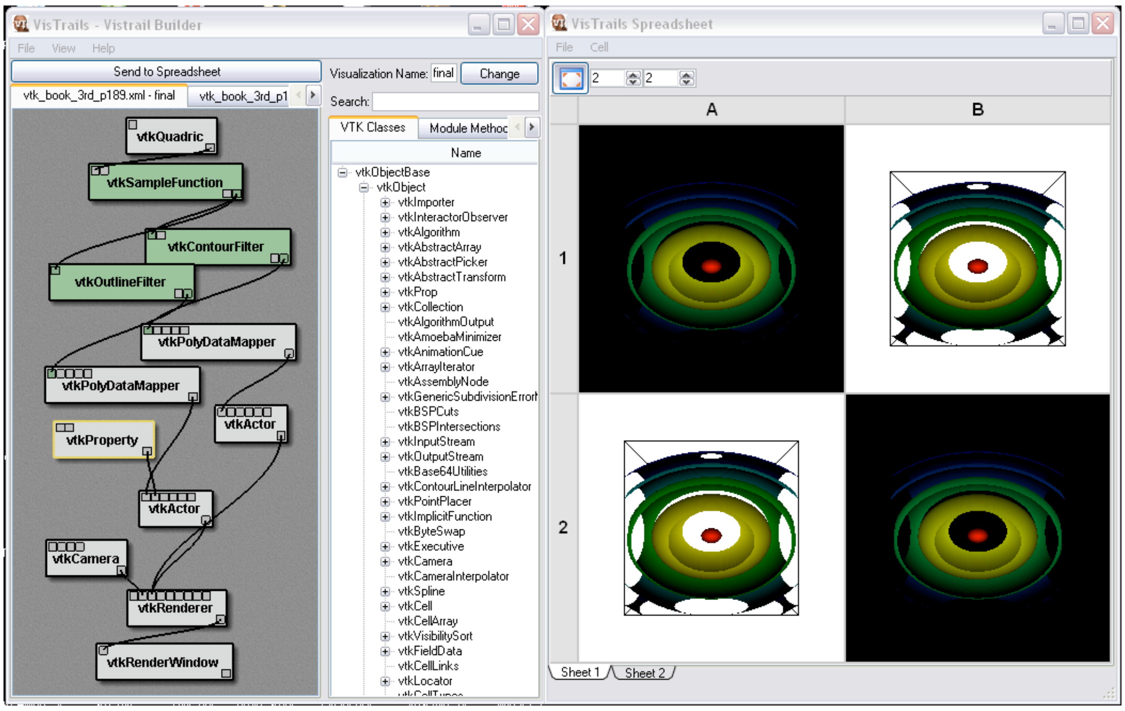
Introduction to the VisTrails System

Erik W. Anderson
Steven P. Callahan
Juliana Freire
Emanuele Santos
Carlos E. Scheidegger
Cláudio T. Silva
Huy T. Vo



The VisTrails Windows

1 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.

2 Rationale and Goals of VisTrails

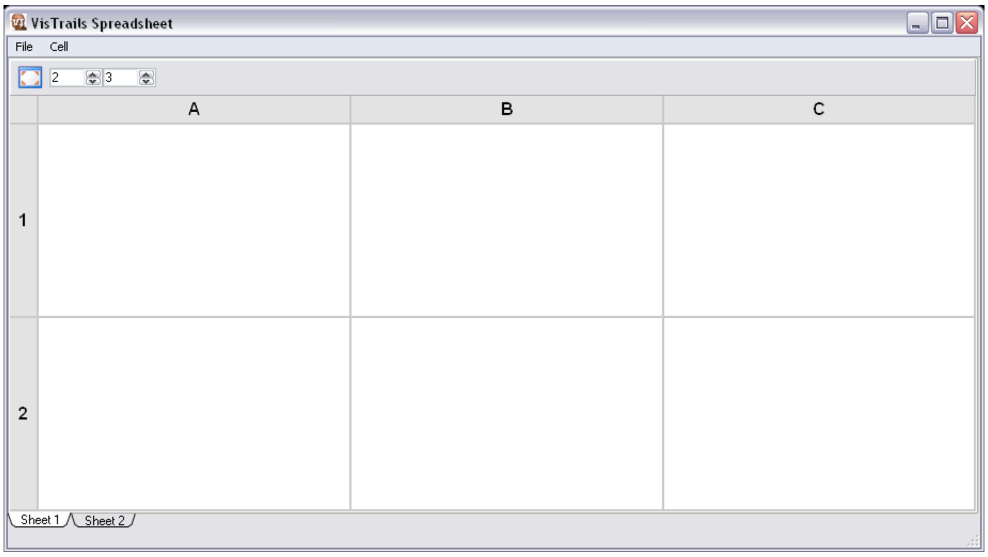
Scientists are now faced with an incredible volume of data to analyze. To analyze and validate various hypotheses, it is necessary to create insightful visualizations of both the simulated processes and observed phenomena. Data exploration through visualization requires scientists to go through several 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.

3 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 dont, it is possible one is hidden behind the other and you may need to move or resize the front window.



The VisTrails Spreadsheet

4 The VisTrails Spreadsheet

The Spreadsheet defaults to eight cells arranged in two rows of three columns. You can change the number of cells in the rows or columns by using the controls in the upper 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.

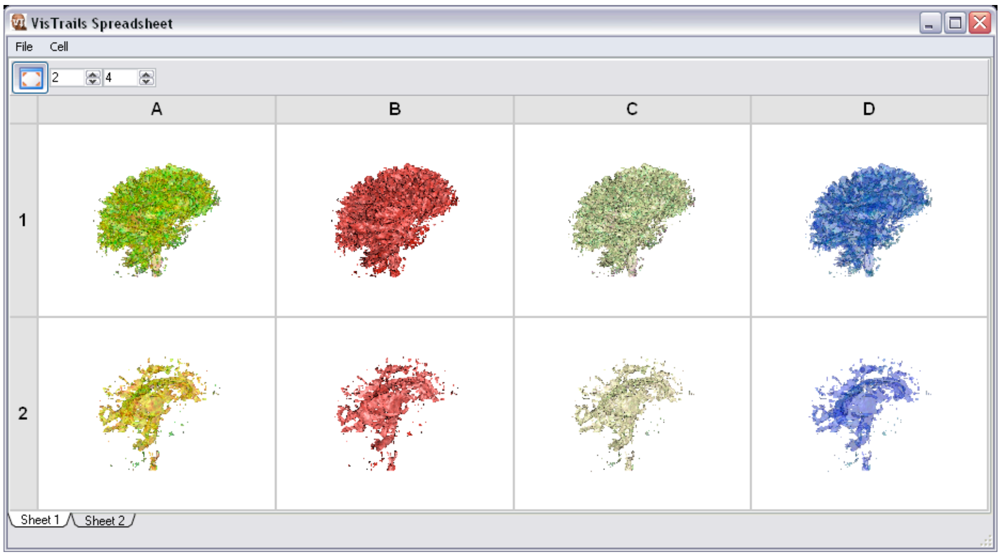
4.1 Loading a VisTrail Into the Spreadsheet

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.



The VisTrails Spreadsheet with the brain study loaded.

4.2 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.

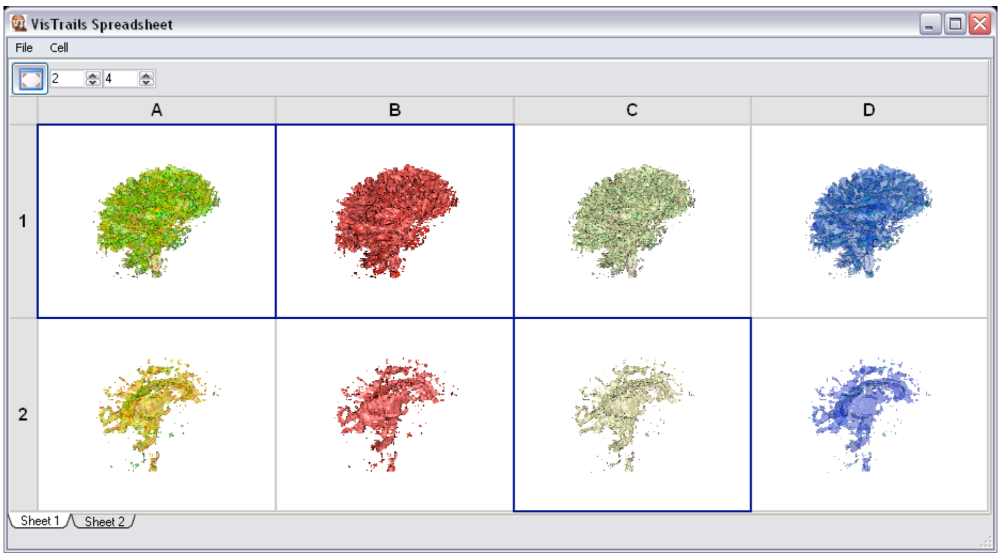
Operation	Windows	Linux	Mac
Rotate	Left Button	Left Button	Button
Translate	Middle Button	Middle Button	*shift* + Button
Zoom	Right Button	Right Button	*command* + Button
Wireframe	*w*	*w*	*w*
Surface	*s*	*s*	*s*

Mouse Events for Viewing

4.3 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 blue 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.



