MADAI Workbench 1.8.0 Tutorial: A Supplement to the ParaView 3.98 Tutorial

Cory Quammen

Contents

1 Introduction

The MADAI Workbench is a visualization program based on ParaView. It uses ParaView's plugin-mechanism to provide features over and above what ParaView provides in addition to all the great features already in ParaView.

As with all visualization tools, the first question you must answer is how to get the your data into the visualization program. The first half of this tutorial starts with some examples of converting example data files into a format that can be imported by the MADAI Workbench.

The second half of this tutorial will walk you through using several of the extensions that the MADAI Workbench provides. These features are not available in a plain installation of ParaView.

The content of this tutorial assumes that you have gone through the ParaView Tutorial for version 3.98 or higher of ParaView. You also need the tutorial data in the ZIP file `VisTutoriaData.zip`.

2 Importing Data

The MADAI Workbench can import many different file types. In this section, we will perform a relatively simple manual conversion of an example text file containing data from a simulation into a VTK (Visualization Toolkit) file that the MADAI Workbench can load.

2.1 Importing a Text File

It is often the case that simulation codes produce output files in a custom format that no other programs can read. In some cases, however, the output format file may be modified slightly to become a file that the MADAI Workbench can read. We will be loading ASCII files written by a particular program that has five data points per line separated by spaces with no header. In this exercise, we are going to edit this file (make a copy first) to turn it into a VTK file. This file format is described in the VTK File Formats PDF.

Copy the file `AR.spec` to `AR.vtk`. Open `AR.vtk` in your favorite text editor and paste the following text into the beginning of the file. Do not include the portions of the line following `//`. Make sure to get the correct number of spaces in each line. Note the space between the `#` and `vtk` in the first line.

```
# vtk DataFile Version 2.0   // File-format indicator string (magic cookie)
Converted Jonathan Lees file // This comment line  must be present
ASCII
DATASET STRUCTURED_POINTS
DIMENSIONS 7461 4096 1       // Tells the number of points in X, Y, and Z
SPACING 1 1 0                // Computed based on number of points and X,Y extent
ORIGIN 0 0 0                 // Centered around (0,0) in X and Y
POINT_DATA 30560256          // 7461x4096x1 total data values
SCALARS AR float 1           // One scalar per value. Name based on file name (AR)
LOOKUP_TABLE default         // Use the default color map
```

After you save the file with your modifications, load it in the MADAI Workbench. It will probably take a long time to parse and load the ASCII file. To make future operations faster, you can save the data as a binary file that will be loaded later. To do this, select **File** → **Save Data** and then change the **Files of type** menu to "VTK ImageData Files (*.vti)". Give the file the name `AR.vti` and click **OK**. Make sure **Data Mode** to Binary and **Compressor Type** is set to None. Now, when you load `AR.vti`, it will read in much faster.

The file `AR.vti` is too large to process on most computers. We will extract a subset of the file to work with. Click on `AR.vtk` in the pipeline browser and add the filter **Extract Subset**. Set the first row of text fields of the **VOI** (volume of interest) to 2000 and 3000, the second row of text fields to 0 and 1000, and the third row of text fields to 0 and 0. Click **Apply**. Now save this file to `AR_small.vti`.

Now do the same modifications to the files `MTM.FVMAT` and `MTM.spec` files to produce `FV_small.vti` and `M_small.vti`. Make sure the fields are called FV and M in the headers.

2.1.1 Combining Different Data Fields

The files you produced in the previous section each contain a single scalar data field. For certain visualization techniques, the separate scalar data fields must be combined into a single dataset. The **Append Attributes** filter makes it possible to combine the different data sets.

Reset the MADAI Workbench and load the files `AR_small.vti`, `FV_small.vti`, and `M_small.vti`. Select all three by clicking the first file in the pipeline browser, then press the shift key and click the last file in the pipeline browser. Next, select **Filters** → **Alphabetical** → **Append Attributes**. This file creates a new object that has all three data sets included in it.

