LCSE
Hierarchical Volume Renderer (HVR)
User's Guide

Software by David H. Porter,
Guide by Paul R. Woodward
April 1, 2002

This guide is intended to introduce the student to the use of the LCSE's HVR software, written by David H. Porter, for the interactive generation of volume rendered images and movies. This software has developed over many years for use with compressed data generated by the PPM gas dynamics code, but it can be applied to a wide variety of other uses. An earlier version of this LCSE software was the BoB volume renderer built by Ken Chin-Purcell after a still earlier and less user friendly version by David Porter. Aspects of this software have also turned up, as a result of collaboration of the LCSE with SGI, in software available through Silicon Graphics for use on their equipment. The present version is targeted at Windows PCs with graphics accelerator cards that support the OpenGL standard.

Installing the HVR software on your PC:

Copy the HVR software files off of the DVD-ROM distributed in class or download them from the LCSE Web site at www.lcse.umn.edu/hvr. Among these files are two command files named SETUP_HVR_APPS.cmd and SETUP_MOVIE_APPS.cmd. After you have placed the HVR files in the directory where you intend to keep them on your hard disk, you need only to double click on both of these command files in order to prepare your machine to run the software. It is a good idea to create shortcuts for the two executable files, hvr_gui.exe and movie_gui.exe, by right-clicking on these files in the windows explorer window and then selecting the item "Send To" from the menu that pops up. From the submenu of this menu, you should select the item "Desktop (create shortcut)" in order to get these two utilities onto your desktop. Then you will never have to remember where you put all these HVR software files. (Of course, you can always find this location again by right-clicking on the desktop icons, selecting "properties" from the pop-up menu, and looking at the "target" listed on the "shortcut" tab.) As we will see, the hvr_gui.exe utility can be used to make movies, and the movie_gui.exe utility can be used to show them.

Hierarchical Volume Data Files, or hv files:

This movie making software operates on data files in a special hierarchical format. These are called hv-files, and they have the extension .hv which distinguishes their file type. We will not discuss in this guide how to generate hv-files, nor will we discuss in any detail the nature of their data format. However, it is very useful to understand basically what is in these files. Each hv-file represents a snap shot of the distribution of a particular variable that has been sampled on uniform rectangular grid blocks. It is easiest to think about a single, uniform sampling grid, although the hv-file format is more general than this. For each grid cell, the average value of the variable is represented by a single byte, which gives 256 levels (if we include the level 0). These 256 levels work very well for visualization, since most people cannot distinguish the difference between smooth gradations of shades of gray that are broken into these 256 discrete levels and gradations that are in fact completely smooth. Nevertheless, it is important to remember that there are only 256 levels in the hv-file data, since we can easily throw away large numbers of these levels by using a scaling of the variable that we are trying to visualize for which the top and bottom levels are attained somewhere on the grid, but so infrequently that the bulk of the cell averages fall within, say, only 20 or 30 of the 256 available levels. Although we will not discuss in this guide how hv-files are generated, the generation process is often done badly, leaving only a few of the 256 levels containing any truly useful or interesting information. Common techniques used to avoid this pitfall are to rescale the physical variable that the hv-file data represents before it is mapped linearly onto the 256 available, equally spaced levels. For example, the vorticity of a fluid flow is nearly zero in most places, with large values concentrated in small structures such as shear layers or vortex tubes. The values near zero have much to tell us about the flow, but unless we take special care, they will effectively be invisible in a typical graphics rendering. To see the structures of vorticity in regions of a flow where it is small as well as in regions where it is large, it is useful to render images not of the vorticity but instead of its arcsinh. The arcsinh of the variable x is given by $\log_e\left(x+\sqrt{x^2+1}\right)$. The arcsinh of a variable compresses the variation of the variable where it takes on large values, so that the variation of the variable near the value zero is much greater in a relative sense. In terms of our 256 levels, we might have had the range of the vorticity from –1 to

1 taking up only, say, 32 of the 256 levels, while the range from –8 to –1 and from 1 to 8 each absorb 112 of our levels. If we represent the arcsinh of the vorticity instead of the vorticity itself with our 256 levels, then 81 of our 256 levels will be used to represent vorticity values between –1 and 1 rather than the 32 levels that result from representing the vorticity directly. (This follows because $\text{arcsinh}(\pm 1) = \pm 0.8814$, while $\text{arcsinh}(\pm 8) = \pm 2.7765$.) An even greater compression of the range of large values can be obtained if instead of the arcsinh we use the tanh of the vorticity. For variables like the density, which are never negative, we can expand the range of small values by using our 256 levels to represent the logarithm of the density instead of the density itself.

Because a variable that will be visualized may have been put through some complex mapping, as in the examples above, before it was represented by the 256 levels in the hv-file, it is important to know this scaling before interpreting the visualizations that are made with HVR. For example, if the region near zero values of the vorticity has been greatly expanded (as can be done by using the square root or even the fourth root of the arcsinh of the vorticity), we may see all manner of fascinating features in the vorticity visualization which actually have very little dynamical significance. A frequent use of such misleading variable scaling is to help locate flaws or bugs in numerical algorithms that are used to generate the simulation data. For good simulation programs, these flaws are only seen at very low levels of significance, but we may nevertheless wish to make them appear larger in order to improve the simulation accuracy still further.

The hv-file is a hierarchical data representation. Within this single file, multiple representations of the variation of the variable on the uniform 3-D grid are stored. At the lowest level of this hierarchy, we have the data on the grid that was used by the simulation. This is the most accurate representation of the data. Because the grid is likely to be fine, containing typically a billion or more cells, the data is decomposed into chunks representing small bricks, typically 64 or 128 cells on a side, of the larger grid. At the next level of the hierarchy, the data is given on a coarsened grid in which each cell represents a $2\times2\times2$ brick of the actual grid. The data values in these coarsened grid cells are again cell averages, and hence they are averages of the average values in the 8 cells of the actual grid that are combined to make this coarsened grid cell. The data on this coarsened grid is again decomposed into chunks, and each chunk of the coarsened grid represents 8 chunks of the actual grid. It is now easy to see how we make a still coarser representation by combining cells together in little chunks of 8 and by combining 64^3 or 128^3 bricks together 8 at a time.[†] In this process of coarsening the grid over and over, we will eventually reach a point at which the multiply coarsened grid is so small that all the data it contains can be stored in a single chunk and rendered into images at interactive speed on even an old (1-year-old, that is) laptop PC. This is the point at which the grid coarsening process stops.

The hv-file contains all of the data representations described above, all chunked up at each level of the grid hierarchy. Placing chunks of the data together in the file makes it possible for a PC to read a chunk from its disk very fast. With the Nvidia GeForce-3 graphics cards, for example, this chunked data must be read into the PC's memory at a sustained speed of 150 MB/s in order for this data movement to keep up with the graphics board's very high rendering speed. For this reason, reading the data in contiguous chunks is absolutely essential. The chunked data format of the hv-file is advantageous also from the standpoint of volume rendering efficiency. If we are looking at a flow field that is described by a one-billion-cell grid, and if this grid is decomposed within the hv-file into 512 bricks, each 128 cells on a side, then for a typical view of the flow some of these bricks will be close by, and we will want to render them in complete detail, while others will be relatively far away, so that they need not be rendered so carefully. Inside the hv-file each such brick has associated with it so-called "metadata" that quantifies how much variation, in levels, there is within the brick. From this information, together with the number of pixels on our screen that will be used to represent this variation, we can decide whether to use the fully detailed data in a group of bricks or whether instead to simply render the brick of the coarsened grid that represents the entire group. This kind of decision can rapidly be made from the metadata and the geometry of the view, and it can save an enormous amount of needless rendering labor and time. It is important to understand this feature of the hv-file format, because these rendering decisions are controlled by user-specified rendering quality parameters which we will discuss later on.

One might think that with all this extra data contained in each hv-file the files would grow very large. This is not the case. To describe the variation of a variable on a grid of 1024^3 cells using 256 levels, we require 1 GB of data. Putting all this data into the hv-file format, we end up with only a little over 1.2 GB. Thus the space cost of the hierarchical data format is only 20%, not much when one considers the rendering time that this format can save.

[†] This description is slightly oversimplified. The chunks in the hv-file actually overlap slightly, so that images can be constructed from them without visual artifacts at the chunk boundaries.

Interactive and Presentation Image Rendering:

A tremendous benefit of the hierarchical structure of the data in an hv-file is the ability to generate images at user-specified levels of quality. With image rendering, as with so many other aspects of life, there is an inverse relationship between quality and speed. HVR can draw images very quickly, as long as we do not require that these images be of high quality. HVR supports a dual concept of image quality. The user may specify quality parameters for both types of HVR images: interactive and presentation images. Interactive images are displayed in the image window of the HVR graphical user interface (GUI), while presentation images can be rendered by a remote cluster of worker PCs directly to a movie file on disk for later animation with the movie_gui.exe utility. If the user desires, presentation images can also be rendered and displayed in the image window of HVR_gui. These images can either be rendered by the user's PC or by a remote rendering engine on the network.

By rendering rough images in the window of the user's PC, HVR allows one to interactively explore a data set in order to determine views that one would like to see at higher quality. These can be single views, or they can be entire movie sequences. They can be generated from a single hv-file, or they can come from an entire sequence of hundreds or thousands of hv-files. The idea is that while the user is rapidly moving around in the data, either changing the viewpoint or the time level of the snap shot that is viewed, only rough images are drawn in the window on the user's screen. These images are of course drawn from the coarsened representations of the data in the hv-files. As soon as the user lingers on an image, the quality of the image is progressively improved. The user may not notice this so much unless he or she lingers for a while, since quality images do take quite some time to draw (37 seconds for a billion voxels @ 38 MB/s disk I/O, or 8 seconds if the disks are really, really fast).

An interesting mode of HVR use is possible when the user's PC gives commands to a rendering cluster that draws images on a paneled display like the LCSE PowerWall. In this case image rendering occurs at roughly the same speed on the PowerWall as on the user's screen, but the images are very much larger, and they contain correspondingly more detail (10 times as much on the LCSE's 10-panel PowerWall).

The Keys to Movie Making with HVR:

HVR is designed to enable the user to very efficiently and rapidly create a series of "key frames" that will define a movie. The user employs the interactive performance of the rough rendering mode of HVR to maneuver rapidly in the data set and to select a series of views. These views are characterized not only by the position of the viewer, the direction of the line of sight, the direction of "up" (i.e. the orientation of the viewer's head), and the positions of near and far clipping planes, but they are also characterized by several other parameters. These include clipping planes oriented relative to the data brick that is being viewed and which move with that brick. They also include the color and transparency (alpha) maps that are used to paint the image. Finally, they include the time at which the data is being viewed. That is, they include the number in the sequence of hv-files that is on view. We will presently discuss how all these view parameters are controlled by the user through the HVR_gui user interface. We list all these view parameters here to drive home the point that when HVR constructs a movie sequence between two key frames created by the user, HVR interpolates a smooth transition between all these parameters. This means that movie sequences can be designated in which, for example, the transparency or color, or both, of the data brick smoothly changes from one set of values to another. Sequences can also be designated in which clipping planes, either "attached" to the object or to the viewer's line of sight (the viewing frustum), gradually move in or out. Sequences can also be designated in which time in the simulation under study advances or retreats. And of course the viewer can move the viewpoint and direction in an animation sequence. HVR allows all these animation techniques to be used, either separately or together. The user specifies the number of interpolated frames between any pair of key frames, so that the timing of these transitions can be very carefully controlled. The result of this tremendous flexibility in HVR movie making is that HVR movies from experienced users working with complex and highly detailed data sets have a very professional look. When made for PowerWalls or stereo displays of either wall or CAVE design, these movies can be stunning.

The Concept of a Movie Path:

A series of key frames that defines an HVR movie can be used in other ways. Almost by definition, such a series of key frames amounts to a specification of what is interesting and/or important in the data set. Together with the many intermediate frames that HVR interpolates automatically, these key frames define a path in the space-time context, or world, of the simulation under study. HVR allows the user to save this path in a condensed form on disk for future reference. For example, this path can be used at a later time to construct a movie of a different variable. Then one has two movies. The first shows what the user selected as the most interesting structures and dynamics in, for example, the vorticity in a certain fluid flow simulation. The second might show what is happening, for example, to the entropy or the temperature of the fluid along this same path in space-time. On a PowerWall, both movies can be displayed simultaneously and viewed side-by-side. This proves to be a very effective means of appreciating and understanding multivariate data, especially when these movies can be stopped by the user at any especially important point, and the user can step carefully frame by frame forward or back through the sequences around that point. This capability relieves the user of the need to construct extremely complex images, showing multiple variables at once in each view, that take months or years of training to properly and rewardingly interpret. For fluid flows, we have found that the most useful information can be extracted from viewing no more than three or four fields of carefully constructed and scaled data. Animating these simultaneously on a PowerWall using movies constructed along the same path is an excellent way for a scientist or group of collaborating scientists to determine the essence of the simulation.

Feature Extraction and the Movie Path:

A classic problem in scientific visualization is that of feature extraction. One has a huge data set, and one would like to extract from it only some relatively small set of interesting features. HVR presents the user with two ways of performing this task. The first is for the user to devise a mathematical formula involving the fundamental variables stored in the data set which results in a quantity that is large in magnitude only at the locations of these interesting features. Using the LCSE's utility A3D, which will not be described in this guide, hv-files are then constructed which represent this special quantity. When HVR is used to visualize this quantity, the areas of interest can be made to be the only regions with high enough opacity to be visible at all. This method is direct and elegant, but it is generally very difficult. The stumbling block is coming up with that formula. Often a scientist knows what is interesting when he or she sees it, but cannot get too specific, and certainly not too quantitative, about what it actually is. This is a situation familiar to all of us. And HVR has a solution for it. HVR makes it so easy and fast to have an image rendering cluster do all the hard and tedious work of generating movies, that the scientist can simply make hv-files and simple movies (defined by simple, even dumb paths) of all the simple functions of the variables stored in the data set that might possibly reveal something interesting. Then, in a relatively short time he or she can review all these movies (typical movies are less than 5 minutes long, even for the largest data sets on earth) and use HVR's ability to maneuver within the data set to define a movie path that visits all the points of interest (in both space and time). In this process, which is straightforward and has been used for many years in the LCSE, it is the human perceptual system that performs the difficult task of interesting feature extraction. The large and tedious task of presenting the vast data volume to the user's visual system is, appropriately, performed by the dumb but powerful machine. For perusing data sets from the world's largest fluid flow simulations, at least, data volumes are not yet large enough to make this procedure inefficient and thus to force us to place intelligence anywhere other than in the user.

Use of Movie Animation to Display Vector and Tensor Fields:

Much thought in the computer graphics community has been devoted to the problem of displaying in a single image data that has more than one value at each point in space. We have just mentioned one technique that HVR enables for overcoming the need to do this. The solution is to create multiple images, each showing one of the values from the group of values at each point of the data set and to show them simultaneously, side-by-side on a PowerWall display. However, vector and tensor fields are special cases of multivariate data. It is usually not helpful to display side-by-side images of, say, the horizontal and vertical components of the velocity in a fluid flow. The mind requires great effort to create from these an impression of the flow speed and direction. It is like adding – a task that does not come naturally to the human mind – as opposed to seeing, hearing, or talking. But we perceive vector fields all the time in a nearly effortless manner. We do this when we view leaves being blown around by the wind or bubbles carried along with the water of a stream. The natural way for humans to perceive vector fields is by animation. That is to say, by movies. Much of the effort devoted to making images of vector fields was invested because of the limitations of publishing in former times. Single pictures could be published, but not movies. No longer. HVR allows one to animate views of any passively advected variable that is contained in the data set. (A passively advected quantity is something like a leaf born by the wind, or a dye carried along by a brook. It is passive, because it does not react dynamically upon the flow. Instead, it simply reveals the flow.) Viewing a passively advected variable gives an immediate perception of the velocity field.

One can also get quite a good impression of a velocity field by viewing a variable that is not passively advected, so long as this quantity moves relatively unchanged over short distances with the flow. An example of such a quantity for a fluid flow is the entropy. (The entropy is a bit like the temperature, for students unfamiliar with this flow variable.) The entropy can change its value as it moves with the flow, but it does so only gradually, as a result of diffusion (which is like thermal conduction) or it does so suddenly at only a few very special locations (shock fronts). Another variable that tends to show the velocity field well is the vorticity. The vorticity, to a first approximation, moves with the fluid. It also changes as it moves, but this very important dynamical effect is secondary. Animating images of the vorticity has a special power for certain kinds of flows, since the vorticity is, after all, derived from the velocity field. The vorticity magnitude is a scalar quantity (one value at each spatial point) that represents a special condensation of the information contained in all three components of the velocity. The dynamical importance of the vorticity is a matter for fluid dynamics experts, but suffice it to say that movies of the vorticity not only show the velocity field rather directly but also reveal some of its most important structures in a very special way.

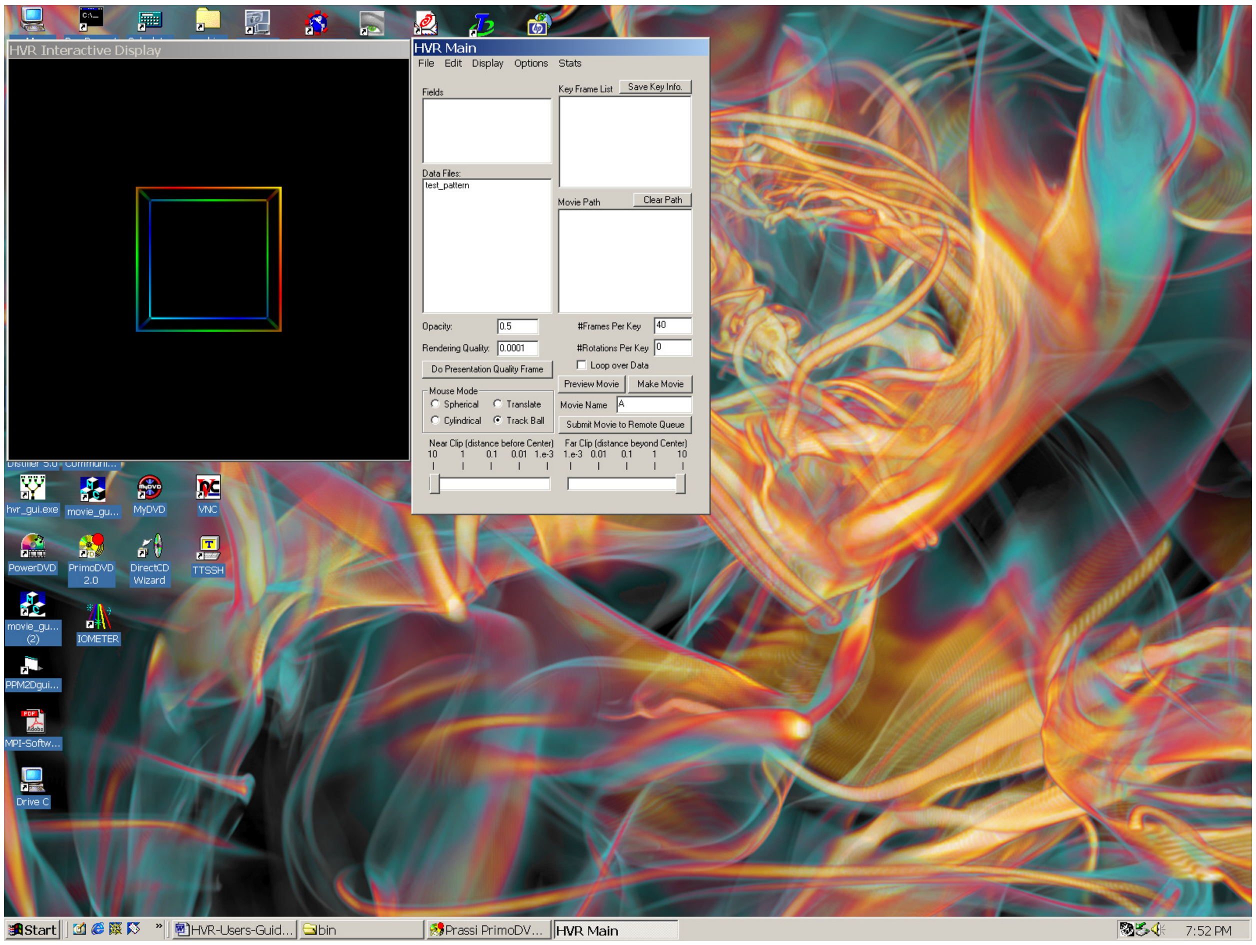
If it is difficult to show a vector field without a movie, it is really difficult to show a tensor field that way. A tensor is something that has even more than the 3 components of a vector at every spatial point. There can be 27 components at every point for a tensor, many of which may not turn out to be independent. To best see how we might visualize a tensor using a movie, it is useful to try to understand why we want to look at tensors anyway. Using fluid flow again to illustrate this, consider the stress tensor. The many components of this tensor quantify what will happen to a little cube of fluid that starts at a given point and travels a short distance with the fluid flow. Because the shape of this little cube can change in so many ways as it flows along, there are lots of components to the stress tensor. But probably what is important to the scientist is whether or not this little bit of fluid is sheared, whether it is compressed or expanded, or whether the flow in this small region has one or another of a small set of classic behaviors, such as a similarity to the flow at a stagnation point at the tip of, say, an airplane, or a similarity to the time reversal of this sort of flow (which is like squeezing toothpaste out of its tube). All these things can easily be identified in a movie animation of either a set of tracer particles in the flow (smoke represents the limit of very many very small tracers) or by a movie of a passively (or almost passively) advected variable like the entropy or a dye. We need the stress tensor, with all its many components, to perform mathematical and quantitative analysis of the fluid flow data, but not to understand what the flow is doing and, even, why. This is a statement of belief with which not all experts will agree. It is not that detailed and quantitative analysis is not important, it is instead that this is not the proper function of scientific visualization.[‡]

Getting Started:

Now that you understand the basic design and function of HVR, let's get started using it. Grab ahold of a representative hv-file from either the disc you were given in class or by downloading it from the LCSE Web site at *www.lcse.umn.edu/hvr*. The illustrations in this guide were made from the following hv-file: v72dT302.hv. This is a snap shot (#302) of the relative temperature distribution (hence the "dT") in a very simplified model of a red giant star of 3 solar masses viewed near the end of its life as a giant. The star is roughly spherical (of course), and you can think of it, if placed at the location of our sun, as extending to about the orbit of the earth. It has a central very hot core (about 100 million degrees K) which will be immediately recognizable in the images we will draw. In reality, the central, heat generating core of such a star would be about the radius of the earth in size, that is, an almost invisible speck at the center of its outer envelope. Instead, we have immobilized all the material in this stellar model out to a radius of one tenth that of the outer envelope. This is a falsification, of course, but it keeps the volume of this region small – only 1% of the volume of the whole star – and it allows us to resolve the flow of material near the core on our computational grid, which for this small data set has only 512^3 cells. In this simplified stellar model, the temperature, when averaged over concentric spheres, runs from millions of degrees at the surface of our enlarged core to just about 3500 degrees K near the surface of the envelope. In the hv-file we will visualize, this average run of temperature with radius has been subtracted out. We use our 256 levels in the hv-file to represent temperature differences relative to this average run. Large values are of course relatively hot, while small values are relatively cold. We have also demanded that the mid level, level 128, corresponds to zero relative temperature.