The VisTrails Spreadsheet with the brain study loaded.

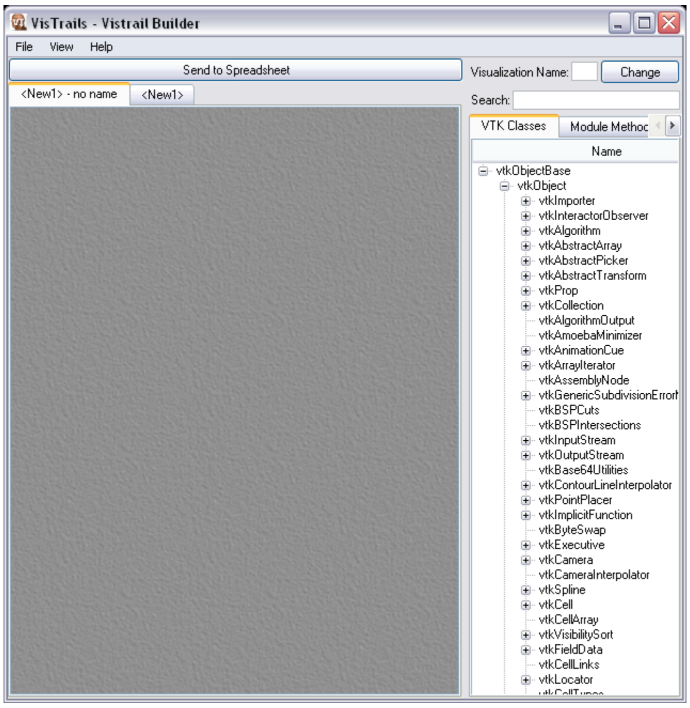
4.4 Synchronising Cells

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.

5 The VisTrails 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.



A blank VisTrails Builder Window.

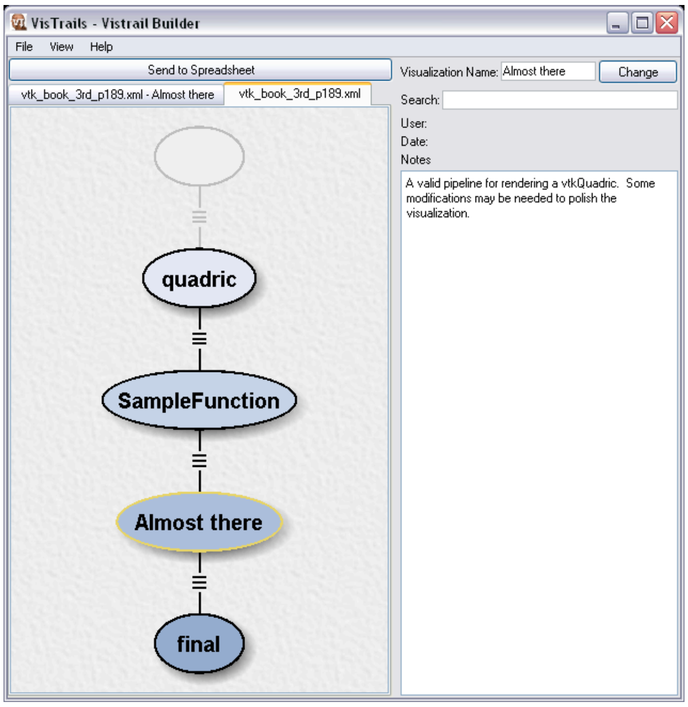
5.1 Resizing the Builder

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 by dragging the dividing bar left or right.

5.2 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 several ovals, the middle one labeled First.



Visualization Name: Almost there
Search:
User:
Date:
Notes
A valid pipeline for rendering a vtkQuadric. Some modifications may be needed to polish the visualization.

A VisTrail Loaded into the Builder Window.

5.3 Viewing a VisTrail in the Builder

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

Operation	Windows	Linux	Mac
Pan	Middle Button	Middle Button	*shift* + Button
Zoom	Right Button	Right Button	*command* + Button

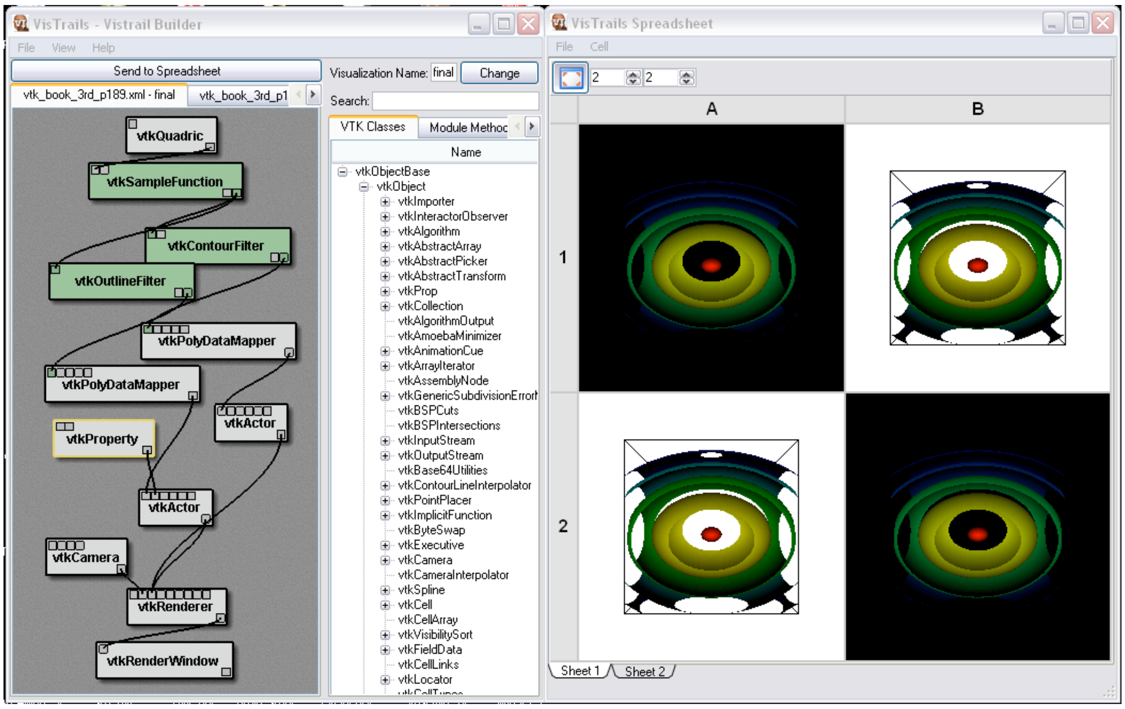
The Builder Mouse Events

5.4 Controlling Visualizations in 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.

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. Alternatively, dragging the version to the desired spreadsheet cell functions similarly to selecting a cell and then pressing the *Send to Spreadsheet* button.



The Spreadsheet containing the last two versions in the history placed by dragging the versions to the spreadsheet cells.

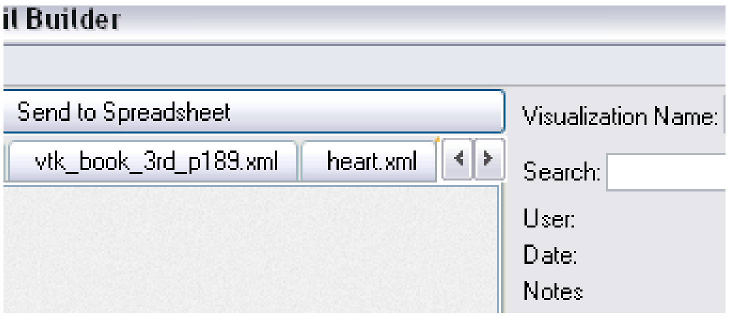
By manipulating the view in the Spreadsheet you will notice the data is a series of ovoid shapes.

5.5 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.