Click on `AppendAttributes1` and choose **File** → **Save Data**. Save this file as a VTI binary file. In the future, you can load just this file and have all three data fields loaded. This is more convenient than having to load each data field file separately.

2.1.2 Visualizing Three Data Fields in 2D

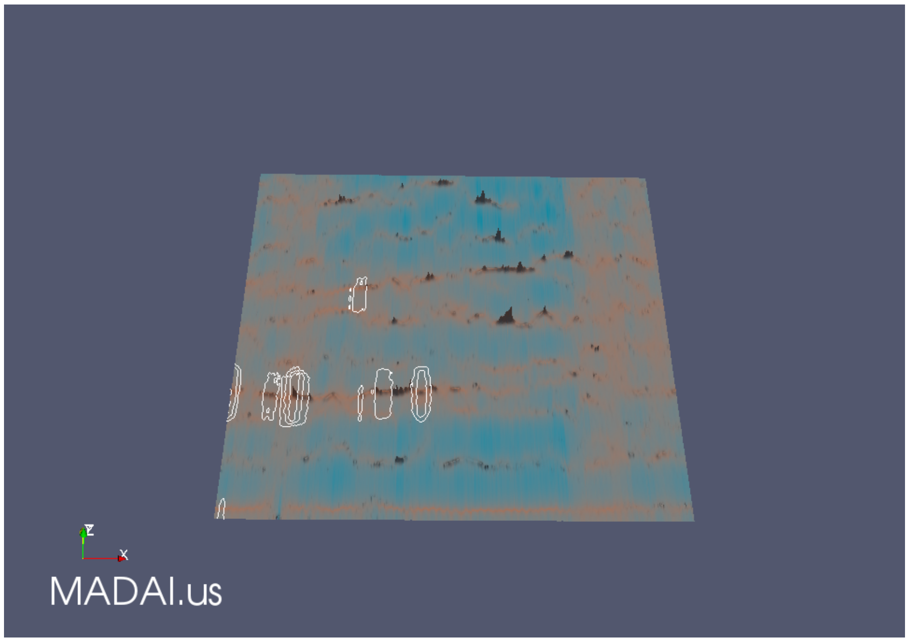


Figure 1: Three data fields with different display channels.

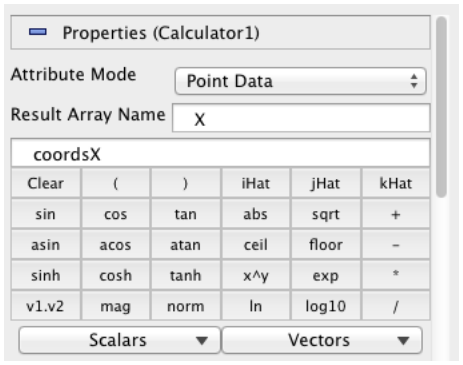
With the `AppendAttributes1` still selected, select **Filters** → **Alphabetical** → **Warp Scalar**. Set the **Scalars** field to `FV` and the **Coloring** field to `AR`. Choose the `AR` for **Coloring**. Change the colormap to an isoluminant one by clicking on the **Edit** and clicking the **Choose Preset** button. Choose the color map "CIELab Blue to Red". Now, add a **Filters** → **Alphabetical** → **Contour** filter to `WarpScalar1` filter. Set the **Contour By** filter to `M`. Delete the default contour value and click the **Add Range** button. Set the number of steps to 5 and click **OK**.

Figure 1 shows the resulting visualization of the three data fields with different display channels (`FV` mapped to height surface, `AR` mapped to color scale, and `M` mapped to isocontour lines).

