Manual for MOOSE Users

© Copyright 2015. All Rights Reserved.

OAK RIDGE NATIONAL LABORATORY
MANAGED BY UT-BATTELLE FOR THE US DEPARTMENT OF ENERGY

ICE Manual for MOOSE Users

Jay Jay Billings, [1a,b] and Alexander McCaskey, [a]Anna Wojtowicz, [a]Jordan H. Deyton[a]

[a]Oak Ridge National Laboratory PO Box 2008 MS 6173 Oak Ridge, TN 37831 USA
[b]The Bredesen Center for Interdisciplinary Research and Education, The University of Tennessee, Knoxville, TN 37996 USA

June 8, 2015

[1]Corresponding author. Telephone: +1 865 241 6308, Email: billingsjj@ornl.gov

Contents

Chapter 1

Using MOOSE with ICE

1.1 Introduction

This document is designed to outline the basic steps of setting up and using the MOOSE plug-ins in ICE. ICE currently supports four MOOSE-based applications: MARMOT, BISON, RELAP-7 and RAVEN. Although this tutorial was created with BISON in mind, the steps for using ICE with MARMOT, RELAP-7 and RAVEN are the same.

There are two different tasks for the input generation and launching of MOOSE products within ICE:

- **MOOSE Model Builder** - Generates a custom input file necessary to launch a MARMOT, BISON, RELAP-7 or RAVEN job.
- **MOOSE Launcher** - Initiates the MARMOT, BISON, RELAP-7 or RAVEN codes to run on a local or remote system using the files generated from the *MOOSE Model Builder*. Also includes the option to run a custom MOOSE-based application.

Any problems should be reported directly to the ICE team by sending an email to the project list, `ice-dev <at> eclipse.org`, or following the instructions to report bugs on our Bugzilla site https://bugs.eclipse.org/bugs/describecomponents.cgi?product=Ice.

1.1.1 Installation and Configuration

Follow the instructions in the Getting ICE article at wiki.eclipse.org/ICE article to download and install the latest version of ICE on your system.

1.1.2 Prerequisites

You should have the MOOSE environment installed, either your local or remote machine. Instructions for installing MOOSE can be found http://mooseframework.org/getting-started/.

Additionally, you should have MARMOT, BISON, RELAP-7 or RAVEN compiled and ready to run. Contact the development teams of these projects on the rules and regulations for obtaining these codes.

1.2 MOOSE Perspective

ICE supports numerous plugins from different areas of science, and as a result it can sometimes be difficult for new users to make sense of all the different windows, panels and tabs in the workbench. To address this issue for MOOSE users, ICE now includes a *MOOSE Perspective* which pares down UI components to only those that are necessary for MOOSE-based plugins.

To access the *MOOSE Perspective*, use the the ICE toolbar at the top and navigate to:

Window > *Open Perspective* > *Other...*

Select *MOOSE* in the window that pops up and click *OK*. Alternatively, you can also access the same pop-up menu by clicking the *Open Perspective* button in the upper right-hand corner of the ICE workbench.

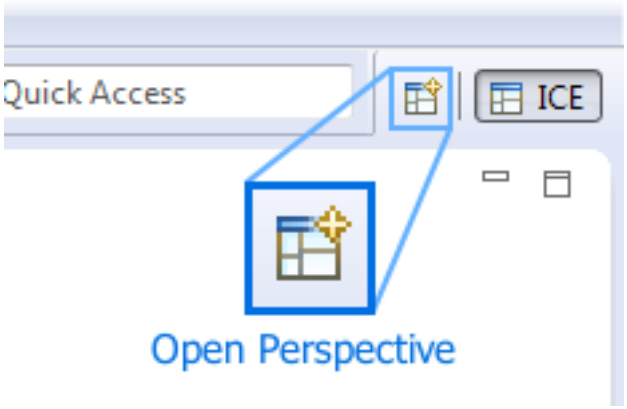


Figure 1.1: The Open Perspective button in ICE for switching to other perspectives such as MOOSE or Visualization.

Once the MOOSE Perspective opens, you should notice the workbench now contains fewer UI components, and resembles something like this:

While it's not necessary to switch to the MOOSE Perspective to use MOOSE-based plugins, it has been included for convenience.

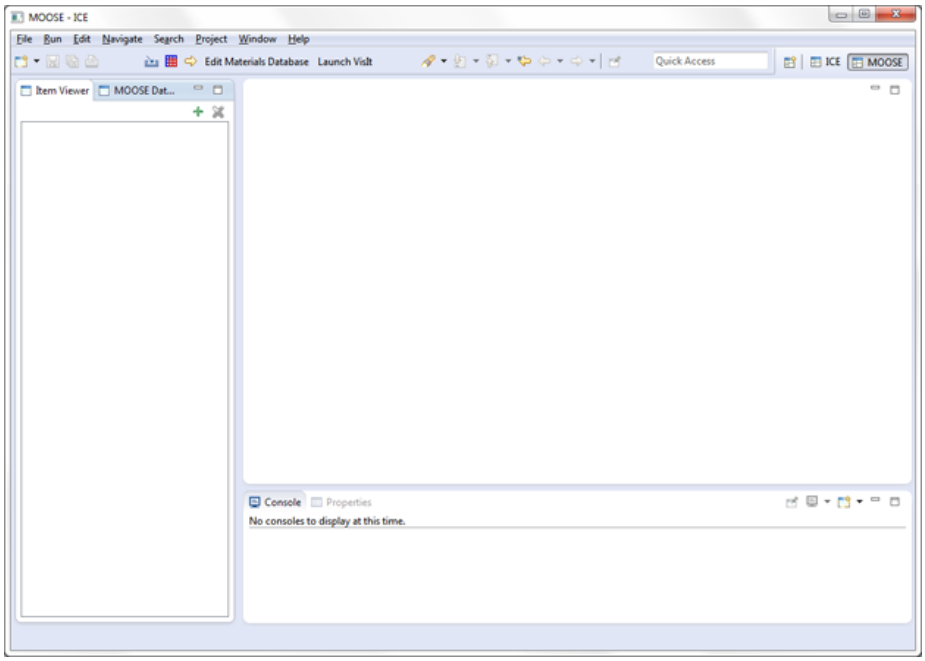


Figure 1.2: The MOOSE Perspective in ICE.

1.2.1 Generating YAML and Action Syntax Files

Like any MOOSE-based GUI, ICE relies on generated YAML and action syntax files to specify the rules of creating input files. To simplify this task, ICE includes a utility that will generate these files (either locally or remotely), clean them up, and place them in the appropriate local directory for ICE to use. All that is required of the user is to specify where and on which machine their MOOSE codes are installed.

This task must be completed at least once for every installation of ICE before the *MOOSE Model Builder* can be used. It's also a good idea to periodically re-run this utility to keep ICE's YAML and action syntax files in sync with any updates to your MOOSE-based codes.

To use this utility, begin by clicking the document-like button in the ICE toolbar:

This will prompt a wizard to pop up requiring a few pieces of information:

- **Hostname**
  This can be a local or remote machine.
- **Execution Path**
  Enter the fully-qualified path to the "trunk" directory of your machine's MOOSE installation. For example, if I have BISON on a machine with the following structure:

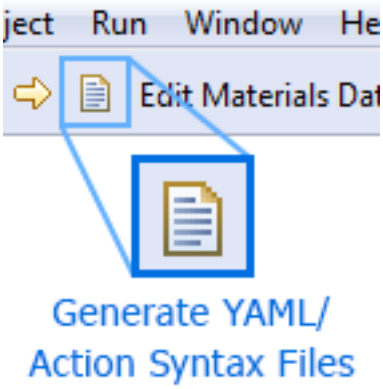


Figure 1.3: The Generate YAML/Action Syntax Files button in the ICE Toolbar.

`/home/user/trunk/bison/bison-opt`

I would set the execution path in ICE to be:

`/home/user/trunk`

If you are launching on a remote machine, also be sure that you have appropriate privileges for the execution path.

- **Login Credentials**
  Your username and password (if required) on the host machine.

Once you have filled out the information, click *Finish*. This will launch a script on your designated machine that checks which MOOSE-based codes you have installed, generates the appropriate files, removes any extraneous header/footer text, and places them in your

`/home/user/ICEFiles/default/MOOSE`

directory where ICE can reference them later.

1.2.2 Creating Input

To create an input file for a MOOSE problem, there are two ways you can get started: **creating an input file from scratch**, or **importing an already existing file to modify**.

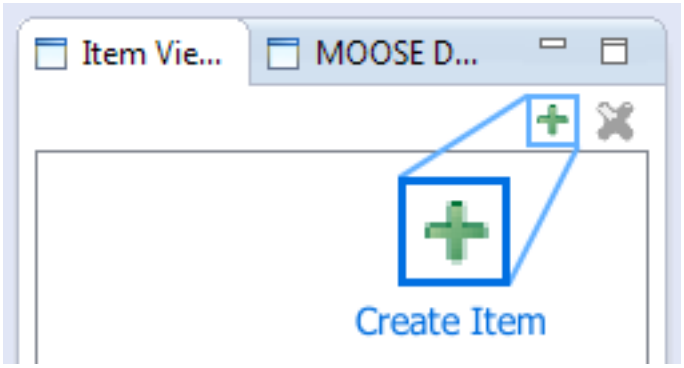


Figure 1.4: The Create Item button in the ICE Item Viewer toolbar.

Creating an Item

If you'd like to start from scratch, begin by clicking the green + button in the *Item Viewer*, located on the left-hand side of the ICE workbench.

This will prompt a window to pop up; select the *MOOSE Model Builder* Item, and click *Finish*.

Alternatively, if you'd like to import an already existing `*.i` file to modify, click the yellow item import arrow located at the top of the ICE workbench.

