The ICE visualization code is divided into four packages, one each for the visualization perspective, generic visualization service code, and one each for the ParaView and VisIt visualization service implementations. The first two center around a single UI piece, with smaller classes geared towards providing specific portions of functionality to it. The other two provide interface implementations and abstract class extensions to the generic visualization API used by the Plot Editor, allowing it to use those modeling programs to plot files.

The current visualization code is well written. Classes are well defined, the code is readable and efficient, and there are generally no problems with design or implementation, other than a few instances noted in the text below. The largest problem is a lack of documentation, with several classes lacking comments or the comments present being insufficient. However, even this is only in a minority of classes. Another concern is that the Visualization Perspective is obsolete, now that the Plot Editor performs the same job with greater extensibility and in a more user friendly way. The perspective should probably be removed from the project. As a minor point of organization, the CSV visualization service may or may not deserve its own bundle instead of a package in org.eclipse.ice.service. The other services are, though on the other hand, the other services are more complex and the service factory implementation uses the CSV service as the default service. Other than this, the code as written is excellent.

Org.eclipse.ice.viz

Org.eclipse.ice.viz

AddFileAction- Contributes the Add File button to the perspective, providing several different options. (The Add____Action classes listed below.) Pressing the button calls the default action, AddLocalFileAction, and the button includes a drop down menu listing the others.

AddFileSetAction- An Action which opens a file dialog, and all selected files are placed in the parent view under a node, which can be clicked to hide or display all files in that set of files. Currently broken, see Bug 471345

AddLocalFileAction- An Action which opens a file dialog, and all selected files are placed in the Visualization File Viewer.

AddRemoteLocalFileAction- An Action which opens a file dialog. The dialog shows files on the machine hosting the connected VisIt session established through the Launch VisIt button (but not connections set up through the preference page.) If the VisIt session is running on the local machine, it calls AddLocalFileAction instead. Selected files are placed in the parent view. (Suggestion: Change UI text to reflect that it is specific to VisIt, not simply adding an arbitrary remote file.)

AddTimeDependentSILOSetAction- An Action which opens a file dialog. The dialog will allow for the selection of a series of .silo files with time steps represented in their names (ie Test001.silo, Test002.silo, etc.) Then a .visit file listing these .silo files in order will be created, and the .visit file will be added to the parent view. (Line 139 has commented out code which should be deleted. It also appears that child and children are created, set, and then never used except within the commented out code, so these local variables should also be removed.)

DeleteFileAction- Contributes the Delete File button to the perspective, allowing the user to delete the selected file from the parent view. (Suggestion: The code is identical to that of DeletePlotAction, except its method takes a ViewPart instead of an IDeletePlotActionViewPart. Since it castes the ViewPart to a VizFileViewer to use the VizFileViewer's removeSelection method, it would be better to make VizFileViewer an IDeletePlotActionViewPart, as DeletePlotAction already calls an IDeletePlotActionViewPart's removeSelection method, and remove the specialized DeleteFileAction.)

DeletePlotAction- Contributes the Delete File button to the perspective, allowing the user to delete the selected plot from the parent view.

IDeletePlotActionViewPart- An interface for classes which remove selected items from a list. (Suggestion: Its name implies that it is an extension of the IViewPart interface, which it is not. It should either be renamed or made to extend IViewPart.) Implemented by CSVPlotViewer and VisitPlotViewer.

PlotEntryContentProvider- A class which allows for other classes to get the names of plot categories from an Entry.

VisualizationPerspective- The Visualization perspective. It registers itself in the Perspectives button on ICE's startup.

VizFileViewer- A view containing a list of files imported into the perspective (using a fileTreeViewer with a ResourceComponent containing the files.) It also contains an add button (using AddFileAction) and delete button (using DeleteFileAction).

Org.eclipse.ice.viz.csv.viewer

AddCSVPlotAction- Contributes the Add Plot button to the perspective, allowing the user to run AddCSVSeriesAction or CreateCSVPlotAction. The default action on pressing the button is AddCSVSeriesAction. (Suggestion: Some of the comments are copied and pasted from AddFileAction and do not reflect AddCSVPlotAction's behavior. These should be changed.)