In our hv-file we have subtracted out the very large general run of temperature with radius because we as scientists already understand this from half a century of 1-D stellar evolution models. (If you, as a student, do not understand this, you might consult any introductory astronomy text.) Because this envelope material is heated from below by the heat coming upwards in the form of light as a result of nuclear reactions in the core, and because this heat is generated much too rapidly to stably diffuse out of the star, there is an unstable run of the entropy with

[‡] Perhaps this point is more easily made using the more familiar example of the proper uses of line plots and of equations. Equations specify detailed and quantitative relationships between variables. Line plots give the scientist a rough impression at a glance of these relationships. If we were to get out our rulers and micrometers, we could recapture the equation from the line plot, but this is usually a waste of time and effort. It is not the purpose of the line plot to be quantitative. Its purpose is instead to encapsulate the information contained in the equation, for which the human mind has no natural means of immediate understanding without many years of training, and to present it, roughly, all at once to the human visual system in a format that can be taken in and comprehended at a glance, with a little but not very much training.



HVR Main

Fields

Key Frame List

Data Files: test_pattern

Movie Path

Opacity: 0.5

#Frames Per Key 40

Rendering Quality: 0.0001

#Rotations Per Key 0

Loop over Data [ ]

Mouse Mode: Spherical, Translate, Cylindrical, [✓ Track Ball]

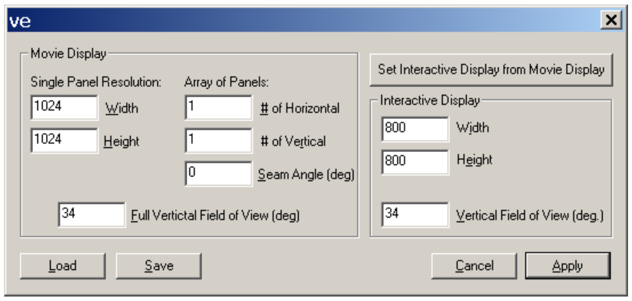
Movie Name A

Near Clip (distance before Center) 10 1 0.1 0.01 1.e-3

Far Clip (distance beyond Center) 1.e-3 0.01 0.1 1 10

depth in the envelope. (Don't worry if you don't understand this. It is included here to satisfy the astronomers who might end up reading this guide.) The result is that regions near the core that are ever so slightly hotter will become buoyant and rise upward in the envelope. The unstable run of entropy guarantees that as they rise they will become ever more buoyant, and hence rise further. Of course this process in any real star will have been going on for literally eons. As a result, the envelope fluid will be in an extremely turbulent state of convection. It was to understand how this convection might be related to the pulsation that stars like this undergo that this simulation with the LCSE's PPM gas dynamics code was performed.

Now that you know the basics about the data in the hv-file we will use as an illustration of HVR techniques, let's get on with it. Double click on the HVR_gui icon that you placed on your desktop to activate the software. A default viewing window of 512 pixels on a side will pop up along with a window that allows you to specify what will be rendered in the viewing window. These windows will appear as shown here at the top of the page. Actually, on your screen these windows might appear larger, if you did not spring for the UXGA screen option (I couldn't resist it). Right at the outset of your session you can, as I do, change the size of this viewing window. Be aware, however, that the rendering speed will be affected by the number of pixels that will have to be filled with your images. I like a setting of 800 pixels on a side, but 640×512 (half of SXGA resolution) is another popular choice. To change the viewing window size, go to the "Display" menu and select the unlikely sounding item "frusta." Don't fall into the trap and select "Quality" because that is something quite different. The dialog box shown at the right will pop up. Like everything else in HVR's user interface, this box takes a bit of explanation. You can see that I have already replaced the default 512 pixel designations by 800's in two places on this form and by 1024's in two others. The viewing window on our screen is referred to here as the "Interactive Display," since the intent is to



Movie Display

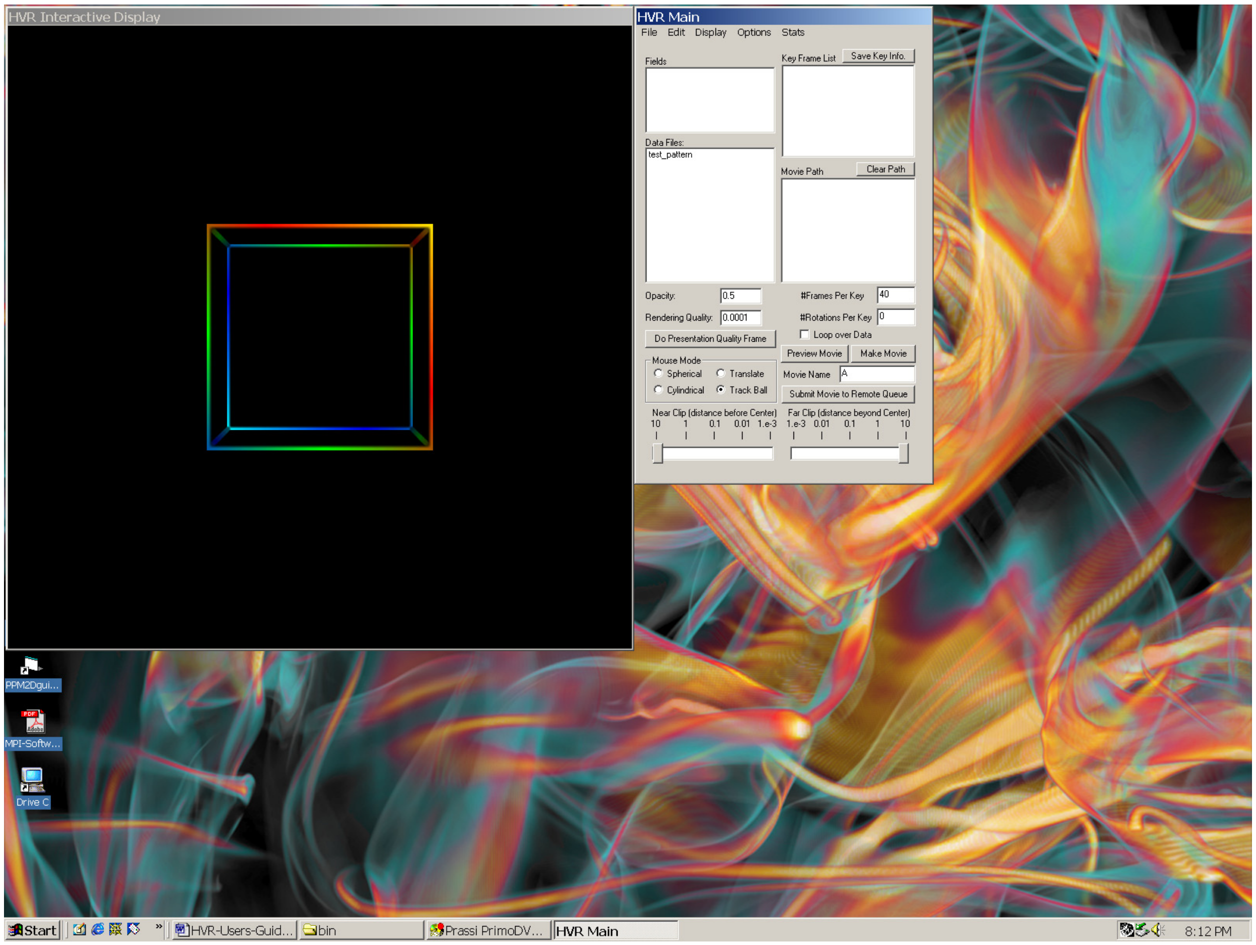
Single Panel Resolution: 1024 Width; 1024 Height

Array of Panels: 1 # of Horizontal; 1 # of Vertical; 0 Seam Angle (deg)

34 Full Vertictal Field of View (deg)

Interactive Display: 800 Width; 800 Height

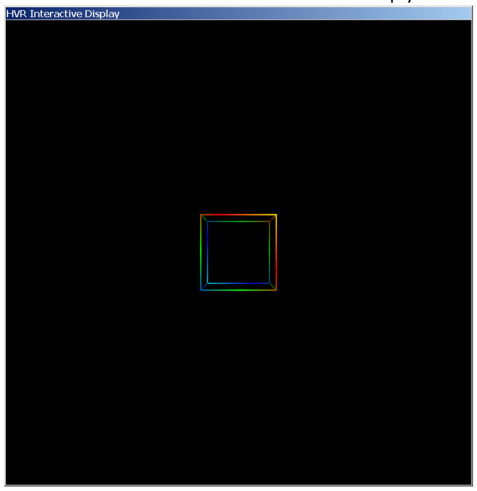
34 Vertical Field of View (deg.)

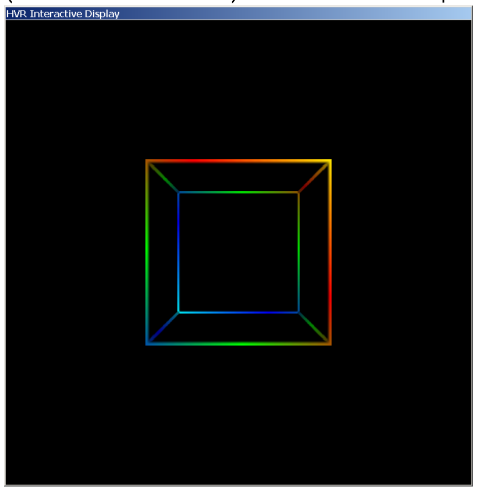


Data Files: test_pattern
Opacity: 0.5
Rendering Quality: 0.0001
#Frames Per Key: 40
#Rotations Per Key: 0
[ ] Loop over Data
Mouse Mode: ( ) Spherical ( ) Translate ( ) Cylindrical [✓ Track Ball]
Movie Name: A

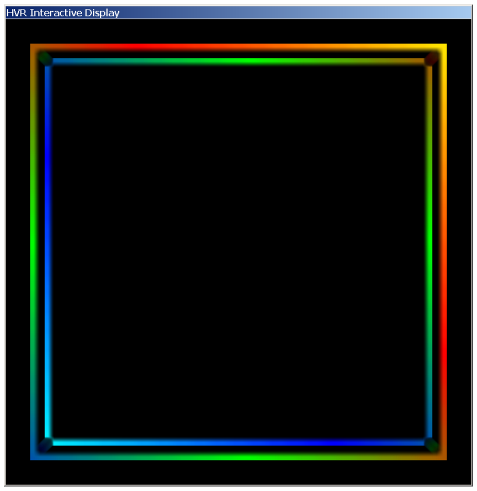
draw quickly into this window in order to support interactive exploration of the data to be visualized. We will see that if we so desire, we can obtain very high quality images in this window, save them on disk as Windows bitmaps, or even generate a movie from the images rendered into this viewing window. Thus for a typical PC laptop machine, these additional modes of use do not correspond to an interactive display. However, this window is almost always more interactive than anything else. Hence its name is appropriate.

You will notice that at the bottom of the settings area for the interactive display is a parameter labeled "Vertical Field of View (deg.)." This label makes enough sense in English, but its relation to what is drawn in the viewing window is not nearly so clear. Actually, it is directly related to the viewing frustum, so we might as well define that now. A frustum is a four-sided pyramid (like the ones in Giza) that has had its top chopped off by a plane cut





parallel to its base. The rectangle of the viewing window defines the ratio of the width to the height of this pyramid's base. The position of the "far clipping plane," which you can set with controls on the main form, defines the distance of this pyramid base from your eye. The near clipping plane, also set with controls on the main form, defines the top of the viewing frustum, while

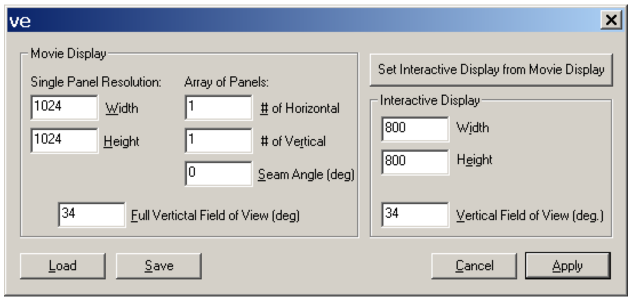


your eye is of course at the apex of the pyramid. The viewing frustum contains what will be visible on the screen. Everything inside the little (or big) bit of the pyramid between your eye and the near clipping plane will be invisible. Also, of course, everything beyond the far clipping plane will be invisible. Setting the positions of these clipping planes therefore allows you to limit what region of your data will be visible, and this will allow you to reduce the confusion that can result when too much stuff crowds the field of view. (Astronomy would greatly benefit if somehow we could install clipping planes. The first infrared satellite, IRAS, was "confusion limited." That is to say that it captured light from so many objects, that it was almost impossible to sort them all out.)

The vertical field of view in degrees, which you can set on the frusta (Latin plural of frustum, like spectrum and spectra) dialog box we have been discussing, is set by default at 34 degrees. This is a natural setting, since by just looking around a room you can easily convince yourself that without rolling your eyes you cannot see outward and up by much more than this. Increasing this setting, which is rather strongly not advised, is like installing a fish-eye lens on a camera. Suddenly the field of view is much larger. To generate the image in the viewing window that is shown at the bottom left on the previous page, we have doubled the default setting, so that the vertical field of view is now 68°. Notice that the little colored cube has become roughly half the apparent size. It is not further from us, but we now have fish or mosquito eyes, so it fills much less of our field of view. If we bring it closer to us, so that it fills roughly the same fraction of our view as before (we will see how to do this presently), we obtain the picture to the right on the bottom of the previous page. Note that now the perspective is more extreme. The apparent difference in size between the near and far faces of the cube has become roughly double the value it had originally. This makes sense of course, but you have to think about it. Or you can simply never fool with the default value for the vertical field of view, a course that has been chosen with great success by generation upon generation of LCSE graduate students.

It is perhaps worthwhile to make a short diversion here. Since we are discussing viewing frusta, it should be mentioned that setting the vertical field of view in degrees to a very small value will result in a classical limit of what is called "orthographic projection." It is said that medical doctors prefer orthographic renderings of 3-D objects, such as babies in the womb. In my opinion, this is a simple case of making a virtue out of perceived necessity. The graphics professionals who first supplied the medical industry with equipment chose to simplify the internal operation of that equipment by allowing it to render only orthographic images. The viewing window at the top of this page was obtained by setting the vertical field of view to only 5°. You can see that even though I have brought the wire-frame cube right up as close as possible, the near and far faces are nearly the same size. While setting the vertical field of view too large is like putting a fish-eye lens on our virtual camera, setting it too small is like looking through a powerful telephoto lens at very distant objects. In the limit in which we look at ever more distant objects through ever more powerful telephoto lenses, we reach orthographic projection. Now you should understand why you never want to do that, even if you are a medical doctor. Surely you have seen sequences in Hollywood movies which were filmed through powerful telephoto lenses showing, for example, cars approaching and approaching while they never seem to get closer. Remember how weird those cars appeared? So just forget about setting the vertical field of view to any small value. Once again, perhaps it is just best to forget about resetting the vertical field of view under any circumstances.

Now let's get back to the "Frusta" dialog box, shown once again at the bottom of this page. The "Movie Display" section at the left takes rather a bit of explaining. The vertical field of view item is as before, but now it is the "full" vertical field of view. The movie display can be either a single screen (or single "panel") or any geometrical array of screen panels. The pixel resolution of each panel is specified in the text boxes at the left, and I have chosen to set both the width and the height to 1024 pixels. This setting would be appropriate, for example, for a CAVE display. For the LCSE PowerWall, one would set the width to 1280 pixels and the height to 1024. This PowerWall has ten panels, 5 in the horizontal dimension and 2 in the vertical, so you would enter a 5 and a 2 in the text boxes beneath the label "Array of Panels" on the form. The "seam angle" in


ve

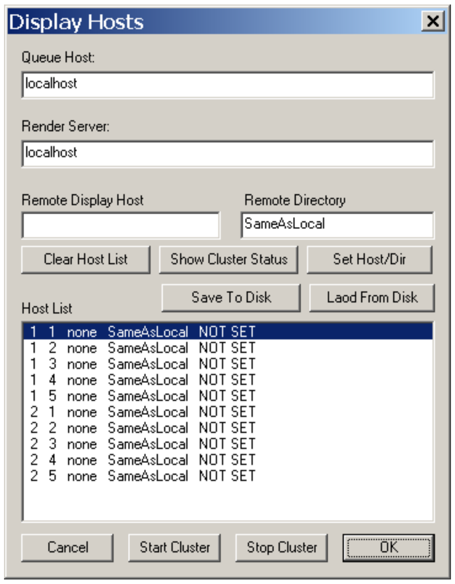
Movie Display
Single Panel Resolution: 1024 Width; 1024 Height
Array of Panels: 1 # of Horizontal; 1 # of Vertical; 0 Seam Angle (deg)
34 Full Vertictal Field of View (deg)

Interactive Display
800 Width; 800 Height
34 Vertical Field of View (deg.)

degrees is the angle by which the panels are rotated as you go from one to the next along the horizontal dimension. For the LCSE PowerWall, which is configured like a bay window, so that it subtends a full 90° in the horizontal field of view (actually, it subtends 112.5°, but that is a detail), has a seam angle of 22.5°. The vertical walls of a CAVE have a seam angle of 90°. Depending upon the target display for viewing your movie, you will fill out all these text boxes appropriately. Note that you can save your settings to a disk file by clicking on the "Save" button at the bottom of this form. And of course you can load saved settings from disk by clicking on the "Load" button. HVR will in fact render movies for any configuration of image panels, including more general configurations that cannot be specified on this simple (?) form. This can be done by constructing a configuration file on disk using a text editor and then loading it into HVR with the "Load" button. This is an advanced technique. Users wishing to learn how to do this should contact the LCSE staff. Suffice it to say that any configuration can be specified, and one may also specify parameters in the configuration file that determine the generation of stereo still or movie images. The HVR software has been used to generate stereo movies for the 6-panel RAVE at Los Alamos National Laboratory with great success (at least the Lab's Deputy Director seemed to like them, but perhaps he was just being polite). In the dialog box as shown on the previous page, I have selected single-panel 1024×1024 pixel movie images.