5.6 VisTrail Module Components

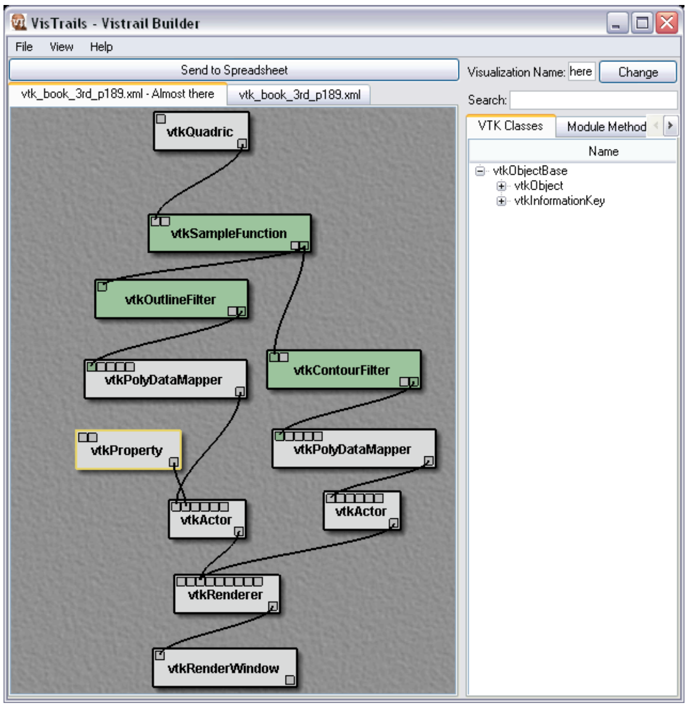
When you selected the First module and sent it to the Spreadsheet, the Builder created another tab VTK_BOOK_3RD_P189.XML First. If you dont see the new tab you may need to use the tab shift buttons to access the new tab.



The tabs illustrated above control the view displayed in the Builder Window.

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.



The pipeline view of the version marked *Almost There*.

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.

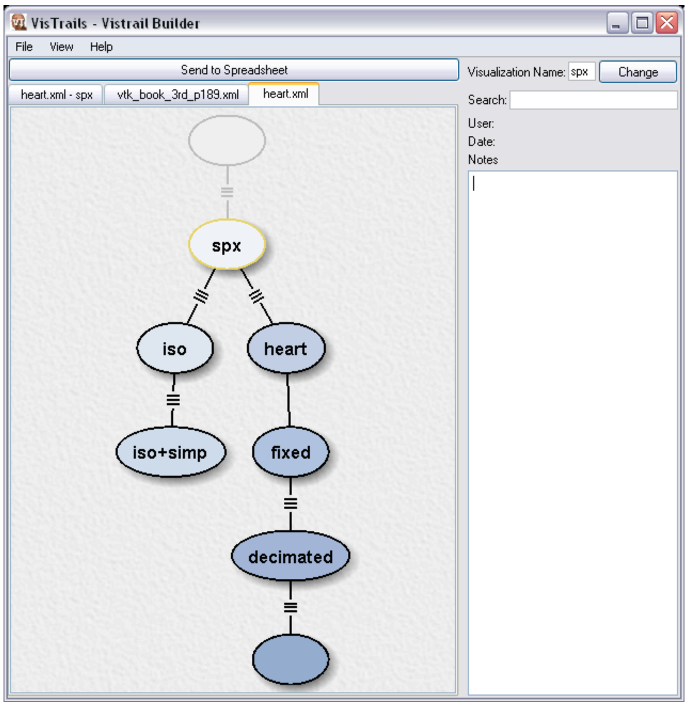
Repositioning a selected module is done by holding down the left mouse button (Mouse Button on Mac) and dragging it to the desired location.

6 VisTrails Modules

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.



The Version Tree after a change takes place to a non-leaf module.

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. Because of this version branching mechanism, all changes made to *any* version in the version tree are monotonic. This monotonicity is important when making changes to large projects collaborated on by multiple people.

6.1 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.

6.2 Working With Modules

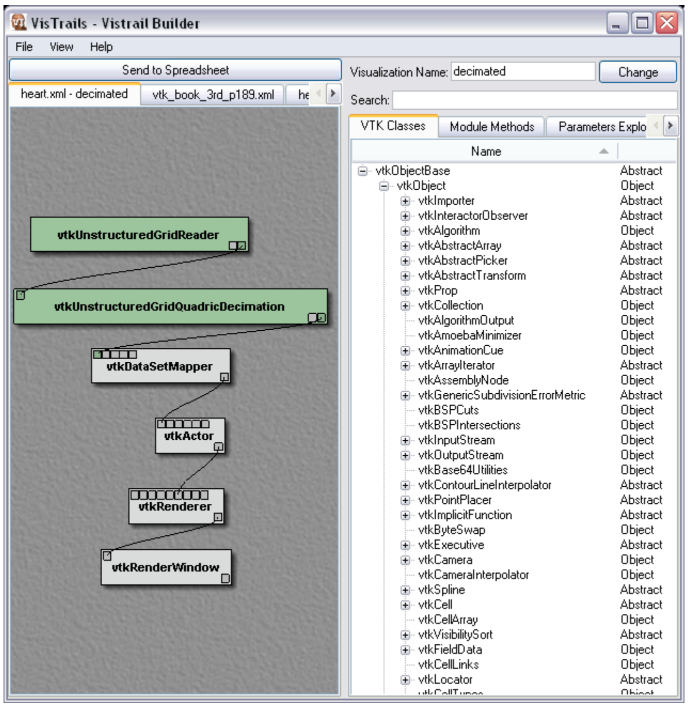
In the visualization module panel, you notice that the modules are connected with lines. This shows the data flow through the modules. Modules can be connected or disconnected, and added or deleted from a visualization.

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*. *Module Methods* is where you can change the parameters of the module. The *VTK Classes* panel contains 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.

6.3 Creating a New Module

Select the *VTK Classes* tab. Click on the plus sign (+) to the left of *vtkObject* to expand the tree.



Visualization Name: decimated

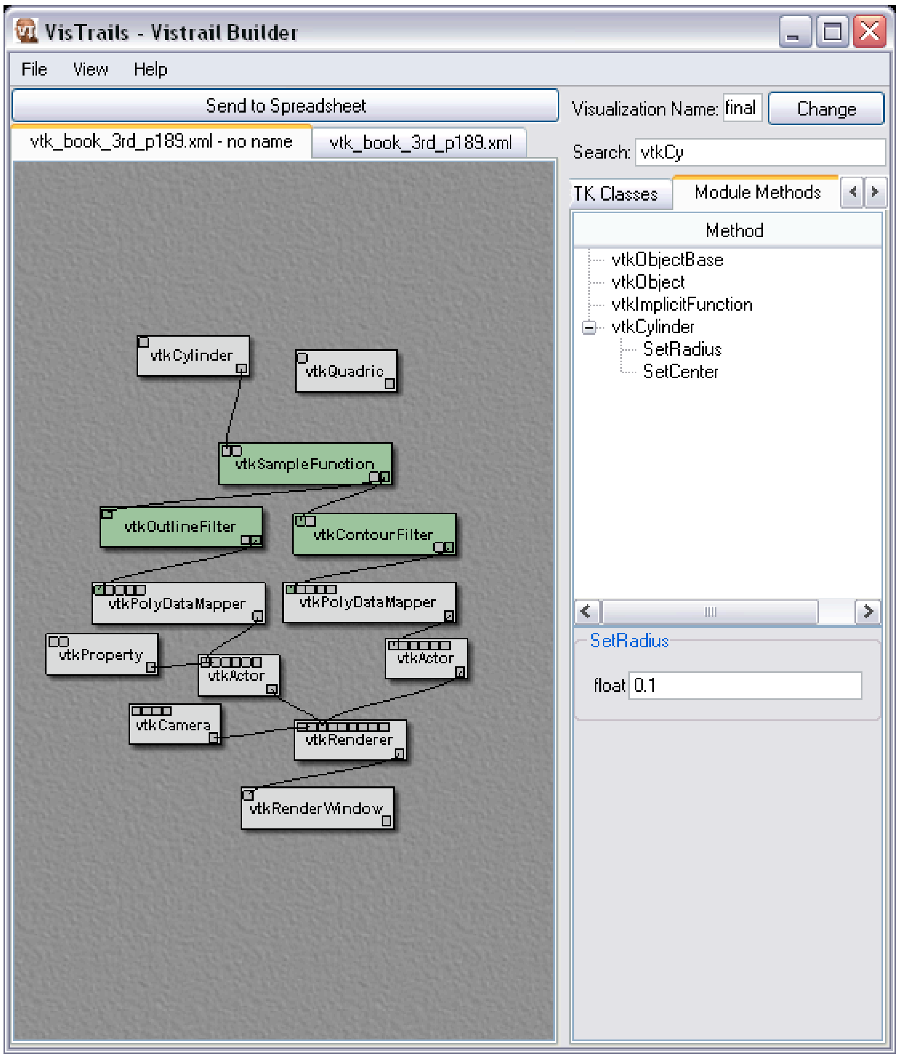
Search:

Name	
vtkObjectBase	Abstract
vtkObject	Object
vtkImporter	Abstract
vtkInteractorObserver	Abstract
vtkAlgorithm	Object
vtkAbstractArray	Abstract
vtkAbstractPicker	Abstract
vtkAbstractTransform	Abstract
vtkProp	Abstract
vtkCollection	Object
vtkAlgorithmOutput	Object
vtkAmoebaMinimizer	Object
vtkAnimationCue	Object
vtkArrayIterator	Abstract
vtkAssemblyNode	Object
vtkGenericSubdivisionErrorMetric	Abstract
vtkBSPCuts	Object
vtkBSPIntersections	Object
vtkInputStream	Object
vtkOutputStream	Object
vtkBase64Utilities	Object
vtkContourLineInterpolator	Abstract
vtkPointPlacer	Abstract
vtkImplicitFunction	Abstract
vtkByteSwap	Object
vtkExecutive	Abstract
vtkCamera	Object
vtkCameraInterpolator	Object
vtkSpline	Abstract
vtkCell	Abstract
vtkCellArray	Object
vtkVisibilitySort	Abstract
vtkFieldData	Object
vtkCellLinks	Object
vtkLocator	Abstract

The expanded tree in the Class Viewer of the Builder Window.