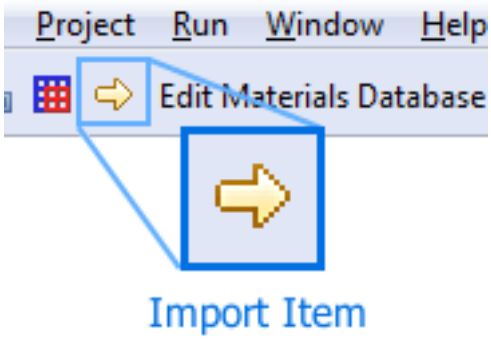


Figure 1.5: The Import Item button in the ICE toolbar.

A wizard will pop up prompting you to specify two things: the `*.i` file you'd like to import from your filesystem, and what kind of *Item* you'd like to import it into.

Use the *Browse...* button to select your input file, and set the item type to *MOOSE Model Builder*. Click *Finish* when you're done and ICE will import the file data.

Regardless of how you decide to begin creating your input file, once the *MOOSE Model Builder* loads, you will see a workbench that looks like the

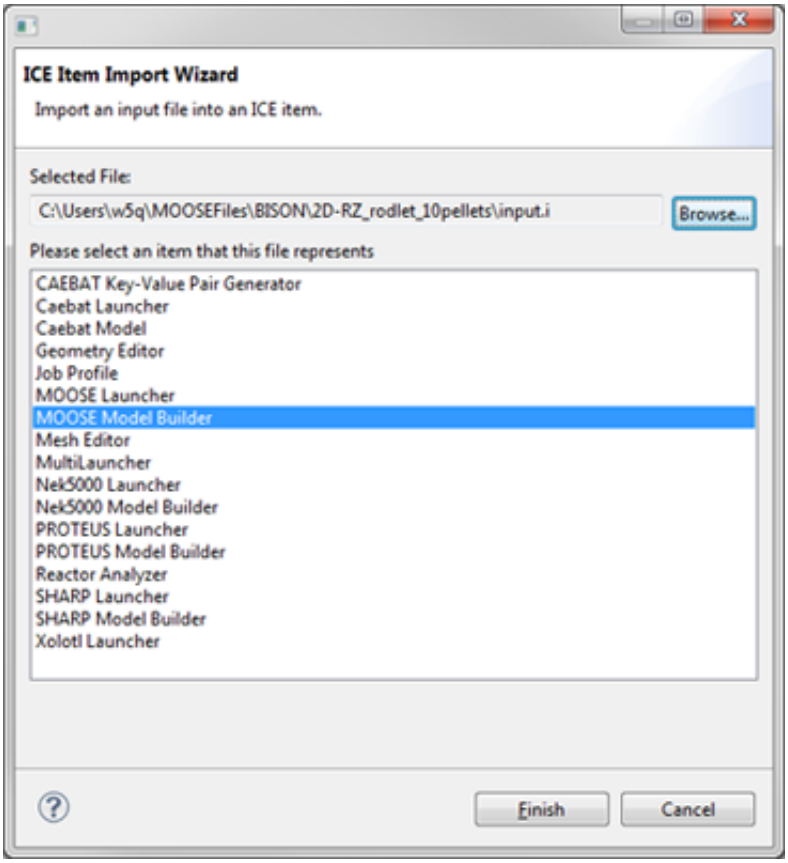


Figure 1.6: The ICE Import Item Wizard. Select MOOSE Model Builder and a input file to import a fully populated MOOSE Model Item.

following. Make note of the three tabs highlighted: the *Tree View*, *Mesh* and *Properties* tabs, as we'll be using them quite a bit in the following section.

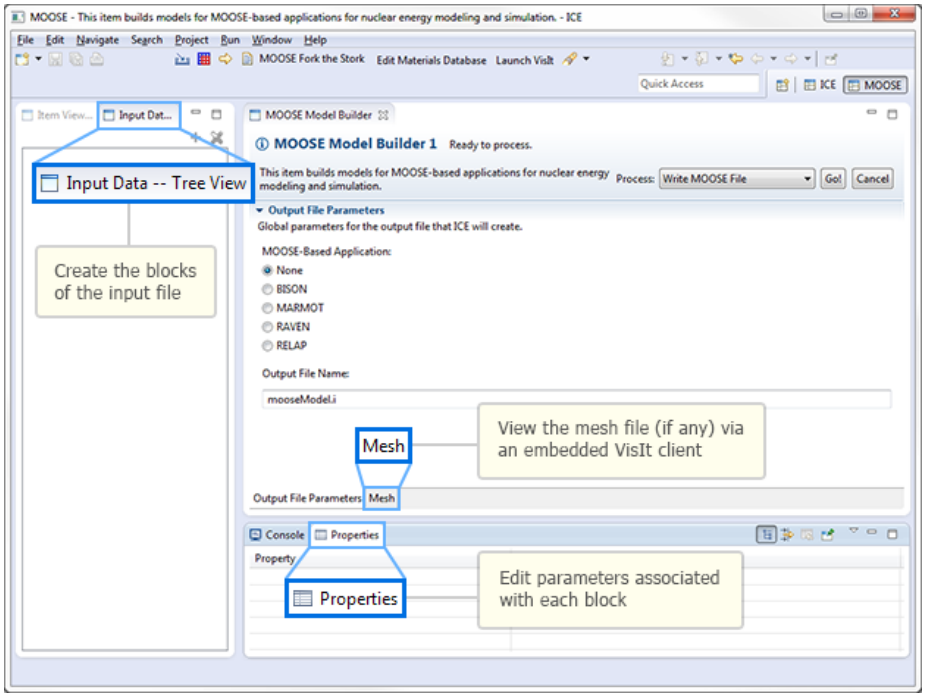


Figure 1.7: The MOOSE Model Builder, blank with no mesh selected.

Constructing the Tree

Before be begin constructing our problem, the first order of business is to specify which MOOSE application we're defining a problem for. Using the radio buttons in the main *MOOSE Model Builder* tab, select one of the options; in this example, we'll be creating a BISON problem.

Once you've made a selection, it must be applied by saving the form. Save by clicking the floppy-disk save icon (or *Ctrl+S*).

At this point, a fully loaded set of blocks should appear in the *Tree View*. If you imported your data from an already existing file, you'll notice the data has been loaded into any blocks that are checked off. If you started from scratch, none of the blocks will be checked off yet.

We're now ready to begin using the *Tree View* to add, delete and modify blocks.

Expanding and Collapsing Subblocks If you imported data from an already existing input file, you will likely notice some block names with a small

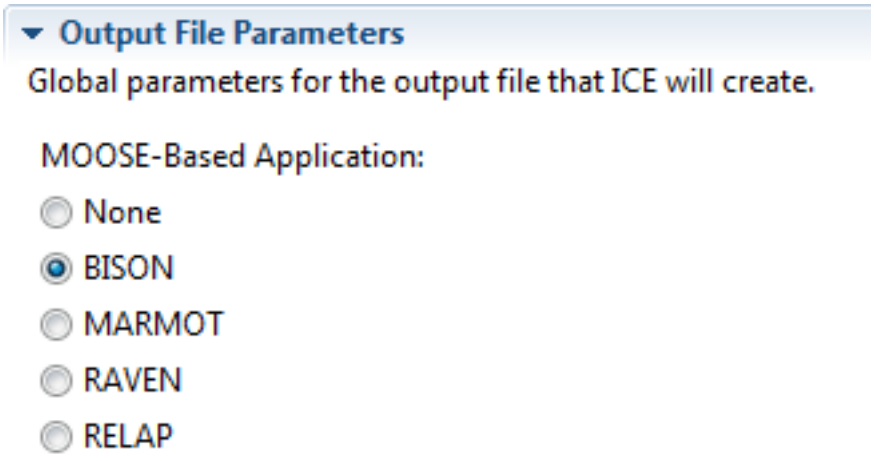


Figure 1.8: The MOOSE App Selection Widget. Select which MOOSE App's YAML tree to display

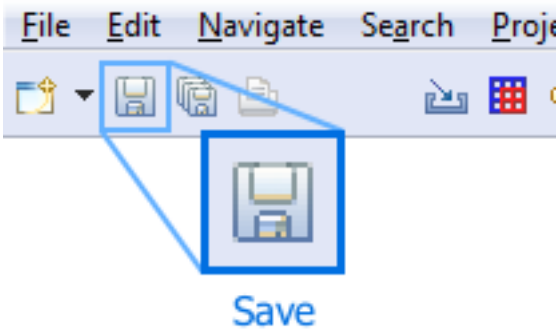


Figure 1.9: The ICE Item Save button in the ICE toolbar.

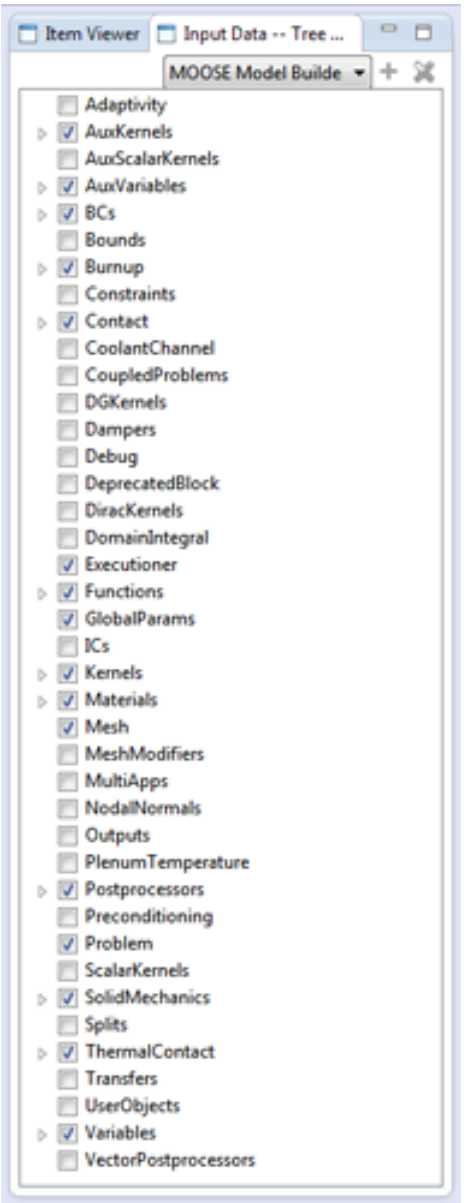


Figure 1.10: The view of a fully loaded MOOSE input file tree.

arrow next to them. This indicates that the block contains subblock structures beneath it. To access these subblocks, simply click the small arrow and the subblocks will expand. Clicking this arrow again will collapse the subblocks.

