Introduction to the VisTrails system

Steven P. Callahan
Juliana Freire
Emanuele Santos
Carlos E. Scheidegger
Cláudio T. Silva
Wayne Tyler
Huy T. Vo

Scientific Computing and Imaging Institute
School of Computing
University of Utah
VisTrails@sci.utah.edu

Summary
This document is a short introduction and tutorial of the VisTrails prototype. This is not meant for widespread public use! We are making this early alpha release of VisTrails available to a select group of potential collaborators to give them a feel of the system and to get early feedback. You should expect broken or missing features and bugs in this release as well as major changes between this and future versions of VisTrails.

Rationale and Goals of VisTrails
Scientists are now faced with an incredible volume of data to analyze. To analyze and validate various hypotheses, it is necessary to create insightful visualizations of both the simulated processes and observed phenomena. Data exploration through visualization requires scientists to go through several steps. They need to select data sets and specify a series of operations to apply to the data to create appropriate visual representations before they can finally view and analyze the results. Often, insight comes from comparing multiple visualizations. Unfortunately, today this process is far from interactive and contains many error-prone and time-consuming tasks. Today, the generation and maintenance of visualization data products is a major bottleneck in the scientific process, hindering not only the ability to mine scientific data, but the actual use of scientific data. VisTrails extends existing dataflow-based visualization systems to streamline the creation, execution and sharing of complex visualizations.

By treating both data products and the workflows used to create the products as first class citizens, VisTrails provides a scalable mechanism for generating a large number of visualizations, comprehensive history management, and systematic maintenance of visualization provenance. VisTrails uses an XML-based dialect to represent visualization pipelines that allows the specifications to be shared and queried. In addition, these specifications are executable and can be used to re-generate images, possibly using...
different parameters. Last, but not least, the availability of formal specifications allows VisTrails to analyze and optimize these pipelines.

Further details of the system are available in the following paper: “VisTrails: Enabling Interactive Multiple-View Visualizations,” by L. Bavoil, S. Callahan, P. Crossno, J. Freire, C. Scheidegger, C. Silva, and H. Vo. This has been published at the IEEE Visualization 2005 conference, and it is available at http://www.cs.utah.edu/~juliana/pub/VisTrails-vis2005.pdf.

Funding for the development of VisTrails comes from the National Science Foundation (grants IIS-0513692, EIA-0323604, CNS-0541560, and OISE-0405402), the Department of Energy, the University of Utah, and the Army Research Office.

VisTrails Windows

Starting VisTrails

VisTrails is available on Windows XP, Mac OSX, and Linux. These versions all have the same functionality and only differ in user interface as noted throughout this document. To install on Windows and Linux, unzip the VisTrails.zip archive to your desired location. On Mac, open the archive VisTrails.dmg, and copy the two directories to your hard drive.

The VisTrails program has two main components, the Builder and the Spreadsheet. The Builder is where an actual VisTrail is built or modified and the Spreadsheet is where you can view the different
visualizations. Upon starting VisTrails you should see both windows. If you don’t, it is possible one is hidden behind the other and you may need to move or resize the front window.

![VisTrails Spreadsheet](image)

**VisTrails Spreadsheet**

**Learning the Spreadsheet**

The Spreadsheet defaults to eight cells arranged in two rows of four columns. You can change the number of cells in the rows or columns by using the controls in the lower left corner. Typing a number into the text boxes or clicking on the up and down buttons of the associated spinners will increase or decrease the number of cells in the Spreadsheet. For now, leave them at the default setting.

You can **resize the Spreadsheet** by placing your cursor on an edge or corner of the Spreadsheet window, holding down the left mouse button and moving the edge or corner of the window.

You can **resize the cells** within the Spreadsheet by placing your cursor on an edge or corner of the grey border that surrounds the cells, holding down the left mouse button and moving the edge or corner of the border. Notice that all the views are resized. By selecting and moving a vertical border edge you can make the cells wider or narrower. Moving horizontal edges resizes the height of the cells. And selecting and moving a grey border corner resizes the cell width and height at the same time.
Resizing the cells in a Spreadsheet.