3 MADAI Workbench Features

3.1 Threshold Points Filter

One type of data you may want to view is point data, especially if your work involves simulating particle systems. You may want to threshold a point set to show only points that have a variable value within some range. There is a problem with the **Threshold** filter provided by ParaView when you try to do this. This filter operates on cells in a dataset. Cells, such as triangle or tetrahedra, are elements that either have a surface area or volume. This is a problem when your dataset consists of points with no associated cells. This can show up in strange ways when you have applied a **Threshold** filter to your dataset, but no filtering appears to have been done.


Properties (Calculator1)

Attribute Mode: Point Data

Result Array Name: X

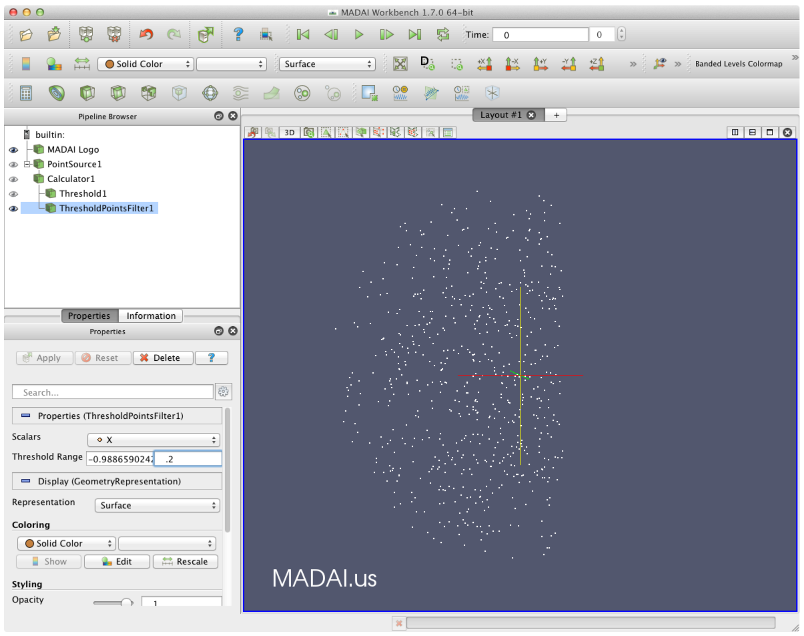
coordsX

Figure 2: Creating a point scalar field for the point data.

To see that this is the case, create a point source by selecting **Sources** → **Point Source**. Change the **Number of Points** to 1000 and **Radius** to 1. Next, create a **Calculator** filter by selecting **Filters** → **Alphabetical** → **Calculator**. Set the **Result Array Name** to "X" and enter "coordsX" in the text field above the calculator buttons (you can also click on the **Scalars** menu and select **coordsX**). Figure 2 shows the settings for the calculator. This creates a point scalar field that has the same value as the x-coordinates of the points.

Now apply the **Threshold** filter (**Filters** → **Alphabetical** → **Threshold**). Select "X" from the **Scalars** menu and uncheck the **All Scalars** checkbox and then click **Apply**. You should still see all the original points in the 3D View. Next, change the **Maximum** value to 0.2. Again, all the original points are displayed, indicating that thresholding did not work. Now delete the `Threshold1` filter.

Now we will perform thresholding that works for point data sets. Select the `Calculator1` filter in the Pipeline Browser again and this time create a **Threshold Points** filter (**Filters** → **Alphabetical** → **Threshold Points**). Select "X" as the scalars and change the right value of the **Threshold Range** to 0.2. This time, the points will be filtered such that points with the X value (and x-coordinate) above 0.2 are removed from the point set, as shown in Figure 3.



Properties (ThresholdPointsFilter1)
Scalars: X
Threshold Range: -0.9886590242 .2
Display (GeometryRepresentation)
Representation: Surface
Coloring: Solid Color
Styling
Opacity: 1

Figure 3: Results of the **Threshold Points** filter.