There is only one more button to explain on the "Frusta" dialog box. But it's a big button. It is labeled "Set Interactive Display from Movie Display." What could this possibly mean? More than you think, I think. At the very least, pressing this button will do the math for you to create an interactive display size that has the same aspect ratio as your chosen panel array (with zero seam angles). This is helpful in getting an impression on your screen of how the PowerWall images will appear. But it is even more helpful when HVR is run in the PowerWall visualization room. In this case, you can get the whole PowerWall (or any desired subset of its image panels) to act like your interactive display window. In addition to your own machine drawing rough images on your viewing window, the LCSE image rendering cluster will draw rough, but really big images on the entire PowerWall, using one rendering PC per image panel. These huge images will be drawn at about the same rate that the rough images can be drawn on your PC's screen. But if you leave the viewing parameters the same for a few seconds, the LCSE cluster will continually improve the image on the PowerWall up to the image quality limit that you specified (we did not discuss this quite yet, but we will soon) for your "Presentation Quality" images. This process is fully interactive, since the image on the PowerWall is drawn at whatever quality that time allows. It is fully interactive even when you explore very large data sets, since all ten machines that are rendering to the PowerWall can read the hv-file data at about 38 MB/s from a 2.8 TB shared disk file system. And the images can be stunning. This mode of HVR use really has to be tried to be believed.

To set up HVR to render interactively on the LCSE PowerWall, you must go to the "Display" menu on the main form and select the item "Hosts." Doing this brings up the dialog box that is shown at the bottom of this page. The "Display Hosts" as this dialog box calls them are just the PCs that will be used to render images to the panels of the PowerWall display. At the LCSE, these host identities are not fixed, since any machine can be set through a video switch to render images onto any panel of the display. Ask for assistance from the LCSE staff before filling in this dialog box, since you never know which machines are available for such a task until you ask. For PowerWall interactive display, you must skip the first two text boxes on this dialog box. Simply ignore them. The dialog box as shown here is what is presented to you if you specify a 5×2 array of panels for your movie images (as discussed above). Note that a default host list consisting of 10 hosts (all now designated as "none") has been generated. The two integers that precede each host name are its x- and y-coordinates in the array of image panels. You may select any one of these entries, although it is possible that no human has ever done this in any order other than going straight down the list (so no other order is sure to work). If, as is shown, the first host is selected (the one for the bottom-left image panel), then you should type into the "Remote Display Host" text box the name of the machine (rend01 or vis-01 are good choices) that will draw to this panel of the display. In the "Remote Directory" text box you should type in the full path to the directory containing the hv-files from which this PC will draw its images. If you designate the same directory for all ten hosts and for your local machine, there will be a lot of thrashing. It is therefore a good idea to have a friend (and it is a less good idea to do this yourself) create multiple copies of the hv-files on separate disk stripe groups (which are labeled by fruits at present in the LCSE). Then you can direct different PCs to read their hv-files from



Queue Host: localhost

Render Server: localhost

Remote Display Host:

Remote Directory: SameAsLocal

Host List

```
1  1  none  SameAsLocal  NOT SET
1  2  none  SameAsLocal  NOT SET
1  3  none  SameAsLocal  NOT SET
1  4  none  SameAsLocal  NOT SET
1  5  none  SameAsLocal  NOT SET
2  1  none  SameAsLocal  NOT SET
2  2  none  SameAsLocal  NOT SET
2  3  none  SameAsLocal  NOT SET
2  4  none  SameAsLocal  NOT SET
2  5  none  SameAsLocal  NOT SET
```

different parallel disk arrays of the LCSE storage area network. Since image rendering in the LCSE today (March, 2002) is strictly I/O limited, putting two copies of the hv-files on separate stripe groups will simply double the overall rendering performance. And 4 or 5 copies will do what you expect.

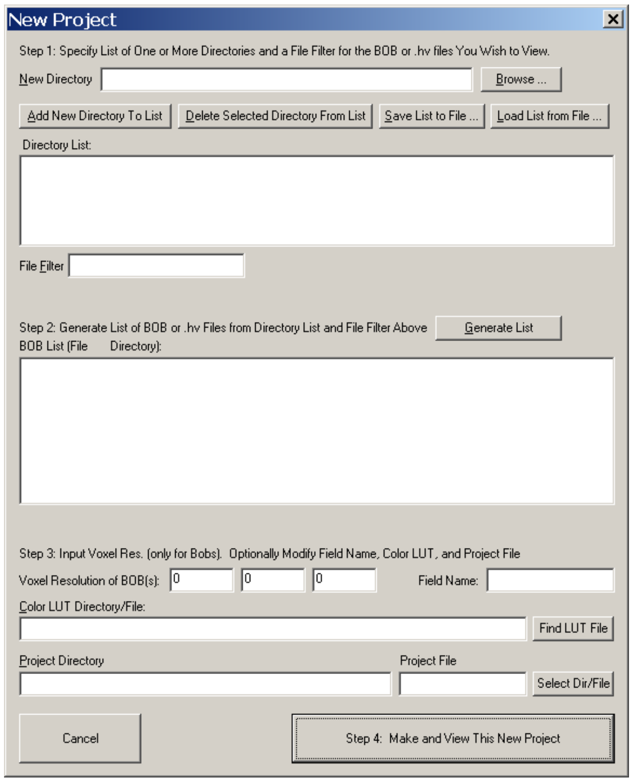
When you have typed in the information for a particular host on your list, just click on the "Set Host/Dir" button on the form to set these parameters. If you make a mistake, you can press the "Clear Host List" and the appropriate thing might happen. When you have all your hosts designated correctly, you can press the "Save To Disk" button to create a disk file containing this information. In a later session, you can save time by reading this configuration back in by pressing the "Load From Disk" button. Once your cluster is specified, you must "start" it if you want anything to happen. You do this by pressing the "Start Cluster" button and then waiting, perhaps a minute, until all the final entries shown in the host list are changed to "ON." At this point, it is advisable to press the "OK" button. You won't see anything on the PowerWall until you click your mouse inside the viewing window on your local display.

This last section about interactive PowerWall visualization is sketchy. This is an advanced technique, and you should not try it out without assistance from the LCSE staff.

Setting Up a Project:

You really can't make any images without first specifying the hv-files containing the data you want to look at. This is done by setting up a "project." A project is defined by a disk file ending in the extension .hvp which lists the locations of other files. At present, a project does not specify the parameters we have just learned how to set with the "Frusta" and "Hosts" dialog boxes. We will presently learn what parameters do correspond to a project, but first we mention the most obvious, namely the list of hv-files that this project will visualize. To designate these files, go to the "File" menu on the main form and select the item "New Project . . ." The dialog box that pops up is shown at the bottom of this page.

This is a complicated form, but there are good reasons for this complexity. Movie making usually requires enormous amounts of data. We may render images from hundreds or even thousands of hv-files. For rendering performance, since this process is always I/O limited, we sprinkle these hv-files out over as many as 20 Fibre Channel Arbitrated Loops (FCALs), or disk stripe groups, so that many files can be read simultaneously at full speed by many rendering PCs. It is therefore complicated to tell the HVR_gui user interface where all these files are located. The "New Project . . ." dialog box handles this task nicely. It is organized into a series of clearly identified steps. The first of these, at the top of the form, is labeled with the instruction: "Specify List of One or More Directories and a File Filter for the BoB or .hv files You Wish to View." (BoB files, or bricks of bytes, are supported for backwards compatibility with the earlier BoB volume rendering utility from the LCSE team, during its 5-year sojourn at the University of Minnesota's Army High Performance Computing Research Center. Don't use them. They have no hierarchical structure and hence cannot be rendered rapidly.)

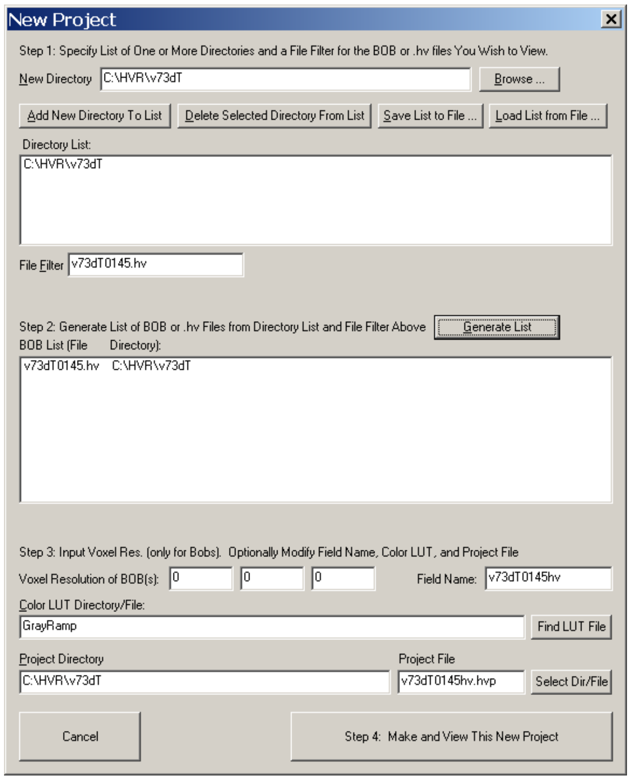

New Project
Step 1: Specify List of One or More Directories and a File Filter for the BOB or .hv files You Wish to View.
New Directory
Browse ...
Add New Directory To List
Delete Selected Directory From List
Save List to File ...
Load List from File ...
Directory List:
File Filter
Step 2: Generate List of BOB or .hv Files from Directory List and File Filter Above
Generate List
BOB List (File Directory):
Step 3: Input Voxel Res. (only for Bobs). Optionally Modify Field Name, Color LUT, and Project File
Voxel Resolution of BOB(s): 0 0 0
Field Name:
Color LUT Directory/File:
Find LUT File
Project Directory
Project File
Select Dir/File
Cancel
Step 4: Make and View This New Project

It is easiest, especially if you don't like typing, to designate your lists of hv-files by clicking on the "Browse . . ." button once for each directory, FCAL, or disk stripe group that contains hv-files you wish to see. In the pop-up window, just browse to the desired directory of the file system (to the right fruit, if you are working at the LCSE) and select any one of the hv-files you want. HVR will then type into the appropriate text boxes on the "New Project . . ." form both the name of the directory (in the top-most text box) and the name of the specific file (in the "File Filter" text box). Now place the cursor in the "File Filter" text box on the form and replace whatever text in the name of the single file that you need to with appropriate wild card characters (usually just a single "*" in the right place will do) in order to specify all the hv-files in the chosen directory that you want to look at. Then click on the "Add New Directory To List" button, and the directory you have designated will appear in the list box near the top of the form, meaning that it is one of your chosen. Now, near the center of the form, click on the "Generate List" button, and *voila* all the files you wanted will appear in a beautiful alphabetical ordering in the list box labeled "BOB List (File Directory):". (This was "Step 2" by the way.)

If all your hv-files are in a single directory, which is generally a bad idea, then you are ready to be promoted to step 3. Otherwise, you need to go back to step 1, browse to another directory, and do the above stuff again. If you have several terabytes of hv-files to view and do not know how to sprinkle them out over the LCSE stripe groups, contact the LCSE staff. The LCSE has little utilities to perform this sort of task (the graduate students asked to do it threatened to quit). Most likely, if you have several TB of data, you will have it somewhere else, since the LCSE does not archive other people's data as a general rule (we have more than enough data of our own). In this case, you will need to bring or send your data to the LCSE. A Gigabit Ethernet link into the LCSE from the world can accomplish this task for you, but you will have to contact Ben Allen at the LCSE in order to get your data through the massive security shield thrown up by the University of Minnesota to protect us from the unknown.

Step 3 is confusing. Don't fall into the trap. To avoid this trap, you need to understand a critical difference between BoB files (bricks of bytes) and hv-files. BoB files, as the name implies, are just bricks of bytes. That's all. They contain no header records, and therefore are not self-describing files. Just bytes. We chose this BoB format many years ago as being potentially universal. Everyone will disagree on what to put in a header record, so we decided to have no header record. It seemed inconceivable to us at the time that a user would not know what the grid resolution of the data was. You need to remember, if you were alive back then, that those were the days when we fought for every additional grid cell and rejoiced when we could scale our grid up from 200 to 250 cells on a side. Not only have those days passed into history, but the hv-file format demands metadata included in the file for its hierarchical data decomposition to be truly useful. Therefore hv-files are self-describing, and if you don't like the format, well, that's just tough. There has to be some format, so we decided that it might as well be this one. (If you need to know how to get your data into the hv-file format, contact the LCSE staff.)

The first set of text boxes for step 3 allow you to specify the x-, y-, and z-dimensions of your grid. You only need to do that if you are working with BoB files. If you have hv-files, this information is redundant and, hopefully, will be ignored (not guaranteed). The next text box on the form allows you to designate a "field name." The idea is that HVR will allow you to visualize scalar fields. If you are a mathematician, this is a familiar term. If you are a physicist, you think instead that HVR will let you visualize a variable. Same thing. HVR_gui will generate for you out of the stem name of the hv-files you selected a "field name," which appears in the text box of this label. From this field name, HVR will construct a field file which will contain a list of all the hv-files that you have selected. HVR also constructs a project file, with a file extension of .hvp, that contains the names and locations of all the disk files, like the field file, that are associated with your project. HVR has the concept of a field file, because it is possible to use HVR to visualize multiple variables (scalar fields), using the same movie paths for each. For now we will consider just one field, and for the moment just one hv-file. At the right you can see the "New Project" dialog box at the point where a single hv-file has been selected, its directory added to the directory list, and the file put into the BoB file list. The field file name has been generated automatically by HVR, as has the project file name and directory location. By default, HVR will place your field and project files in the same directory that contains the hv-files you select. This is not often where you want them to be. To change this location, simply click on the "Select Dir/File" button near the bottom of the form. But first, and you must do this first until this constraint is removed by better HVR program design, check the contents of the text box just above this on the form. There HVR has specified that the default simple grayscale color look-up table (LUT) will be associated


New Project

Step 1: Specify List of One or More Directories and a File Filter for the BOB or .hv files You Wish to View.

New Directory: C:\HVR\v73dT

Directory List:
C:\HVR\v73dT

File Filter: v73dT0145.hv

Step 2: Generate List of BOB or .hv Files from Directory List and File Filter Above

BOB List (File Directory):
v73dT0145.hv C:\HVR\v73dT

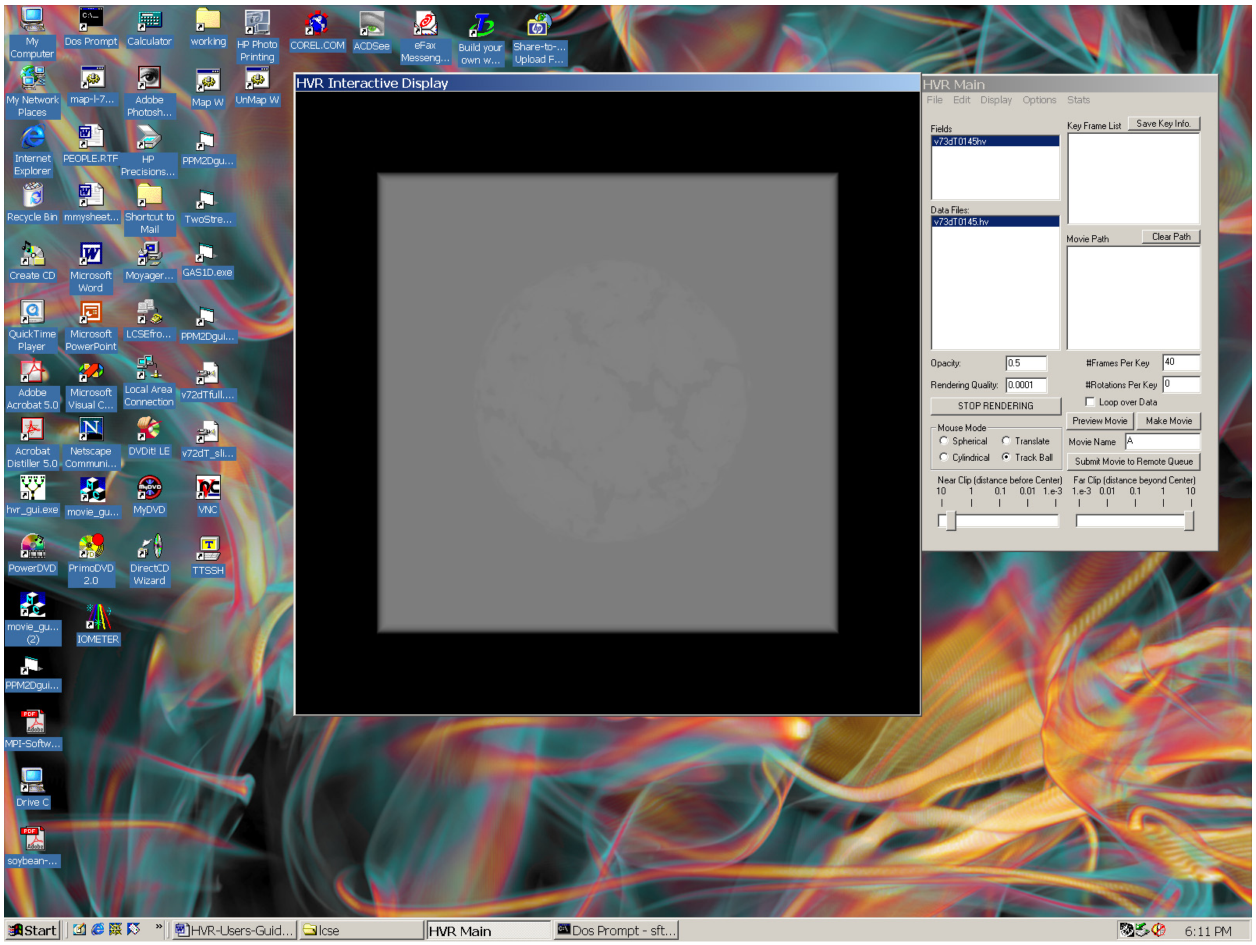
Step 3: Input Voxel Res. (only for Bobs). Optionally Modify Field Name, Color LUT, and Project File

Voxel Resolution of BOB(s): 0 0 0 Field Name: v73dT0145hv

Color LUT Directory/File: GrayRamp

Project Directory: C:\HVR\v73dT

Project File: v73dT0145hv.hvp

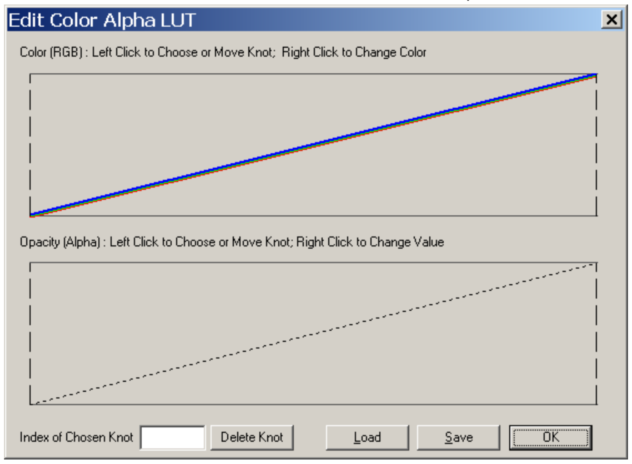


Fields: v73dT0145hv
Data Files: v73dT0145.hv
Key Frame List
Movie Path
Opacity: 0.5
Rendering Quality: 0.0001
#Frames Per Key: 40
#Rotations Per Key: 0
Loop over Data
Mouse Mode: Spherical, Translate, Cylindrical, [✓ Track Ball]
Movie Name: A
Near Clip (distance before Center): 10 1 0.1 0.01 1.e-3
Far Clip (distance beyond Center): 1.e-3 0.01 0.1 1 10

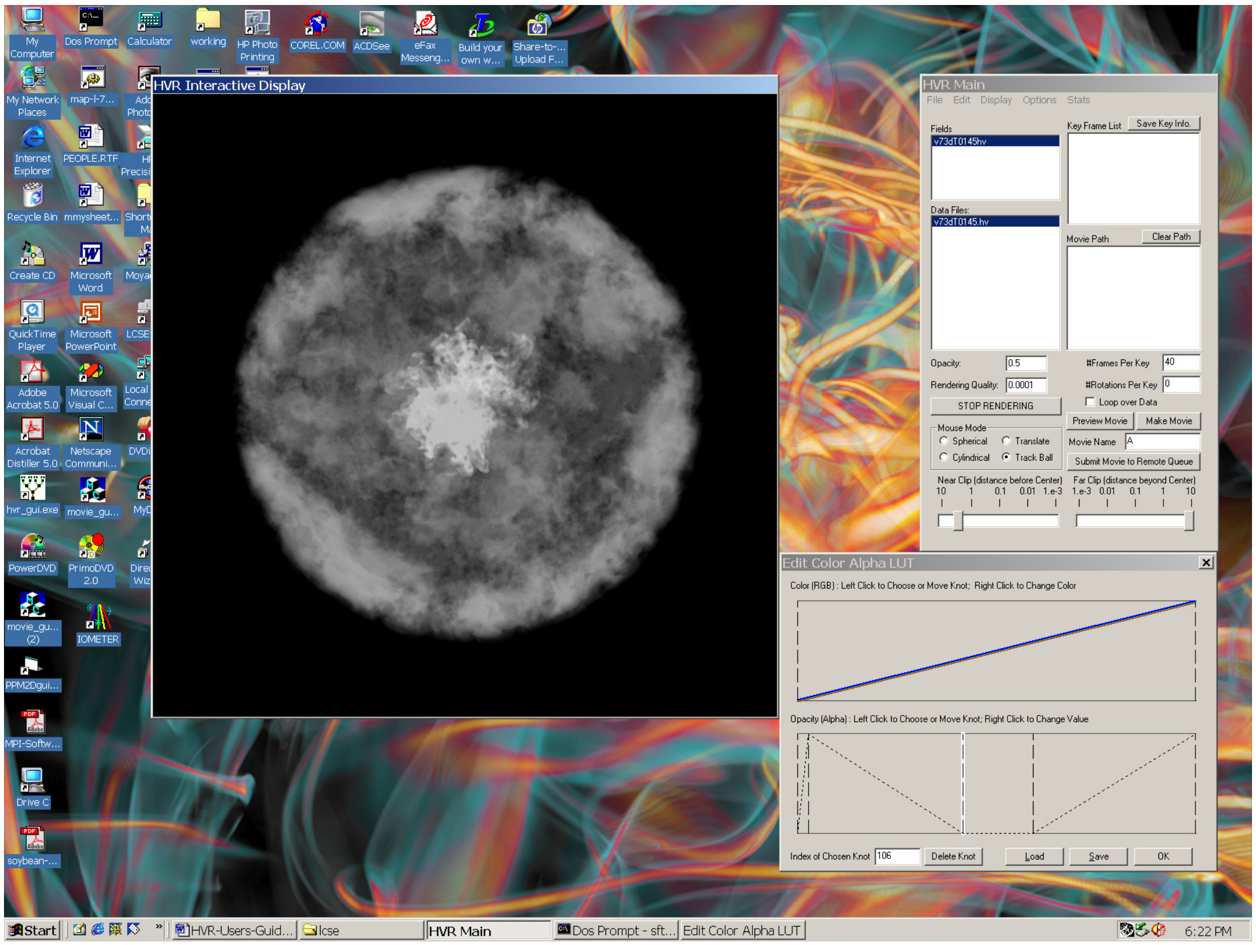
with your project. If you have a favorite LUT-file, you should click on the "Find LUT File" button and navigate to it to specify that it should be used with the project instead. For this example, we will just use the grayscale LUT-file. Now we can click on the button at the bottom of the form, labeled "Step 4: Make and View This New Project."

