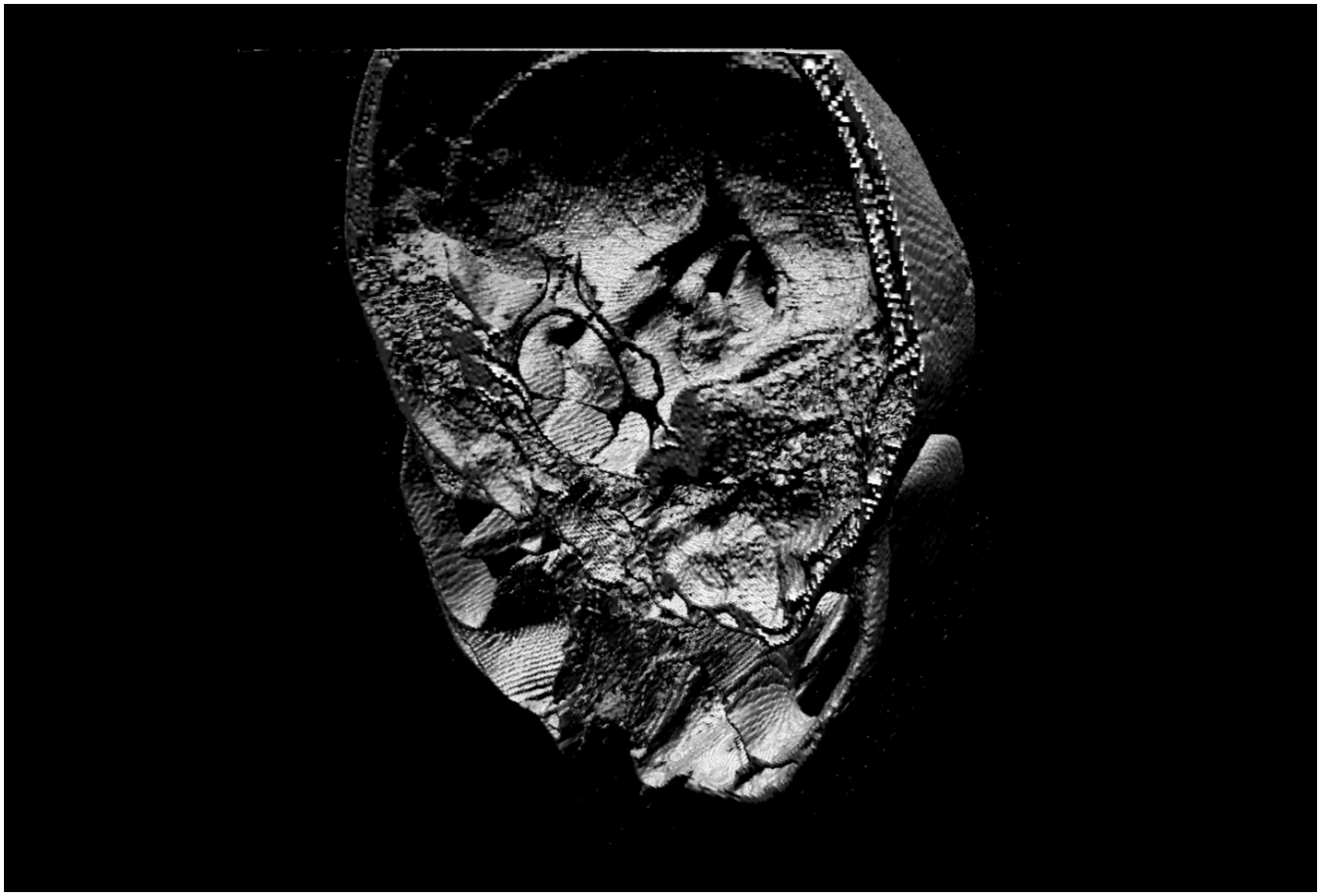

Figure 12: *CT scan rendered without transfer functions and rendered with first hit and Phong diffuse lighting*