**Loading a VisTrail into the Spreadsheet**

In the Spreadsheet, select **File/Open** from the menubar. Browse to the **VisTrailsData** directory located inside the **VisTrails** directory.

Select the **brain.vis** folder in the **VisTrailsData** directory and click the **Ok** button. This will load a VisTrail that uses all eight cells in the default Spreadsheet.

There might be a pause before the first cell in the first row is loaded, but notice the remaining cells in the row are quickly updated with variants of that data. The same process occurs in the second row. This is VisTrails caching at work. After the data is initially loaded, modifying and updating the data occurs very quickly.

Your Spreadsheet should look like the image below.
VisTrails Spreadsheet with brain.vis loaded.

### Adjusting the view

The view of the data can be dynamically manipulated with the mouse and keyboard. Common viewing commands are given below. Note that you can adjust the view in each cell independently.

<table>
<thead>
<tr>
<th>Command</th>
<th>Windows</th>
<th>Linux</th>
<th>Mac</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rotate</td>
<td>Left Mouse Button</td>
<td>Left Mouse Button</td>
<td>Mouse Button</td>
</tr>
<tr>
<td>Translate</td>
<td>Middle Mouse Button</td>
<td>Middle Mouse Button</td>
<td>shift + Mouse Button</td>
</tr>
<tr>
<td>Zoom</td>
<td>Right Mouse Button</td>
<td>Right Mouse Button</td>
<td>command + Mouse Button</td>
</tr>
<tr>
<td>Wireframe Mode</td>
<td>“w”</td>
<td>“w”</td>
<td>“w”</td>
</tr>
<tr>
<td>Surface Mode</td>
<td>“s”</td>
<td>“s”</td>
<td>“s”</td>
</tr>
</tbody>
</table>

### Selecting a cell

You can select a cell by holding down the `ctrl` key on your keyboard and left clicking the mouse in the desired view (`ctrl` + Mouse Button on Mac). You can select multiple cells by continuing to hold down the Ctrl button while choosing other cells. A red border around a cell indicates that it has been selected. Deselecting a selected cell is done in a similar manner.

Multiple cells can be selected simultaneously using the sync buttons at the top of each column and at the front of each row. Pressing one of these buttons will select or deselect all cells in the entire row or column. The smaller button in the top left corner of the Spreadsheet selects or deselects all the cells.
VisTrails Spreadsheet with multiple cells selected.

**Syncing cells**

You can synchronize multiple cells so any rotation, zoom or translation commands that are given in one cell are reflected in the other selected cells. After selecting multiple cells, hold down the `ctrl` button and right click on one of the selected cells (`ctrl + command + Mouse Button on Mac`). A popup selection box will appear with **Clear**, **Sync Selected Cells**, and **Unsync All Cells** options. Choose Sync Selected Cells then adjust the data in one of the selected cells. You will notice that the other selected cells adjust to match the view of the selected cell.

You can unsync the cells by selecting Unsync All Cells from the popup selection box.
Learning the Builder

The Builder is used to create or modify VisTrails. The left side of the Builder is where you can view a VisTrail as a history tree of visualizations. The right side is where you find the building blocks to create and modify a visualization in a VisTrail.

Resizing the Builder

You can resize the Builder window in the same way as the Spreadsheet by selecting and moving a window edge or corner. You can also move the divider between the left and right panels of Builder.

Loading a VisTrail into the Builder

In the Builder, select File/Open Vistrail from the menubar. Browse to the VisTrailsData directory located inside the VisTrails directory.
Select VTK_BOOK_3RD_P189.XML from the VisTrailsData directory and click the Open button. This will create a new tab on the left side of the Builder. To see the VisTrail, click on the tab labeled VTK_BOOK_3RD_P189.XML. You will see a VisTrail history tree with three ovals, the middle one labeled First.

