OpenVDB Viewer – OpenGL
User Manual

Author: Callum James
13/02/2014

Contents

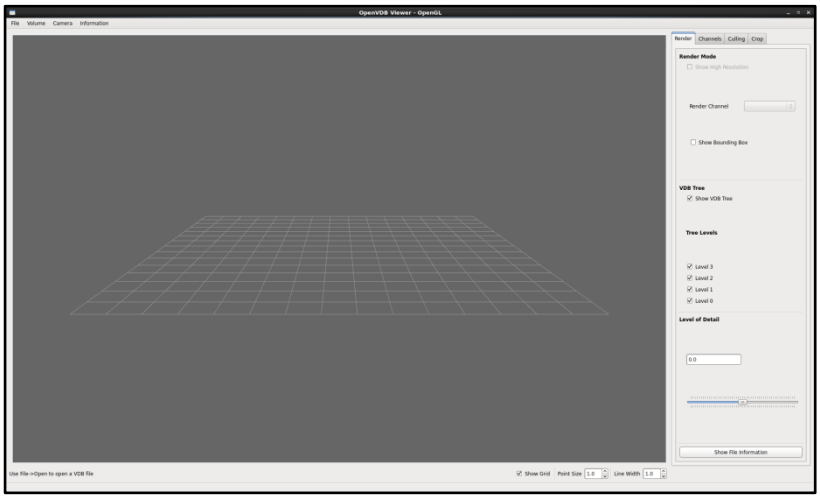
1. Overview

Welcome to the OpenVDB Viewer built in OpenGL and GLSL. The purpose of this application is to be able to view and analyse VDB files and their attributes. OpenVDB is a voxel system used to represent volumetric data as efficiently as possible. The files can often be large, full of data and difficult to analyse. Whilst their is integration within other applications for the OpenVDB system, this is a standalone application to view and break down the data stored within these files.

The application has multiple tools that you can use to view the data in different ways. These range from simple view controls to move around the loaded volume to remove specific data to help find anomalies or simply view certain ranges of data through colour or removal of points.

The UI has been developed within Qt and each part of it with examples of how it affects the data and how to use it.

To obtain some test files, visit http://www.openvdb.org/download/ to download some simple and complex files to use within this application.