AddCSVSeriesAction- An Action which allows the user to add a drawn series from the selected CSV plot's file to the plot. It uses a SelectIndependentVarDialog to prompt the user to select an

independent variable from the file, an AddPlotDialog to prompt for the series type, a SelectTimeDialog to select the time step if more than one is available, and a SelectFeatureDialog to prompt for the dependent variable. (Suggestion: SelectFeatureDialog has functionality allowing the user to choose arbitrary variables to plot against each other, but this is never used because the SelectIndependentVarDialog already chose one of the variables. Perhaps SelectFeatureDialog should entirely replace SelectIndependentVarDialog. Also, AddPlotDialog allows for the selection of multiple plot types, but currently all but the first are disregarded. AddCSVSeriesAction should add a new series for each selected series type.)

AddPlotDialog- A dialog box prompting the user to select from the available hard-coded plot types for CSV plots. It allows the selection of multiple types. Also allows the user to cancel the operation.

CreateCSVPlotAction- An action that creates a new PlotProvider from the selected CSV Plot and opens a new view using it to display the plot. It uses a SelectIndependentVarDialog to prompt the user to select an independent variable from the file, an AddPlotDialog to prompt for the series type, and a SelectFeatureDialog to prompt for the dependent variable. (Suggestion: See AddCSVSeriesAction for comments on the use of SelectFeatureDialog and AddPlotDialog.)

CSVDataTableViewer- A view containing the data from a CSV file, formatted as a table. It does not appear to be currently used by ICE.

CSVPlotViewer- A view containing a list of CSV plots created by AddCSVPlotAction (using a TreeViewer with a RescourceComponent containing the plots. It also contains an add button (using AddCSVPlotAction) and a delete button (using DeletePlotAction.) Clicking on one of the plots brings the tab containing the associated CSV Plot Editor to the front, assuming one exists. There are buttons to move to the next or previous plot, and a play button (using PlayAction) to automatically switch to the next plot at a user selected speed, for the purposes of animating a plot over time.

DataTableContentProvider- A content provider that extracts data from a CSV file in a structured way for the CSVDataTableViewer. This class lacks full documentation and is used only by the unused CSVDataTableViewer.

DataTableLabelProvider- A class that extracts the label of a column as a string. It lacks documentation and is used only by the unused CSVDataTableViewer.

PlotTreeContentProvider- A class that holds a map of PlotProviders with their titles, and can extract relevant information from a PlotProvider, PlotTimeIdentifierMapping, or SeriesProvider. For a PlotProvider, this is an array of PlotTimeIdentifierMappings containing its child time steps and the PlotTreeContentProvider saves the PlotProvider by its title for later use. For a

PlotTimeIdentifierMapping, this is the parent PlotProvider, or null if the PlotTreeContentProvider has never been passed that PlotProvider before. For a SeriesProvider, this is two Strings containing its x axis features and y axis features. This class lacks full documentation.

PlotTreeLabelProvider- A class that, when passed a PlotProvider, PlotTimeIdentiferMapping, SeriesProvider, or String, will return a String containing an appropriate label for that object. For a PlotProvider or SeriesProvider, it is the plot/series's title. For a PlotTimeIdentifierMapping, it is the time step. A String is simply returned. The class lacks full documentation.

SelectFeatureDialog- A dialog box which provides a list of independent variables and a list of dependent variables, prompting the user to select which will graphed against which. It can also cancel the operation.

SelectIndependentVarDialog- A dialog box which provides a list of features which could serve as the independent variable for a graphed series, and prompts the user to select one. It can also cancel the operation.

SelectTimeDialog- A dialog box which provides a list of time steps which could be graphed, and prompts the user to select one. It can also cancel the operation.

Org.eclipse.ice.viz.visit

AddVisitPlotAction- Contributes the add plot button for the VisIt Plot Viewer. When pressed, it checks that VisIt is connected and displays an error dialog if it is not. Then, it opens an ExposedCheckTreeDialog to prompt the user for which plot(s) in the file to create. The selected plots are then added to the list. This file also contains the class ExposedCheckTreeDialog, which is a dialog box containing a list of all plots available in a provided VisIt file. The plots are sorted into one node per category. Multiple plots can be selected. The user can select all, deselect all, or select or deselect an entire category at a time. The operation can also be canceled. (Suggestion: Move ExposedCheckTreeDialog to its own file.)

LaunchPythonScriptDialogAction- Contributes the execute Python script button for the VisIt PlotViewer. When pressed, it checks that VisIt is connected and displays an error dialog if it is not. Then, it opens a VisItPythonDialog to allow the user to input Python scripts for VisIt.