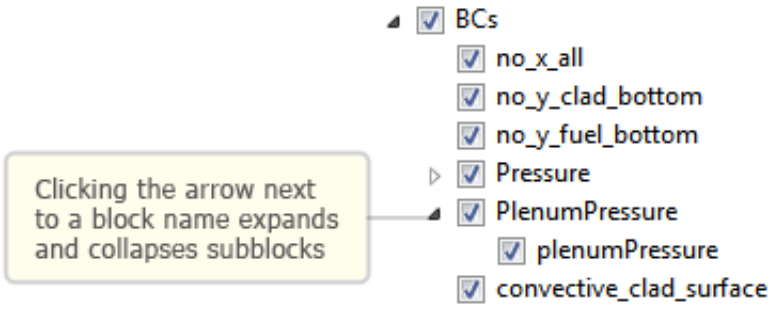


Figure 1.11: You can expand the nodes that have corresponding children in the input file.

Adding and Deleting Blocks If you'd like to add a subblock, first select the parent block you'd like to add it to. Next, click the green + button located at the top right-hand corner of the *Tree View*. Alternatively, you can also right-click the parent block, and select *Add Child* from the context menu that appears.

Doing this will prompt a pop-up dialog to appear. This dialog contains a list of all possible subblocks that can be added, according to rules outlined in the YAML files generated earlier. Each block that appears in the list has its own unique set of parameters, with the exception of `BlankBlocks`, which are—as their name suggests—blocks that are totally devoid of any data.

Once you've selected a subblock to add, click *OK*, and it will be added in the *Tree View*. Use the arrow next to the parent block to expand/collapse the list of subblocks.

Similarly, to delete a block, select the particular block you'd like to remove, and click the red "x" button located at the top right-hand corner of the *Tree View*. You can also delete a block by right-clicking on it and selecting *Delete Child* from the pop-up context menu.

If the red "x" button is greyed-out for a particular block, this means that deletion has been disabled. This is true for all top-level blocks.

Renaming Blocks To rename a block, right-click on the block in question and select *Rename* from the context menu that appears. If the renaming option is greyed-out from the context menu, this means that renaming has been disabled for this block. Once again, this is true for all top-level blocks.

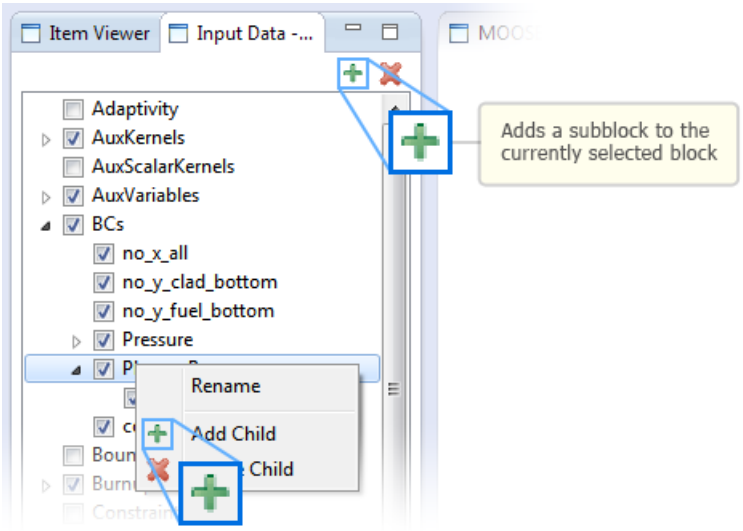


Figure 1.12: Add a child block to the top-level MOOSE input file tree by selecting the green add button.

Editing Parameters

Each block has a set of parameters associated to it. The default list of parameters for each block is drawn from the YAML file that was generated earlier. To add, remove or modify parameters associated to a block, first select a block from the *Tree View*. A table of parameters will then appear in the *Properties* tab referenced earlier.

The parameters table displays a parameter name, value and the option for an in-line comment, which can all be edited directly in this table. When written out to file, each parameter will be written to one line in the form:

```
name = value     # comment
```

Additional parameters can be added by clicking the + button to the right of the parameter table; parameters can be similarly deleted by clicking the "-" icon.

Next to each parameter row, you'll also notice an *Enabled* checkbox. Toggling this option on means that the parameter associated to it will be written out normally in the file created. Toggling this option off will cause the entire line to be commented out, and thus, won't be used during the problem runtime.

Note that if you created a *MOOSE Model Builder* by importing data from an already existing input file, ICE automatically parses any in-line comments or commented out lines (non-sensitive to leading whitespaces). This will be reflected in the *Enabled* and *Comment* columns of the parameters table.

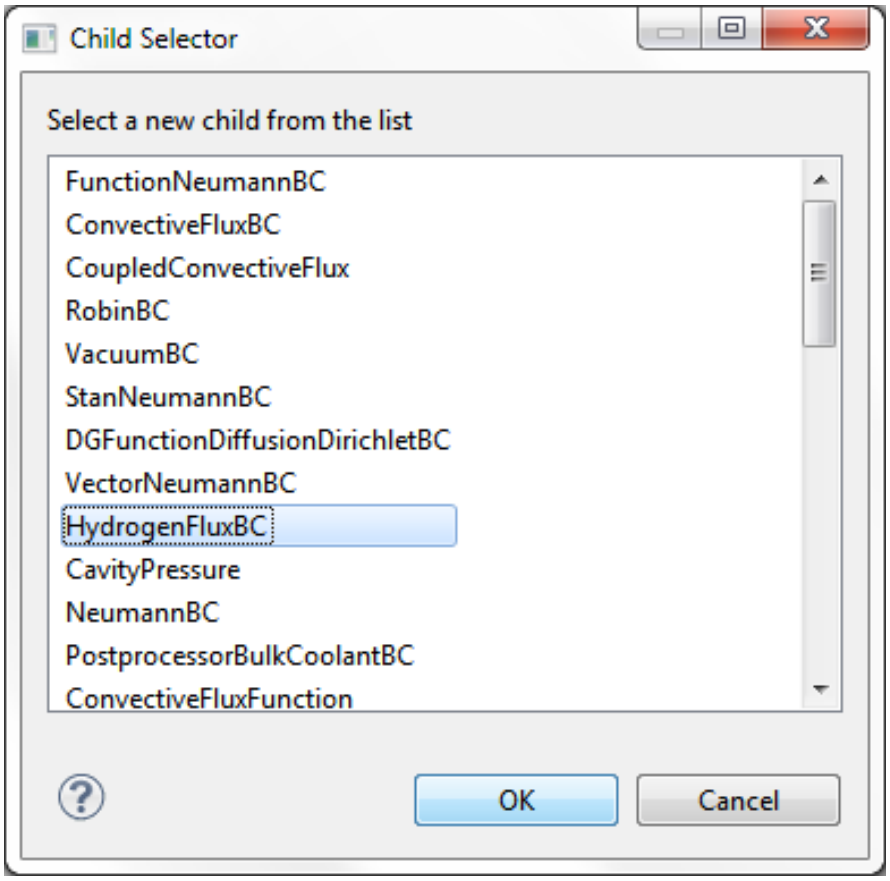


Figure 1.13: View of the possible children that can be selected for a BC block. Select one to add it to the tree.

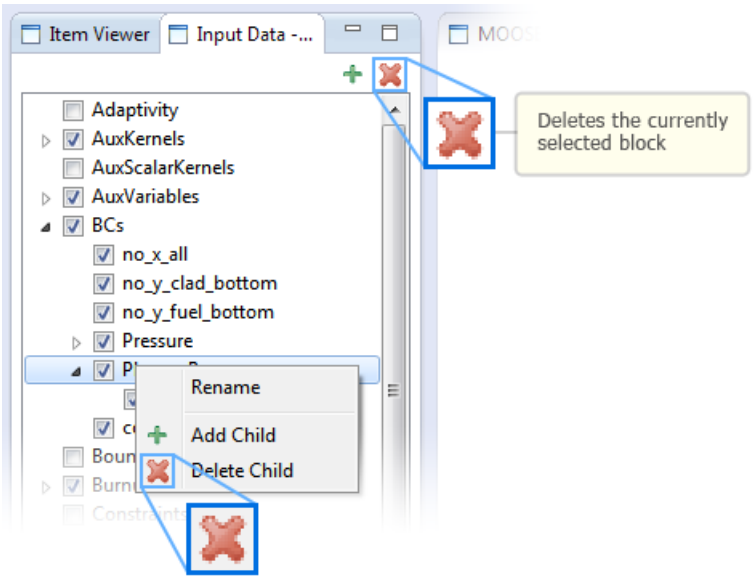


Figure 1.14: Delete a child block by selecting the red delete button.