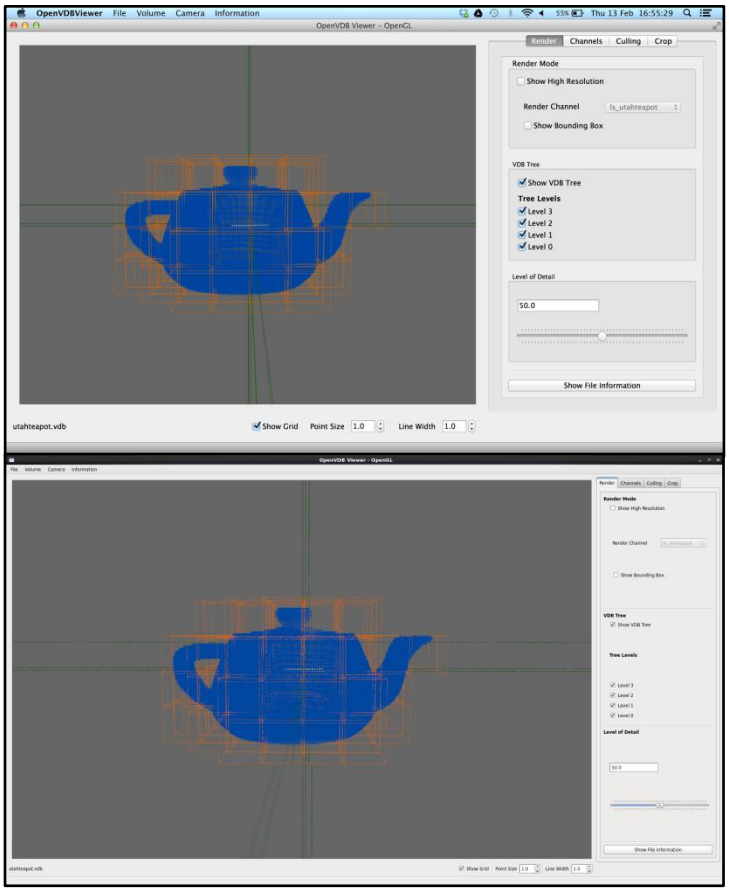
OpenVDB Viewer Running on Linux

2. Supported Platforms

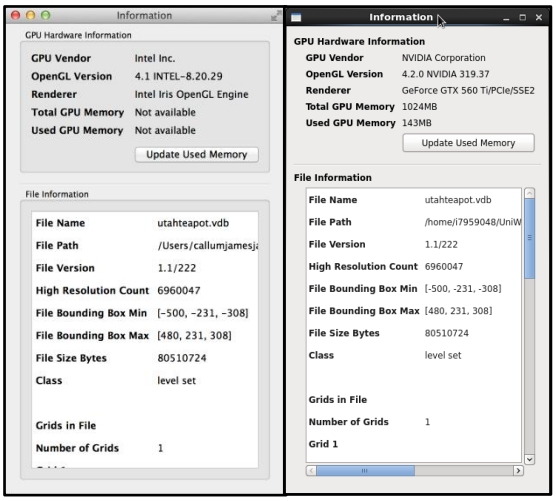
OpenVDB Viewer is not limited to Linux. It has also been built to run on MacOSX (10.9 Mavericks). Currently Windows is not supported for building or running the application. This however is something queued for future development and will be supported in the future.

The way the application works on both Linux and MacOSX is identical with only a couple of infinitesimal UI logic differences. These differences will be pointed out within this manual as they appear.

Visually, the viewer running on the different platforms will have different styles which are of course platform specific and generated by the OS. The layout in general is similar however and the UI logic the same. Below are some images of the viewer running on both RHEL 6.4 (Santiago) and MacOSX 10.9 Mavericks to demonstrate the visual differences found between the platforms.



OpenVDB Viewer on MacOSX (top) and Linux (bottom)



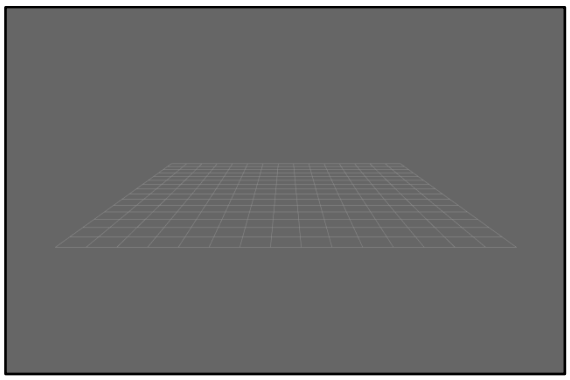
Information Window on MacOSX (left) and Linux (right)

3. The Interface

The interface for this viewer has been developed using QtGUI and has been designed to be as simple to use as possible with as little clutter as could be achieved. It has been built to use sliders and drop down boxes as much as possible to quickly select different attributes and values to manipulate and analyse. Currently, it does not implement any user interaction directly onto the widget other than rotations. This is however something for the next iteration of the application.

All attribute interaction and viewing has been put into 4 simple tabs on the right of the application to group and organise all similar tools together. A few simple GL specific options have been placed below the render window to allow for quick changes if desired.

Attention was also given to avoid using too many windows, simply to keep the application clean and easy to use.



Render Window

The main feature of the UI you will notice is the render window on the left. This is where all VDB files will be shown and attributes rendered. This is your window into the world of OpenVDB!

Whilst all of the attribute interaction is done through UI elements outside of the render window, a couple of features exist within the window itself.

By moving your cursor over the render window, left mouse clicking and dragging, you are able to rotate the VDB tree or volume itself within the world. This tumbling of the volume allows inspection of all angles.

To move the volume or tree around the world, hover your cursor over the render window, right click and drag. The volume will move with your cursor.

The camera is also moveable within the application. To tumble the camera, again hover your cursor over the render window, hold the Alt key, left mouse click and drag. This time the camera will rotate around either the origin of not VDB file is loaded or about the volume if one has.