3.2 Binning Filter

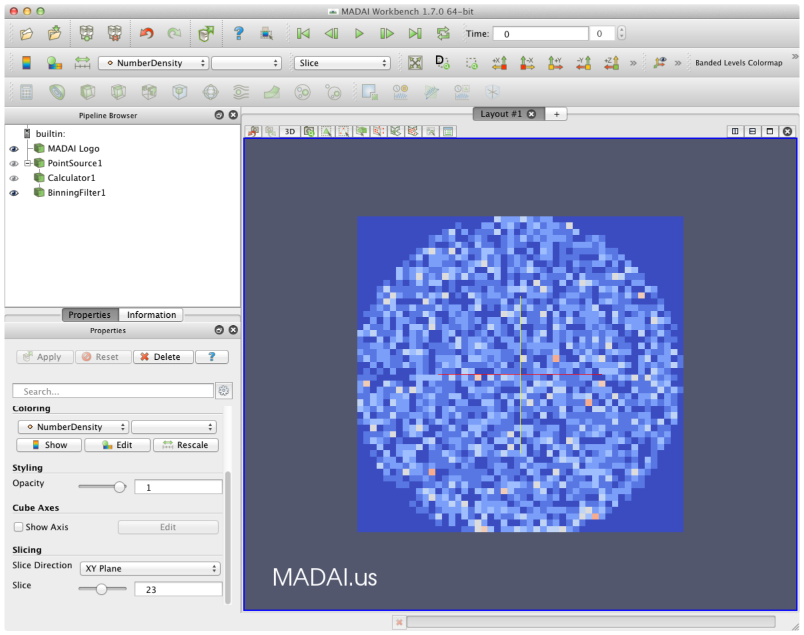
When you have a large number of points, it may be difficult to estimate certain quantities such as local density of points because of *overplotting*. Overplotting occurs when there are fewer pixels than points to display; more than one point will map to one pixel in the display, and you will see only the nearest point.

In cases of overplotting, you can instead compute an intermediate dataset that stores density information by using the **Binning** filter. The **Binning** filter provides a fast way to create a regular 2D or 3D grid representing density from a point set.

If you have completed the exercise from the previous section, delete the and `ThresholdPoints1` filters from the pipeline by right-clicking them and choosing the menu item **Delete**. Now click on the `PointSource1` source and change the **Number of Points** to 100,000. Now apply a **Binning** filter to it (**Filters** → **Alphabetical** → **Binning** filter). The default settings are fine. Click **Apply**. Now set the representation to `Slice` and set the **Slice Direction** to XY Plane. Move the **Slice** to the middle of the range. In the **Coloring** section, choose the `NumberDensity` point field in the popup menu and click the **Rescale** button. You should see an image similar to Figure 4.

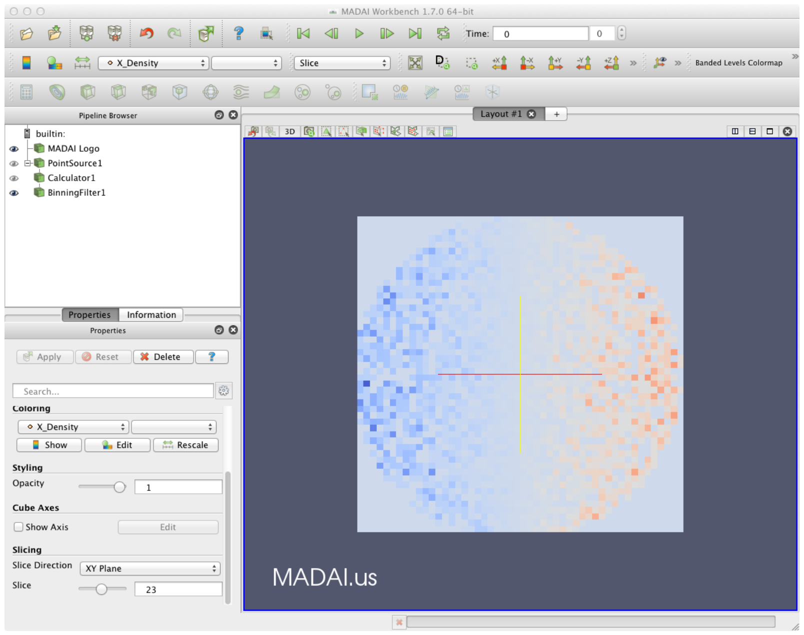
The `NumberDensity` field gives the number of points that fall within each grid cell. In addition to the `NumberDensity` field, the binning filter also sums up each point data field. Change the point data field in the **Coloring** section to `X_Density`. The value of each grid point in this filter's output is the sum of all the `X` values of the points that fall within the cell (see Figure 5).