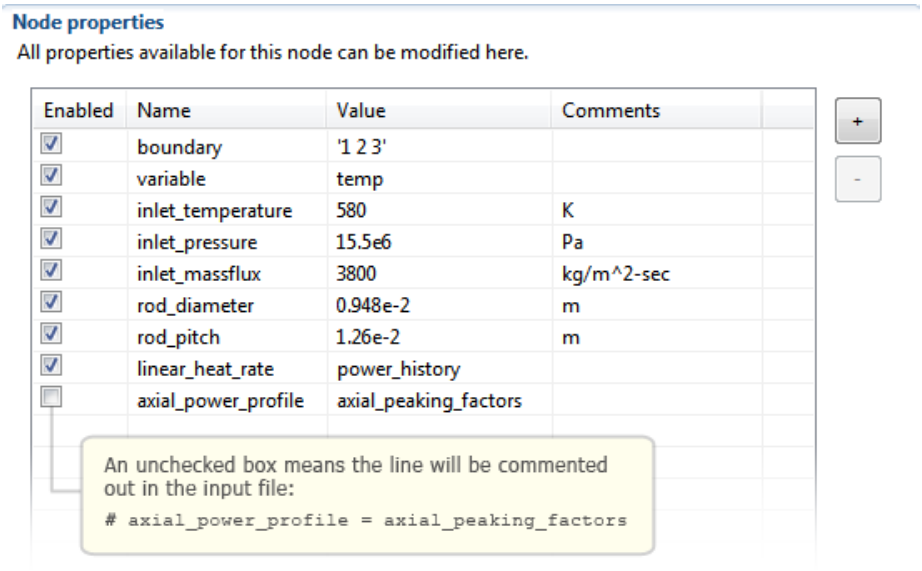


Figure 1.15: Clicking on a block in the MOOSE tree will display that block's set of edittable properties.

Lastly, some blocks contain a special `type` parameter, such as the `Executioner` block, which affects the list of parameters associated to the block. In these special instances, an additional drop-down menu will appear in the *Property* tab, above the parameters table.

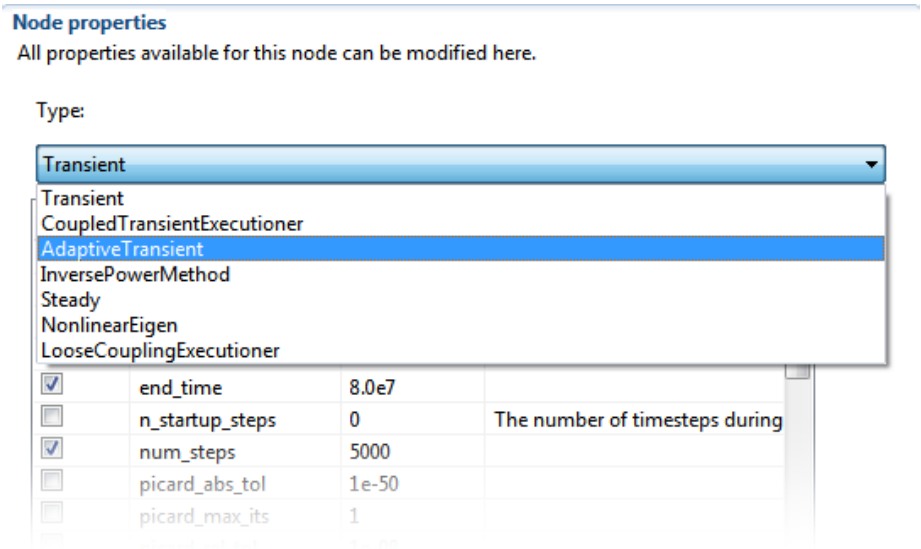


Figure 1.16: Some blocks have an associated adaptive type, to change that type just select it from the dropdown at the top of the Properties view.

By setting the `type` of the block ICE automatically repopulates the parameter table according to the rules defined by the YAML specification.

Viewing the Mesh File

If your MOOSE problem uses a mesh file, it can be viewed in an interactive embedded VisIt client. More information about how to correctly configure a VisIt session in ICE can be found in the documentation in the embedded VisIt visualizations in ICE following this section on using MOOSE.

To begin, make sure your `Mesh` block contains a parameter named `file` by clicking on the `Mesh` block in the MOOSE tree and examining its parameters in the *Properties View*. The `file` parameter should be set to the name of a `*.e` file located in your `/home/<user>/ICEFiles/default` directory. You can either manually place the mesh file in that folder or import it using the import button in ICE's main toolbar. If you need to add a `file` parameter, do so now, and then save the form by clicking the floppy-disk save icon (or *Ctrl+S*).

When you save the MOOSE Model, ICE will attempt to load the file specified by the `Mesh` block's `file` parameter as a *Resource*. To view the mesh, open the *MOOSE Model Builder*'s *Mesh* tab, then double-click the mesh file *Resource*

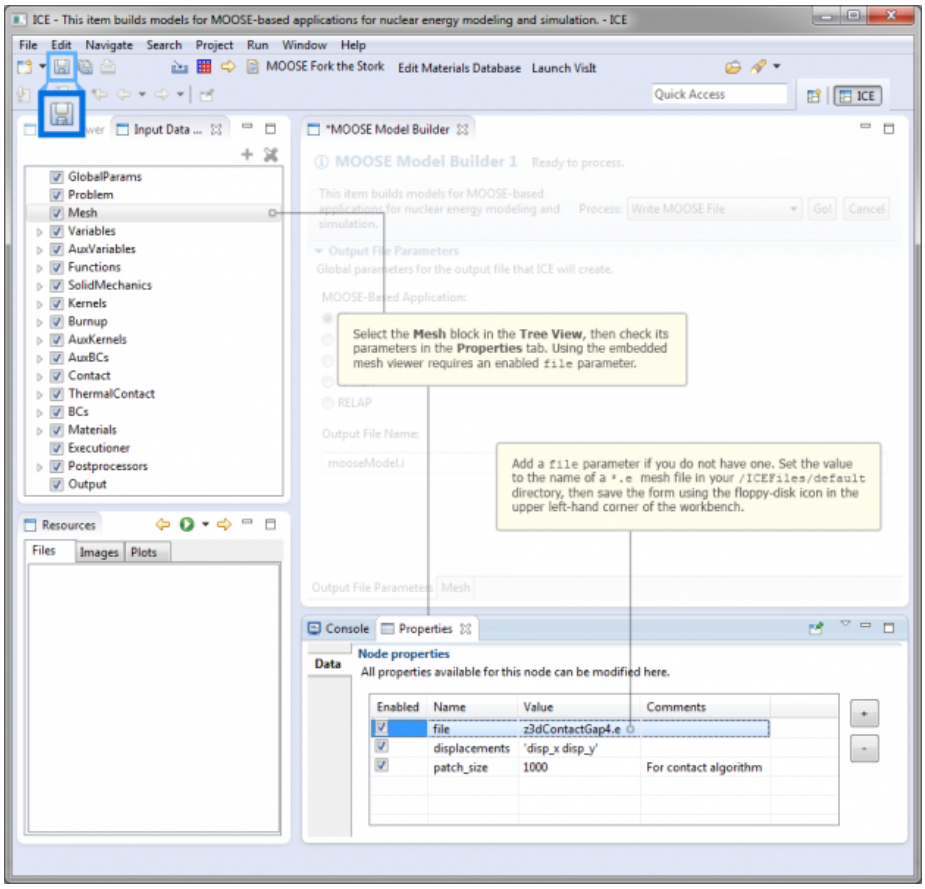


Figure 1.17: The specialized MOOSE Model Mesh view.

in the *Resources View*. If the file can be read by one of ICE's *Visualization Services*, e.g., VisIt, the first available mesh in the file will be opened in the *Mesh* tab's *Resource Page*.

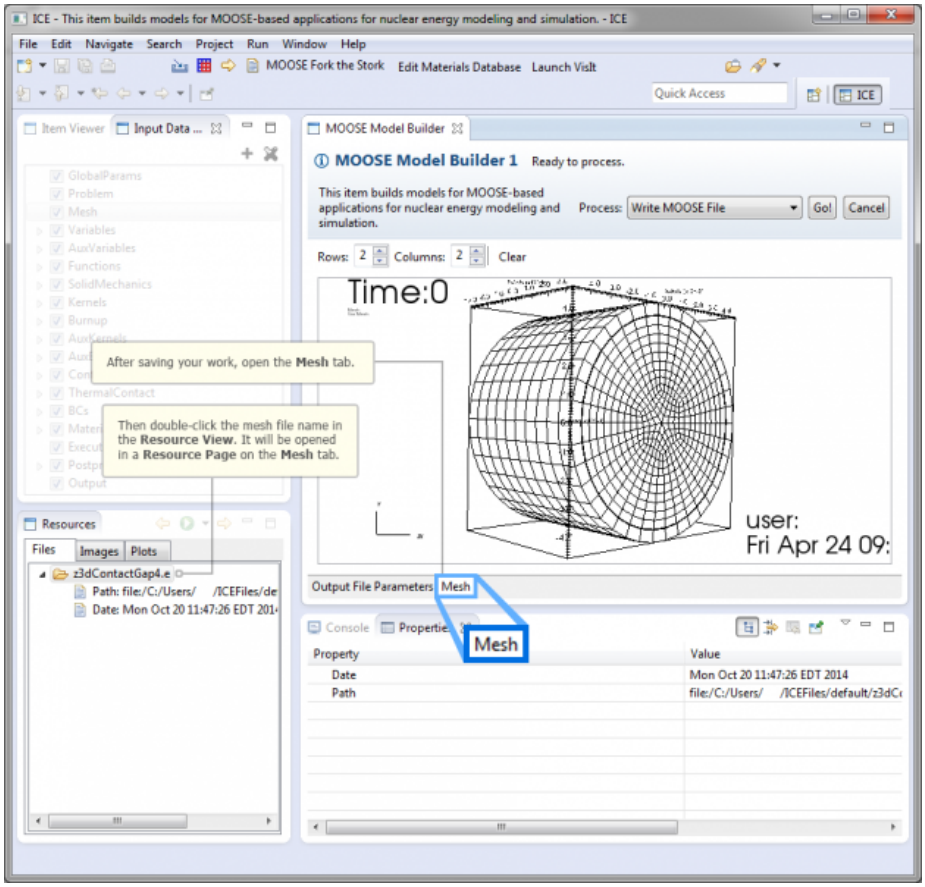


Figure 1.18: The MOOSE Mesh View with selected mesh file displayed in the MOOSE Model Builder Mesh tab.