To zoom in and out from the volume or the origin, scroll up and down on the mouse wheel with your cursor over the render window. You will be able to zoom in to a certain point before it stops, but able to zoom out as far as you like! If you are using the application on a laptop with no external mouse, to perform this zoom, scroll up and down the trackpad with two fingers (also left and right on a macbook device).

Below the render window, there are three small simple controls. The show grid check box will toggle the drawing of the reference grid within the scene.

The spin box for the Point size will change the size of rendered points in the high resolution volume.

The line width spin box will change the width of lines drawn in the scene.

The bottom left of the interface displays the file name of the currently opened VDB file when one is opened and instructions on how to open a file when not.

3.1 Shortcuts

A number of shortcuts have been implemented into the viewer for your convenience. A list of these follow. Please note that the shortcuts are slightly different on Linux as they are on MacOSX. These differences are denoted by a slash with the former being Linux and the later MacOSX. For example if a shortcut was Ctrl+C on Linux and cmd+C on MacOSX, it will be denoted

Ctrl+C/cmd+C - ...

OpenVDB Viewer Shortcuts:

- Mouse click and drag - rotate the volume
- Alt+Mouse click and drag - rotate the camera
- Ctrl+O/cmd+O - Open a file
- Ctrl+Shift+C/cmd+Shift+C - Clear the scene
- Esc/cmd+Q - Quit application
- Del/Del - Remove high resolution mesh if loaded
- Ctrl+R/cmd+R - Reset Volume transform
- R/R - Reset scene (volume transform and camera)
- Shift+R/Shift+R - Reset Camera

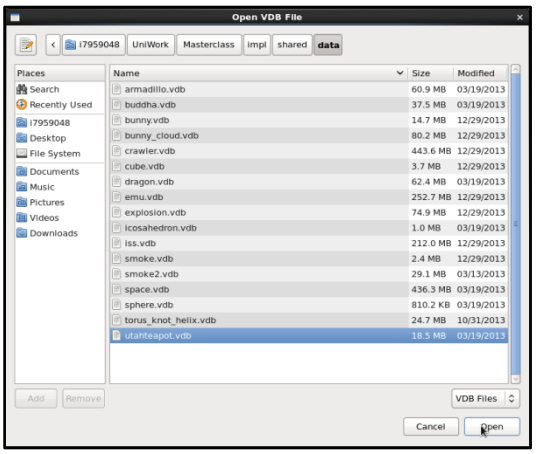
4. Menu System

The OpenVDB Viewer has a simple menu bar system comprising of 4 options. Each of these options contains sub options that can be used to interact with the file and the control of the camera. The complete menu hierarchy is as follows:

- File
  - Open
  - Clear
  - Exit
- Volume
  - Remove
    - Remove High Resolution Data
  - Reset Transform
  - Reset Scene
- Camera
  - Reset Camera
- Information
  - Information
- Help (Linux Only)
  - About OpenVDBViewer

4.1 File Menu

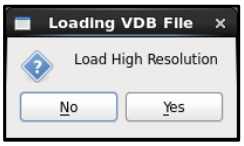
The file menu contains the options to open, clear and exit. These are fairly self-explanatory.



File Dialogue

The open option will bring up a file dialogue that is looking for .vdb files. Once selected and opened, a dialogue will appear asking if you wish to load the high resolution data found in the file or not. If no is selected, the loading of the VDB tree will proceed and will be a quick load. If yes is selected, a warning will appear ensuring you want to load high resolution data and the effect it can have on memory and performance. Unfortunately due to the size of some files and its structure, it is unavoidable that some files will take a considerable time to load and could affect computer performance.

The clear option will do exactly that, clear the entire scene and remove the file from memory. You will be returned back to the default camera position with an empty world and just the reference grid. From here you can then load in another file or exit the application. Files can be opened when other files are open, as the



Check for load

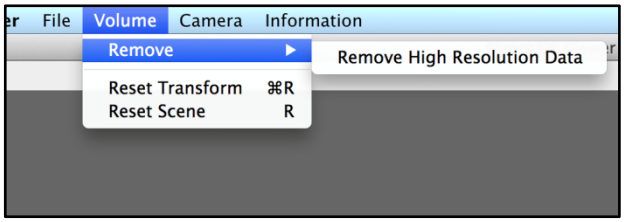
previous file will be cleared first.