To get the average density of the points in the grid cell, you can divide the `X_Density` field by the `NumberDensity` field. To do this, apply a **Calculator** filter to `BinningFilter1`, set the **Result Array Name** to `AverageX`, and set the text field to "X_Density / NumberDensity" (without the quotation marks). Choose the `AverageX` field in the **Coloring** section



Time: 0
Coloring: NumberDensity
Styling: Opacity 1
Cube Axes: Show Axis
Slicing: Slice Direction XY Plane
Slice 23

Figure 4: Results of the **Binning** filter.



Time: 0
Coloring: X_Density
Styling: Opacity 1
Cube Axes: Show Axis
Slicing: Slice Direction XY Plane
Slice 23

Figure 5: The `X_Density` field computed by the **Binning** filter from the `X` field.

and click the **Rescale** button. The image will look largely the same as Figure 5. The main difference is that the range of the `AverageX` is now close to [-1,1]. You can confirm this by clicking on **Information** tab. Under **Data Arrays**, you will see a table showing field names, field data types, and the numeric ranges of the fields. Look at the table row for field `AverageX`. It is slightly smaller than the range [-1,1]. Because this is in a sense the average x-coordinate value from the points in each bin and the points were sampled from a sphere with radius 1, this range makes sense.

3.3 Gaussian Scalar Splatter Filter

Whereas the **Binning** filter computes discrete summation of points within a grid cell, the **Gaussian Scalar Splatter** filter computes smoother density fields by spreading the values associated with each point according to an isotropic 3D Gaussian "splat" function whose integrated energy is 1. The 3D Gaussian splat's size is controlled by the standard deviation parameter of the Gaussian and is specified in the units of the space in which the data is embedded (e.g., meters). The splat size determines how far the influence of each point reaches and how smooth the resulting scalar field.

The **Gaussian Scalar Splatter** operates on each point in the input dataset. For each input point, the splat is centered at the point. The integrated portion of the splat that intersects a grid cell in the output is computed and added to the value at the cell. After each point is processed, each cell in the output has a number density similar in concept to the number density field produced by the **Binning** filter. In fact, this filter produces the same results as the binning filter as the size of the splat approaches 0. Each point data field `F` is multiplied by this integrated value and added to a new data field called `F_Density`. An important feature of this filter is that if you integrate a dataset field produced by the filter and integrate the dataset field input to this filter, the answer is the same (within some tolerance). This is true no matter the resolution of the output dataset.

Click on the `PointSource1` filter and change the **Number of Points** to 1000. If you do not change this number to 1000, it will take too long to compute.

Next, add a **Gaussian Scalar Splatter** filter (**Filters → MADAI → Gaussian Scalar Splatter**) to `Calculator1`. Leave the **Standard Deviation** setting at 0.1, but change the **Sample Dimensions** to 25 25 25. (***WARNING***: if you don't change the **Sample Dimensions** to 25 25 25, you will be wait a *very* long time for the filter to finish.) Click the **Apply** button. Next, change the representation to `Volume`, color the data by `NumberDensity`, and click the **Rescale** button. You should see an image similar to the left side of Figure 6.