### Viewing a VisTrail in the Builder

The view of the VisTrail can be changed with the following mouse events.

<table>
<thead>
<tr>
<th></th>
<th>Windows</th>
<th>Linux</th>
<th>Mac</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pan</td>
<td>Middle Mouse Button</td>
<td>Middle Mouse Button</td>
<td>shift + Mouse Button</td>
</tr>
<tr>
<td>Zoom</td>
<td>Right Mouse Button</td>
<td>Right Mouse Button</td>
<td>command + Mouse Button</td>
</tr>
</tbody>
</table>

### Sending a visualization from the Builder to the Spreadsheet

Before sending a visualization from the Builder to the Spreadsheet, it is recommended you select the cell or cells in the Spreadsheet where you want the data to appear.

In the Builder, select the visualization labeled First. You will notice that the oval becomes highlighted and is now ready to be sent to the Spreadsheet.
VisTrail version tree

Above the VTK_BOOK_3RD_P189.XML tab is a button labeled Send to Spreadsheet. Clicking this button sends the selected visualization to the Spreadsheet.
VisTrails Spreadsheet with VTK example

By manipulating the view in the Spreadsheet you will notice the data is a series of ovoid shapes.

**Expanding and Collapsing the Version Tree**

By default the Builder does not show the entire VisTrail. To see all the modules in the VisTrail, select **View/View Complete Version Tree** from the menubar. When this is checked, the entire VisTrail is shown in the left panel. When it is unchecked, only those visualizations with names are shown. Selecting **View Complete Version Tree** again shows the smaller version of the VisTrail.

**VisTrail module components**

When you selected the First module and sent it to the Spreadsheet, the Builder created another tab **VTK_BOOK_3RD_P189.XML – First**. If you don’t see the new tab you may need to use the tab shift buttons to access the new tab.
When you select this visualization definition tab, the left panel will display the visualization pipeline used to create the data seen in the Spreadsheet.

Viewing manipulation of the visualization pipeline is done in the same manner as the VisTrail history.

To select a module in the visualization, left click on it (Mouse Button on Mac). When a module is selected, it becomes highlighted and its parameters are shown in the right panel.
Repositioning a selected module is done by holding down the left mouse button (Mouse Button on Mac) and dragging it to the desired location.

**Making changes to a visualization module**

To change the parameters of a module, select a text edit box in the right panel and type in a value. The labels to the left of each text edit box indicate the parameter input type (double – number with a decimal point or int – whole number) and the name of the parameter.

After typing in new values for the parameters, you can choose to update the VisTrail, or update it and send it to the Spreadsheet. When a module is changed, a new instance of the visualization with the changed parameters is added to the VisTrail.

To see this, change one of the parameters for the `vtkQuadric` component (as selected above) and click on the **Update** button. Now go back to the `VTK_BOOK_2RD_P189.XML` tab panel and turn on the **View Complete Version Tree**. You will see two branches from the visualization label `First`. The longer branch is the original VisTrail, the shorter branch is the visualization you just created by changing the parameter and updating the VisTrail.
To see what effect the parameter changes have on the data, click on the **Update and Send** button. Then check the Spreadsheet to view the modified data.

**Using the VisTrails version tree**

As you make changes to the modules of a visualization, the instances are automatically added to the VisTrail. This allows you to go back to a previous version (higher up in the tree), and use a different set of parameters to modify the data without losing any of the changes you have already made.

To make a visualization you like easily accessible, you can assign it a name. This makes the visualization visible in the collapsed form of the version tree.

Select the visualization you want in the extended version tree (the module should be highlighted). In the right panel at the top, there is a text edit box with the label **Visualization Name**. Type in a name for the module and select the **Change** button to the right. This will place that name in the selected visualization in the version tree.