LaunchVisItHandler- This class sets up and launches the LaunchVisItWizard, then, if the user did not cancel the operation, opens a VisitEditor, passing the configuration from the LaunchVisItWizard to it.

LaunchVisItWizard- A wizard for opening a VisIt connection. It is essentially a wrapper around LaunchVisItWizardPage.

LaunchVisItWizardPage- A wizard page for connecting to VisIt. It allows for three options. In the first, a local VisIt connection is established. The user is asked for the path to VisIt's folder and, optionally, a port and/or a password for the connection. In the second, VisIt is launched on a remote machine. In addition to the information from the first option, the user must specify the remote hostname and may optionally specify a proxy by url and port number. In the third, ICE connects to an already running VisIt session. This requires a hostname, a port number, and a password, as well as, optionally, a proxy by url and port number. The user can also cancel the operation. The file also contains the following classes and interfaces:

ICheckComposite, an interface for composites which may be associated with a check box and provides an API to check if that box is selected.

PortComposite, an ICheckComposite that contains a text box for specifying a connection port number and a check box for enabling/disabling it.

PasswordCompositte, an ICheckComposite that contains a text box for specifying a password and a check box for enabling/disabling it.

HostComposite, an ICheckComposite that contains a text box for specifying a hostname and a check box for enabling/disabling it.

GatewayComposite, an ICheckComposite that contains texts boxes for the proxy url and proxy port number and a check box for enabling/disabling them. (Suggestion: Possibly move these classes to their own files.)

VisItEditor- An editor for displaying models rendered in a VisIt session. It takes as input the VisItEditorInput provided by a LaunchVisItHandler and sets up a connection based on its connection settings, if that connection does not already exist. It displays the graphics sent to it through its VisIt connection, and uses a VisItMouseManager to provide mouse input for modifying the model to the VisIt session. It registers several listeners with its VisItSwtWidget to handle this mouse input. The mouse wheel zooms, clicking and dragging rotates the model, and clicking and dragging while shift and/or control is pressed moves the model.

VisItEditorInput- An IEditorInput implementation for use with the VisItEditor. It contains information on the VisIt connection to use in the editor, provided by the LaunchVisItHandle.

VisItMouseManager- This class creates and manages a new thread for use in handling mouse input for a VisItEditor. It passes mouse input to the VisItEditor's VisItSwtWidget for processing.

VisItPlotViewer- A view containing a list of VisIt plots created by AddVisItPlotAction (using a TreeViewer with a RescourceComponent containing the plots. It also has an add button (using AddVisItPlotAction), a delete button (using DeletePlotAction), and an execute python script

button (using LaunchPythonScriptDialogAction). It also has buttons to manually step through the time steps of the plot, forwards or backwards, to automatically play the time steps of the plot as an animation, or to stop the animation. There is also a combo box displaying the plot categories available for the currently drawn plot. Changing this combo box or double clicking on a new plot causes the VisIt session to draw a new model accordingly.

VisItPythonDialog- A dialog which provides a console to send the connected VisIt session Python commands to parse. Output from VisIt is also displayed in the console. A “load from file” button to import scripts from a file is also available.

Icons

The following icons are available in this bundle:

Add.png, a green plus sign.

Delete_X.png, a red X.

Launch.png, a sheet of paper

Plugin.xml

The plugin provides the VisItEditor, an editor for VisIt plots

The plugin specifies the Visualization Perspective. This perspective has a VizFileViewer in the upper left, and a CSVPlotViewer and VisItPlotViewer stacked on top of each other in the center left.

The plugin adds the “Launch VisIt” button to Eclipse’s toolbar. This button runs LaunchVisItHandler.

Org.ecplise.ice.viz.service

Org.eclipse.ice.viz.service

AbstractVizPreferenceInitializer- An extension of AbstractPreferenceInitializer which adds an IPreferenceStore class variable to hold the preferences for a service and a getter method for that variable.

AbstractVizPreferencePage- An extension of FieldEditorPreferencePage that automatically sets the preference page name based on the associated IVizService and provides a getter method for the IVizService.

AbstractVizService- An abstract implementation of IVizService. It adds an IPreferenceStore to contain the service's preferences and a Set of Strings detailing the supported file extensions. It also includes getter methods for these variables and an implementation of createPlot that checks the URI argument's extension and throws an error if the service does not support it.