For more information about embedded visualizations like this or about ICE *Resources* and *Resource Pages*, please see the documentation in the embedded visualizations in ICE section following this section on using MOOSE.

Viewing the 3D Plant (RELAP-7 only)

For RELAP-7 problems, ICE includes an interactive 3D *Plant View* to visualize the physical representation of the problem. To access the *Plant View*, click the *Plant View* tab located at the bottom of the main *MOOSE Model Builder* panel.

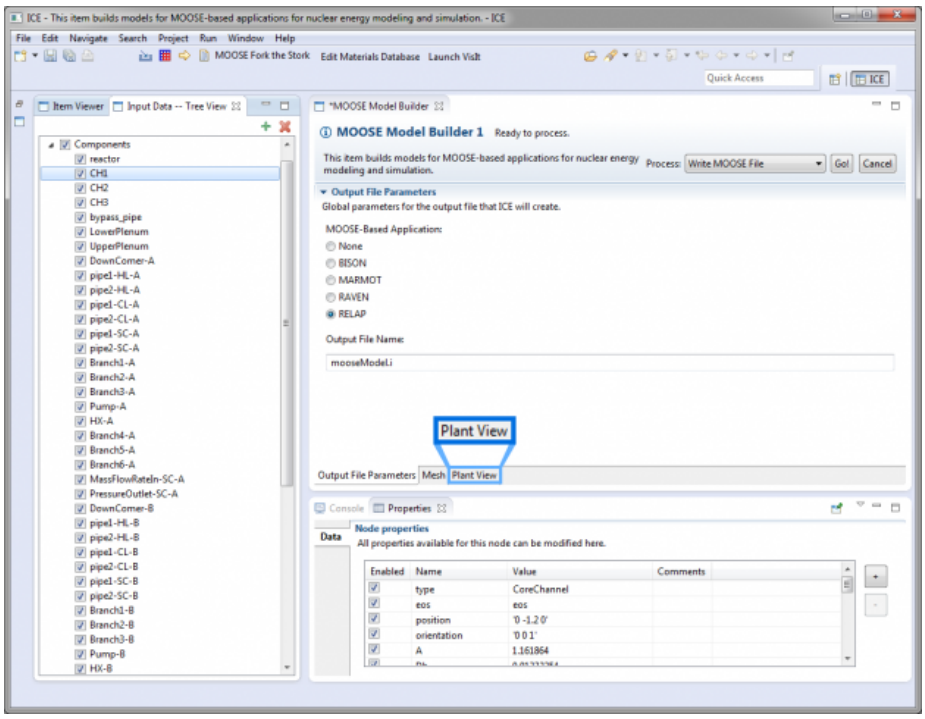


Figure 1.19: If this is a RELAP-7 model, ICE displays a custom Plant-View tab.

This view draws data from the `Components` block in the *Tree View*; if you have valid components in the `Components` block, then they should begin rendering now. Before we go on, there are a few things to note about using the *Plant View*. First, only components that are currently enabled in the *Tree View* (i.e. have a checkmark) are rendered. This way, components can easily be turned on and off without having to delete them entirely. And secondly, the *Plant View* updates in real time; any changes that you make to components in the *Tree View* will be reflected immediately such as adding, removing, re-positioning or re-orienting components.

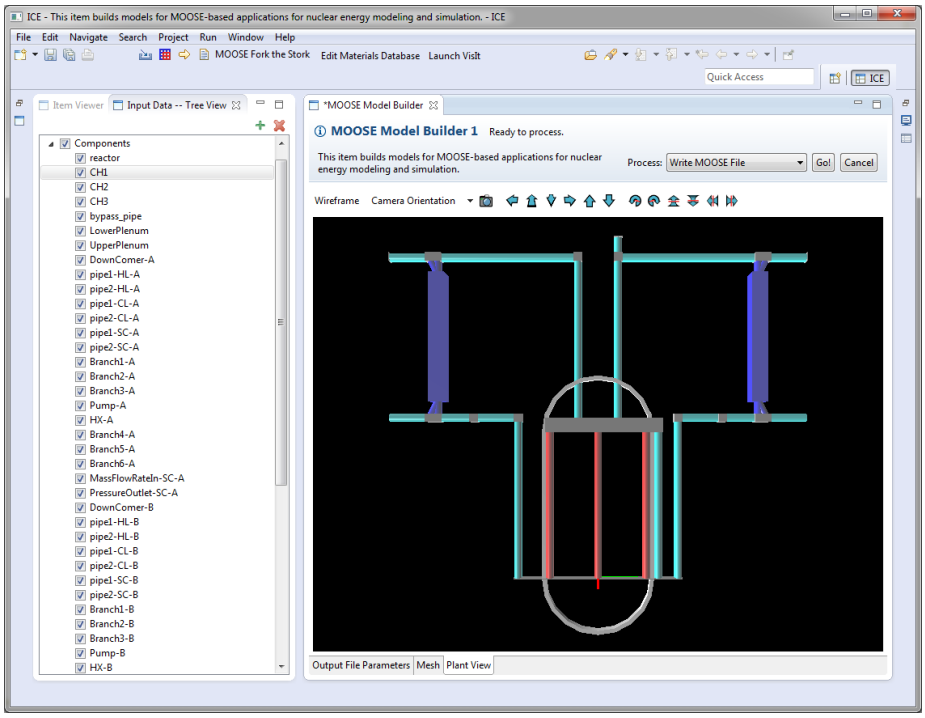


Figure 1.20: The ICE RELAP-7 Plant-View

Once your plant components have rendered, you can move around the 3D space by using the arrow buttons in the toolbar above the viewing window (hover over the button for a description of what it does), or by using the following keyboard controls:

Table 1.1: Camera Keyboard Controls

Movement	Key	Rotation	Key
Left	A	Roll right	Q
Right	D	Roll left	E

Movement	Key	Rotation	Key
Up	C	Pivot left*	
Down	Space	Pivot right*	
Forward	W	Pivot up*	
Backward	S	Pivot down*	

* The camera viewing-angle can also be pivoted by left-clicking and dragging

Lastly, the *Plant View* has 3 other tools located in the toolbar above the 3D viewing window, to the left of the movement/rotation buttons.

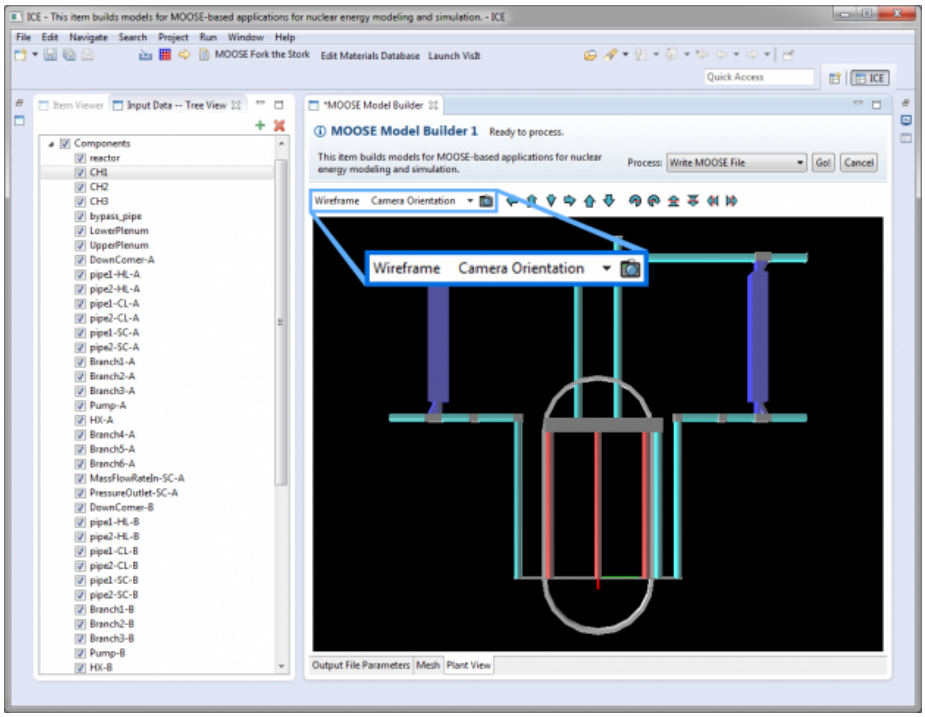
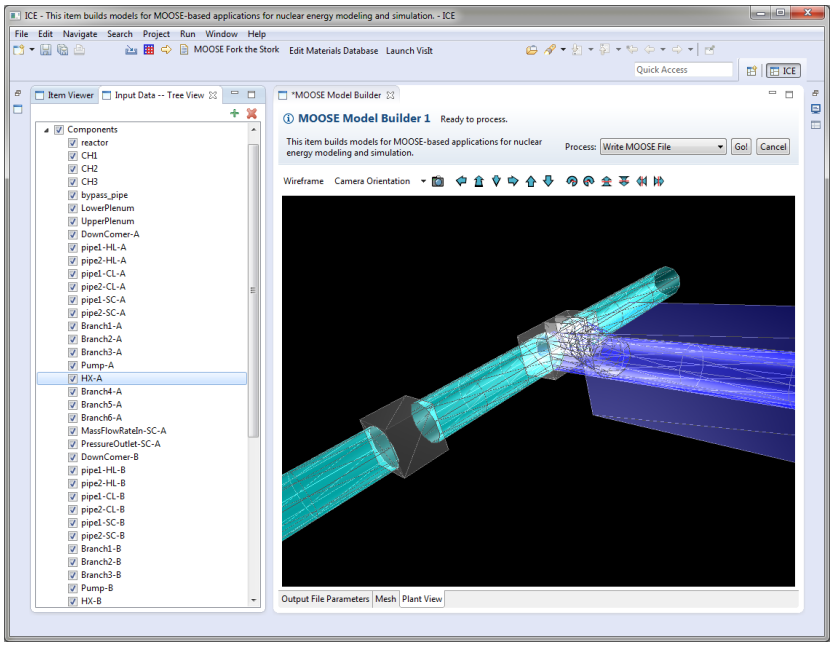


Figure 1.21: A view of the ICE Plant-View controls.