Editing the Color and Opacity Maps:

At this point, our screen looks like the image on this page. The gray square in the center of the viewing window is the volume rendered image of our data. If you look closely at this gray square, you can faintly discern a circular feature within it. The data in our hv-file is the relative temperature in our red giant star model, as was explained earlier. For this data, the level zero relative temperature difference is set to the level number 128, which corresponds to a mid-gray, half-way between white and black. Let's look at this default color look-up table. To do this we need only select the item "Color/Alpha LUT" from the "Edit" menu. The pop-up window that appears is shown here at the right. The top line plot in this window shows the values of the red, green, and blue color levels as functions of the level in the data (ranging from 0 to 255). The plots for these three primary colors are drawn in their respective colors. Here they all appear on top of each other, because for shades of gray the R, G, and B values are all equal. This default color table sets each color level to the data value. We will soon see how to change these settings.


Edit Color Alpha LUT
Color (RGB) : Left Click to Choose or Move Knot; Right Click to Change Color
Opacity (Alpha) : Left Click to Choose or Move Knot; Right Click to Change Value
Index of Chosen Knot
Delete Knot
Load
Save
OK

In the lower line plot in the "Edit Color Alpha LUT" window, the opacity, or alpha value, is shown as a function of the data value. This also is a straight line



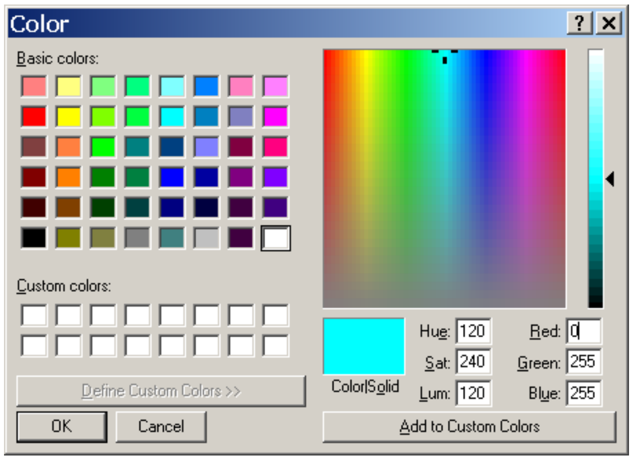
Fields: v73dT0145hv
Data Files: v73dT0145.hv
Key Frame List
Movie Path
Opacity: 0.5
Rendering Quality: 0.0001
#Frames Per Key: 40
#Rotations Per Key: 0
[ ] Loop over Data
Movie Name: A
Mouse Mode: ( ) Spherical ( ) Translate ( ) Cylindrical [✓ Track Ball]
Near Clip (distance before Center): 10 1 0.1 0.01 1.e-3
Far Clip (distance beyond Center): 1.e-3 0.01 0.1 1 10

Color (RGB) : Left Click to Choose or Move Knot; Right Click to Change Color
Opacity (Alpha) : Left Click to Choose or Move Knot; Right Click to Change Value
Index of Chosen Knot: 106

from zero opacity at zero data value to an opacity of 255 (totally opaque) at data value 255. This is our problem with the display of our data. Relative temperatures near zero occur in our cubic problem domain outside of our red giant star, and these are all represented by mid-level gray at 50% opacity. A simple fix which will let us see through these uninteresting exterior regions, where after all there is literally nothing to see, is to make these data levels transparent. This is easily done, but it will take a bit of explaining. The result of doing it is shown in the image of our screen above. Here the changed opacity map is also shown.

To generate the new opacity map shown in the image above we need to specify 3 "knots," or special data values at which we can prescribe desired values of the opacity. Setting the first knot is easy. All we need to do is to click at any desired horizontal location in the opacity plot, and a knot will immediately be inserted there. By holding the left mouse button down at this location, we can drag the knot to the left or right, so that it is positioned precisely where we want it. There is a rule about knot creation. Once you have positioned your first and only knot, if you click the left mouse button at any location that is closer to one side of the opacity line plot (to level 0 or to level 255) than it is to the present knot location, a new knot will be inserted where you clicked. Otherwise, the existing knot will be moved to the horizontal position where you clicked. This takes some getting used to. In this fashion I created 3 knots in the opacity plot. To set the values of the opacity at these knots, it is only necessary to click the right mouse button while the cursor is located at the knot. Then you can drag the opacity level up or down, as desired. In the opacity map that is shown above, I set a knot on each side of the central data value (level 128) and I set the opacity at each of these knots to zero. Then I let the opacity increase linearly from these knot locations to completely opaque near level 0 and level 255. The result is that the empty region around the model star is now transparent, as is also all the gas within the star that is near the average value of the temperature for its depth.

It takes a little bit of understanding of how the volume rendered image is created in order to interpret the image properly. The background, where I have no data, is black. Since I have made the empty space around my star but still within my cubical region of the simulation grid transparent, I see through to this black background. Therefore, you might easily assume that the regions within the stellar envelope that appear black are regions where I

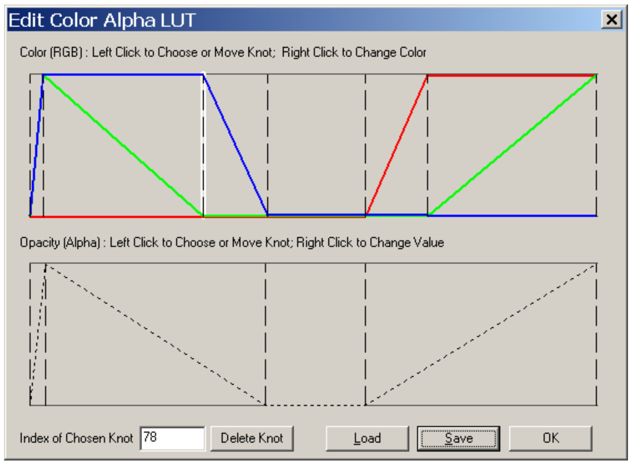


Hue: 120
Sat: 240
Lum: 120
Red: 0
Green: 255
Blue: 255

see right through the star to the black background. Wrong. In this rendering, black is considered a color, not the absence of color. Material in the stellar envelope that is much colder than average at its depth will, according to my color map, be represented as black. My opacity map also says that this material will be opaque. This means that this black material will blot out any lighter material behind it. So it will act rather like a dark cloud that absorbs star light from the Milky Way and therefore appears as a dark ink blot on the sky. This representation in the image of both empty space and also cold material as black is confusing. It is not a good idea. We should change it.

The change we need is easy to make. First, we introduce five knots in the color map in just the same fashion that we did with the opacity map. Again, at each knot location we right-click in order to change the value there. But this time the value is really a color, or a set of three values, one each for the level of red, green, and blue. To help us, the "Color" dialog box pops up. This dialog box is shown at the upper right. It is a standard element of the Windows operating system, so perhaps you are familiar with it already. In any case, it presents to you a matrix of standard colors to choose from, and also a rectangular color gamut. The slider at the right allows you to choose fully saturated colors by selecting the middle value, or colors that are blended with different amounts of white (the higher values) or black (the lower values). Hues are selected by clicking at appropriate locations within the rectangle. I personally find this color rectangle rather unhelpful, so I generally tend to type the actual levels for red, green, and blue into the text boxes provided at the right on the form. The color you have chosen is shown in the little rectangle in the lower portion of the form, and if you like it, you have only to click on "OK" and this color is applied to the knot you had selected on your color map. When you have constructed a color map that you like, you may save it to a disk file by clicking on the "Save" button at the bottom of the "Edit Color Alpha LUT" dialog box. By clicking on the "Load" button there, you may load in a color table from a previous session. The result of all these actions is shown at the bottom of this page. The color map uses "cold" colors, blues and aquas, to represent relatively cold temperatures in the stellar envelope, and it uses warm colors, reds and yellows, to represent relatively warm temperatures.

The image that results, after turning the star on a vertical axis by about 90° is shown at the top of the next page. The black and white image of the same view of this data is shown in reduced size below it. It is much more difficult to interpret, and of course it is also much less attractive aesthetically. This hv-file data comes from a simulation carried out by the LCSE team with the PPM gas dynamics code on a grid of a billion cells. Good images can come from good rendering software like HVR, but they can never come from bad data. This particular image reveals a large-scale asymmetry of the stellar envelope: one side is relatively cool while the opposite side is relatively warm. Movie animations that show us the velocity field (relative temperature changes rather slowly as the gas moves, so that animating a sequence of images like this gives an excellent perception of the velocity field) reveal that the gas of the envelope is streaming past the hot central core , striking it from the right in this image and flowing around and away from it on the left. The gas is heated by its journey past the core, as is immediately evident by the flame-like structure to the left of the core in the image.

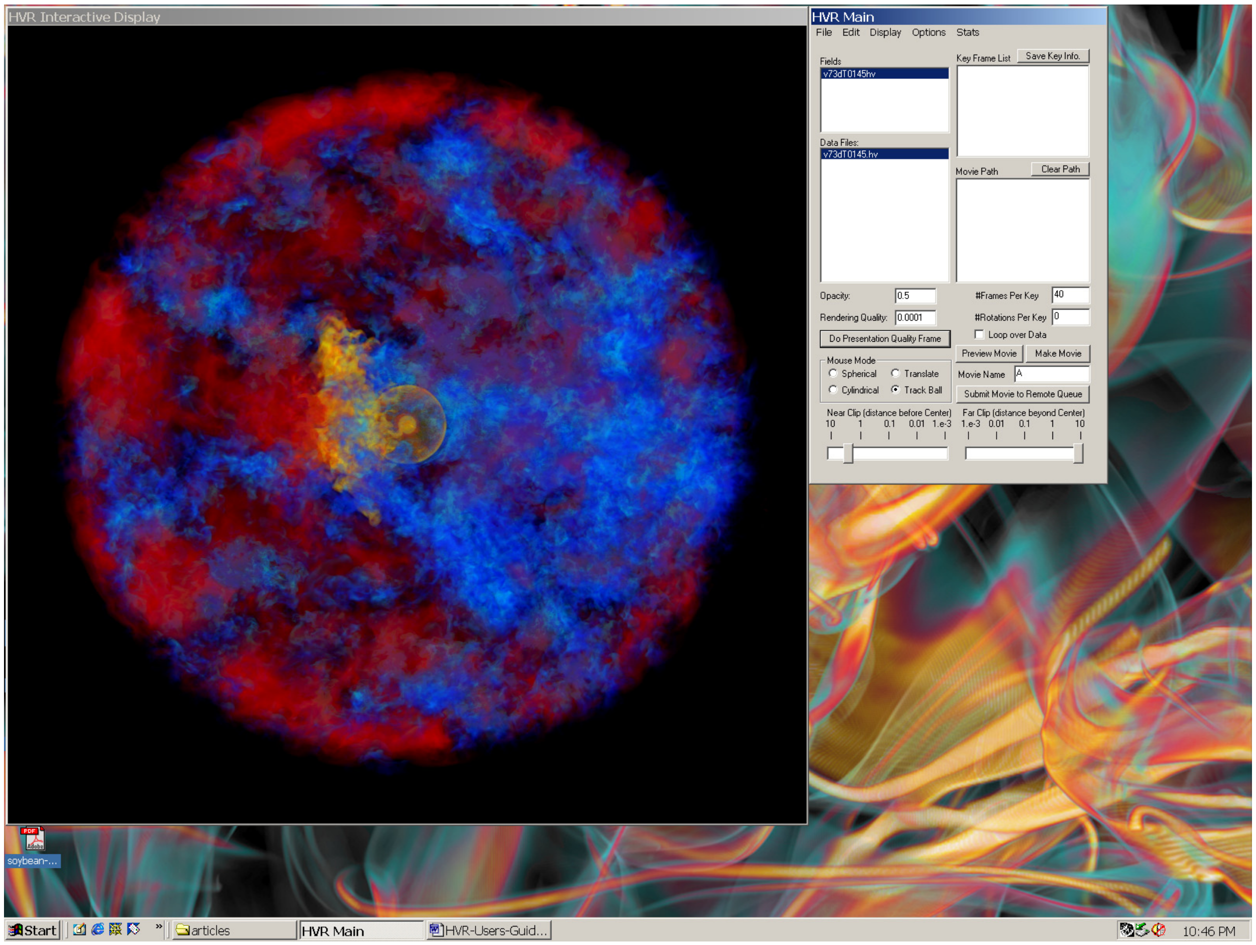

Edit Color Alpha LUT
Color (RGB) : Left Click to Choose or Move Knot; Right Click to Change Color
Opacity (Alpha) : Left Click to Choose or Move Knot; Right Click to Change Value
Delete Knot
Load
Save
OK

Index of Chosen Knot 78

Changing the View Point:

A moment ago I said that I had rotated the star about a vertical axis in order to make the image on the next page. I did this because the default view for this data had us looking at the star so that only the warm side was facing us, and the spherical shape of the core itself was obscured by the hot, flame-like tongues of gas rising from near its surface. But how did I turn the star so that we could look at it from a different angle?

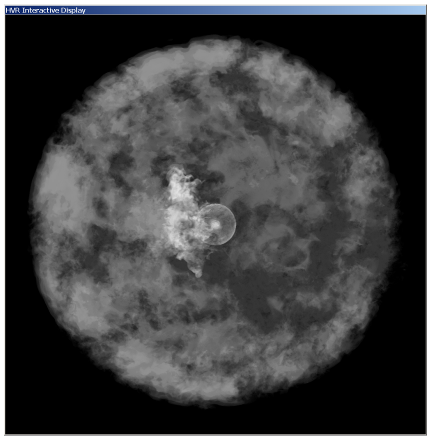
On the main form there is a section labeled "Mouse Mode" and which contains four buttons only



Fields: v73dT0145hv
Data Files: v73dT0145.hv
Key Frame List
Movie Path
Opacity: 0.5
Rendering Quality: 0.0001
#Frames Per Key: 40
#Rotations Per Key: 0
[ ] Loop over Data
Mouse Mode: Spherical, Translate, Cylindrical, [✓ Track Ball]
Movie Name: A
Near Clip (distance before Center): 10, 1, 0.1, 0.01, 1.e-3
Far Clip (distance beyond Center): 1.e-3, 0.01, 0.1, 1, 10

one of which may be selected at a time. By default, the "Trackball" button is selected. In this mode, mouse movements cause the problem domain, in this case the entire star and the cubical shaped bit of empty space included in the domain with it, to move. All such movements in this trackball mode are constrained in the following way. The center of the problem domain, which is the center of our star, must move along a line extending from the viewer directly back into the center of the viewing window. To move the star forward and back along this line, hold down the right mouse button and drag the mouse downward (to move the star away) or upward (to move the star toward you). In this case, I chose to move the star closer until it just filled the viewing window, but I could easily have moved it closer, so that I would actually be inside it. On the next page, a close-up image of the region near the central core is shown that was made by going inside the star in this fashion.

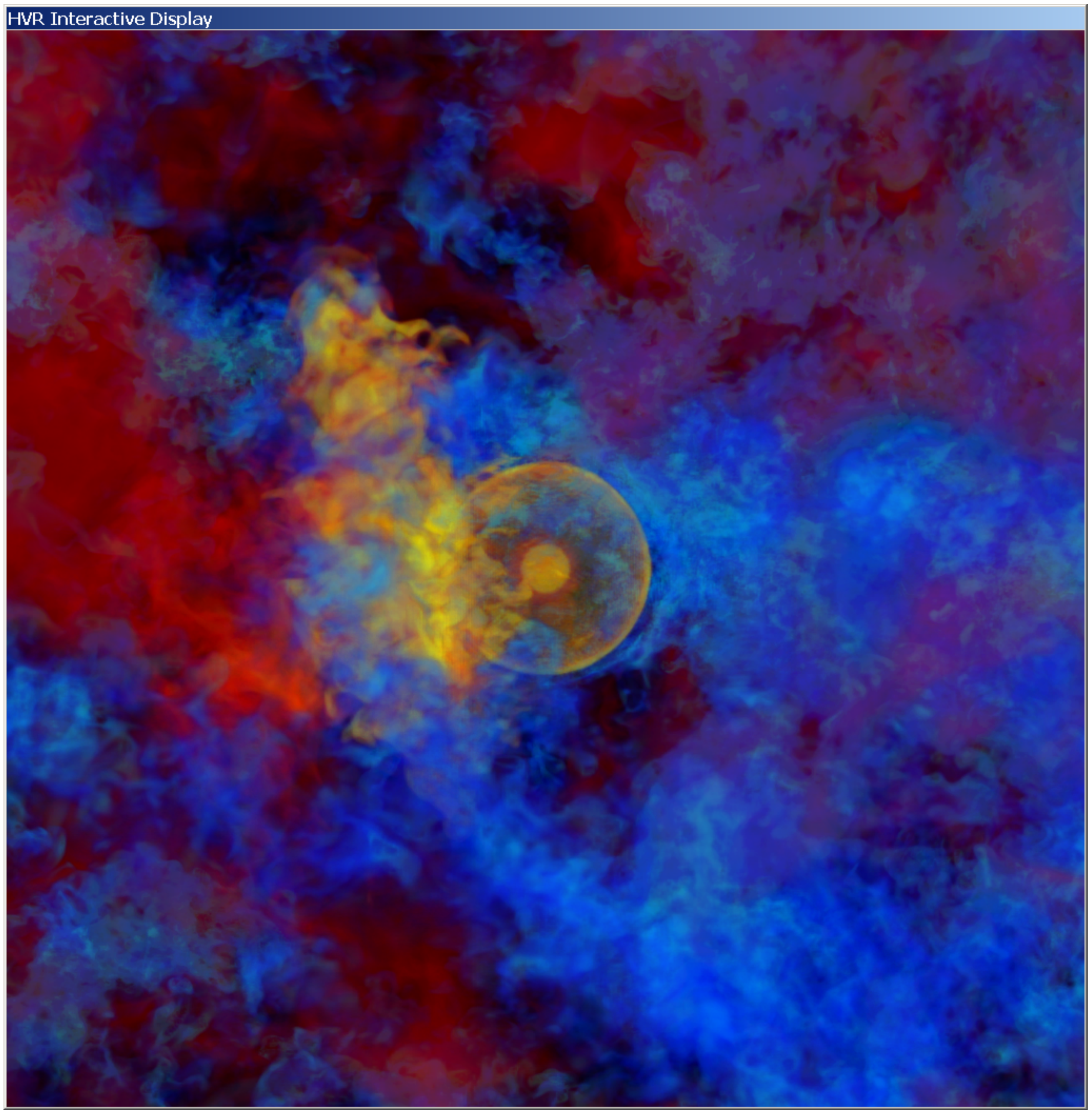
I also rotated the star, so that I could see the stream of cool gas over the core from the side. To rotate the problem domain (and the star in this case) around any axis passing through its center, just hold down the left mouse button and drag the mouse in the direction that you want to be that of an equatorial rotation. Just try this a bit, and you will get the hang of it very quickly.