The exit option will clear all data currently in memory and safely shut the application down.

4.2 Volume Menu

The volume menu contains the options to Remove->Remove High Resolution Data, Reset Transform and Reset Scene.

The Remove->Remove High Resolution Data option will look for any high resolution data that has been loaded in. If found, it will then clear only this data, meaning that if you wish to view the high resolution points from this point on you will need to reload them back in. It does not remove the basic file information and the VDB tree as the file is still loaded in the application. This option can be used to help increase computer performance at the loss of data.



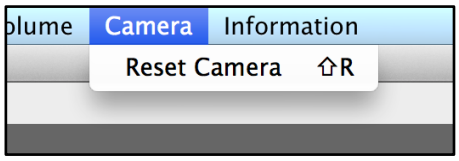
Volume menu

The Reset Transform option will reset any volume or VDB tree loaded into the application to its original position as if just loaded in from file.

The Reset Scene action will reset both the volume or VDB tree and the camera with relation to the volume loaded. This is useful if you have got a little over excited with the tumbling of camera and volume and become lost within the world needing some orientation.

4.3 Camera Menu

The camera menu contains only one option, reset camera. This will reset the camera in relation to your current file context. If you have a VDB file loaded into to the application, resetting the camera will reset it back to focus on the file. If there is not file loaded, it will return the camera to focus on the origin as it is when you first run the program.



Camera menu

4.4 Information Menu

The information menu also only contains one option, the information option. This will bring up the information window, giving you data about the hardware you are using, real time updates on the current used GPU memory (which can be manually updated by clicking the button on the window) and when a file is loaded information on the file. This window can be bought up at any time (also from the Render tab) either when a file is loaded or not.

4.5 Help Menu

The Help menu only exists on the Linux version of the application and contains a link to the about window for the viewer. This contains version and author information as well as contact information for the developer. On Mac, this can be found in OpenVDBViewer->About OpenVDBViewer.

5. Tab System

The OpenVDB viewer used a tab system to hide and organise most of the tools and options available to use. These tools have been grouped into sub headings to make use of the application easier and finding the tools as simple as possible. The Render Mode tab is the default tab when first running the file. The tabs currently implemented into the viewer are:

- Render
- Channels
- Culling
- Crop

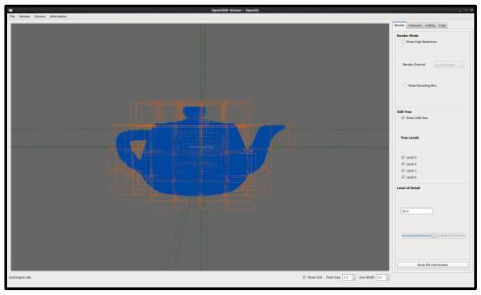
5.1.1 Render – Render Mode

The render tab contains a couple of simple options on how to render the file.

The first of these is whether or not to show the high resolution volume data. If the data is already loaded and present in the cache, the data will simply be toggle on and off as you select the tick box. If however the data is not yet loaded, it will prompt you that it first needs to be loaded in. Once loaded, it will again act as a simple toggle of showing and hiding the high resolution data until the data is cleared.

The render channel drop down box is automatically populated with the channels that are present within the file. From this you are able to choose the channel to render in the high resolution points.

The show bounding box check box will toggle the bounding box being drawn to the render window. The bounding box is always loaded even with the basic load and so is always available to you.



VDB Tree rendered

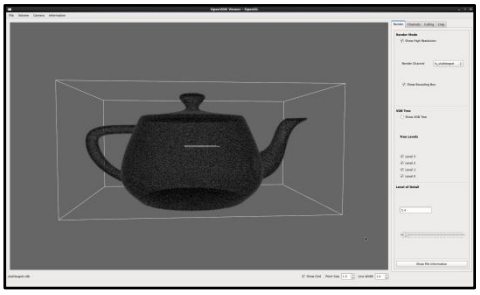
5.1.2 Render – VDB Tree

The next section is the VDB tree group. In here you can control the drawing of the VDB voxel tree. The Show VDB Tree check box is a global control to toggle the rendering of the whole tree. The

tree level options allow you to turn on and off specific levels of the tree if you want to look in more detail at one or another.