BasicVizServiceFactory- An IVizServiceFactory. It contains a preferenceStore for the factory's preferences. Upon registering an IVizService, it registers the PlotEditor as the default editor for files with that service's supported extensions.

IPlot- An interface for plots. It contains a map of plot types available for the plot, as well as functions to get the number of axes, get and set the plot's editable properties, get the data source and source host, and check if the source is remote. It is implemented by MultiPlot.

IVizService- An interface for visualization services. It has functions to get the service's name and version, get and set the connection properties, connect to the service, disconnect from the service, and use the service to create an IPlot. It is implemented by AbstractVizService.

IVizServiceFactory- An interface for a factory for IVizServices. It has functions to register and unregister a service, get the names of services, get a service, and get the default service. It is implemented by BasicVizServiceFactory.

MultiPlot- An abstract implementation of IPlot which allows for the same IPlot to be drawn in multiple parent Composites. It contains the used IVizService, a URI for the data source, and a map of Composites to their child PlotRenders. It has functions to create and update PlotRenders.

PlotEditor- An editor which displays a visualization model using an arbitrary IVizService. It takes as input a fileEditorInput, retrieves the appropriate IVizService from a VizServiceFactoryHolder based on the input's file extension, and uses it to draw a plot. In the case that a given extension can be handled by more than one available IVizService, it creates a PlotEditorDialog to prompt the user for which one to use. It includes a toolbar which can change the plot's category and type or close the editor. (Suggestion: Some code is commented out and should be removed.)

PlotEditorDialog- A dialog box that prompts the user as to which IVizService to use. It takes as input a list of names, displays them in a combo box, and provides an OK button for the user to finalize their selection.

PlotEditorInput- An IEditorInput for PlotEditors. It contains an IPlot.

PlotRender- This class creates and manages a composite in which to display a drawn IPlot. In case of error, it will create an error image along with text describing the error in place of the plot.

VizPerferenceInitializer- An extension of AbstractVizPreferenceInitializer. It overrides initializeDefaultPreferences to set autoConnectToDefaults to true, setting the default behavior of the service to automatically connect to a default connection if one is available.

VizPreferencePage- An extension of AbstractVizPreferencePage which manages the main Visualization Preferences page. It contains a field to toggle whether or not to automatically connect to visualization services' default connections on startup.

Org.eclipse.ice.viz.service.connections

ConnectionAdapter- An abstract implementation of IConnectionAdapter. It provides default implementations of many functions to give simple functionality and console messages. Most importantly, it implements connect and disconnect, allowing it to use its connection and track the ConnectionState.

ConnectionManager- A class which maintains an IConnectionAdapter along with a list of associated IConnectionClients and a ConnectionTable of connection properties. IConnectionClients added to the manager are automatically connected to the ConnectionManager's IConnectionAdapter and their IConnectionAdapter is set to null when they are removed from the ConnectionManager. When the connection properties in the ConnectionTable are changed, the IConnectionAdapter connects, disconnects, or disconnects and reconnects as appropriate.

ConnectionPlot- An IConnectionClient that extends MultiPlot. It has an IConnectionAdapter to connect it to a remote visualization service, along with getter and setter methods for it and a function to alert the ConnectionPlot that the connection has been updated. Also, its setDataSource implementation checks the file's validity before setting it as the data source.

ConnectionPlotRender- An extension of PlotRender for use with a ConnectionPlot. When it encounters an error during rendering, it includes a link to the connection's preference page along with the normal PlotRender error message. It validates the connection before rendering the plot.

ConnectionState- An enum of states for a connection. It includes Connecting, Connected, Failed, and Disconnected.

ConnectionTable- An IKeyManager that extends TableComponent. The table lists connections along with a key (the connection name). The information stored for each connection is the connection name, the host, the port number, the host OS, the file path to the service, the username, and the password. (Suggestion: The class's documentation of the table row template does not match the actual implementation. The documentation needs to be updated.)

IConnectionAdapter- An interface for wrapping a connection. It has functions to connect and disconnect (either blocking or non-blocking), its properties, its key, its state, its host, its port, getter and setter methods for the connection's properties, and a function to check if it is a remote connection.

IConnectionClient- An interface which extends IUpdateableListener. It allows the class to associate itself with an IConnectionAdapter. It has a function to set the IConnectionAdapter.