- **Wireframe** - Clicking this will toggle the plant's wireframe on and off; this allows you see how the meshes for the rendering engine are constructed.



In certain cases, this may be useful for verifying the plant model before running a simulation. For instance, pipe components in RELAP-7 have a property called "n_elems" representing the number of cylindrical cross-section slices along the pipe's length. By switching to wireframe mode, the user can see the same number of cylindrical sections stacked together that represent the pipe's physical construction.

- **Camera Orientation** - You can re-orient the rendering engine's default camera view by clicking on the *Camera Orientation* button. The default orientation—YZ—is the standard physics orientation with the Y axis increasing to the right, the Z axis increasing upwards, and the camera looking at the YZ-plane along the positive X axis.

  In the same "Camera Orientation" menu, we provide two alternative default orientations: the XY orientation shows the XY-plane from along the positive Z axis, and the ZX orientation shows the ZX-plane from along the positive Y axis. Selecting *Reset to current default* will snap the camera back to the origin view should you ever get lost.

- (***Camera Icon* - Save Image**) - Clicking this will prompt a pop-up window to save a `.png` image of the current *Plant View* to your local filesystem.

Creating the File

Once you have edited your blocks and associated parameters to your liking, the last step is to write them to file.

Any top-level block in the *Tree View* with a checkmark next to it will be written to file, and any top-level blocks without a checkmark will not. Similarly, any *sub*-blocks with a checkmark will also written to file, however, any sub-blocks *without* a checkmark will still be written to file but commented out. Ensure that you've correctly checked/unchecked all the necessary blocks. Save your work by clicking the floppy-disk save icon (or *Ctrl+S*).

In the main *MOOSE Model Builder* tab, specify the name of the file you'd like to write in the *Output File Name* field. Next, in the top right-hand corner, set the *Process* drop-down menu to "Write MOOSE File", and click the *Go!* button.

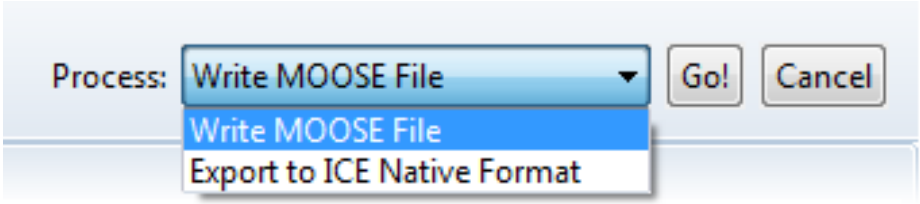


Figure 1.22: The Write File Entry for specifying the name of the input file you'd like to write.

This will write the contents of your *Tree View* to the specified filename, and will be placed in your `/ICEFiles/default` directory. If you wish to review the file before moving onto the next section, you can do so by using the ICE toolbar and navigating to:

File > Open File...

1.2.3 Launching a MOOSE Job

Once you've generated appropriate input files, launching a MOOSE job is a relatively simple task.

To get started, click the green + button in the *Item Viewer* once more to create a new ICE Item.

Select *MOOSE Launcher* from the menu that pops up and click *Finish*.

A form will appear in the main ICE workbench area. This form contains the information necessary for launching a MOOSE problem.

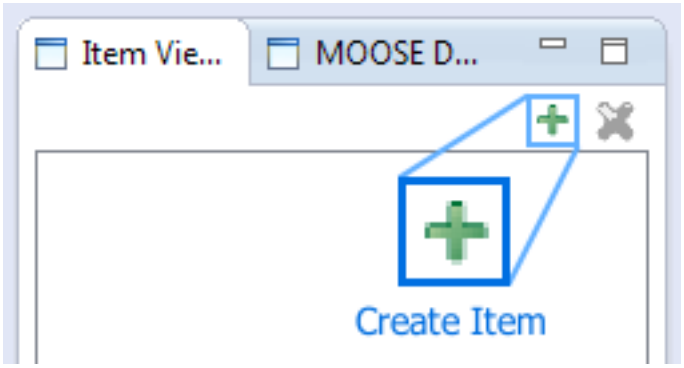


Figure 1.23: The Create Item button in the ICE Item Viewer toolbar.

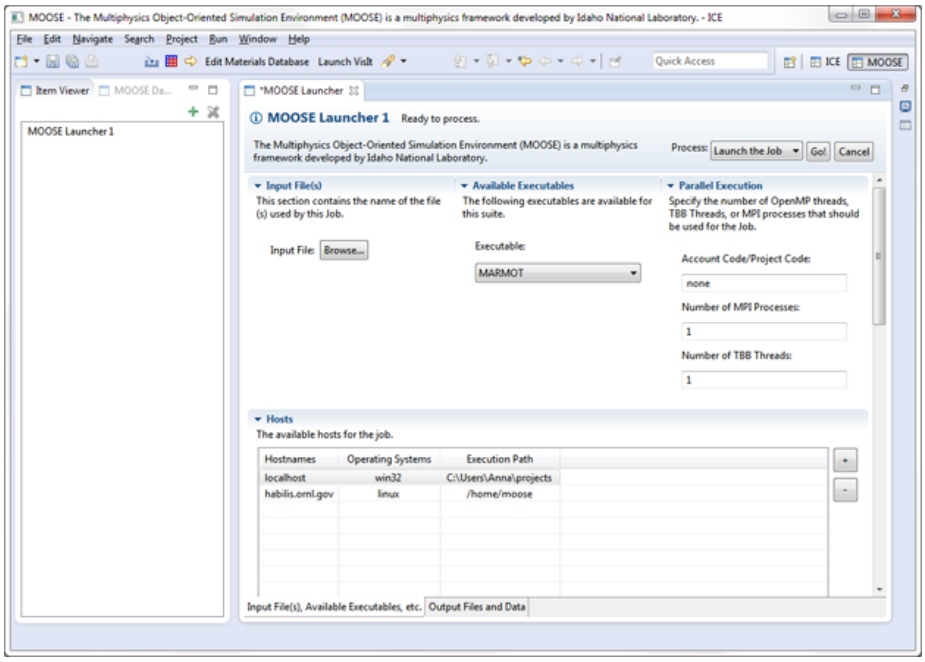


Figure 1.24: The MOOSELauncher View.

Selecting the Input File(s)

From the *Input File(s)* drop-down menu, select an appropriate MOOSE input file. This drop-down menu displays all `*.i` files that were in the `/ICEFiles/default` directory at the time the *MOOSE Launcher* was created. If you created your own input file in the previous step using the *MOOSE Model Builder*, this file should appear in the list of available files.

If you'd like to use an input file not found in this list, click the *Browse...* button; a file browser will pop up for you to locate the file you'd like to use. Once you've selected a file, it will be imported into the `/ICEFiles/default` directory.

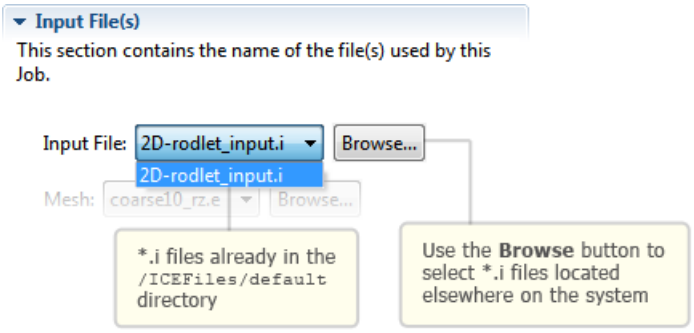


Figure 1.25: Select the input file to launch with the specified MOOSE application. You can browse the file system for the input file if it is not located in your ICEFiles/default workspace.

At this time, the *MOOSE Launcher* will scan the `*.i` file for any references to additional files, such as a mesh, peaking factors, power history, etc. If any additional file dependencies are found, the *MOOSE Launcher* will dynamically render additional file entries for your to set in the same manner.

Selecting the MOOSE Product

You will need to indicate which MOOSE-based application you'd like to run. From the list of *Available Executables*, simply select one of the available applications.

If you'd like to launch a custom MOOSE application not listed, you also have the option of doing so. Select *Custom executable name* from the drop-down menu, and enter your application's name in the text field that appears.

If we were to use the example in Figure 1.27, ICE would attempt to launch an executable called `PUMA-opt` (and not `puma-opt`). To learn more about creating your own custom MOOSE-based applications in ICE, read the Developing MOOSE Applications with ICE section of this tutorial document following this section on using MOOSE.