Setting the Clipping Planes:

Before we go on to discuss the other mouse modes, let's talk about the two sliders at the bottom of the main form. These are confusing, because the order in which you use them is important. And of course that feature is not apparent from just looking carefully at the form. But it is an intended feature, and therefore is unlikely to be "fixed." Learn to love it.

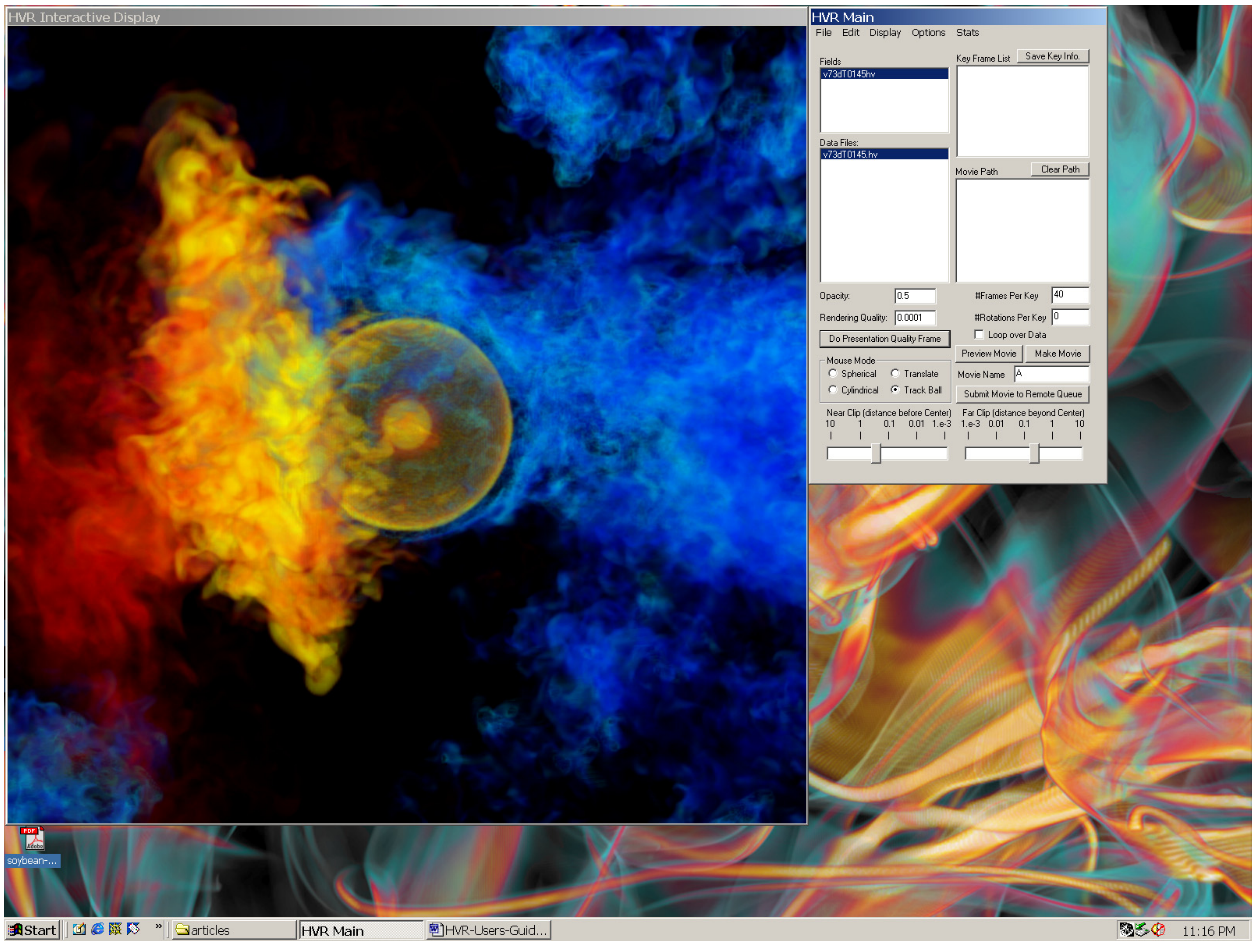
The slider that you want to manipulate first is the one on the right. Never forget this. You will not be reminded again. What's special about



the slider on the right is that when you move it, the slider on the left moves too, in a precisely symmetrical fashion. The slider on the right allows you to move both the near and the far clipping planes in tandem so that they make invisible all but a thinner or thicker slice of the problem domain that is centered on the center of the domain.

The image on this page was made by simply moving in toward the center of the star and rendering the view from there. If this were one frame from a movie animation, we would get good depth perception by seeing the clouds of gas move past each other, either in front or behind. However, as a still image it is a bit confusing. We see too much stuff in this single view, and we do not have enough information in the form of visual cues to figure out what is located where. To get a clearer view, we need to clip away some of the confusing foreground and background material. This is easy. We just move the right slider bar at the bottom of the main form to the left. The result is the much more understandable view on the next page. In this view there is no question that a distinct stream of cooler gas is passing over and around the central core, and being heated in that process.

Note that the labels along the slider bars indicate that the placement of the slider really determines the logarithm of the slice thickness, in units in which a slice of thickness unity just includes the entire problem domain.



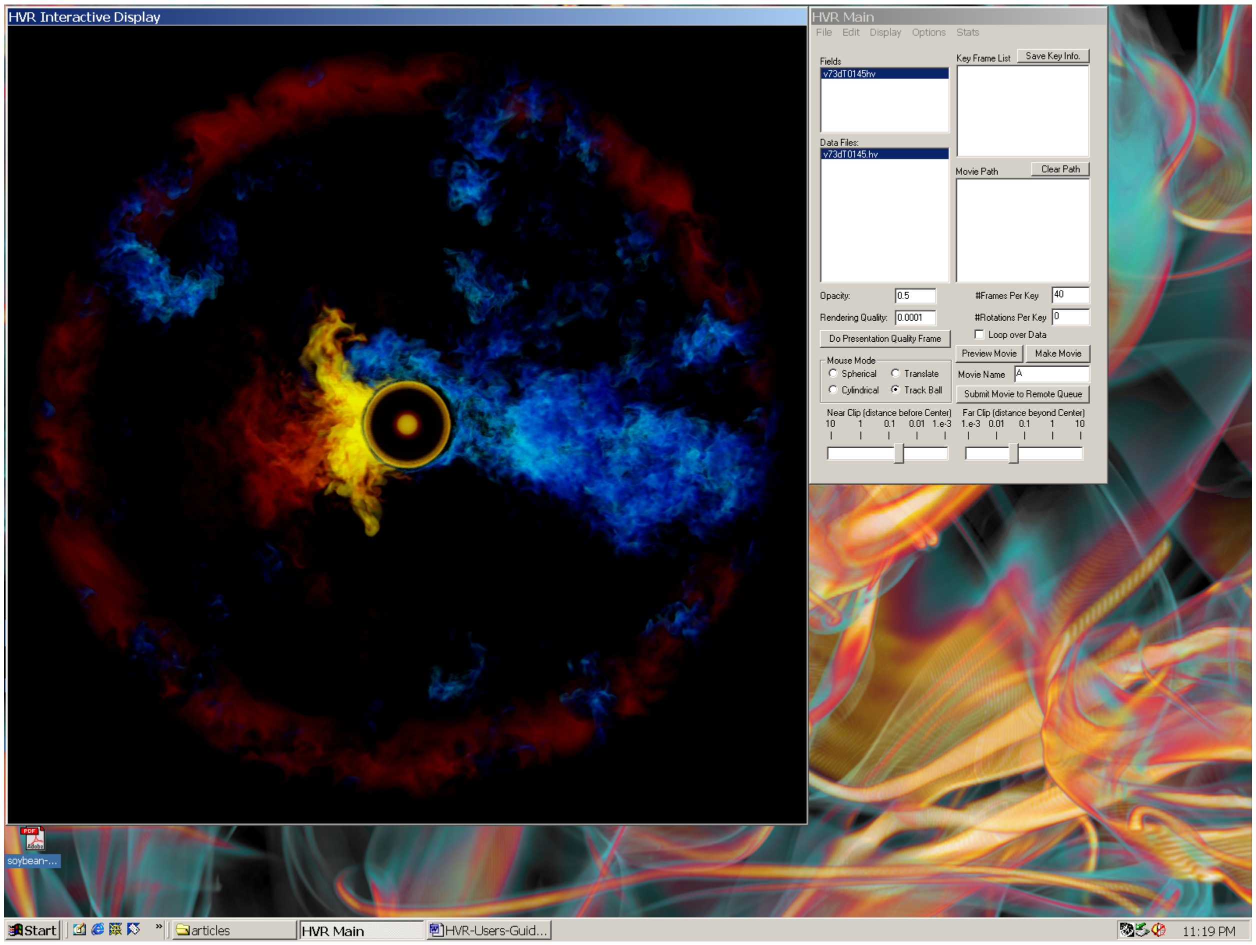
Fields: v73dT0145hv
Data Files: v73dT0145.hv
Opacity: 0.5
Rendering Quality: 0.0001
Mouse Mode: Spherical, Cylindrical, Translate, [✓ Track Ball]
#Frames Per Key: 40
#Rotations Per Key: 0
Loop over Data: ☐
Movie Name: A
Near Clip (distance before Center): 10 1 0.1 0.01 1.e-3
Far Clip (distance beyond Center): 1.e-3 0.01 0.1 1 10

This feature affords us very fine control of the slice thickness as it draws in very close to the center of the problem domain. We can adjust this slider so accurately that we can come up with a slice that is just one grid cell thick. Of course, it we make the slice very thin, we may have to readjust the opacity map so that this very thin slice is easily visible. A thin slice of the whole star is shown at the top of the next page. Here it is clear, especially since we have first rotated the star so that this slice goes through the right part of it, that the stream past the core is no local phenomenon but instead part of a dipolar convection flow that is global in scale.

The clipping planes controlled by the sliders at the bottom of the main form are always oriented perpendicular to the line of sight. This means that if we use the mouse to rotate the star, the clipping planes will not rotate with it. Instead, different sections of the star will pass through the thin slice of space in which things become visible. One can animate a sequence like that, but it is usually quite confusing. This is such a weird thing to do that the movie viewer is usually unable to imagine what is going on. This is just rather unnatural. In the real world, once you have sliced away part of an object, it stays sliced away. You can rotate the slice of the object if you want, and view it from an angle, but you can't rotate the slicing planes through the object. Things like that just don't ever happen.

To avoid the confusion just discussed, HVR provides the capability to designate clipping planes that move along with the problem domain (in this case, they move along with the star). This kind of clipping plane cannot be oriented arbitrarily, like the other sort we have been discussing. These clipping planes that move with the object must lie along the grid planes of the data set. This limitation is usually helpful. Because we will view the slices we create in this way from all sorts of angles, it is useful in interpreting the resulting images to know that the problem domain has been sliced along its grid planes.

To designate object-oriented clipping planes, to borrow a computer science expression, select the item "Subregion" from the "Edit" menu. This will cause the dialog box to pop up that is shown at the bottom of the next page. This is a wonderful dialog box that lets you set the clipping planes in three complementary ways. If you simply want to take the clipping planes that you have set using the sliders on the main form and make these, or the



Fields: v73dT0145hv

Key Frame List

Data Files: v73dT0145.hv

Movie Path

Opacity: 0.5

#Frames Per Key: 40

Rendering Quality: 0.0001

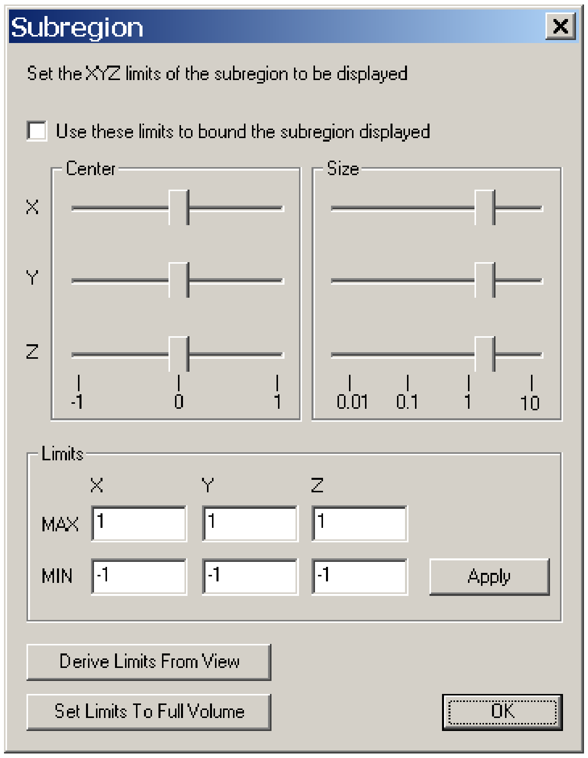
#Rotations Per Key: 0

☐ Loop over Data

Mouse Mode: ○ Spherical ○ Translate ○ Cylindrical [✓ Track Ball]

Movie Name: A

planes closest to them that are aligned along grid planes, into object-oriented clipping planes, simply click on the button at the bottom which is labeled "Derive Limits From View." This might actually work. Otherwise, moving our attention upward on the form, you can type in numbers for the maximum and minimum values of the grid coordinates x, y, and z that you want to make visible. Once again, the value unity here refers to the boundary of the problem domain. Or if you like sliders, you can manipulate the sliders for x, y, and z on this form. Here you get a slider for the slice thickness, with logarithmic scaling to give you fine adjustment capabilities for thin slices, and you also have a separate slider to adjust the position of the center of your slice. What more could you desire?



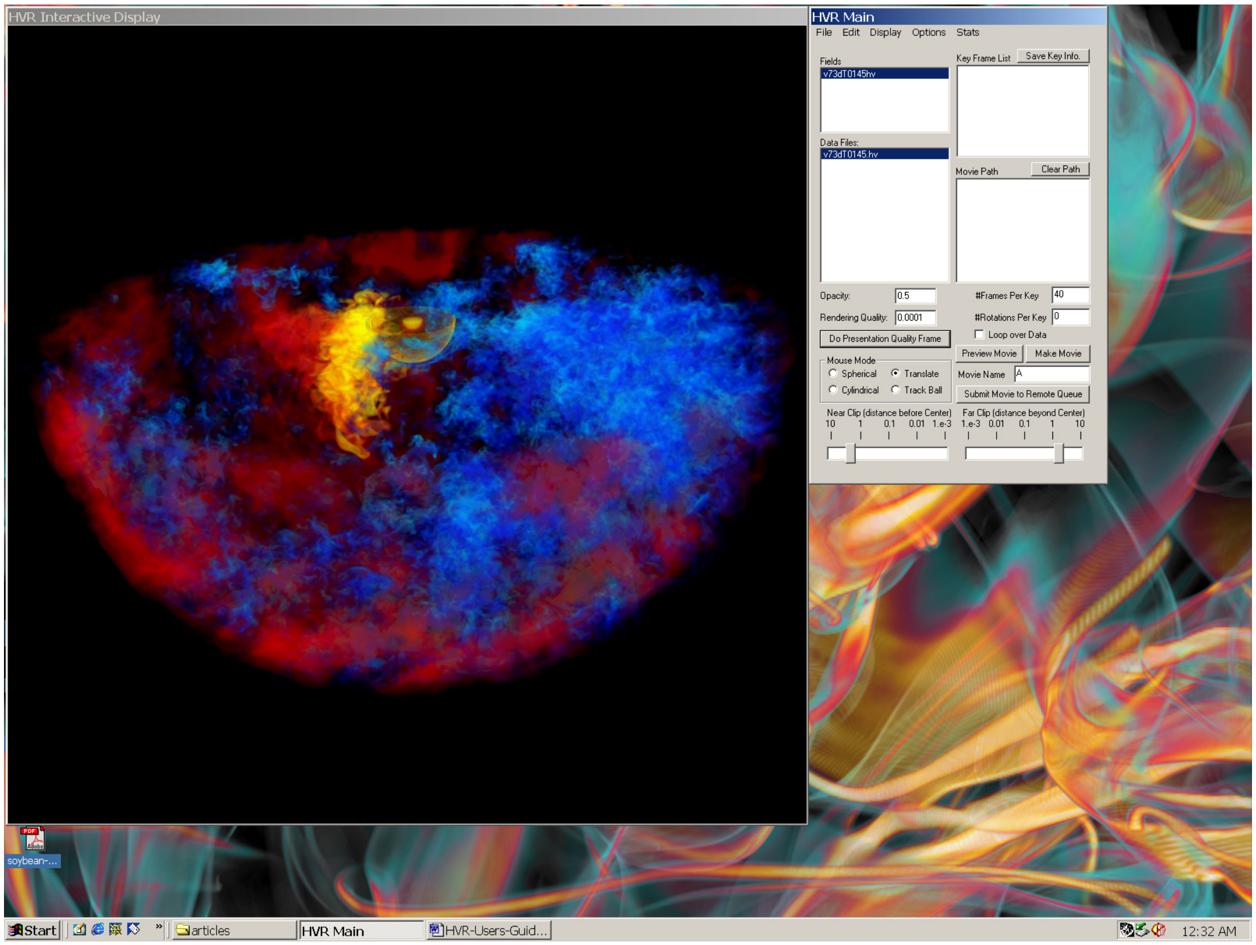
Set the XYZ limits of the subregion to be displayed

☐ Use these limits to bound the subregion displayed

Limits

	X	Y	Z
MAX	1	1	1
MIN	-1	-1	-1

To illustrate how this feature can be used, I show on the next page a rendering of the star with half of it sliced away. This kind of a view would be useless unless we could look at the cut-away from an angle, as in this image. This would be impossible to do with clipping planes that are always oriented perpendicular to our line of sight. Just to keep things interesting and confusing, you can combine the effects of both sorts of clipping planes in any way you wish.



Fields: v73dT0145hv

Data Files: v73dT0145.hv

Key Frame List

Movie Path

Opacity: 0.5

Rendering Quality: 0.0001

#Frames Per Key: 40

#Rotations Per Key: 0

[ ] Loop over Data

Movie Name: A

Mouse Mode: ( ) Spherical [✓ Translate] ( ) Cylindrical ( ) Track Ball

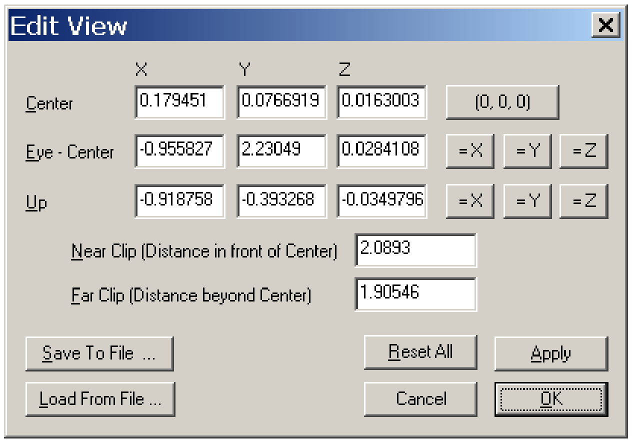
Near Clip (distance before Center): 10, 1, 0.1, 0.01, 1.e-3

Far Clip (distance beyond Center): 1.e-3, 0.01, 0.1, 1, 10

To obtain the view that I show on this page, I had to raise the visible hemisphere of the star up in the viewing window, so that it would be more or less centered within it. I did this by selecting the "Translate" button in the "Mouse Mode" section of the main form, and then I simply dragged the star upward in the window. The translate mouse mode is very useful when you are dealing with sliced domains or when you are making extreme close-up views from within large domains.

Fine Adjustment of the View Parameters:

While we are discussing dialog boxes with lots of places for you to type in numbers, let's talk about the "view." HVR allows you to establish fine control over the parameters that define your view of the problem domain. You can type in precise numbers in the "Edit View" dialog box, shown here for the view that is displayed at the top of this page. You can save the view parameters to a disk file by clicking on the "Save To File . . ." button on this form, and you can load the view parameters from such a disk file by clicking on the "Load From File . . ." button located just below it. You don't have to type any numbers, of course, since these values are automatically set by HVR as a result of all the manipulations we have been discussing. So you can simply save them in order to avoid ever having to go through all those manipulations again. This is extremely useful when you want to create identical views of several different variables or of data from several different simulations for a precise visual comparison.