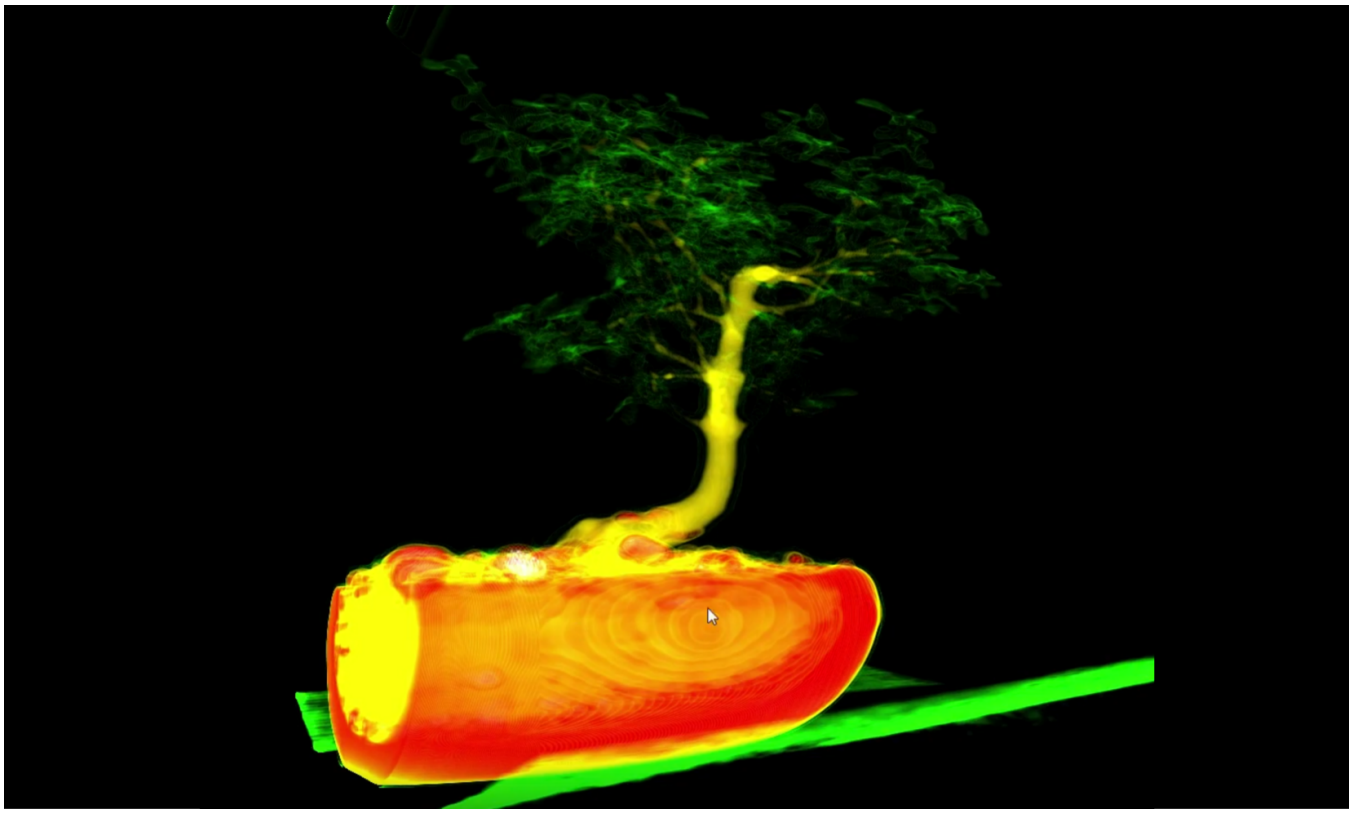

Figure 13: *Ray marching on a high quality volume dataset of a bonsai tree with transfer functions*

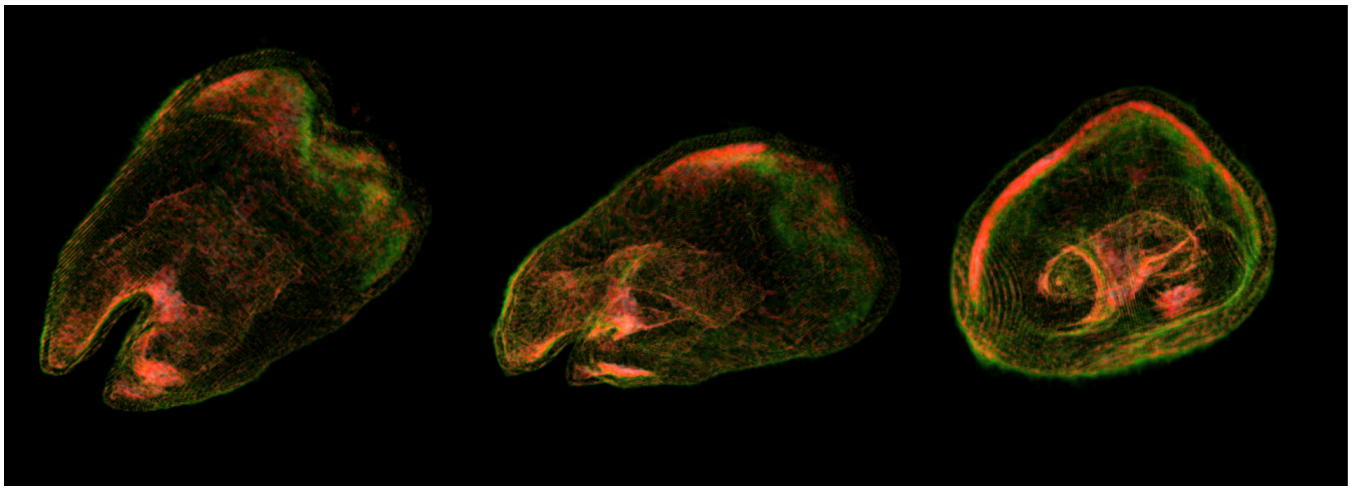

Figure 14: *Multiple renders of the tooth datasets in Realvol.*