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.



Dragging and dropping a class from the class viewer adds the *vtkCylinder* module to the dataflow.

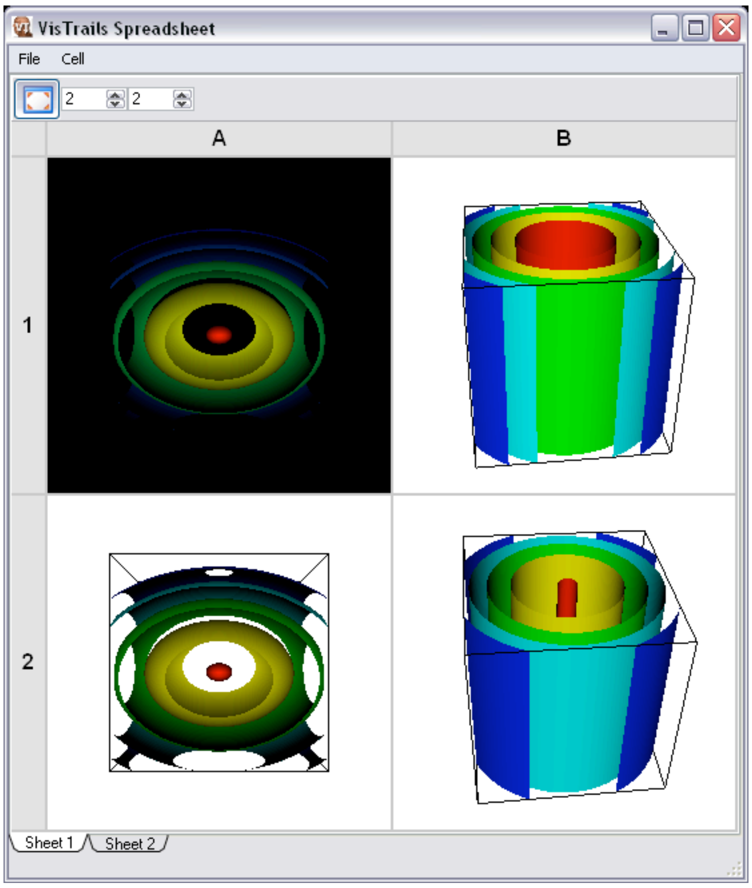
6.4 Creating a Dataflow From Modules

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 spreadsheet resulting from replacing the vtkQuadric module with a vtkCylinder. Note: The two cylinders differ only in a parameter change of their radius.

The input ports of the module will only accept connections from correct output ports. Dropping a connection on a module will cause it to snap to the nearest appropriate port. However, when a module accepts multiple ports of the same type, care must be take as to how the connection is made. The easiest way to ensure proper connectivity is to begin the connection at the module with multiple ports of the same type and drag it to the appropriate endpoint. To determine the exact port to begin at, simply hover the mouse cursor over the port to query and a small note will be displayed with information about the port in question.

6.5 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.

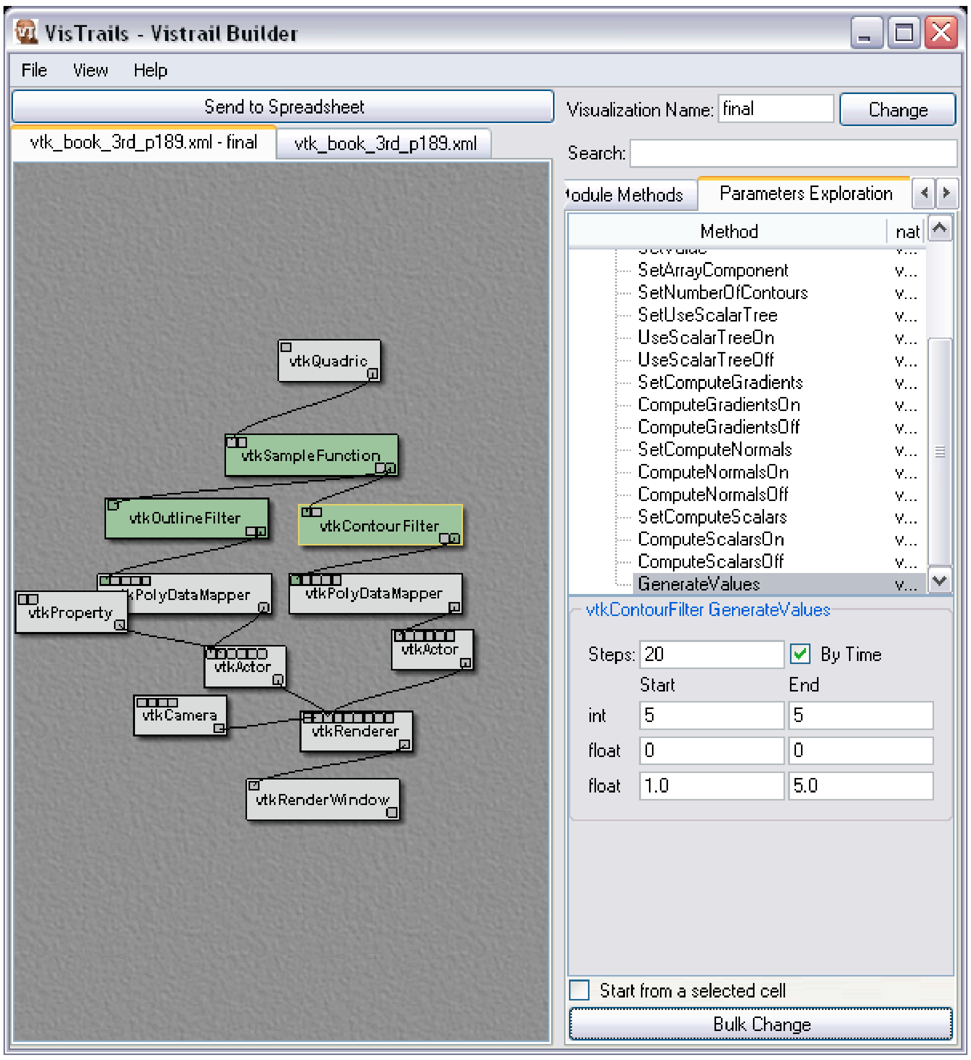
To modify a parameter from its default setting, left click (Mouse Button on Mac) on the word *SetRadius* and continue to hold down the mouse button. *SetRadius* becomes highlighted. With the mouse button still held down, drag the cursor to the area directly below the *Update* and *Update and Send* buttons. 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.

7 Advanced Features

The above instructions outline the basic functionality and use of VisTrails. Below, we discuss the advanced features of the system including Parameter Exploration, Macros, Visual Diffs, and User Tracking.

7.1 Parameter Exploration

Often, an entire space of parameters needs to be visualized in order to fully explore a dataset. VisTrail's Parameter Exploration functionality allows easy and complete exploration of a parameter. To perform an exploration, click on the *Parameter Exploration* tab located above the Class/Module Pane in the Builder. This will open the Parameter Exploration pane. This window in the builder looks and functions like the *Module Methods* pane discussed earlier. However, when a module's method is dragged to the bottom section of the pane, additional parameters are availabed. Each method's parameters contain a *From*, *To*, and *Steps* field. These fields indicate and control the values defining the parameter space to explore. The data is processed through the pipeline once for each parameter value defined by the From, To, and Steps fields. The parameter values are interpolated from *From* to *To* generating *Steps* values. The Parameter Exploration feature is capable of performing n-dimensional explorations where each dimension represents a change to a different parameter in a module.