IKeyChangeListener- An interface for a listener for an IKeyManager. It has a function to update the IKeyChangeListener when a key has been changed, specifying its old and new values.

IKeyManager- An interface which manages a set of key Strings. It has functions to check if a key is available, get all keys, iterate to the next key in the order, and to register and unregister IKeyChangeListeners.

KeyEntry- An extension of Entry. It is an entry that is managed by an IKeyManager. When the KeyEntry's value is set, it claims that value as a key in the IKeyManager or gives an error message if the value is invalid as a key or that key already exists in the IKeyManager.

KeyEntryContentProvider- This class standardizes the interaction between an IKeyManager and KeyEntry. It can get the allowed value types, the next key for use as a default value, and a list of all still available keys from the IKeyManager for the KeyEntry.

Launcher- This class creates a window containing a ConnectionTable. It lacks full documentation and the window is named "VisIt Tester." It appears to be test code, and should either be made into a formal SWTBot test or be removed.

PortEntry- An Entry that only accepts integers within a given range, as specified by the PortEntry's PortEntryContentProvider.

PortEntryContentProvider- A content provider for a PortEntry. It forces the PortEntry to only accept ranges of integers between 1024 and 65535 as valid value ranges.

SecretEntry- An extension of Entry that, by default, sets the Entry's secretFlag to true.

Org.eclipse.ice.viz.service.csv

CSVData- An implementation of IData. It holds information for a CSV data point. It holds the point's position, value, uncertainty, units, the feature it belongs to, and the tolerance for use in double compassions.

CSVDataLoader- A class for loading a CSV file into A CSVDataProvider. After extracting features, uncertainty, and units from the file header, if available, each value within the file is used to

create a CSVData. A set of multiple files can be loaded simultaneously, placing all the CSVData into a single CSVDataProvider. (Suggestion: The load and loadAsFileSet functions appear to have duplicate code. There should be a single function for extracting header information and another for creating a CSVData, and these should be called by load and LoadAsFileSet instead of having copy and pasted code.)

CSVDataProvider- A holder class to contain CSVData in a structured way. It contains a map of times to a map of features to CSVData data points. A default time is used if none is provided. It contains the time units, a data source, the current time, and a list of independent variables. Data can be retrieved for a particular feature at a particular time. There are also functions for returning all time steps for which a give feature has data, a list of all features, the total number of time steps which have data, the uncertainties of all data at a given time for a given feature, all features present at a given time, and all time steps which have data.

CSVPlot- An IPlot for use with the CSVVizService. It has the plot's data source, a CSVDataProvider to contain the plot's data, and a map of drawn plots. It takes as input a URI to a csv file, and loads the file's data into a CSVDataProvider. It can then draw a plot in a DrawnPlot. This file also includes the DrawnPlot class. DrawnPlot creates a plot based on data from a CSVDataProvider. It also has functions to clear the plot, or add or remove a series from it. (Suggestion: As per a TODO comment, CSVPlot should extend MultiPlot and make use of a PlotRender instead of DrawnPlot.)

CSVPlotEditor- This class is an editor for CSV files. It displays a plot based on the data from a PlotProvider. It also includes a slider to allow the user to change which time step out of a series of plots to display. This class lacks full documentation.

CSVVizService- An extension of AbstractVizService. It handles .csv files by creating a CSVPlot from the input file and loading its data.

PlotProvider- A content provider for a CSVPlot. It holds a list of SeriesProviders, each of which provides the content for a single series. These SeriesProviders can be accessed by the time step they are assigned to and the list of time steps with an associated SeriesProvider can also be retrieved. It also contains information on the plot title, axes titles, time units, and whether or not the plot is a contour.

PlotTimeIdentifierMapping- This class holds a title for a plot along with the time for a series. It lacks full documentation.

SeriesProvider- A wrapper for a CSVDataProvider. It contains the time step of the series, the series' title, the names of the series' x and y features, and the series' type.

Org.eclipse.ice.viz.service.internal

VizServiceFactoryHolder- A holder class for an IVizServiceFactory. It can register, unregister, and get an IVizServiceFactory.

Org.eclipse.ice.viz.service.paraview.web

HttpParaViewWebClient- An IParaViewWebClient. It provides a way for ICE to communicate with a ParaView http web client. It can send a method name and JsonObject to a remote ParaView client, and receive a JsonObject in response, returning a new JsonObject if it receives an improperly formatted response. It can also send rendering instructions, an event, or call a method and receive a Future<JsonObject> in response. This class lacks full documentation.