5.1.3 Render – Level of Detail

The level of detail slider is exactly that. When high resolution data is loaded, this slider will change the amount of points being drawn to the screen. Due to rounding errors causing holes in the volume, there is less change near the top as the points chosen to draw become similar. Near the lower end there is a more distinct change and can help ease the viewing of the file. As of currently, the level of detail does not improve or decrease performance of the

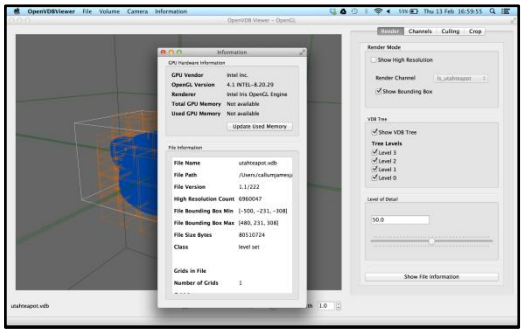


High resolution points, low LOD

application as the entire calculations are done on the GPU. This is however a topic for future development.

5.1.4 Render – Show File Information

Exactly the same as the information menu action, this button will simply bring up the information window for your viewing pleasure.

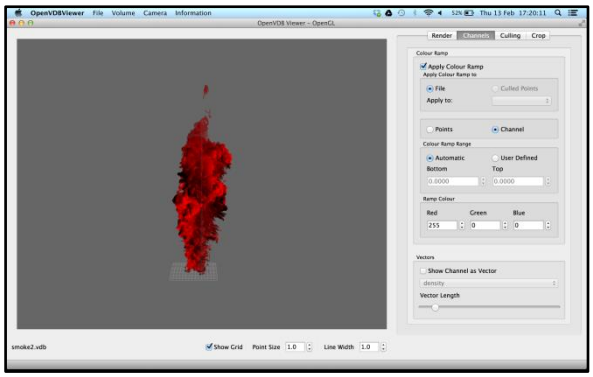


Information Window

5.2.1.1 Channels – Colour Ramp: Apply Colour Ramp (to)

OpenVDB Viewer has the ability to apply a colour ramp to the VDB volume high resolution mesh. A colour ramp looks at the extremes of the channel (so maximum and minimum values in all axis) and then shades each colour as a ratio of these values as a colour. This is clearly visible if applying a colour ramp to the points, as the lower in the x range the darker the colour will be, and the higher the brighter. This can also be applied to the channel data and combined with culling to filter out those problem points or locate points of a certain value.

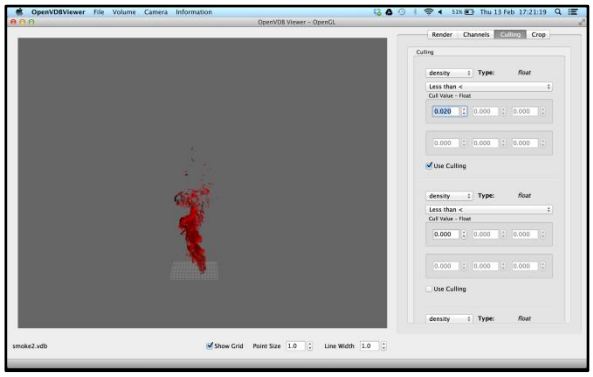
To enable colour ramping, simply click the Apply Colour Ramp check box. Once checked, you will see an immediate change to the rendering of the high resolution data. The colour ramp when applied to the file, will be calculated on the currently selected render channel from the render tab.



Colour ramp on vector channel

The option to apply the colour ramp to either the file or the cull will change the values passed to the shader. If it is applied to the file, it will take the extremities of the bounding box as its lower and upper end. If applied to culled points, it will take the lower and upper values of this cull as the range to ramp through.

The drop down box below the Culled Points selection will allow you to apply the colour ramp to any active culls on the file. The drop down box is automatically populated from active culls and will change the colour ranges shown on the file.