	X	Y	Z
Center	0.179451	0.0766919	0.0163003
Eye - Center	-0.955827	2.23049	0.0284108
Up	-0.918758	-0.393268	-0.0349796

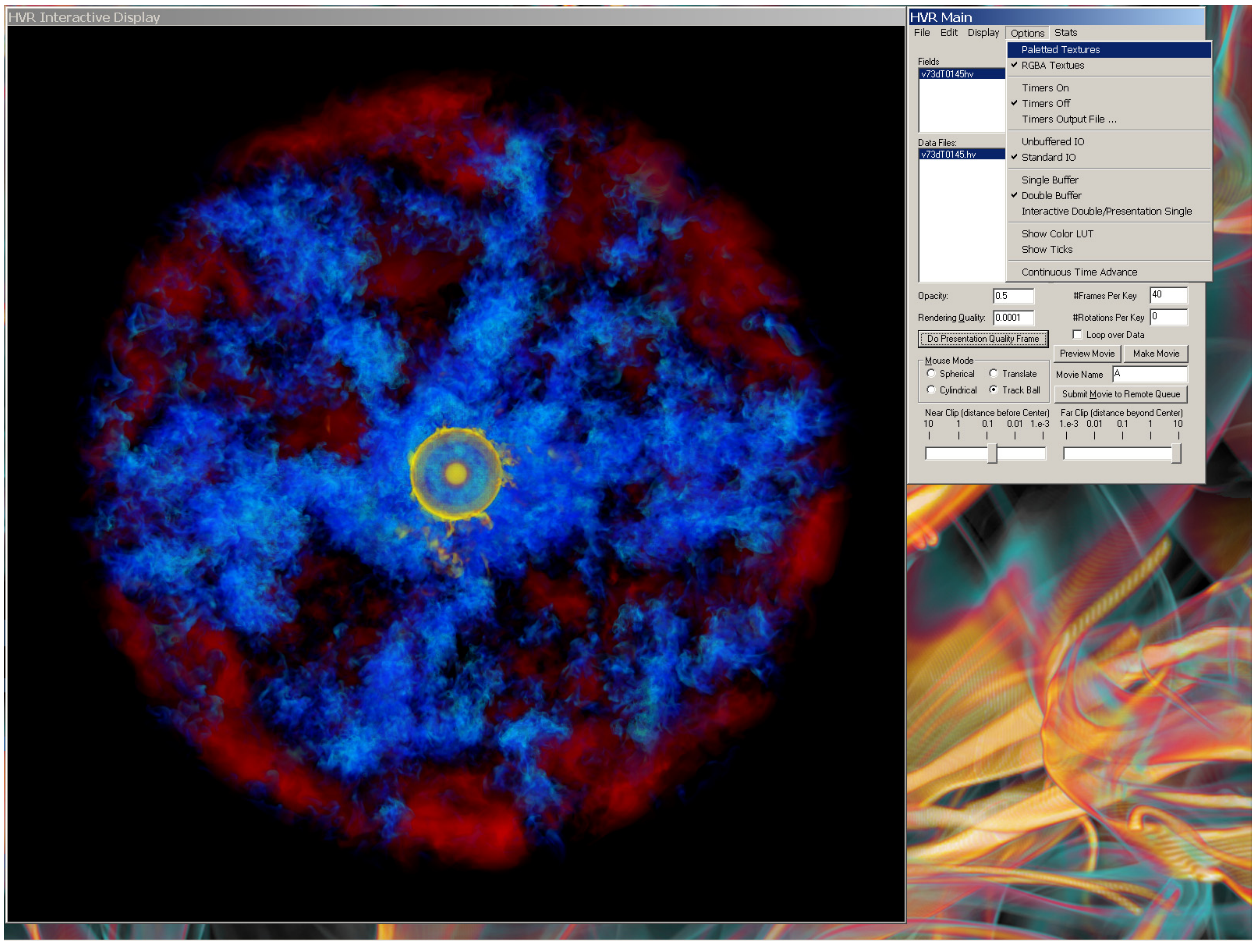
Near Clip (Distance in front of Center): 2.0893

Far Clip (Distance beyond Center): 1.90546

I am tempted to skip telling you what all these numbers mean. You are smart. You can guess. Just remember that the problem domain runs from -1 to 1 in x by definition (and the other dimensions are scaled to match). The first line of x, y, and z values is the position of the center of the field of view. This is not the position of the center of the problem domain. Instead, it defines a sort of "look at" point. This is the point about which rotations will occur when the mouse mode is set to trackball mode. It is also the point that will lie at the center of any thin slice you make by setting the near and far clipping planes very close together by manipulating the right-hand slider bar on the main form. The coordinates of the eye center come next. This is the position that you are looking from. These two points now define a line, which is the line of sight to the center of the field of view. It only remains to specify which way is up, and the viewing parameters, except for the clipping plane positions, given further down on the form, are complete. The coordinates for "up" are actually the components of a unit vector pointing up.

__Making a Movie:__

Now that we have discussed the many features of HVR that let you create any still view of your data that you might possibly wish, let's make a movie. But first, and this is really important, you should make sure that you have selected the proper rendering options. Our first movie will be made right on your own machine, so these options will apply to that and not to the equipment at the LCSE. In the picture of my screen below, I have gone to an 1150^2 viewing window, so it looks really big, and have taken the initial view of the star, which is looking directly into the upwelling warmer gas, and have clipped off the front hemisphere by moving the near clipping plane in toward the middle (see the position of the left-hand slider on the main form). This shows us the cooler hemisphere, viewed with all the plumes of cool gas plunging inward toward the core and, near the core, toward us. This entire screen capture was the only way I could show you the "Options" menu, and I apologize that the view of the star hogs the screen. The text in the menu options still looks readable on my screen, so if you can't make it out, perhaps you need reading glasses. I have shown all the default options checked, and I am in the process in this view of selecting the "Paletted Textures" option. This is important. So, what are paletted textures? A texture, and we will consider a 2-D texture for simplicity, is a bitmap that can be mapped by the graphics hardware of your PC onto any



Fields: v73dT0145hv

Data Files: v73dT0145.hv

Opacity: 0.5

Rendering Quality: 0.0001

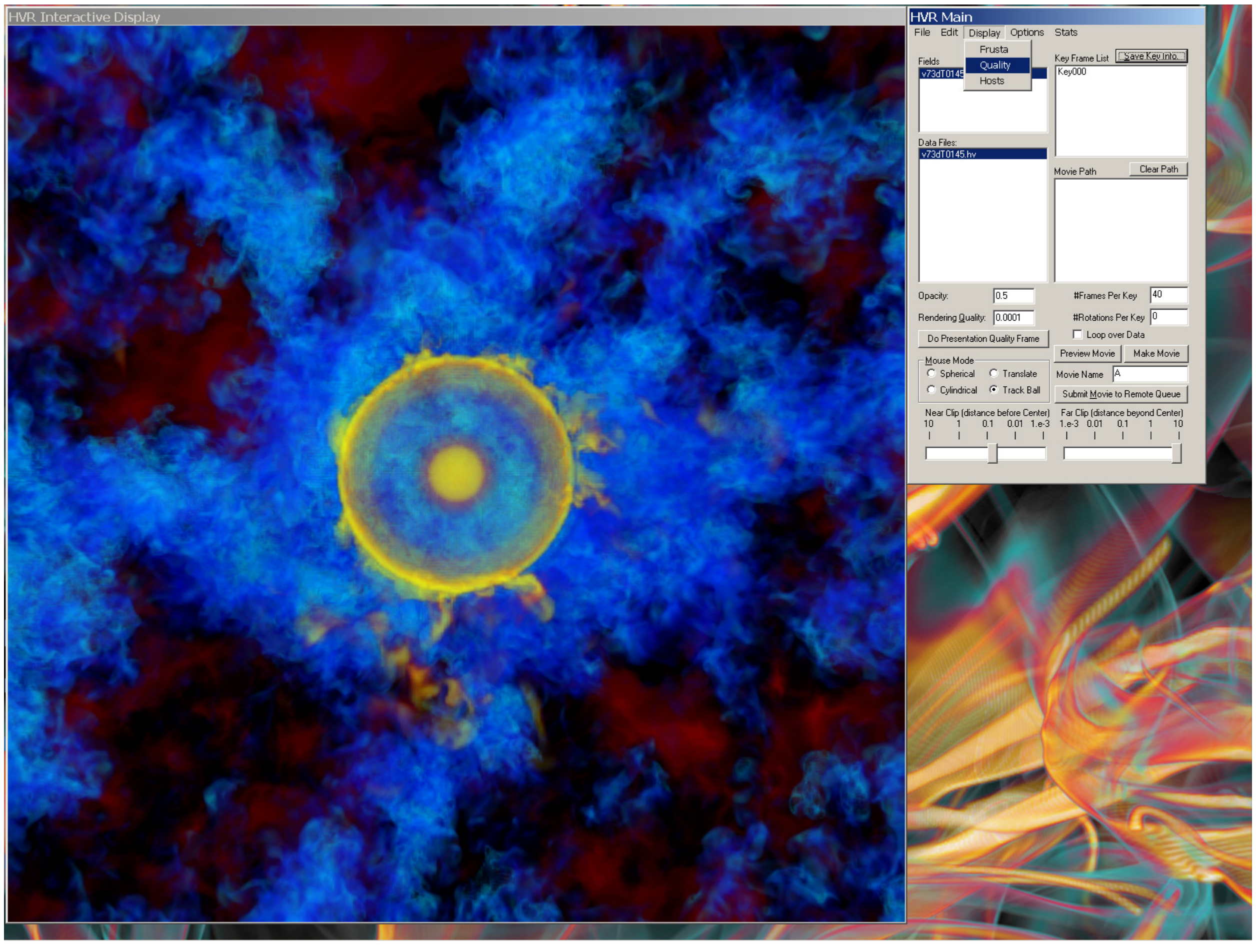
#Frames Per Key 40

#Rotations Per Key 0

☐ Loop over Data

Mouse Mode: Spherical, Translate, Cylindrical, [✓ Track Ball]

Movie Name A



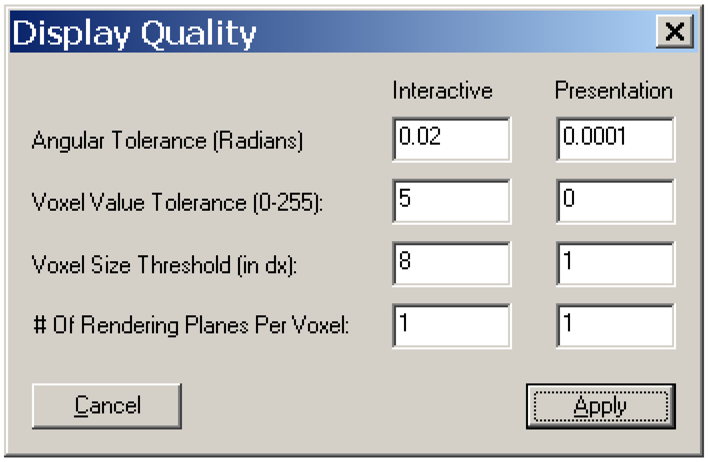
Fields: v73dT0145
Key Frame List: Key000
Data Files: v73dT0145.hv
Movie Path
Opacity: 0.5
Rendering Quality: 0.0001
#Frames Per Key: 40
#Rotations Per Key: 0
[ ] Loop over Data
Mouse Mode: ○ Spherical ○ Translate ○ Cylindrical [✓ Track Ball]
Movie Name: A
Near Clip (distance before Center): 10 1 0.1 0.01 1.e-3
Far Clip (distance beyond Center): 1.e-3 0.01 0.1 1 10

desired surface. A standard RGBA texture has 4 bytes, one each for red, green, blue, and opacity (alpha), at each texel. A paletted texture has only one byte, for a color given in a color look-up table (LUT), at each texel. If moving all this data around inside your machine is limiting the rendering speed, it seems pretty clear that paletted textures will be better. Since my laptop PC supports paletted textures, I am selecting this option before making a movie, because rendering speed will really begin to matter once I set the machine off to rendering several hundred frames. You might wonder why paletted textures are not the default option, since they are so cool. Well, that is because when you select this option and your machine does not support it, experience shows that your machine is likely to crash. Your machine is a PC, so you expect this sort of thing. And it's worth it to find out if you can use this fast rendering mode. Try it – once. But no law suits. You were duly warned.

None of the other options on this menu turn out to be of much use. Try them out if you wish, and you can discover what they do. But there are better uses of your time.

There is another menu that we really must explain before you begin to make a movie. This too is very important. It can save you many hours of time. So pay attention.

In the screen view shown on this page, I have generated another beautiful view of the star, this time from a much closer view point. Before going to this closer view, I clicked on the "Save Key Info" button at the top right on the main form. This saved the view on the previous page as a key frame, so that I can use it to make my movie. In the screen view above, you can see that I have selected the "Quality" item from the "Display" menu. There is the beautiful close-up view in the viewing window which you can appreciate in all its glory. But this view took a little while to generate, at least on my laptop, and that's why I'm showing you the "Quality" dialog box on the next page. No one could ever accuse me of not loving quality, but I have to admit that it has its place. That might be in this document, but it is certainly not in a rough cut of the movie I might or might not want to see. I could waste an enormous amount of time rendering a fabulously detailed movie sequence from this billion-voxel hv-file only to determine upon its first viewing that I really did not, for example, want to get up this close to the stellar core. Or that


Display Quality

	Interactive	Presentation
Angular Tolerance (Radians)	0.02	0.0001
Voxel Value Tolerance (0-255):	5	0
Voxel Size Threshold (in dx):	8	1
# Of Rendering Planes Per Voxel:	1	1

I really would have preferred to have zoomed in four times more slowly and smoothly. Or a host of other possible objections. This is why it is really, really important to set the image quality parameters before you set a movie rendering.

HVR provides all the flexibility you might require to control the quality of the image rendering for your movie. This is accomplished by setting the parameters listed on the "Display Quality" dialog box shown here at the right. As with the other aspects of HVR, this takes a bit of explaining. As we discussed near the beginning of this guide, HVR supports two display modes (and, although we are not using this feature right now, multiple displays). These are the interactive display and the presentation display. We will be generating our first movie right on your own machine, so both of these displays will be the same physical device, the viewing window on your screen. But they will represent two different image rendering modes. The interactive one will be fast and rough, while the presentation one will be slower and better. How fast and rough and how much slower and better is up to you. And your choice will depend upon what you are doing.

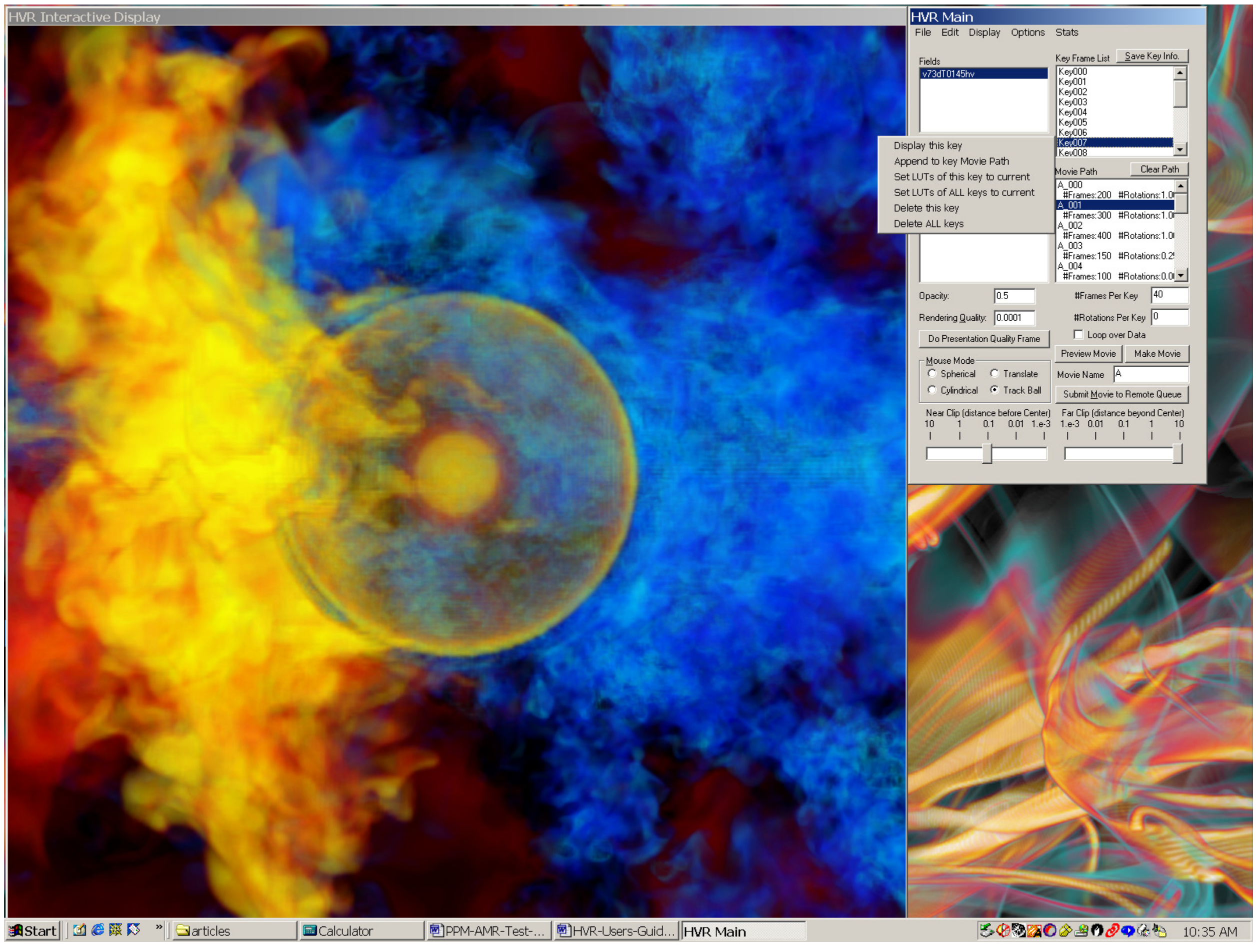
At the moment, we are about to make our first movie. It is incredibly unlikely that this movie will be any good. Therefore, we should set the rendering quality with this understanding in mind. The simplest way to control the rendering quality of your movie is through the setting for the voxel size threshold. This parameter is set in the third row of text boxes on the "Display Quality" form. To understand what these settings mean, we must remember that the data stored in the hv-files is hierarchically organized. At the lowest level, data values in grid cells are stored chunk by chunk. At the next level up, we have data values in coarsened grid cells which each represent $2\times2\times2$ cubes of the original grid cells. Up one additional level we have data values for still coarser grid cells which represent $4\times4\times4$ cubes of the original grid. Note that for the interactive display the default value of the voxel size threshold is given as 8 in the "Display Quality" dialog box. This means that the image renderer will not use grid representations of the hv-file data that involve cells finer than $8\times8\times8$ cubes of original grid cells. Even in a very large data set, there can't be too many of these very coarse grid cells. Therefore images rendered at this quality specification should be produced in a split second. The voxel size threshold of 1 that is prescribed by default for the presentation quality display demands that the image rendering process will use data in its very finest representation within the hv-file. For a large data set, this may take some time. If we simply set this voxel size threshold to the value 2, we should get quite acceptable images for a rough cut of a movie, and we should get them quite fast (perhaps as much as 8 times faster).

The angular tolerance, in radians, which can be specified in the first row of text boxes on the "Display Quality" form, is a related concept to the voxel size threshold. It is a means of specifying how much detail we want our images to contain. The default interactive setting is 0.02. This means that unless a grid cell (a voxel) subtends an angle of at least 0.02 radians when viewed from the eye position, the renderer will go to the next coarser grid for its data. It turns out that this angular tolerance is related to how you set the vertical field of view (in degrees), which was discussed much earlier, on pages 7 and 8. Of course, you set the vertical field of view (or, if you heeded my admonition, you refrained from setting the vertical field of view) in degrees, while this angular tolerance needs to be specified in radians. So you need to know that a radian is about 57°. Therefore the angular tolerance of 0.02 that is set by default corresponds to 57/50 = about one degree. If you left the default vertical field of view, set at 34°, alone, as I encouraged you to do, and if you are using the default, square image window size of 512×512 pixels, then each pixel on your display subtends $34/512=0.066°=0.066/57$ radians = 0.0011 radians. The angular tolerance of 0.02 radians that is now set in the Display Quality window shown on this page therefore corresponds to roughly 20 pixels on your screen. So images rendered from voxels that subtend no less than this angle will be pretty rough. If you have a fast machine, you might want to set this angular tolerance to, say, 0.01 (about 10 pixels on the default image window). HVR looks at both the angular tolerance and the voxel size threshold to determine how far down its voxel brick hierarchy to go in order to render the image at the desired level of detail. If you know what you're doing, and you can work out the math as above, the angular tolerance is the more useful specification of image quality. You can set the voxel size threshold to 1, which will always allow HVR to delve down to the finest level of detail in the data, and then set the angular tolerance to a desired number of pixels in your display window. Then your images will be equally rough when you have the entire data brick in your field of view and when you are rendering a close up view of just a small subregion of the data brick. If you don't want to think about the math, or if you don't have your calculator handy, or if you can't remember how many degrees are in a radian, or you are just lazy, then you can simply set the voxel size threshold.