Now collapse the version tree. The new module is visible along with the original **First** module.
**Working with visualization modules**

In the visualization module panel, you notice that the modules are connected with lines. This shows the data flow through the modules. Modules can be connected or disconnected, and added or deleted from a visualization.

To see how this works, we will change the original data from the `vtkQuadric` module to a `vtkCylinder` module.

In the right panel, there are two tabs, one labeled **Module Methods** and the other labeled **VTK Classes**. The Module Methods is where you can change the parameters of the module. The VTK Classes contain modules that can be added to your visualization. In our `VTK_BOOK_3RD_P189.XML` example, everything comes from the VTK Classes. However, data from external sources can also be used.

**Creating a new module**

Select the **VTK Classes** tab. Click on the plus (+) to the left of `vtkObject` to show the sub-objects.

![Clicking on the + expands vtkObject](image)

Use the scroll bar on the right of the VTK Classes tab to scroll down to `vtkImplicitFunction`. Expand that by clicking on the plus (+) to the left. The third item down is called `vtkCylinder`. An alternate
method for finding the class is to use the Search box at the top of the window by typing the class name directly into the text box.

Left click (Mouse Button on Mac) on the word *vtkCylinder* and continue to hold down the mouse button. *vtkCylinder* becomes highlighted. With the mouse button still held down, drag the cursor over to the visualization panel on the left and release the mouse button. A new module, *vtkCylinder*, is added to the visualization panel.

![The vtkCylinder component added to the visualization panel](image)

**Connecting and disconnecting modules**

To change the data source from *vtkQuadric* to *vtkCylinder*, you need replace the output of the first with the second. Notice that the line connecting each of the modules starts and ends in a small box at the top or bottom of the modules.

To disconnect the *vtkQuadric* output from *vtkSampleFunction*, place the cursor over the small box at either end of the connection line. Click and hold down the left mouse button (Mouse Button on Mac). Drag the end of the line away from the module and release the mouse button. The connecting line will disappear.
To connect the **vtkCylinder** output to the **vtkSampleFunction** input, place the cursor over the small box in the lower right corner of the **vtkCylinder** module, click and hold down the left mouse button (Mouse Button on Mac). Drag the cursor away from **vtkCylinder** and a line will appear. Drag the end of the line to the left most small input box in the upper left corner of the **vtkSampleFunction** module and release the mouse button. The line now connects **vtkCylinder** and **vtkSampleFunction**.

To check that you were successful, click on the **Send to Spreadsheet** button at the top of the visualization panel. The data in the cell shows a series of cylindrical shapes.

The input ports of the module will only accept connections from correct output ports. To determine the port type, right click (**command** + Mouse Button on Mac) on the input or output box to display an overlayed window with the port information.

Accessing module parameters

You will notice that when you select the **vtkCylinder** component in the visualization panel on the left, there are no parameters to adjust in the lower right panel on the right. Only the parameters that have been modified by the user are displayed to prevent clutter.

To modify a parameter from its default setting, left click (Mouse Button on Mac) on the word **SetRadius** and continue to hold down the mouse button. **SetRadius** becomes highlighted. With the mouse button still held down, drag the cursor to the area directly below the **Update** and **Update and**
Send buttons. Then release the mouse button. A parameter text edit box is shown for SetRadius. You can enter a new radius size for the vtkCylinder component. To see the results of the new radius in the Spreadsheet, press the Update and Send button.

![Creating a parameter text edit box in the Module Methods panel](image)

**Case Study 1: Isosurface simplification**

An important visualization tool for exploring volumetric data as a result of imaging devices or scientific simulation is isosurfacing. An isosurface is the set of all locations within the volume that correspond to a given value. Often a data set is too large to visualize interactively, thus simplification techniques are employed to reduce the amount of data drawn. In this case study we will analyze the difference between simplifying a volume before extracting an isosurface with simplifying the isosurface after it has been extracted from the volume. For this case study we will use a tetrahedral mesh with scalar values representing the temperature within the volume.