Visualization Name: final

Method	nat
SetArrayComponent	v...
SetNumberOfContours	v...
SetUseScalarTree	v...
UseScalarTreeOn	v...
UseScalarTreeOff	v...
SetComputeGradients	v...
ComputeGradientsOn	v...
ComputeGradientsOff	v...
SetComputeNormals	v...
ComputeNormalsOn	v...
ComputeNormalsOff	v...
SetComputeScalars	v...
ComputeScalarsOn	v...
ComputeScalarsOff	v...
GenerateValues	v...

vtkContourFilter GenerateValues

Steps: 20 [✓ By Time]

	Start	End
int	5	5
float	0	0
float	1.0	5.0

[ ] Start from a selected cell

The Parameter Exploration Pane. Similar to the Module Methods Pane, this panel allows exploration of a multi-dimensional parameter space.

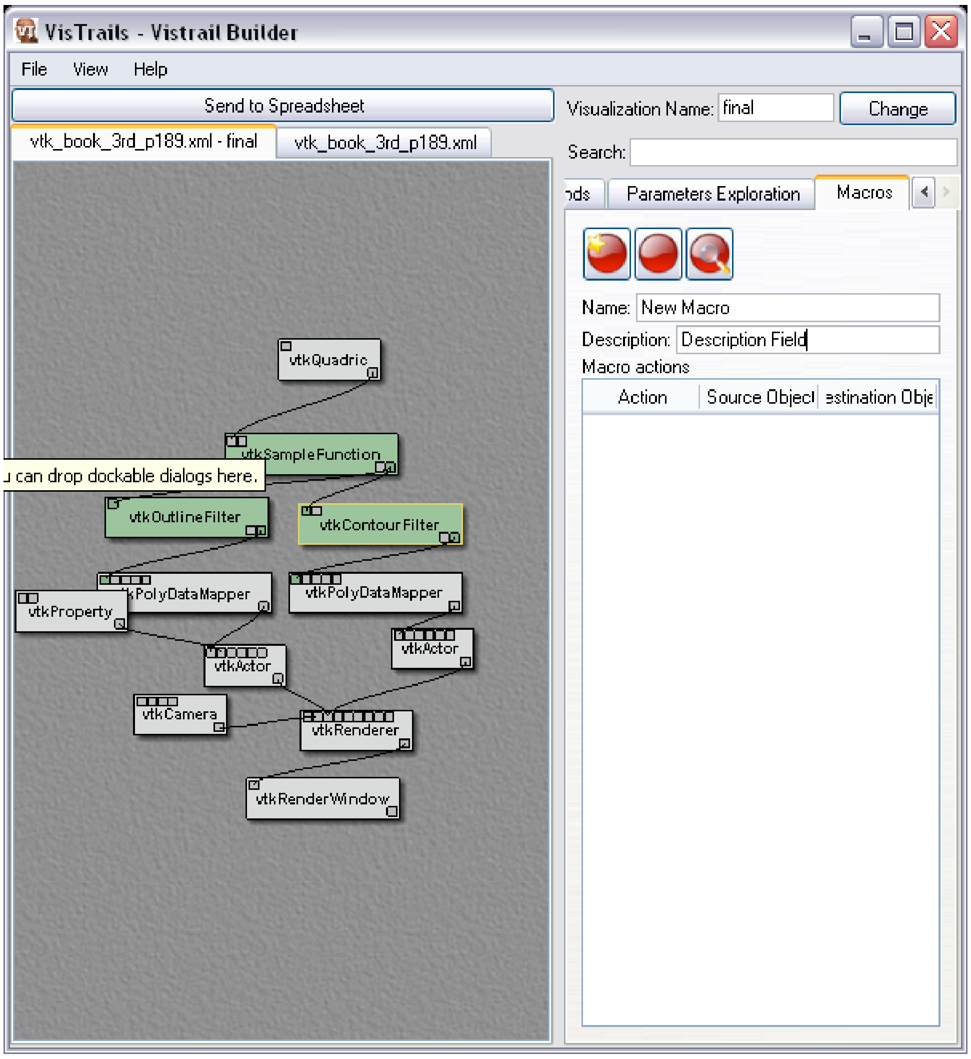
Typically, the processing of the data results in a visualization. Selecting a cell in the Spreadsheet indicates the starting position for the resulting images. This active cell, selected as described earlier, acts as cell (0, 0) when generating the visualizations for a parameter exploration.

By selecting the *By Time* checkbox, the resulting visualizations generated by the Parameter Exploration will be composed in time forming an animation. Once the visualizations are generated, an animation is displayed at the appropriate cell location(s) and can be controlled by pressing the *ctrl* key in the cell you wish to manipulate. The cell that has focus at the time will be overlayed with additional controls. In addition to the standard animation controls, *Play, Pause, and Restart*, VisTrails provides a slider that controls animation speed.

7.2 Macros

Many times, changes made to a specific version of a pipeline are applicable to many others. For example, modifications to the camera position and orientation should be applied not only to the version being changed, but also to any other version in the history tree containing a vtkCamera module. *Macros* are a simple and concise way of making and saving modifications to a pipeline in such a way that they can be automatically applied to another version.

To create a Macro, change to the *Macro Tab* in the same way that the Parameter Exploration or Class Pane tabs were selected. The Macro Tab is substantially different from other panes available.



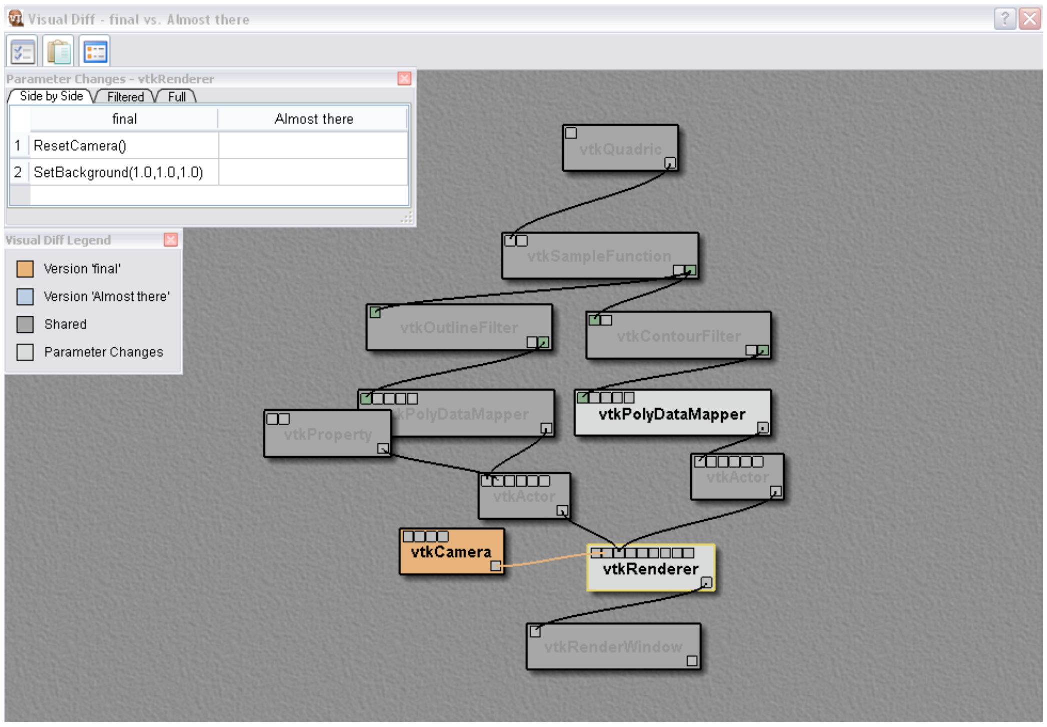
The Macro Pane. This view not only allows macros to be recorded, but displays the Actions they contain.

Macros in VisTrails are best thought of as exactly what they are; a series of actions performed on a pipeline. These actions can be operations such as parameter modification of a module to creation and connection (or deletion and disconnection) of a new or existing module in the pipeline. Thinking of macros in this way gives VisTrails a powerful new tool that can be applied not only to network modification, but to the creation of a set of basic processing pipelines.

To begin recording a macro, open the *Macro Pane* in the VisTrails Builder Window and select the *Create New Macro* button to begin the process of creating a macro. After naming and describing the new macro, press the *Record Macro* button. This will begin recording any actions performed on the pipeline. At this point, all the macro will contain any operations performed on the pipeline being modified. If the pipeline is empty, then any actions created to build the pipeline will be recorded in the macro. If the pipeline is already populated with some modules, any changes of parameter or module creation or deletion operations will be recorded. Once the macro is finished being added to, simply return to the *Macro Pane* and press the *Stop Recording* button. This will finalize the macro and allow it to be saved for later use on other history versions or even on new VisTrails!