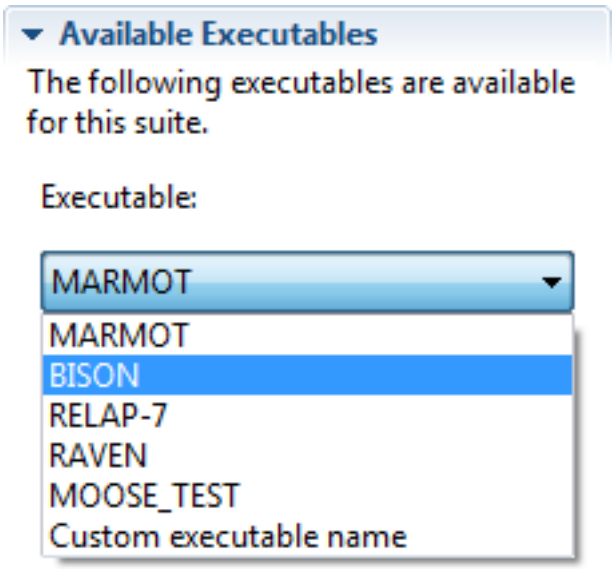


Figure 1.26: A list of the available MOOSE applications to choose from.

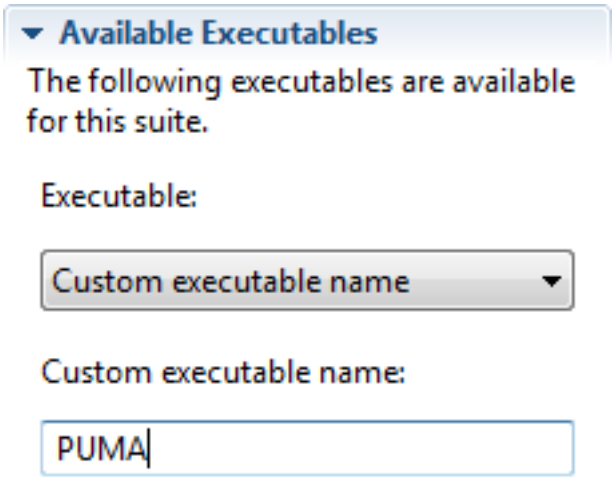


Figure 1.27: You can now specify a custom executable!

Specifying a Hostmachine

From here, the next step is to tell ICE which machine your MOOSE code will be run on, either locally or remotely. A list of hosts used at ORNL is displayed by default, however, additional hosts can be added by clicking the + button to the right of the *Hosts* table.

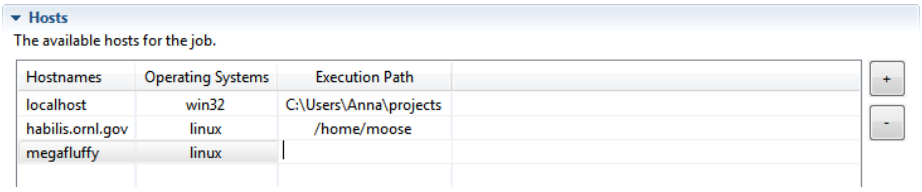


Figure 1.28: The MOOSELauncher Hosts table. Specify the hosts that you'd like to launch the application on.

When adding hosts, set the *Execution Path* to the trunk directory of the machine's MOOSE installation. For example, if I'm launching BISON on a machine with the follow structure:

```
/home/user/trunk/bison/bison-opt
```

I would set the execution path in ICE to be:

```
/home/user/trunk
```

If you are launching on a remote machine, also be sure that you have appropriate privileges for the execution path.

Setting Parallel Execution (Optional)

Optionally, if you'd like to take advantage of parallel processing, you may specify the number of MPI process and/or Intel Thread Building Block (TBB) threads.

To use multiple MPI processes, change the marked field to an integer value anywhere between 1 and 10000. Note that `mpirun` must be specified in the host machine's `PATH` variable. If you choose not to change this field, the default value of 1 MPI process is used.

To use multiple TBB threads, change the marked field to an integer value anywhere between 1 and 256. Note that the host machine must have Intel TBB support. If you choose not to change this field, the default value of 1 TBB thread is used.

Launching the Problem

Once the input file(s), host, and any parallel execution options are specified, save your settings. If you make any subsequent changes to the *MOOSE Launcher* form, you will have to re-apply them by saving the form in the same way.

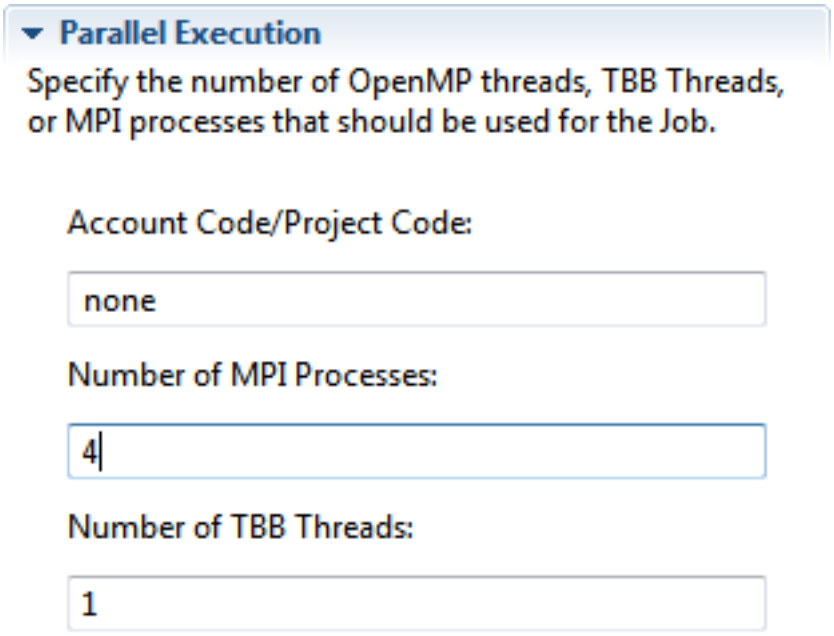


Figure 1.29: You can specify how many MPI processes, OpenMP threads, or Intel TBB threads to use in your application launch.

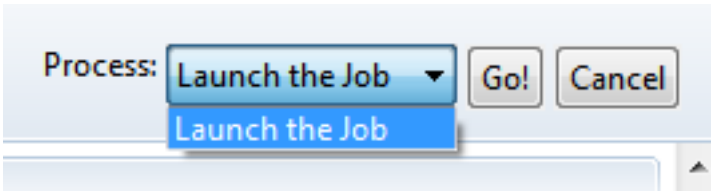


Figure 1.30: Select the Launch the Job action and click Go to launch your MOOSE application.

Lastly, use the *Process* menu in the upper right-hand corner; select the *Launch the Job* task from the drop-down menu and click the *Go!* button.

Depending on your host machine's configuration, you may be prompted for login credentials.

You should shortly begin seeing the standard console output in ICE as your problem begins to solve.

Chapter 2

Embedded Visualizations in ICE

2.1 Introduction

This document describes the embedded visualization capabilities available in ICE. Embedded visualizations in ICE aim to provide users with immediate yet simple visualization of simulation input and output well within the simulation workflow. In other words, simulation input and output can be visually verified or validated without leaving the associated Model Builder or Job Launcher in ICE. For more advanced or complex visualization tasks in ICE, please see the document on Visualizing Output with ICE.

2.2 Resources and Resource Pages

ICE *Items* may include any number of *Resources*, which may include any type of file like meshes, pictures, shell scripts, `.csv` files, or even plain text files. For example, a *Model Builder*, which is used in ICE to configure simulation input, may at some point include a plain text input file as well as associated data files or meshes. Likewise, a *Job Launcher*, which actually launches the simulation, may produce any type of output data, which may also include `.csv` files, binary files, or even more mesh files.

Before proceeding, you should familiarize yourself with the *Resources View* (highlighted in blue on the left) and the idea of a *Resource Page* (highlighted in red, while its tab is highlighted at the bottom) as shown in the image below.

The Resources View

Some *Items* in ICE store references to these resources for easy access to the user. In these cases, the resources are presented to the user in the *Resources View*, which is part of the default ICE perspective.

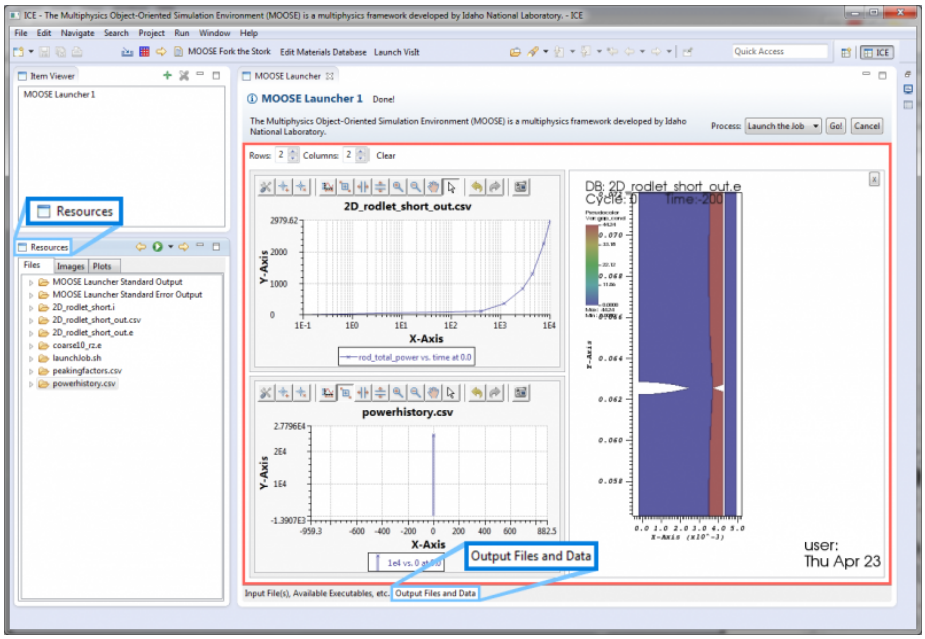


Figure 2.1: The new ICE Resource View showing generated output files that can be visualized natively in the view.