Colour ramp on channel cull

The final option here is to apply the colour ramp to the points (coordinates) or to the channel. If applied to the coordinates it will simply ramp based on their position in the world. If applied to the channel, it will ramp on their channel data, allowing you to easily analyse points channel values.

5.2.1.2 Channels – Colour Ramp: Colour Ramp Range

The range of the colour ramp will change how you view the points and how you can interpret the data. Here you are presented with two options.

Automatic colour ramping range will calculate the extremities of either the selected cull or the entire file and simply ramp between these ranges.

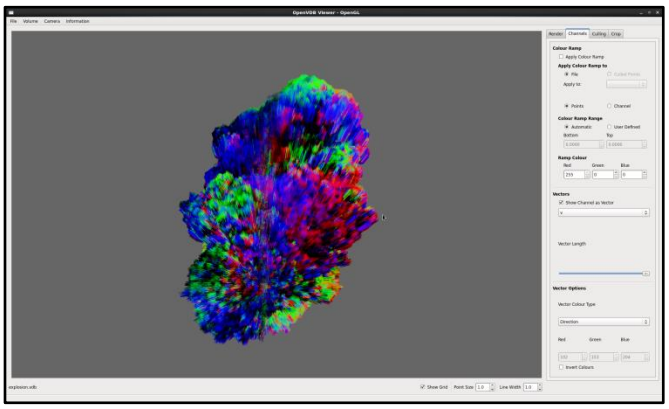
The user defined colour ramp will take in the two values in the Bottom and Top spin box as the lower and upper values respectively. It will then ramp between these values. These spin boxes have a minimum value of -10000 and a maximum of 10000.

5.2.1.3 Channels – Colour Ramp: Ramp Colour

You can manually set the colour of your colour ramp by setting the Red, Green and Blue values in this group.

5.2.2 Channels – Vectors

As well as viewing the channel data as a colour, you can also view it as direction vector. To enable this view, check the Show Channel as Vector box. This will not only show the channel data as a vector in the render window, but also enable the drop down box below. This will allow you to choose the channel to display as a vector, meaning you can view one channel as a colour and one as a vector, all at the same time!



Channels rendered as vectors

The slider below controls the length of the vectors. To change the length for display purposes, slide the slider left and right. The length of the vectors will update in real time.

5.2.3 Channels – Vector Options

The vector options gives you the ability to change the colour of the vectors being rendered.

The vector colour type drop down lets you choose between rendering the vectors as a single colour (which can be set in the Red, Green and Blue boxes below) or as the colour of the channel data at that point. This will colour the vectors the same colour as the point data.

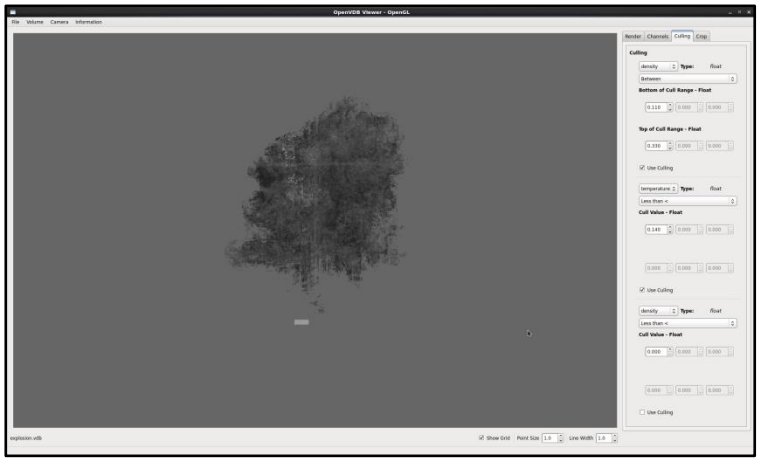
The invert colour option allows you to invert the colours, allowing a different view at the file.

5.3 Culling

OpenVDB viewer allows you to apply culls (or filters) to the high resolution data set. Up to three culls can be applied at the same time on any of the channels present within the loaded file. Each one of these culls is controlled by one group on this tab. This manual will describe one of these cull groups, but the same will apply to all three of them.