Now change the **Standard Deviation** value to 0.2 and click **Rescale**. Notice how the splatted result is smoother on the right side of Figure 6.

Finally, color the slice by the `X_Density` field and click the **Rescale** button. Figure 7 shows the result.

3.4 Ensemble Surface Slicing Representation

In the ParaView tutorial, we have seen that several representations are available for displaying data in ParaView (`3D Glyphs`, `Outline`, `Points`, `Surface`, `Surface with Edges`, `Wireframe`). Another representation is suitable for displaying a group of surfaces using a

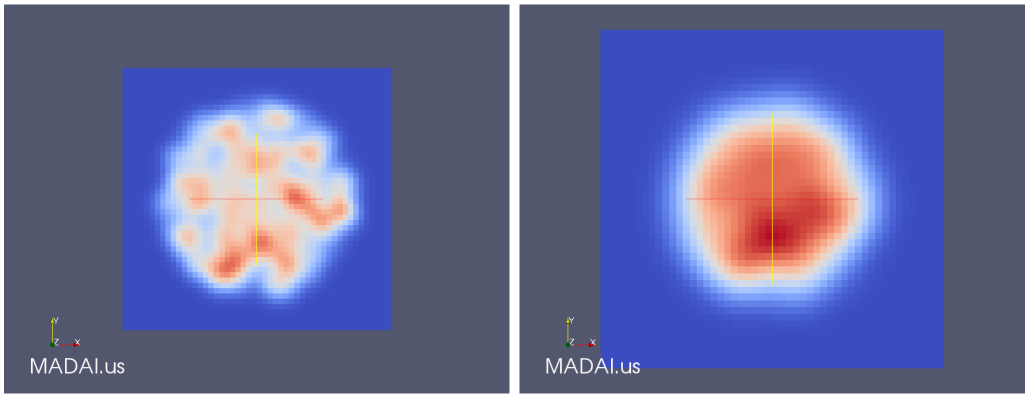


Figure 6: Point density produced by the **GaussianScalarSplatter** filter (`NumberDensity` field) with standard deviation 0.1 (left) and standard deviation 0.2 (right).

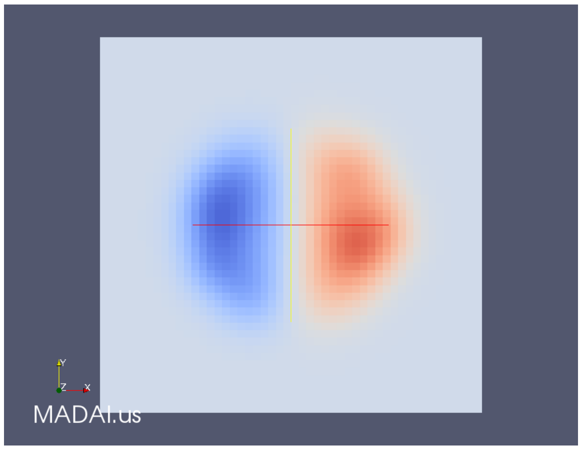


Figure 7: `X_Density` produced by the **Gaussian Scalar Splatter** filter with standard deviation 0.2.

technique called Ensemble Surface Slicing [1]. In this technique, more than one surface embedded in the same space are all displayed at the same time. However, only slices from each surface are displayed. These slices are interleaved in such a way that the slices from all the surfaces are mostly visible (some slices from one surface may be occluded by slices from other surfaces depending on the shape and viewing angle.

To try this technique, open the files `apple.ply`, `banana.ply`, `cherry.ply`, and `grape.ply`. The `Ensemble Surface Slicing` representation operates on a set of surfaces grouped together into what is called a multiblock dataset. A multiblock data set is a hierarchy of other datasets. To group the files, select them all in the pipeline browser by selecting the first file in the browser, then press the shift key and select the last file. Next, group them together with **Filters** → **Alphabetical** → **Group Datasets**. The resulting display will show the union of the surfaces, the same as what you would get just by having all the surfaces' visibility turned on.