7.3 Visual Diffs

One of the most important ways to analyze the differences between two files is by using a Diff. VisTrails provides a *Visual Diff* System to visually compare two pipeline versions in a history. To compare two different versions, simply click and drag one version in the history tree to another. In doing so, a new Diff Window will appear allowing a fast visual comparison between the two versions.



	final	Almost there
1	ResetCamera()	
2	SetBackground(1.0,1.0,1.0)	

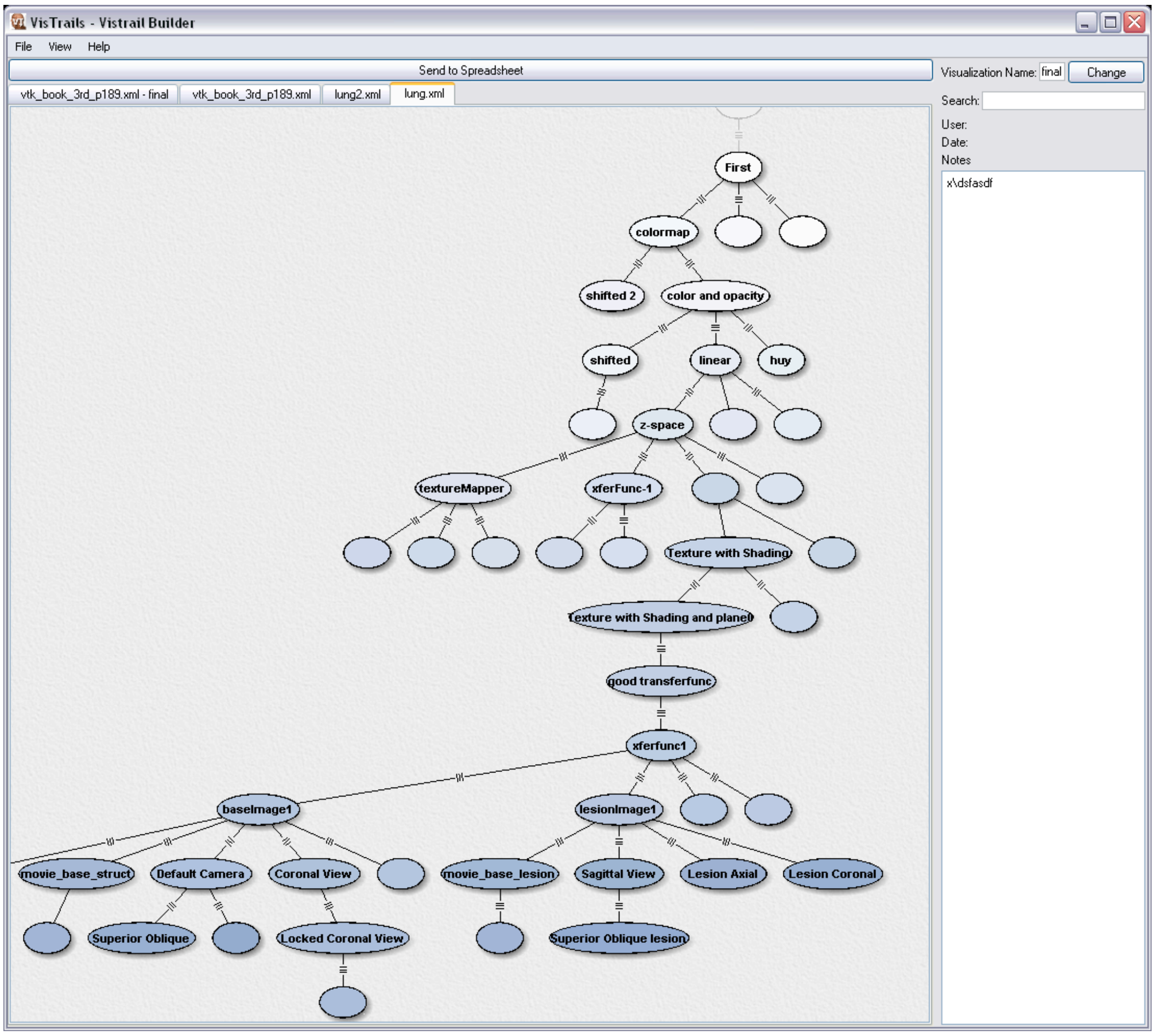
🟧	Version 'final'
🟦	Version 'Almost there'
⬛	Shared
⬜	Parameter Changes

The Visual Diff Window. The legend is shown giving clues as to how the modules are related as well as the window describing the parameterization differences of two modules.

A legend is availabed to denote the various colors in the Diff Window. Darkened modules indicate modules and parameterizations found in both pipelines being analyzed. Light-colored boxes indicate modules that differ only in their parameters. Orange boxes denote modules contained in one pipeline but not in the other.

7.4 User Tracking

Tracking the progression of a VisTrail is often important to analyze the evolution of a data processing pipeline. The most obvious results of VisTrail's workflow management system is the coloring of the Version Tree. The timeline of the pipeline is indicated by the saturation level of each version in the tree. Pipelines most recently used or changed will appear with a higher saturation level than others. In this way, the workflow of a single user or group of users can easily and quickly be analyzed. Furthermore, versions created by other users can be colored differently. This color scheme can be set by the user, but by default, nodes in the version tree colored blue represent versions created by the current user while all other versions are colored differently.



A large version tree developed by a single user. Nodes with a darker color represent versions most recently created.

If additional analyzation is required, each version is also tagged with a specific creation time, user that created it, and can be tagged with text augmenting the description of the version. These fields are also capable of being searched and queried using the *Search* field at the top of the pane. This search functions in the same way as it does in the *Class Pane* with a few small caveats. While searching, the field in which to examine against the *regular expression* search term can be explicitly named by preceeding the search term by the name of the field to search and a colon. *E.g. Date: May*

8 Visualizing Quadrics: A Case Study

In this section we will form a VisTrail based on visualizing a small set of quadrics. The step-by-step instructions are meant to fully illustrate the mechanisms discussed in the previous sections as well as familiarize you with some of the more advanced functionality offered by the system. In this tutorial, we will first create a new VisTrail and add modules to it to form a valid processing pipeline. After completing the pipeline, we will illustrate parameter changes and their effects on the version tree. Finally, a pipeline will be formed allowing us to perform a multi-dimensional parameter exploration.

8.1 Creating a New VisTrail

Upon opening VisTrails, you are presented with an empty spreadsheet and a blank builder window. To create a new VisTrail to begin the visualization or data processing task, simply click on *File* at the top left of the menubar on the Builder Window. From there, click the menu entry *New* to create a new VisTrail. The Builder Window will contain two tabs; the tab <*New1*> contains the Version Tree for the pipeline while the other tab is the Pipeline View for the builder.

8.2 Adding Modules to the Pipeline

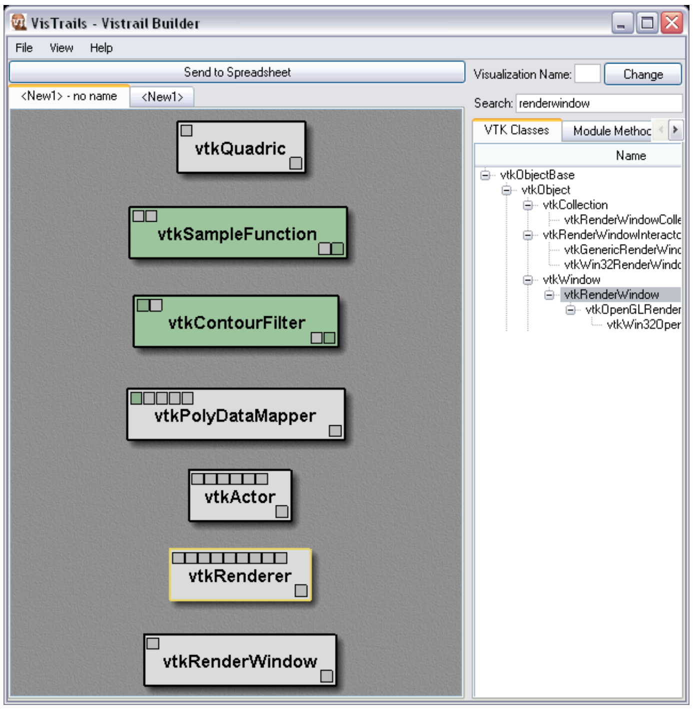
At this point, we must begin to add modules to our processing pipeline. To do this, make sure the *VTK Classes* tab is active in the rightmost pane of the Builder Window. Now, we are ready to search for the approprate classes, or modules, to add to our pipeline. In the *Search Field* type in vtkQuadric. You will notice that the vtk classes will be searched and displayed as the *regular expression* in the field is processed. Once the vtkQuadric module is located in the VTK Classes pane, simply click-and-drag it over to the Pipeline Viewer area. This module generates data in the form of quadric expressions. However, if a data file was to be processed, the appropriate vtk file reader can be inserted in its place by searching for it in the same manner described above. Once the file reader is incorporated in the network, the SetFileName parameter must be set as described in the *Module Parameterization* section below.