Setting the interactive display quality parameters will affect the rendering quality and speed for views you create in your image window as you fiddle around trying to find the right key frames for your movie. These settings will also affect the quality and animation speed of the movie preview that you will obtain by clicking on the "Preview Movie" button on the main form. The presentation display quality settings will affect the speed and quality of image rendering when you click on the "Do Presentation Quality Frame" button or on the "Make Movie" button on the main form. The default settings for these rendering parameters are somewhat extreme. They assume that you are trying to make a movie for the LCSE PowerWall. If you are not doing that, and you only want a nice movie to view on your laptop PC or to make into a compressed AVI file that you might put on a Web site, you could waste a lot of time by leaving these settings at their default values. Think about it. Suppose you are rendering from an hv-file representing a cube 1024 voxels on a side and you have left the presentation image size at its default value of 512×512 pixels. With the default field of view of 34^o, each pixel on your screen subtends an angle of 0.0011 radians. But the default angular tolerance for the presentation image is set to 0.0001 radians! This is absurd. HVR will lovingly render gobs of detail that you could not possibly see on your screen, and it will take lots of time, especially if you are making your movie at the last minute before your presentation on battery power while flying to your talk. With a little thought, after bringing up the desk calculator window on your laptop, you can fix this. Going through the math given above, you can determine that you might as well reset your angular tolerance to 0.001 radians. There will be no visible difference in the movie that HVR will generate, but you might get it done before the stewardess forces you to shut off your laptop (on my laptop, experimentation shows that, using this technique, I can get a beautiful movie of 600 images from a billion voxel flow snap shot while flying from Minneapolis to Denver, and my laptop is a year old).

So, what are those other two display quality settings for? They are for experts. I advise you to ignore them. But if you have to know, read this paragraph. The second on the list is the voxel value tolerance. If a coarsened representation of the data (a brick of those little $2\times2\times2$ or $4\times4\times4$ voxel cubes) has been formed by blending individual voxels at a higher resolution that varied in color value by no more than this voxel value tolerance, HVR will not bother using a finer representation of the data, regardless of the value of the angular tolerance or the voxel size threshold. For data that was worth computing in the first place, this display quality setting is usually irrelevant. If there is no color variation in your data, why on earth are you bothering to visualize it? The last display quality setting on the list is something else. It specifies the number of rendering planes per voxel. If you have a lot of voxels, you will find the default setting of 1 quite acceptable. If you don't have very many voxels, you are probably not using HVR anyway, since there are commercial packages out there that you can buy that work fine as long as you have hardly any data. This setting is truly for experts. Be aware. Demanding 2 or, oh gosh, even 4 planes per voxel will force HVR to slice through your voxel brick twice or even 4 times as many times, compositing this increased number of planar images together to form the final image. If you are viewing your data brick from a 45^o angle (the worst case), this will cause the edges of these little image planes to blend together better along the edges of your brick in the image. But watch out. HVR runs on PCs. And in 2002 (or perhaps I had better say in 2001) PC game cards do not support more than 256 color levels per texel. This means that if you have al lot of these little image planes to composite in forming your final image, you could run into rounding errors in the arithmetic involved. The old SGI image rendering engines had 16-bit texels, so you could bump this display quality setting up to 8 planes per voxel (which was definitely overkill) to enjoy the near disappearance of all visual artifacts. Don't try this on a PC, unless, after this writing, the game card manufacturers can think of nothing better to do with the millions of transistors available to them on their graphics processors than to go to 16-bit texels. We at the LCSE have gone through our withdrawal from 16-bit texels over a year ago, and we can attest that there is life, and even pleasure, in a thoroughly 8-bit graphics world. We think you will agree.

Let's get on with it and make a movie. We've selected the paletted textures option, we set our diplay rendering quality parameters, and we've got our frusta all defined. Let's define some key frames and get going. This can be easy, or it can be tedious, depending upon your system, your data, and your mood. I don't want to go through it with you in full detail (when I did this with my class of freshmen, some of them fell asleep). So let's just look at a movie I have already set up. It's definition is seen (sort of) on the screen capture on the next page. In the key frame list on the main form, you can see that I have defined a large number of key frames. I set each one by manipulating the view and setting the color and alpha look up tables as I have described earlier. When I got these just the way I wanted them, I clicked on the button at the upper right on the form, labeled "Save Key Info." Then all these parameters were saved in the key frames file associated with my project. To modify any of them afterward, all I need to do is to select the appropriate key frame on the list. On the screen capture on the next page, you can see that I have selected Key007. Before grabbing this image, I selected the option "Display this key" from the list on



Fields: v73dT0145hv

Key Frame List: Key000, Key001, Key002, Key003, Key004, Key005, Key006, Key007, Key008

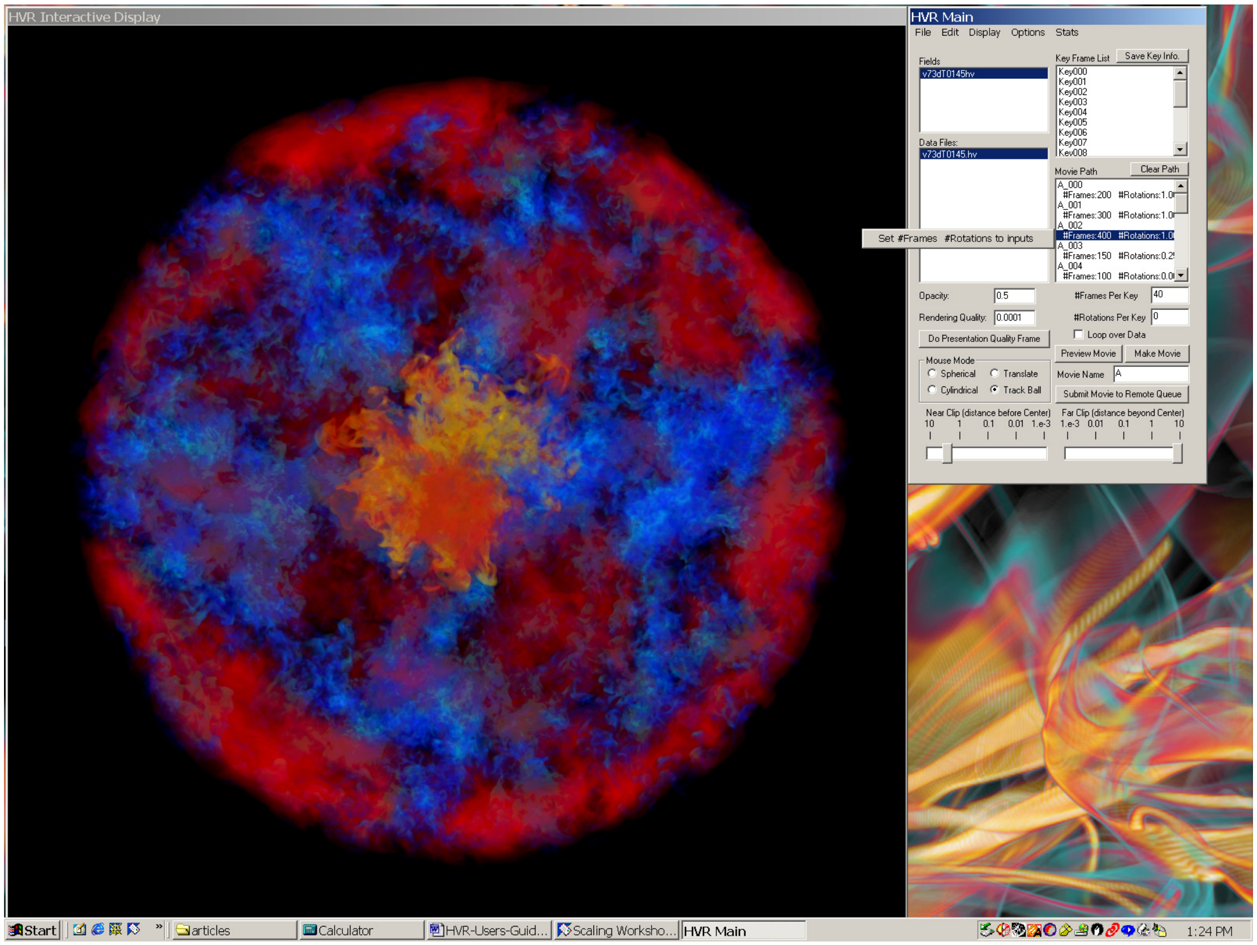
Movie Path:

```
A_000
  #Frames:200  #Rotations:1.0
A_001
  #Frames:300  #Rotations:1.0
A_002
  #Frames:400  #Rotations:1.0
A_003
  #Frames:150  #Rotations:0.2
A_004
  #Frames:100  #Rotations:0.0
```

Opacity: 0.5
Rendering Quality: 0.0001
#Frames Per Key: 40
#Rotations Per Key: 0
Loop over Data: ☐
Mouse Mode: Spherical, Translate, Cylindrical, [✓ Track Ball]
Movie Name: A
Near Clip (distance before Center): 10 1 0.1 0.01 1.e-3
Far Clip (distance beyond Center): 1.e-3 0.01 0.1 1 10

the pop-up menu to the left of the key frame list. That caused HVR to render the key frame, and it appears in the image window. The other options on the pop-up menu are pretty self explanatory. The second option on the list is to "Append to key Movie Path" (which might more understandably have been expressed as "Append this key to Movie Path"). Selecting this option will add this key frame at the end of the list in the "Movie Path" window on the form. I could achieve the same effect by simply double clicking on Key007 in the "Key Frame List" window on the form. We will discuss the movie path in a moment. The next two options on the poop-up menu allow me to modify the color and opacity settings for this key frame, or for all the key frames. By selecting these options, the color and opacity settings that correspond to whatever image is displayed in the image window are set to replace those of this selected or of all key frames.

I can construct a movie path, that is, a sequence of key frames with specified numbers of interpolated frames in between, by double clicking on a sequence of the key frames that I have defined and which are listed in the "Key Frame List" window on the main form. Between each pair of such key frames in my movie path, a number of intermediate frames will be interpolated. At present, a simple linear interpolation is performed, so that the viewing parameters and the color and opacity look-up tables change smoothly from the one frame to the other. The number of such intermediate frames is set in the "# Frames Per Key" text box, located just below the movie path window on the main form. Below this text box is the "# Rotations Per Key" text box, which allows us to specify that the viewing position will rotate around the "look-at" point a given number of times in going from the one key frame to the next. This rotation option is often used when the two key frames are identical (i.e. the same entry on the list in the key frames window), but this is not a requirement. Also, as the movie in this example illustrates, the number of specified rotations can be a fraction. Between key frames A_003 and A_004 (the dumb movie name "A" is the default, which I could have changed by typing a reasonable name into the "Movie Name" text box on the main form), I have specified that a quarter rotation should occur. This quarter rotation will proceed at a somewhat slower apparent speed than the single rotation before it, since for that whole rotation I specified 400 frames, while for the quarter rotation I specified 150. To make this requested quarter rotation work, I can produce the second key frame, A_004, from the first by carefully dragging the star around by 90° using the "trackball" mouse mode. If I get the



Fields: v73dT0145hv
Data Files: v73dT0145.hv

Key Frame List: Key000, Key001, Key002, Key003, Key004, Key005, Key006, Key007, Key008

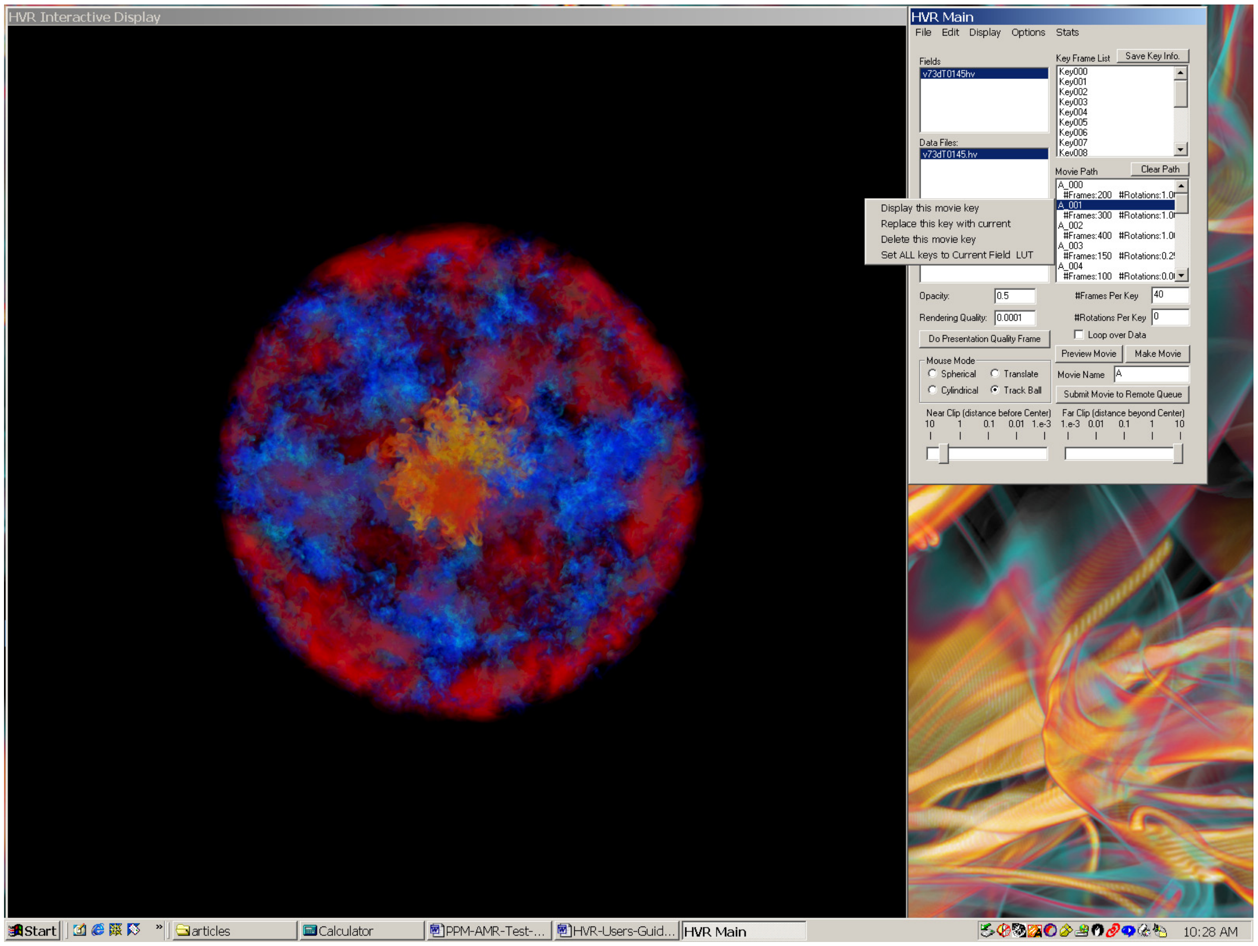
Movie Path:

```
A_000
  #Frames:200   #Rotations:1.0
A_001
  #Frames:300   #Rotations:1.0
A_002
  #Frames:400   #Rotations:1.0
A_003
  #Frames:150   #Rotations:0.2
A_004
  #Frames:100   #Rotations:0.0
```

Opacity: 0.5
Rendering Quality: 0.0001
#Frames Per Key: 40
#Rotations Per Key: 0
☐ Loop over Data
Mouse Mode: Spherical Translate Cylindrical [✓ Track Ball]
Movie Name: A
Near Clip (distance before Center): 10 1 0.1 0.01 1.e-3
Far Clip (distance beyond Center): 1.e-3 0.01 0.1 1 10

angle slightly wrong, HVR will understand and still interpolate intermediate frames along a circular arc. However, a safer mode for making this specification is to bring up the detailed numerical view parameters for the first key frame, A_003, and to type in new ones that correspond to a precise 90° rotation in the desired direction.

If for any reason I want to change the specification of the number of intermediate frames and/or rotations, I can select the appropriate interval in my movie path, as is shown above, and by selecting the single list item that pops up, "Set # Frames # Rotations to inputs," I can reset this information to whatever values I have previously entered into the "# Frames Per Key" and "# Rotations Per Key" text windows on the main form. There is a further refinement in the movie example here (you can view the movie by downloading it from the LCSE Web site at www.lcse.umn.edu/hvr), since the key frames at the beginning of the movie correspond to increasingly close up views. This means that the interpolated path of the viewing position will actually spiral in toward the satr as the movie progresses. HVR handles this with no problem. Later in this movie, the viewing position is held fixed between two key frames, while the clipping planes are moved so that the foreground material of the stellar convective envelope is clipped away to an ever greater degree. HVR interpolates the clipping plane position in the intermediate frames, so that this transition is smooth in the finished movie. There are also segments in which only the color and opacity look-up tables are changed between key frames. HVR generates interpolated look-up tables for the intermediate frames. These interpolations are performed by interpolating the positions and values associated with the "knots," or special, user-specified values that define the look-up tables. This fact means that the two look-up tables used for the two key frames had better have the same number of such knots. If they don't, results are unpredictable, and the LCSE will not be held responsible for the consequences. You can prevent such problems easily by inserting an additional key frame, with no interpolated intermediate frames, so that the discontinuous change from, say, 4 knots to, say, 6 is accomplished between two key frames without any intermediate ones. Then you can have gradual transitions between two 4-knot look-up tables or between two 6-knot tables without fear of bizarre color changes. It is often useful that you can save your look-up tables in disk files by simply choosing the "Color/Alpha LUT" item on the "Edit" menu of the main form and then clicking on the "Save" button on the pop-up window. You can reload such a saved look-up table from the disk file by clicking on the "Load" button on this "Edit Color Alpha LUT" window. You can also save and reload viewing parameters in a similar fashion by selecting the "View" item from the "Edit" menu.