The first drop down box lets you select the channel to apply the cull to. When selecting a channel, the application will inform you of the type the channel is to the right of the drop down box.

The drop down box below this gives you three options for the type of cull you wish to perform. The first of these is a less than cull. This will remove all points that are below the specified value in the enabled value box below. The second is a between cull. This will activate another value spin box, the top one for the lower value, the bottom for the upper value. This will cull all points that are not within that range. The final is the greater than cull, which will remove all points above the specified value.



Double cull, on channel density and temperature – original data set is the one shown on the previous page for vector rendering

If a channel is selected that is a vector type instead of a scalar, all three value spin boxes will be enabled for you to enter the x, y and z values respectively for the cull.

The range of these value boxes is determined by the extremities of the channel you have chosen.

Finally, the check box below the value inputs, the Use Culling check box, is used to enable the use of that cull or not.

5.4.1 Crop – Crop Model

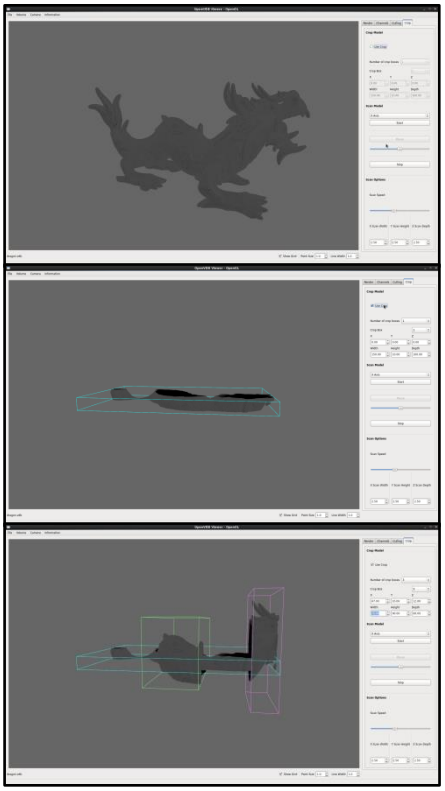
The final main tab is the Crop tab. Within here you are able to use and position multiple crop boxes to view only segments of the total high resolution data.

The crop model section is used to apply static crop boxes, up to 3, onto your model. To enable the use of crop boxes, simply click the Use Crop check box at the top of the tab. A single blue crop box will appear and the data points cropped to its constraints.

To add more crop boxes, use the Number of Crop Boxes drop down to select how many. You can have up to a maximum of 3 crop boxes at once.

To move or change the size of the individual crop boxes, use the Crop Box drop down below the number of boxes selection and select the crop box you wish to modify. Once selected, use the input values below to change its attributes. You will see it update in real time.

The range of these value boxes is determined by the extremities of the channel you have chosen.



Dragon data volume with no crop (top), 1 crop (middle) and 3 crops (bottom)

5.4.2 Crop – Scan Model

Another feature implemented by the OpenVDB viewer is the ability to 'scan' your high resolution data. When a scan is running, a pre-sized crop box based on the bounding box is moved across the data within its bounding box. This scan can be run in all three axis but only 1 at a time.

To select which axis, use the drop down box. When happy hit start and a single crop box will begin to move across the data points, cropping as it goes. You can start another scan at any time by hitting start again on any axis.

To pause a scan, simply hit the pause button. This will enable the slider below, allowing you to manually and precisely move the crop box to any point of interest you want. To resume the scene, simply hit the resume button and the scan will continue in its original direction.

To stop any scan, just hit the stop button.

5.4.3 Crop – Scan Options

Whilst a scan is running or when it is not, you can change some options.

The Scan Speed slider will change how fast the automatic scan moves across the bounding box. This can be adjusted to see those erroneous or interesting data points.

The three input boxes below the speed slider are used to set the size of the scan in each axis. As the scan will automatically orientate the crop box to the axis it is scanning, you can change them separately for each possibility.