To allow the pipeline to function properly, we must add more modules to it. Add the following modules in the way described above:

- vtkSampleFunction

- vtkContourFilter
- vtkPolyDataMapper
- vtkActor
- vtkRenderer
- vtkRenderWindow

After adding these modules, your builder should be similar to the one shown below.



The modules described above are added to the pipeline view, but remain unconnected.

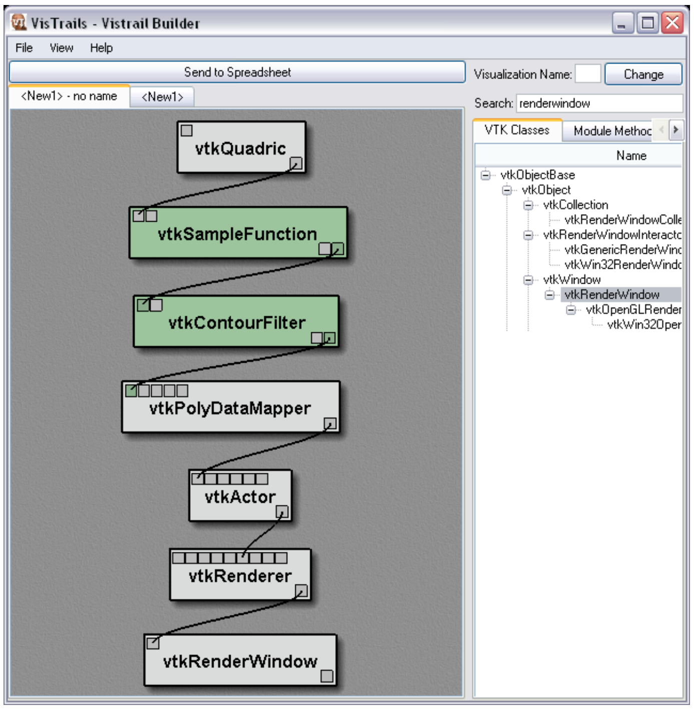
8.3 Connecting Modules in the Pipeline

Now that we have all the modules necessary to process our data, we must connect them properly to fully form our processing pipeline. Each module box has a set of inputs, located in the upper-left hand corner of the box, and a set of outputs, located in its lower-right hand corner. By connecting inputs to outputs we will form a valid VTK pipeline to visualize the data being generated by the vtkQuadric module. In order to connect two modules together, click-and-drag the appropriate output box contained in the module to the module using it as its input. For example, by clicking and dragging the output box of the vtkQuadric module to the vtkSampleFunction module, a connection will be made between the two modules. This connection is indicated by a solid black line. Now that we have a connection between the vtkQuadric module and the vtkSampleFunction module, we must finish connecting our pipeline.

To finish connecting the pipeline add connections between the following modules:

- vtkSampleFunction → vtkContourFilter
- vtkContourFilter → vtkPolyDataMapper
- vtkPolyDataMapper → vtkActor
- vtkActor → vtkRenderer
- vtkRenderer → vtkRenderWindow

After connecting these modules together, your pipeline should look similar to the one shown below.



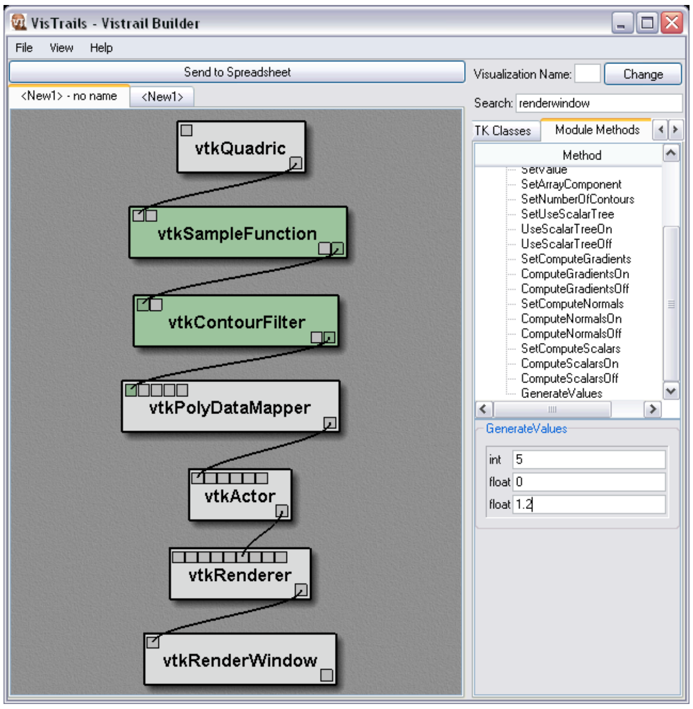
The modules described above are connected to each other in the pipeline view.

8.4 Module Parameterization

Now that we have a valid processing pipeline, we must now set a few parameters in order to properly visualize the data. Parameters for the pipeline are set on a *per-module basis*. So, in order to tune a parameter, select the module containing the parameter to change by left clicking on it. You will notice it is highlighted in yellow when it is selected. Now, once the appropriate module is selected, click on the tab in the rightmost pane labelled *Module Methods*. Once the Module Methods pane is active, a list of all methods supported by this module are displayed. To change a parameter in this module, select the method governing the parameter to be changed, in this case the *Generate Values* method and drag it to the lower section of the pane. This module should have the parameter of type *int* set to the value 5, the first parameter of type *double* set to 0, and the second to the value 1.2.

We also need to set some other parameters. In the vtkQuadric module, the *SetCoefficients* function should have the values 0.5, 1.0, 0.2, 0.0, 0.1, 0.0, 0.0, 0.2, 0.0, 0.0 set. The *SetSampleDimensions* function in the vtkSample-Function module should have all three values set to 50.

At this point, your Builder Window should look similar to the one shown below.



Search: renderwindow

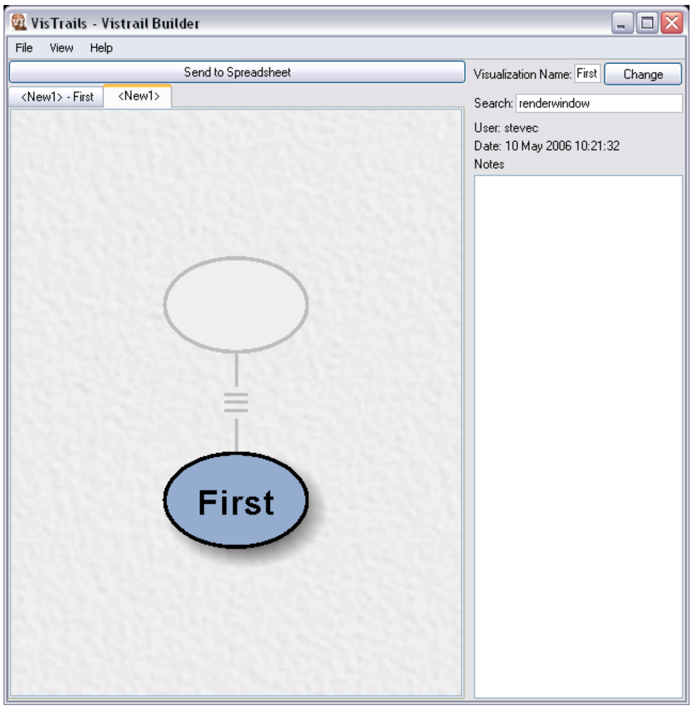
int	5
float	0
float	1.2

One of the modules *Module Methods* panes after parameters have been set in it.

Now that the modules are connected and have a working set of parameters, the pipeline is ready to be visualized. Pressing the *Send to Spreadsheet* button will send the current pipeline with the set parameters to the VisTrails Spreadsheet. After reviewing the visualization, we should create a *Named Version* to capture the changes we have made this far.

8.5 Naming Versions and Saving VisTrails

Before this VisTrail is further altered, we should create a new *Named Version* in the history tree to capture the changes we have made to the pipeline. To do this, simply enter a short descriptive name in the *Visualization Name* field in the upper-right corner of the Builder Window. Pressing the *Change* button commits this change and creates a new version in the history tree. This can be seen by switching to the History View by clicking on the appropriate tab over the Pipeline Builder. The history tree should look something like the one shown below.



The version tree after creating a *Named Version*.

If you want to save the work done up to this point, simply click on the *File* button on the menubar of the Builder Window and select the *Save* or *Save As* option to save the current VisTrail.