Fields: v73dT0145hv

Data Files: v73dT0145.hv

Key Frame List: Key000, Key001, Key002, Key003, Key004, Key005, Key006, Key007, Key008

Movie Path:
```
A_000
  #Frames:200  #Rotations:1.0
A_001
  #Frames:300  #Rotations:1.0
A_002
  #Frames:400  #Rotations:1.0
A_003
  #Frames:150  #Rotations:0.2
A_004
  #Frames:100  #Rotations:0.0
```

Opacity: 0.5

Rendering Quality: 0.0001

#Frames Per Key: 40

#Rotations Per Key: 0

Loop over Data

Mouse Mode: Spherical Translate Cylindrical [✓ Track Ball]

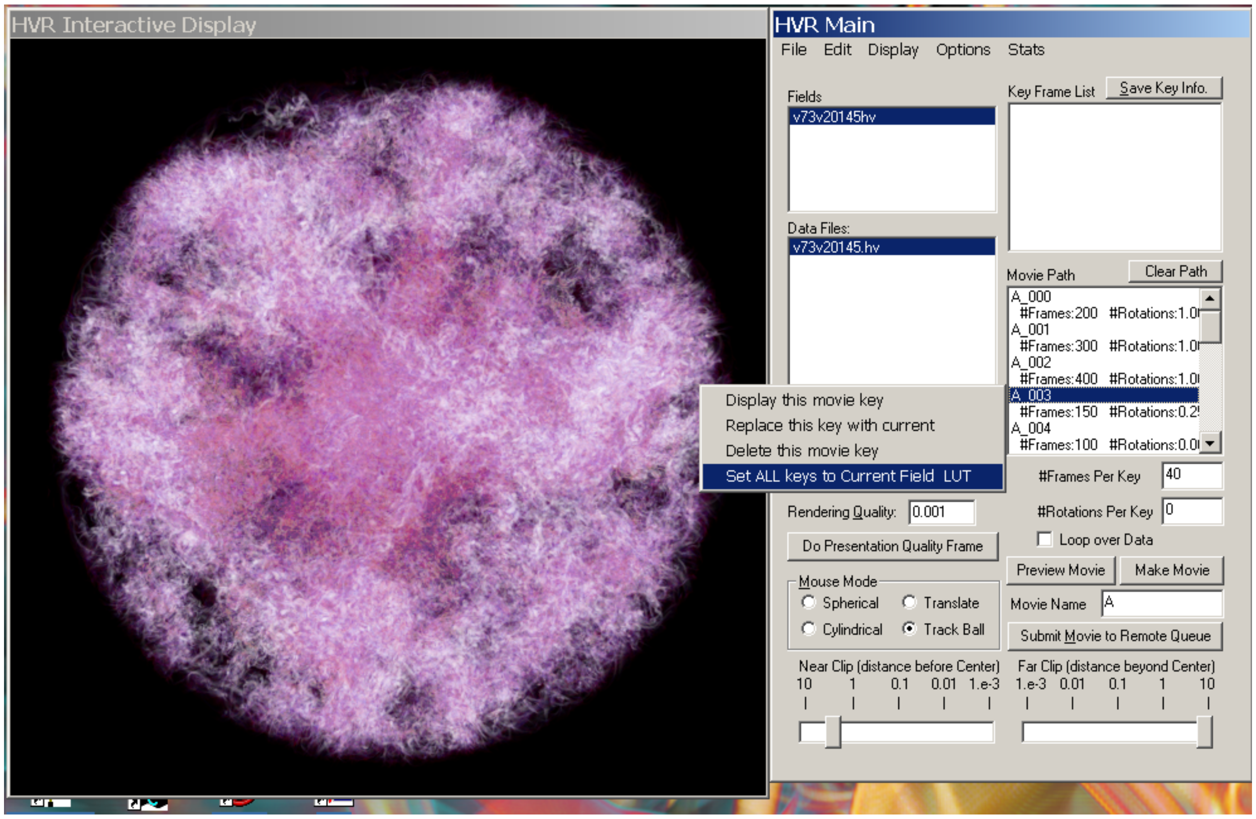
Movie Name: A

Near Clip (distance before Center): 10 1 0.1 0.01 1.e-3

Far Clip (distance beyond Center): 1.e-3 0.01 0.1 1 10

Your movie path and your key frames (with their associated viewing parameters, clipping plane locations, and color and opacity look-up tables) are saved for you with your project. You do not need to save separate disk files describing particular views or look-up tables unless you want to do that. You can also edit your movie path. One way to modify the path is to change the number of frames and/or rotations between a pair of key frames, as we have discussed above. Another way is to redefine key frames in the path. This can be done by selecting a key frame you want to change, and then selecting one of the options that pop up on the menu as shown above. These options allow you to display the key frame, after which you might modify some of its parameters, and then you can reset it to the new frame that you would then have in the image window by selecting the key frame once more and then selecting the pop-up option "Replace this key with current." You may also simply delete this key frame. The final option on the pop-up menu is "Set ALL keys to Current Field LUT" (it should really say "Field & LUT"). This takes some explaining. It is intended to accommodate the situation in which you would like to use the same movie path to view different variables (which David Porter, the author of HVR, thinks of as fields). You might, as we have done in this example, set up a movie path to display the relative temperature differences at each height in the stellar envelope. But then it would be nice to be able to generate a matching movie, with an identical number of frames from identical views, to look at, for example, the vorticity. If you could animate both of these movies side by side and in a synchronized fashion (and of course you can, using the LCSE movie display software), then you would be able to determine visually to what extent the temperature and vorticity disturbances are correlated. This can be done using HVR, although it is not as easy as it could be.

There are several quite different ways in which you can use HVR to create movies of multiple variables all sharing the same path. In this context, we view the movie path as a path of the viewer through space time. This path does not only include the viewer's position, but also the direction of the view and the "up vector." To create compatible views, we also want to keep the same clipping plane positions. The idea is that we look at each variable in the same way, focusing on the same portion of the object, but we will generally not use the same color and opacity tables for different variables. You might think that the easiest way to use the path we determined while working with one variable for a movie of another variable would be to save the movie path in a .path file by


HVR Interactive Display
HVR Main
File Edit Display Options Stats
Save Key Info.
Clear Path
Display this movie key
Replace this key with current
Delete this movie key
Set ALL keys to Current Field LUT
Do Presentation Quality Frame
Preview Movie
Make Movie
Submit Movie to Remote Queue

Fields: v73v20145hv

Data Files: v73v20145.hv

Key Frame List

Movie Path

```
A_000
 #Frames:200  #Rotations:1.0
A_001
 #Frames:300  #Rotations:1.0
A_002
 #Frames:400  #Rotations:1.0
A_003
 #Frames:150  #Rotations:0.2
A_004
 #Frames:100  #Rotations:0.0
```

#Frames Per Key: 40

Rendering Quality: 0.001

#Rotations Per Key: 0

Loop over Data [ ]

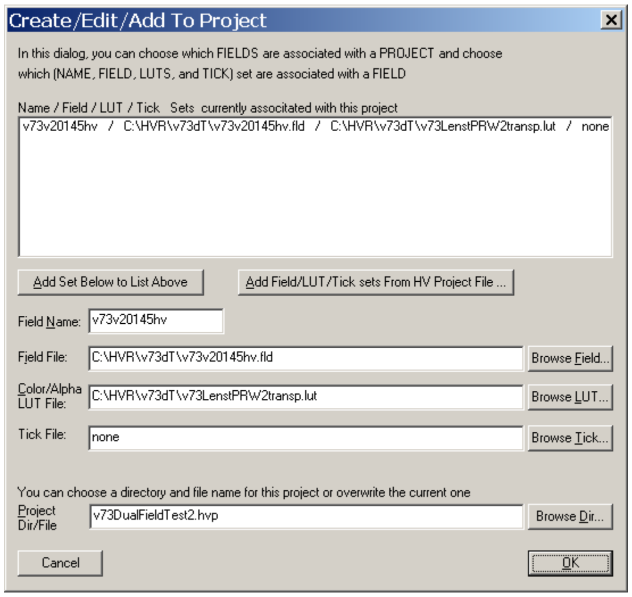
Mouse Mode: Spherical ○ Translate ○ Cylindrical ○ [✓ Track Ball]

Movie Name: A

Near Clip (distance before Center): 10 1 0.1 0.01 1.e-3

Far Clip (distance beyond Center): 1.e-3 0.01 0.1 1 10

choosing the option "Save Movie Path . . ." on the "File" menu of the main HVR form after we are done working with the first variable, and then loading this path into a second project we have subsequently set up to visualize the hv-files of a second variable. This is precisely what we have done before making the screen capture shown above. The trouble with this fairly direct approach is that the movie path contains references to the specific hv-files and color tables in addition to the simple viewing parameters. The key frames in the movie path point to the wrong set of hv-files. If we select any of these keys, as shown above, and then select the option "Display this movie key" at the top of the pop-up menu, we will get an image of the relative temperature fluctuations, not of the vorticity (actually, in this case, the vorticity squared). So instead, as shown above, we select the option at the bottom: "Set ALL keys to Current Field LUT." We might hope that this selection would cause all the movie path keys to refer to the same variable and color and opacity look-up table as are shown in the image window. No luck. Instead we get an error (you and I might call this a bug, but the HVR author does not).


Create/Edit/Add To Project
Add Set Below to List Above
Add Field/LUT/Tick sets From HV Project File ...
Browse Field...
Browse LUT...
Browse Tick...
Browse Dir...
Cancel
OK

In this dialog, you can choose which FIELDS are associated with a PROJECT and choose which (NAME, FIELD, LUTS, and TICK) set are associated with a FIELD

Name / Field / LUT / Tick Sets currently assocaitated with this project

v73v20145hv / C:\HVR\v73dT\v73v20145hv.fld / C:\HVR\v73dT\v73LenstPRW2transp.lut / none

Field Name: v73v20145hv

Field File: C:\HVR\v73dT\v73v20145hv.fld

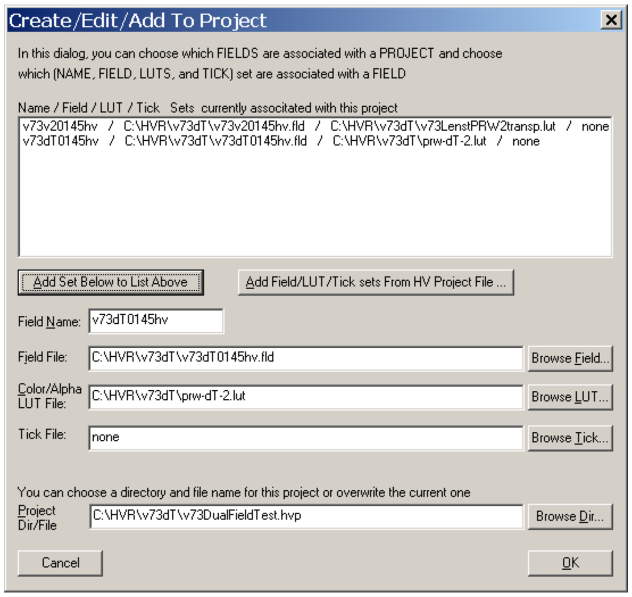
Color/Alpha LUT File: C:\HVR\v73dT\v73LenstPRW2transp.lut

Tick File: none

You can choose a directory and file name for this project or overwrite the current one

Project Dir/File: v73DualFieldTest2.hvp

The trick here is to read into this new project the old "field" that refers to the hv-files used in the imported movie path. This is somewhat easily done. We merely need to select "Edit project . . ." from the "File" menu on the HVR main form. The "Create/Edit/Add To Project" window shown at the right pops up. In the main text window, we can see the vorticity field that we originally set up this particular project to visualize. In the text boxes below, this same field originally appeared, as is shown at the right. However, I can change the entries in the lower text boxes on this window so that they refer instead to the relative temperature data that I used in an earlier project to define my movie path. At the top of the next page, this same window is displayed after I have made those alterations. Note that I may or may not want to change the name of the project in the bottom text box of this window before clicking "OK." To add the modified entries in the lower part fo the form to the list of fields in the main text window, I must click on the “Add Set Below to List Above” button, which I have done just before capturing the window’s appearance at the right here. Once I click “OK” on this window, a new field will appear in the list of fields on the main HVR form. The field I had originally displayed, in this case the vorticity, is still displayed in the image window. If I click on the relative temperature field name in the “Fields” window of the main form, I will get the same view as now in the image window, but for the newly designated field and for its default color and opacity look-up table. But I do not want to bring this relative temperature field up at this point. This is the old field that I am trying to replace in the movie path with the vorticity field. After all this, this replacement is finally easy to do. I just select any movie key frame, and then select on the pop-up menu the last option, “Set ALL keys to Current Field LUT.” This time, this process works as one might expect (if the label had said “Field & LUT”). Violà! I have my same movie path for the new variable and for its default color and opacity look-up table.



Create/Edit/Add To Project

In this dialog, you can choose which FIELDS are associated with a PROJECT and choose which (NAME, FIELD, LUTS, and TICK) set are associated with a FIELD

Name / Field / LUT / Tick Sets currently associtated with this project

```
v73v20145hv  /  C:\HVR\v73dT\v73v20145hv.fld  /  C:\HVR\v73dT\v73LenstPRW2transp.lut  /  none
v73dT0145hv  /  C:\HVR\v73dT\v73dT0145hv.fld  /  C:\HVR\v73dT\prw-dT-2.lut  /  none
```

Field Name:	v73dT0145hv
Field File:	C:\HVR\v73dT\v73dT0145hv.fld
Color/Alpha LUT File:	C:\HVR\v73dT\prw-dT-2.lut
Tick File:	none

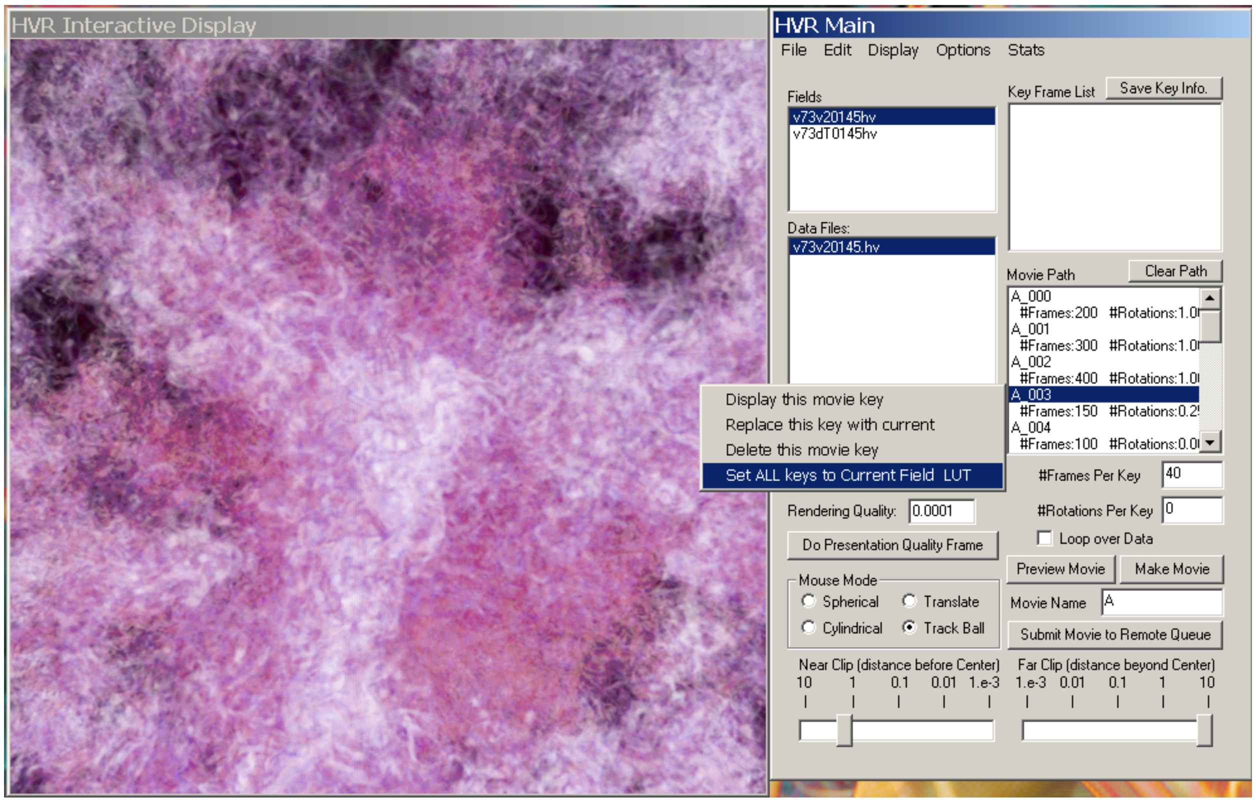
You can choose a directory and file name for this project or overwrite the current one

Project Dir/File	C:\HVR\v73dT\v73DualFieldTest.hvp

Add Set Below to List Above
Add Field/LUT/Tick sets From HV Project File ...
Browse Field...
Browse LUT...
Browse Tick...
Browse Dir...
Cancel
OK

The main form after all of these laborious manipulations is shown below. At this point, the thing to do is to check out each of the key frames of the movie. I don’t want to change any of the viewing parameters, but I may very well want to change the color and opacity look-up tables, since this is a new variable. In this example, I chose to change these tables for the few key frames in the middle of the movie where I have sliced much of the star’s envelope away, so that I am looking at only a thin slice through the middle of the star. For these key frames, I need to increase the opacity values, so that the thin slice is bright and clearly visible.

Once I have made sure that I like all the key frame colors and opacities, then I am ready to preview my movie. To do this, I might want to revise the quality settings for the interactive display, so that the preview is not just a



HVR Interactive Display

HVR Main

Fields:
v73v20145hv
v73dT0145hv

Data Files:
v73v20145.hv

Movie Path

```
A_000
  #Frames:200   #Rotations:1.0
A_001
  #Frames:300   #Rotations:1.0
A_002
  #Frames:400   #Rotations:1.0
A_003
  #Frames:150   #Rotations:0.2
A_004
  #Frames:100   #Rotations:0.0
```

#Frames Per Key	40
#Rotations Per Key	0
Rendering Quality:	0.0001
Movie Name	A

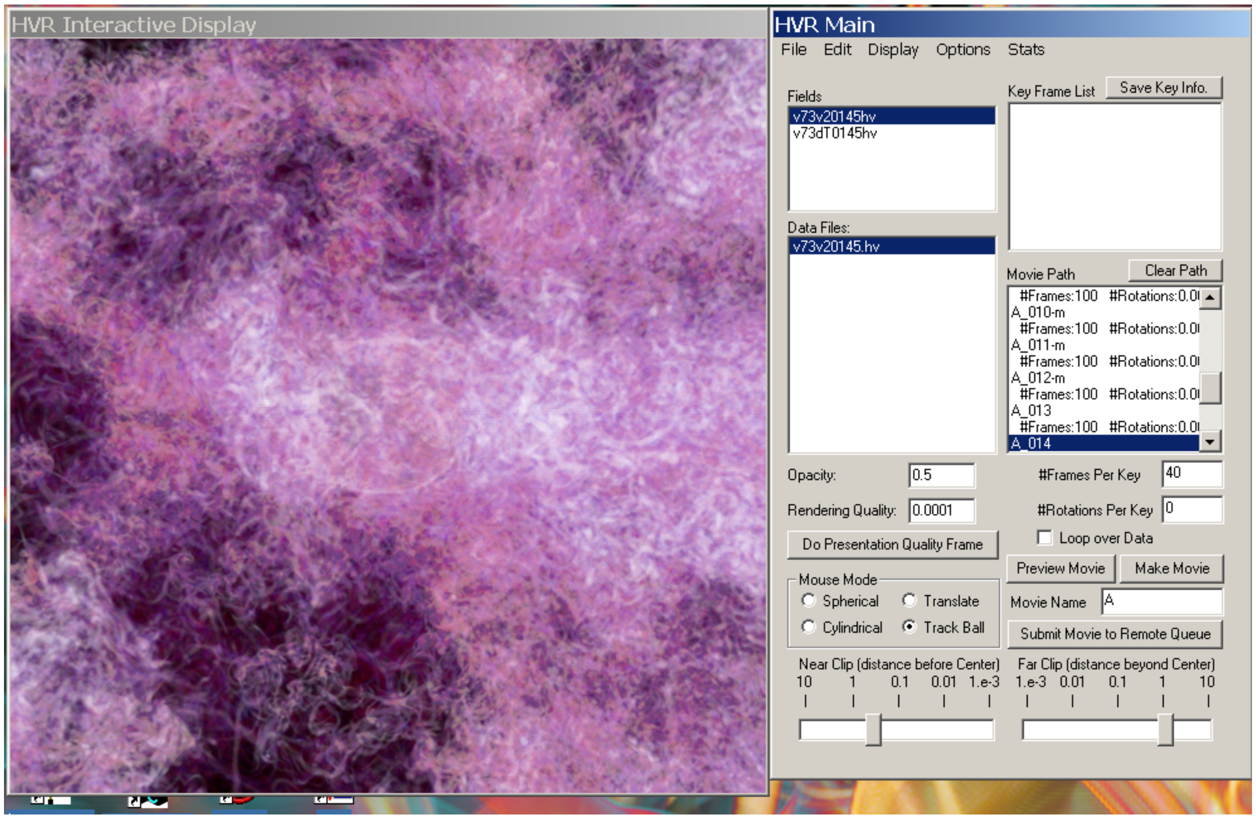
☐ Loop over Data

Mouse Mode: ○ Spherical ○ Translate ○ Cylindrical ● Track Ball

File
Edit
Display
Options
Stats
Key Frame List
Save Key Info.
Clear Path
Display this movie key
Replace this key with current
Delete this movie key
Set ALL keys to Current Field LUT
Do Presentation Quality Frame
Preview Movie
Make Movie
Submit Movie to Remote Queue
Near Clip (distance before Center)
10 1 0.1 0.01 1.e-3
Far Clip (distance beyond Center)
1.e-3 0.01 0.1 1 10

simple blur on my screen. For the movie in this example, I went a bit overboard and specified 3300 frames. So the preview takes rather a while. I wish David had put in an "Abort Preview" button. But the preview doesn't take anywhere near as long as rendering the presentation quality movie. To get that rendering, I need only click on the "Make Movie" button on the main form. HVR will think unaccountably long, figuring out what all 3300 frames should be, I guess, and then it will prompt me for a name and directory for the movie file. I like to use a .movie file extension for my movies, but you can give your movie any name you like (I would always avoid inserting spaces in file names used by software written by Unix people – just a warning). The presentation movie making process is really long compared to the preview, so it does have an abort button, in case you suddenly realize that this is just not your movie or in case you simply can't wait any longer.

Making movies with the LCSE movie queue:



Fields
v73v20145hv
v73dT0145hv

Data Files:
v73v20145.hv

Key Frame List

Movie Path
```
  #Frames:100   #Rotations:0.0
A_010-m
  #Frames:100   #Rotations:0.0
A_011-m
  #Frames:100   #Rotations:0.0
A_012-m
  #Frames:100   #Rotations:0.0
A_013
  #Frames:100   #Rotations:0.0
A_014
```

Opacity: 0.5
Rendering Quality: 0.0001
#Frames Per Key 40
#Rotations Per Key 0
☐ Loop over Data
Mouse Mode: ○ Spherical ○ Translate ○ Cylindrical [✓ Track Ball]
Movie Name A