IParaViewWebClient- An interface for communicating with a ParaView web client. It has functions to connect or disconnect and functions to send a rendering instruction, event, or method call to the ParaView client. It has no documentation.

Org.eclipse.ice.viz.service.preferences

CustomScopedPreferenceStore- A class for managing IEclipsePreferences. It can check if a node exists, retrieve a node, remove a node or value, and save changes to the preferences. It can also get, set, and remove encrypted preferences as ISecurePreferences.

DynamicComboFieldEditor- An editable combo box. Its allowed and currently set values can be programmatically changed.

EntryCellContentProvider- An ISecretContentProvider. It is a content provider that sets a cell's contents based on the value of an Entry and the tooltip to the Entry's description. It can provide a combo box to the cell if the Entry has a discrete list of allowable values, and obscure the cell's text for secret Entries.

EntryCellEditingSupport- A wrapper for an EntryCellContentProvider, providing methods to change the underlying Entry's value. This class lacks full documentation.

ISecretCellContentProvider- An interface that provides functions to check if a cell's contents are secret and provide a secret character (default "*") to display in place of the cell's actual value.

TableComponentCellContentProvider- A implementation of ICellContentProvider that takes an ICellContentProvider and an index. It overrides the ICellContentProvider methods to take a list as input, then call the underlying ICellContentProvider's method on the object in the list specified by TableComponentCellContentProvider's index. This class lacks full documentation.

TableComponentCellEditingSupport- An extension of EntryCellEditingSupport that takes an EntryCellEditingSupport and an index. It overrides the EntryCellEditingSupport methods to take a list as input then call the underlying EntryCellEditingSupport's method on the object in the list specified by TableComponentCellEditingSupport's index. This class has no documentation.

TableComponentComposite- A composite for displaying a table of content from a TableComponentContentProvider. It also provides buttons to add or delete rows. This class lacks full documentation

TableComponentContentProvider- A class that links a TableComponent with a view, updating the view when the TableComponent changes. It uses EntryCellContentProviders to populate the table's content, and EntryCellEditingSupport to add editing functionality.

TableComponentPreferenceAdapter- A class which handles the interaction between a CustomScopedPreferenceStore and the TableComponent which will display the store's preferences. It can load preferences from the CustomScopedPreferenceStore into the TableComponent, save the TableComponent's content into a CustomScopedPreferenceStore, or clear the TableComponent's CustomScopedPreferenceStore of data.

VizConnectionPreferencePage- An extension of AbstractVizPreferencePage. This class adds a ConnectionTable as well as keeps track of keys from connections loaded from the CustomScopedPreferenceStore. It provides a UI including a DynamicComboFieldEditor populated with default connections and a TableComponentComposite to allow editing of preferences. This class lacks full documentation.

OSGI-INF

vizFactory- This package provides the BasivcVizServiceFactory as a IVizServiceFactory service. It consumes IVizService services with the BasicVizServiceFactory.

vizServiceFactoryHolder- This package consumes IVizServiceFactory services with the VizServiceFactoryHolder.

(Suggestion: CSVVizService does not advertise itself as an IVizService, which breaks the pattern established by the other IVizServices. It is instead added into BasicVizServiceFactory by hardcoding. BasicVizServiceFactory should consume CSVVizService as normal, and set the default service by checking for CSVVizService's name when registering a new service.)

Plugin.xml

This package provides a Visualization preference page, implemented by VizPreferencePage and initialized by VizPreferenceInitializer.

Org.eclipse.ice.viz.service.paraview

Org.eclipse.ice.viz.service.connections.paraview

ParaViewConnectionAdapter- An extension of ConnectionAdapter<VtkWebClient>. It overrides the functions for opening and closing a connection and setting connection properties to work with VtkWebClient.

Org.eclipse.ice.viz.service.paraview

ParaViewPlot- An extension of ConnectionPlot<VtkWebClient>. It provides a ParaView specific implementation for creating a PlotRender and finding the plot types.

ParaViewPlotRender- An extension of ConnectionPlotRender<VtkWebClient>. It has a IConectionAdapter<VtkWebClient> with a connection to the ParaView client which is rendering its plot. It displays the model from the ParaView client in an InteractiveRenderPanel and has a toolbar which allows the user to change the plot type and representation. This class lacks full documentation.