**Step 1:** Load a VisTrail with a simple rendering pipeline
- Start a fresh session of VisTrails
- Load the CASE STUDY 1.XML VisTrail from the Vistrail/VisTrailData directory. This VisTrail contains one visualization called spx.
• Select the spx visualization and send it to the Spreadsheet with the **Send To Spreadsheet** button at the top of the Builder. The first cell in the Spreadsheet should now contain this visualization.

---

**Step 2: Extract an isosurface**

**Step 2:** Create an isosurface of the data.

- Switch to the **CASE_STUDY_1.XML – spx** tab in the Builder to show the visualization pipeline.
- Use the **Search** command in the right panel to find the **vtkContourFilter** class in the **VTK Classes** tab. Make sure that the **VTK Classes** tab is selected.
- Drag the **vtkContourFilter** into the left panel containing the other modules.
- Change the connection between the **vtkUnstructuredGridReader** and the **vtkDataSetMapper** by dragging the line from the output port of the **vtkUnstructuredGridReader** to the output port of the **vtkContourFilter**.
- Next, connect the reader to the contour filter by clicking in the last output of the **vtkUnstructuredGridReader** and dragging the connection to the first input port of the **vtkContourFilter**.
- To set the value of the isosurface, select the **vtkContourFilter** module and drag the **SetValue** method from the **Module Methods** tab in the right panel to the empty space below the **Update** and **Update and Send** buttons.
- Now that the isosurface parameters are exposed, change the **i** parameter to 0 and the **value** parameter to 0.9 and press the **Update and Send** button. The Spreadsheet should now contain a visualization of an isosurface of the volume in the second cell.
- Name this visualization **isosurface** by using the **Visualization Name** text box in the top right of the Builder.
Step 3: Simplify the extracted isosurface

**Step 3: Simplify the isosurface**

- Search the VTK classes for the `vtkQuadricDecimation` class and add it to the visualization.
- Remove the connection between `vtkContourFilter` and `vtkDataSetMapper` modules and connect the last output of `vtkContourFilter` to the input of `vtkQuadricDecimation` as well as the last output of `vtkQuadricDecimation` to the input of `vtkDataSetMapper`.
- Select the `vtkQuadricDecimation` module and find the module method `SetTargetReduction` and drag it into the empty space below the Update and Update and Send buttons.
- Change the value to 0.9 and press the Update and Send button to see the new visualization in the Spreadsheet.
- Name this visualization `isosurface+simplification` by using the Visualization Name text box in the top right of the Builder.
Step 4: Return to our previous visualization and simplify the mesh before isosurface extraction

• Select the CASE_STUDY_1.XML tab in the left panel of the Builder. Here you should see the named visualizations spx, isosurface, and isosurface+simplification.
• Select the isosurface visualization and return to the CASE_STUDY_1.XML – isosurface tab in the Builder. This should return to the visualization pipeline that did not include the vtkQuadricDecimation module.
• Add a vtkUnstructuredGridQuadricDecimation module from the VTK Classes tab in the right panel of the Builder to our visualization.
• Place it the module between the vtkUnstructuredGridReader and the vtkContourFilter.
• The connection between the vtkUnstructuredGridReader and the vtkContourFilter should be removed and a connection should be created between the last output port of vtkUnstructuredGridReader and the input port of vtkUnstructuredGridQuadricDecimation as well as the last output port of vtkUnstructuredGridQuadricDecimation and the first input port of vtkContourFilter.
• As before, find the SetTargetReduction method in the Module Methods tab and set the value to 0.5, then send the visualization to the Spreadsheet with the Update and Send command.
• Name this visualization simplification+isosurface by using the Visualization Name text box in the top right of the Builder.
Results of Case Study 1 from left to right: Original mesh, unsimplified isosurface, simplified isosurface, and isosurface from a simplified mesh.