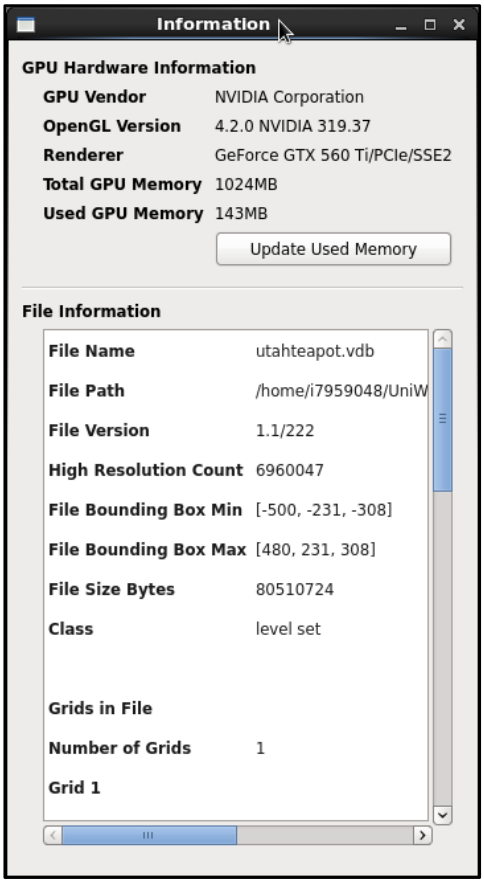
6. Information Window

When the viewer is run, it query's the connected hardware for certain information about itself. Such information as the device name and OpenGL version are fetched. These are then displayed within the information window. It is also within this window that all file information is displayed once a file has been loaded.

To access the information window, either use the Information menu option or click the Show File Information button in the render tab. This will bring up the window, or show it if it is already open. The value for the currently used GPU memory should update by itself if using an NVidia device. However if it does not due to other applications taking up memory, you can hit the Update Used Memory button on the information window to get a fresh look at how much you are using.

The information shown for the files is roughly structured to match the channels present and information available. The first section is general information about the file and the data within. The next sections focus on the channels, with each channel information displayed separately. Next information about the VDB tree within the file is shown. Finally, a dump of the raw metadata from the file is written to the table, just to make sure we haven't missed anything!

To close the information window, simply hit Esc/cmd+Q on MacOSX or press the close button on the top of the window.



Information window filled with information

7. Known Issues

Unfortunately there a couple of known issues with the OpenVDB Viewer in its current state. I have documented them here to make you aware of any possible strange behavior, and that they have been noted to be fixed for the next development stages.

The first of these is when loading in very large files with a very high point count for the high resolution data set. The error here is a hardware issue and limitation. On all machines currently used for development, including one a 2GB GPU, a couple of the files do run out of GPU memory before they are able to draw the points. This can either cause the application to close or only a portion of the points being drawn.

Another known issue is when closing the application, very rarely it will suffer a segmentation fault. As it occurs so rarely, I have yet to pinpoint the problem and am still trying to catch it in action.

The quality of the line drawing on Linux is also in question. Currently they draw blotchy and broken whereas on MacOSX they draw cleanly and sharp. This is something I am looking into fixing.

Another issue I am aware of is on MacOSX, not only does the line width value have no effect but the points start at such a small size, it can be difficult to see the data. This is again something I will be fixing in future developments.

8. Future Developments

Whilst there are of course many things I could implement into this application in the future, there are a few things I will be focusing on first.

The first of these is the ability to not only render the volume data as a mesh, something inbuilt into the OpenVDB library but I did not consider crucial for this version of the viewer, but also be able to export VDB volumes as meshes and read in meshes and export as VDB volumes.

I also intend to implement a frame system, where a series of frames of VDB files can be loaded in and then played back to the user. At each frame each of the implemented features will be available and be able to be used as playback is occurring.

Development for Windows is also something I want to focus on soon, to make this application useful on all platforms.

User interaction on the render window is also something I wish to implement to make manipulation of the data easier and faster. This may include re sizing of the crop boxes or querying certain points or groups of points.

As a bit of future thinking, I also want to look at a tool to be able to load in multiple VDB files and conduct certain processes on them such as combining them or performing a Boolean difference on them and then using the result in some way, such as exporting as a new VDB file.