ParaViewPreferenceIntiializer- An extension of AbstractVizPreferenceInitializer. It has no actual code, only a stub function definition containing commented out code.

ParaViewPreferencePage- An extension of VizConnectionPreferencePage for ParaView.

ParaViewVizService- An extension of AbstractVizService. It has a ConnectionManager to hold the connections to ParaView clients. It creates a plot directly from an URI, automatically passing the plot to a connection.

OSGI-INF

This package provides a ParaViewVizService as an IVizService.

Plugin.xml

This package provides a ParaView preference page. This is initialized by ParaViewPreferencePage and initialized by ParaViewPreferenceInitializer.

Org.eclipse.ice.viz.service.visit

Org.eclipse.ice.viz.service.connections.visit

VisItConnectionAdapter- An extension of ConnectionAdapter<VisItSwtConnection>. It has VisIt specific implementations to create a connection and set connection properties.

VisItConenctionTable- An extension of ConnectionTable. It has a connection template for use in connecting to VisIt.

Org.eclipse.ice.viz.service.visit

ConnectionPreference- An enum that holds properties for a VisIt connection. It is not used.

VisItMouseManager- A class which manages the listeners for a VisItPlotRender. Listeners exist for clicking and moving the mouse, for scrolling the mouse wheel, and for keyboard button presses. Clicking and dragging rotates the model, scrolling the wheel zooms, and the clicking and dragging while the control and/or shift key is pressed moves the center of the model.

VisItPlot- An extension of ConnectionPlot<VisItSWTConnection> It has VisIt methods for creating a plot render and finding plot types.

VisItPlotRender- An extension of ConnectionPlotRender<VisItSwtConnection>. It creates a composite featuring a VisIt model from a VisItConnectionAdapter. It also populates a context menu with plot categories and types the current model can switch between. (Suggestion: getPreferenceNodeID returns ParaView's preference id, not VisIt's. This should be fixed.)

VisItPreferenceInitializer- An extension of AbstractVizPreferenceInitializer. It has no actual code, only a stub function definition containing commented out code.

VisItPreferencePage- An extension of VizConnectionPreferencePage with functions overridden to be specific to the VisIt service.

VisItVizService- An extension of AbstractVizService. It has a ConnectionManager to hold connections to VisIt clients. It creates a plot directly from an URI, automatically passing the plot to a connection.

Org.eclipse.ice.viz.service.visit.widgets

BinarySearchTree- A node in a binary search tree. It contains an index for the node, the node's contents (a double), and left and right children. The root node also has a list of all values stored in the tree. The tree can be searched for a given value, returning either the value nearest to it or the index of the node containing the value nearest to it.

ComboDialog- This class is a dialog containing a combo box. It has a list of allowed values, and can be set to either editable or read only. If editable, the user can write in their own input, and there are functions to validate the user's text against a set of allowed values. The operation can be canceled, in which case the value of the dialog is set to null.

TimeSliderComposite- This class is a composite with controls for displaying a plot's time steps. It consists of a bar, buttons, and a text field. The bar is a timeline of the available time steps. The user can drag the cursor to set when in the timeline to display, using a BinarySearchTree to snap to the nearest time step. The text field allows the user to explicitly enter a time step to display. There are forward and back buttons to display the next or previous time step, respectively. There is also a play button, which causes the display to automatically advance to the next time step and, if pressed again, stops the playback. An option button provides a list of framerates to allow the user to control how fast it moves between time steps and also can open a ComboDialog which allows the user to input a custom framerate.

Icons

Nav_backward- A yellow arrow pointing left. Used for the TimeSliderComposite's back button.

Nav_forward- A yellow arrow pointing right. Used for the TimeSliderComposite's forward button.

Nav_go- A green triangle pointing right. Used for the TimeSliderComposite's play button.

Resume_co- A vertical yellow bar and green triangle pointing right. Unused.

Suspend_co- Two vertical yellow bars. Used for the TimeSliderComposite's pause button.

Thread_obj- A green triangle pointing right with an orange circle to its upper right. Used for the TimeSliderComposite's options button.

OSGI-INF

vizService.xml- This package advertises VisItVizService as an IVizService.

Plugin.xml

This package provides the VisIt preference page. It is implemented by VisItPreferencePage and initialized by VisItPreferenceInitializer.