The results of our case study show the error that is introduced from the simplification process both before and after isosurface extraction. To view the different results simultaneously, you can select the sync button at the right of the row of cells. This provides an easy tool for interactively comparing different regions of the visualizations.

Case Study 2: Medical Imaging

In this case study, we explore the use of some simple imaging techniques to display an image. This comparative visualization helps a scientist decide the visualization that represents the original data the best.
Step 1: Load a simple image reader
- Start a fresh session of VisTrails.
- Open the **CASE_STUDY_2.XML** from the **VisTrails/VisTrailData** directory in the builder.
- Select the **image** visualization from the version tree and go to the **CASE_STUDY_2.XML – image** tab in the Builder.
- Press the **Send To Spreadsheet** button to view the bitmap image in the Spreadsheet. The image is almost completely black.
Step 2: View the image with a geometry based approach

- Remove the `vtkImageMapper` module and the `vtkActor2D` module from the visualization by selecting it and pressing the delete key.
- Find the `vtkImageDataGeometryFilter` from the VTK Classes tab in the right panel using the Search text box.
- Add `vtkImageDataGeometryFilter` to the visualization by dragging it from the right panel to the left.
- Find the `vtkPolyDataMapper` module from the VTK Classes tab in the right panel using the Search text box.
- Add `vtkPolyDataMapper` to the visualization by dragging it from the right panel to the left.
- Find the `vtkActor` module from the VTK Classes tab in the right panel using the Search text box.
- Add `vtkActor` to the visualization by dragging it from the right panel to the left.
- Connect the last output port of the `vtkBMPReader` module to the input port of the `vtkImageDataGeometryFilter` module.
- Connect the last output port of the `vtkImageDataGeometryFilter` module to the first input port of the `vtkPolyDataMapper` module.
- Connect the output port of the `vtkPolyDataMapper` module to the fourth input port of the `vtkActor` module.
- Connect the output port of the `vtkActor` module to the first input port of the `vtkRenderer` module.
- Press Send to Spreadsheet to view the results in the Spreadsheet.
• Name this visualization *geometry-based* by typing into the **Visualization Name** text box at the top of the right panel and pressing the **Change** button.

---

### Step 3: Image-based approach

**Step 3:** View the image with an image-based approach

- View the VisTrail version tree by selecting the **CASE_STUDY_2.XML** tab at the top of the right panel in the builder.
- Select the *image* visualization and change back to the visualization tab called **CASE_STUDY_2.XML - image** by selecting the tab.
- Select the **vtkImageMapper** module by clicking on it.
- Select the **Module Methods** tab at the top of the right panel and Find the **SetColorWindow** by using the **Search** text box.
- Drag **SetColorWindow** to the space below the **Update** and **Update and Send** buttons.
- Change the double parameter to 255 and press **Update**.
- Find the **SetColorLevel** method by using the **Search** text box.
- Drag **SetColorLevel** to the space below the **Update** and **Update and Send** buttons.
- Change the double parameter to 127 and press **Update**.
- Find the **SetRenderToRectangle** method by using the **Search** text box.
- Drag **SetRenderToRectangle** to the space below the **Update** and **Update and Send** buttons.
- Change the int parameter to 1 and press **Update**.
- Select the **vtkActor2D** module in the visualization.
- Find the **SetWidth** and **SetHeight** methods in the **Module Methods** tab and drag them to the area below the **Update** and **Update and Send** buttons.
- Change the w and h parameters to 1 in **SetWidth** and **SetHeight**.

---

24
• Press the **Update and Send** button to see the visualization in the SpreadSheet.

![VisTrots - Spreadsheet](image)

**Results of Case Study 2 from left to right:** Original image, image using geometry-based approach, and image using image-based approach.

The results of this case study show the difference between an image that is mapped to polygonal data as scalars and showing it directly by changing the scale of the colors. The advantage of the geometric approach is that it can be viewed in 3D space, whereas the image approach is static.