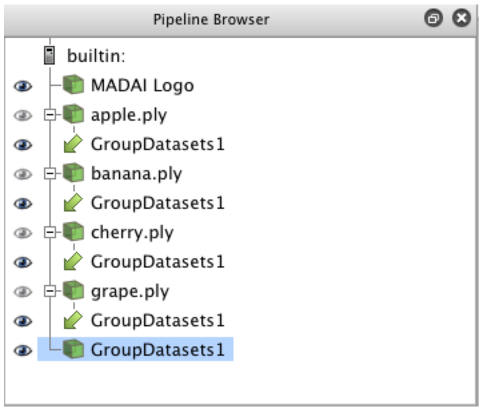
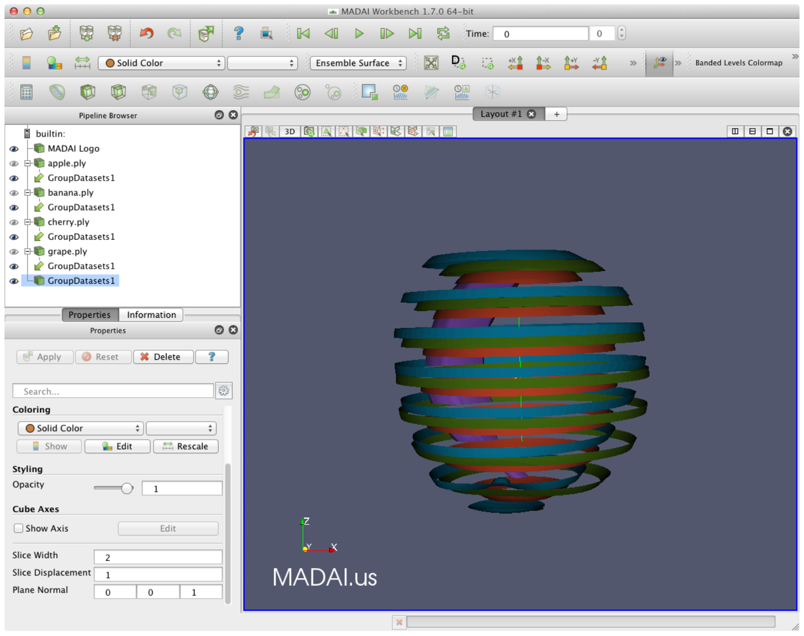


Figure 8: The fruit datasets `apple.ply`, `banana.ply`, `cherry.ply`, and `grape.ply` grouped together.

Next, change the representation of the grouped surfaces to `Ensemble Surface Slicing`. Change the **Slice Width** setting to 2 and the **Plane Normal** to 0, 0, 1. You should see an image similar to Figure 9. Each surface is colored by a different color. These colors are automatically chosen to be perceptually well-separated in color space but isoluminant to avoid obscuring the perceived shapes. You can try changing the **Plane Normal** to different values to change the orientation of the slices and the **Slice Width** to change the width of the slices.

References

[1] Oluwafemi S. Alabi, Xunlei Wu, Jonathan M. Harter, Madhura Phadke, Lifford Pinto, Hannah Petersen, Steffen Bass, Michael Keifer, Sharon Zhong, Chris Healey, and Russell M. Taylor II. Comparative visualization of ensembles using ensemble surface slicing. pages 82940U–82940U–12, 2012.



```
builtin:
  MADAI Logo
  apple.ply
    GroupDatasets1
  banana.ply
    GroupDatasets1
  cherry.ply
    GroupDatasets1
  grape.ply
    GroupDatasets1
  GroupDatasets1
```

Coloring: Solid Color

Styling
Opacity 1

Cube Axes
Show Axis

Slice Width 2
Slice Displacement 1
Plane Normal 0 0 1

Figure 9: The fruit datasets displayed with the ensemble surface slicing technique.