8.6 Modifying A Pipeline

VisTrail's *Action-based Provenance* management system records all changes made to a visualization regardless of how minor. This can be seen by performing a single parameter change on our pipeline. A good example of this is a change to the vtkRenderer module's SetBackgroundColor method. Set the background color to white by setting all three of this method's parameters to 1.0. Pressing the *Send to Spreadsheet* results in a visualization equivalent to the previous image with only the background color changing from black to white. Create another new Named Version for this visualization as you did previously. Now, looking at the version history, we can see that another node has been added indicating that actions were performed to generate a new, meaningful visualization. Changes can continue in this manner for any group of changes made to a pipeline including adding modules, deleting modules (Pressing the *Delete* key will delete the active module), or changing a module's parameters.

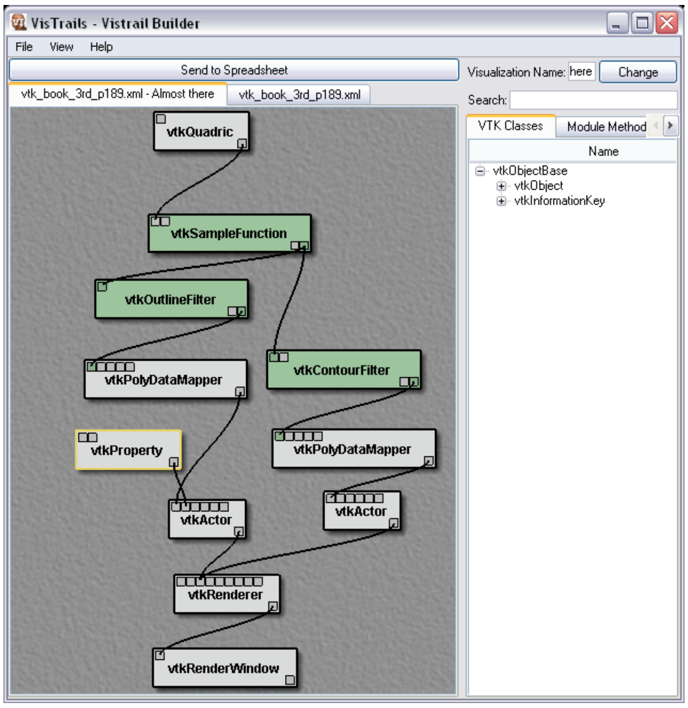
Now, we want to modify the previous version's pipeline. Select the first *Named Version* we created previously in the History View. You will notice that active versions in this view will be highlighted in yellow. Switching to the pipeline builder, we can see that nothing has changed from our original pipeline, and the background color for the vtkRenderer module remains unset. Now, we want to add a bounding box to the visualization. To do this, we must add the following modules in the same manner discussed previously:

- vtkOutlineFilter
- vtkPolyDataMapper
- vtkProperty
- vtkActor

This modules must be connected as follows:

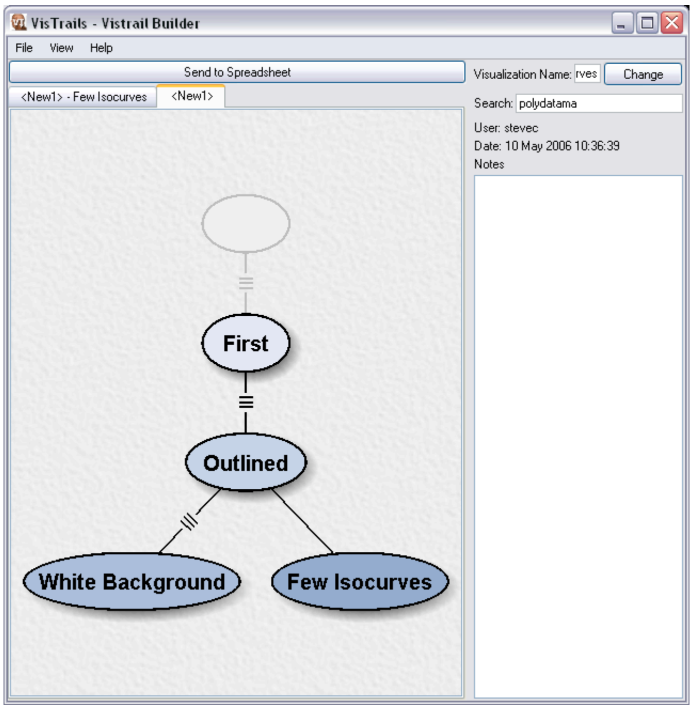
- vtkSampleFunction → vtkOutlineFilter
- vtkOutlineFilter → vtkPolyDataMapper
- vtkPolyDataMapper → vtkActor
- vtkProperty → vtkActor
- vtkActor → vtkRenderer

At this point, your processing pipeline should look like the one below.



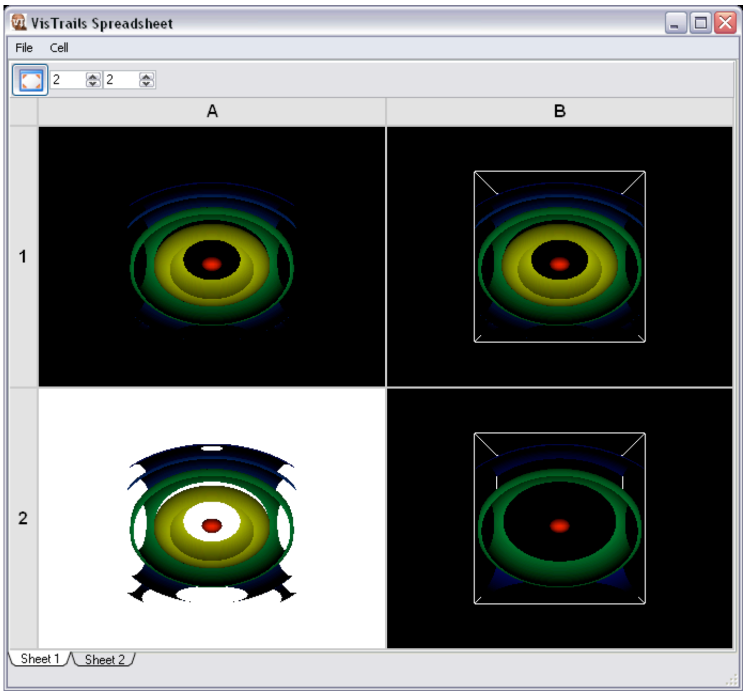
The fully connected visualization pipeline for the quadric example.

By creating another *Named Version* for this altered pipeline, the Version History viewer displays a *branching* history. This immediately allows us to see that this new version we just created came directly from the first pipeline we created instead of the pipeline generating a white background.



The version tree after creating several *Named Versions* descending from a single parent.

At this point, you should have a history tree containing 4 *Named Versions* each capable of producing a visualization. Below is an image of a spreadsheet containing these four different visualizations. To re-create this, simply drag each *Named Version* from the version tree to the appropriate spreadsheet cell.



The spreadsheet containing visualizations of all the *Named Versions* in our history tree.

9 References and Papers

All papers regarding VisTrails can be found at the VisTrails homepage located at: http://www.sci.utah.edu/∼vgc/vistrails/

[3, 6, 1, 2, 4, 5]

References

[1] E. Anderson, S. Callahan, J. Freire, E. Santos, C. Scheidegger, C. Silva, and H. Vo. Visualization in radiation oncology: Towards replacing the laboratory notebook. Technical Report UUSCI-2006-017, SCI Institute–University of Utah, 2006.

[2] L. Bavoil, S. Callahan, P. Crossno, J. Freire, C. Scheidegger, C. Silva, and H. Vo. Vistrails: Enabling interactive multiple-view visualizations. In *IEEE Visualization 2005*, pages 135–142, 2005.

[3] S. Callahan, J. Freire, E. Santos, C. Scheidegger, C. Silva, W. Tyler, and H. Vo. Vistrails: A short tutorial. http://www.sci.utah.edu/ vgc/vistrails/pub/vistrails-tutorial.pdf.

[4] S. Callahan, J. Freire, E. Santos, C. Scheidegger, C. Silva, and H. Vo. VisTrails: Visualization meets Data Management. In *ACM SIGMOD*, 2005. To appear.

[5] S. Callahan, J. Freire, E. Santos, C. Scheidegger, C. Silva, and H. Vo. Managing the evolution of dataflows with vistrails *(Extended Abstract)*. In *IEEE Workshop on Workflow and Data Flow for Scientific Applications (SciFlow)*, 2006. To appear.

[6] S. Callahan, J. Freire, E. Santos, C. Scheidegger, C. Silva, and H. Vo. Using provenance to streamline data exploration through visualization. Technical Report UUSCI-2006-016, SCI Institute–University of Utah, 2006.