The *Resource View* is actually a tree, where each resource has its own node. If you expand a resource node, it will reveal the underlying file's path along with its date of last modification.

Resource Pages

If an ICE *Item* includes resources, it will have a designated tab for showing those resources.

- For *Model Builders*, this tab could be named anything. For instance, in the *MOOSE Model Builder*, the tab for viewing resources is called *Mesh*.
- For *Job Launchers*, this tab will be named *Output Files and Data*.

Click on the tab to open it and view the *Item*'s *Resource Page*. Each *Resource Page* is capable of showing plain text files or embedded visualizations based on the selected resource's file type.

Viewing a Resource

To open a resource, go to the *Resources View* and double-click its item in the list of available resources. The way ICE handles this selected resource depends on the following:

1. If the file can be opened using one of the visualization services, it will be added to the active *Resource Page*.
2. If the file can be opened using a plain text editor, it will be opened in a new *Text Editor* in ICE.
3. If the file cannot be opened, the *Resource Page* will attempt to render it through a browser widget. This handles some basic files like certain image types.
4. If the file cannot be opened in the browser, the browser will prompt you to save the file. In this case, you can either save it, or cancel and open the existing file at the location specified in the *Resources View*.

This document will not discuss the latter three situations any further. The next sections will describe the embedded visualizations from the first scenario.

Controlling Embedded Visualizations

Each *Resource Page* can display a number of embedded visualizations at any time. Selected plots are displayed in a grid defined by the *Rows* and *Columns* widgets in the toolbar (highlighted in blue in the image below) near the top of the page, although embedded plots will take up as much space as they can. The default grid is two by two, although the number of rows and columns can be changed at any time.

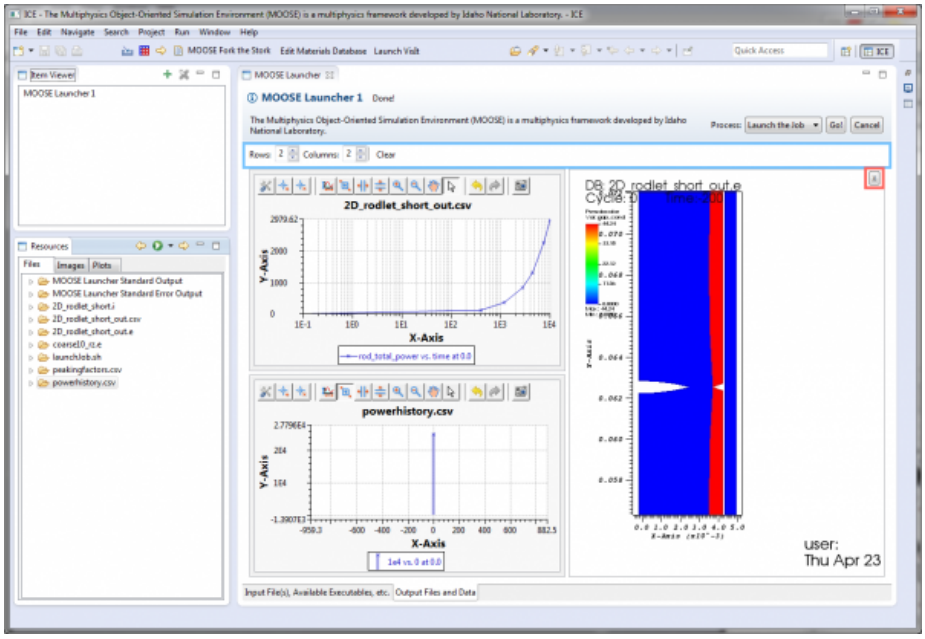


Figure 2.2: The Resource Page grid of plots can be modified through the rows/columns toggle buttons.

Adding Plots to the Grid To add a new plot to the grid, simply go to the *Resources View* and double-click the desired file that can be rendered by one of ICE's visualization services. Note that the order in which plots appear in the grid depend on the order in which they are added to the grid, with the grid moving left-to-right, top-to-bottom.

Removing Plots from the Grid To remove an individual plot, you have two choices:

1. You can hover the mouse cursor over the plot, then click the "X" button that appears (highlighted in red in the image above).
2. You can right-click somewhere inside the plot, then click *Remove*.

You can also remove all plots at once by clicking the *Clear* button in the *Resource Page*'s toolbar. This is located next to the widgets for controlling the size of the grid.

Right-click Menus Every plot drawn inside the *Resource Page* includes a right-click menu that provides the following basic options:

1. **Remove** - Removes the plot from the *Resource Page*
2. **Set Plot Type** - Provides nested sub-menus that let you set what is displayed in the plot.

The contents of the *Set Plot Type* sub-menu depends on both the file and the visualization service used to plot it. Generally, you first choose the plot *category* from the first sub-menu and then a *type* from the category sub-menu.

Additional menu choices may be available depending on the visualization service that provides the plot.

2.3 Visualization Services

VisIt

Prerequisites To use the embedded visualization service for *VisIt*, ICE requires a local installation of VisIt (minimum version 2.8.2) developed by Lawrence Livermore National Laboratory.

Preferences

Getting to the Preferences Using the *VisIt* visualization service for viewing complex 2D or 3D mesh data requires little initial configuration. Assuming the VisIt prerequisite is installed, use the the main menu bar at the top of the window and navigate to:

Window > *Preferences*

In the *Preferences* dialog, navigate to:

Visualization > *VisIt* (highlighted in blue in the image below)

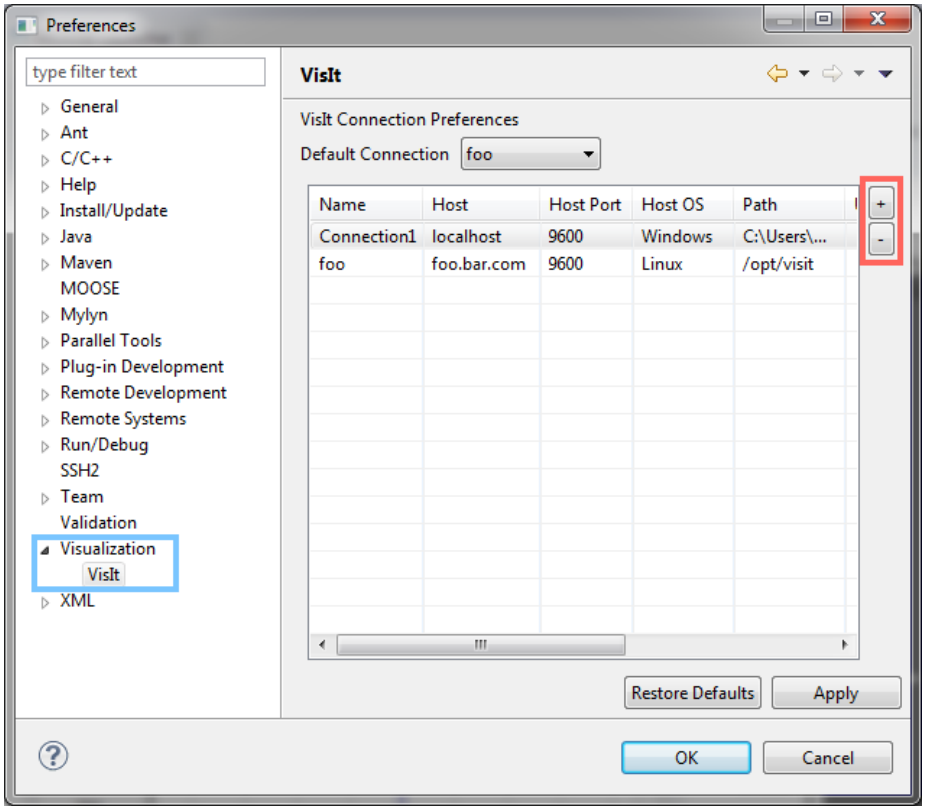


Figure 2.3: The new VisIt Preferences Page.

Adding Connections You will need to press the + button (highlighted above in red) to the right of the table to add a new entry. Fill out the new row in the table. Generally speaking, the default values are fine, but the *Path* will need to be updated to point to the folder containing your VisIt executable. You can copy and paste the path into this field and press *Enter*.

Setting the Default Connection Currently, only one connection to VisIt can be used at a time. If you only have one configured connection, it is automatically selected as the default. However, if you have multiple configured connections, you will need to set the *default* connection by choosing from the drop down above the table.

Removing Connections To remove a connection, click on any cell in its row in the table, then click the "-" button (highlighted above in red) on the right of the table. You can select multiple connections by holding *CTRL* while you click them, then clicking the "-" button.

Applying the Connection Preferences When you have finished updating your connection configurations for VisIt, you can click *OK* to apply the changes and close the *Preferences* dialog. Alternatively, you can apply the changes immediately by clicking *Apply*. It is then safe to close the *Preferences* dialog in any valid way, e.g., by clicking the dialog's close button or by clicking *Cancel*.

Opening a VisIt Plot The *Resources View* will pass any ICE resources pointing to `.e` (Exodus) or `.silo` files to the *VisIt Visualization Service*. If the visualization service is configured and running and if the file is valid, then the *Resource Page* will open a view of the resource powered by the default VisIt connection, as in the image of a battery mesh SILO file below.

An example VisIt plot embedded in a *Model Builder* can be seen in Figure 2.4.

The embedded VisIt view can be rotated by clicking and dragging and zoomed in and out with the mouse wheel.

Right-click Menu The image below shows the context menu available when right-clicking somewhere inside the VisIt view.

The plot types available in the context menu depend on the data available in the VisIt-compatible file. In the case of this SILO file, the only plot categories are *Meshes* and *Scalars*.

The context menu also includes the ability to change how the VisIt plot is rendered. The representations available depend on the current plot type. A full list of supported "representations" is listed below for each plot category.

- Materials
  - Boundary *(default)*

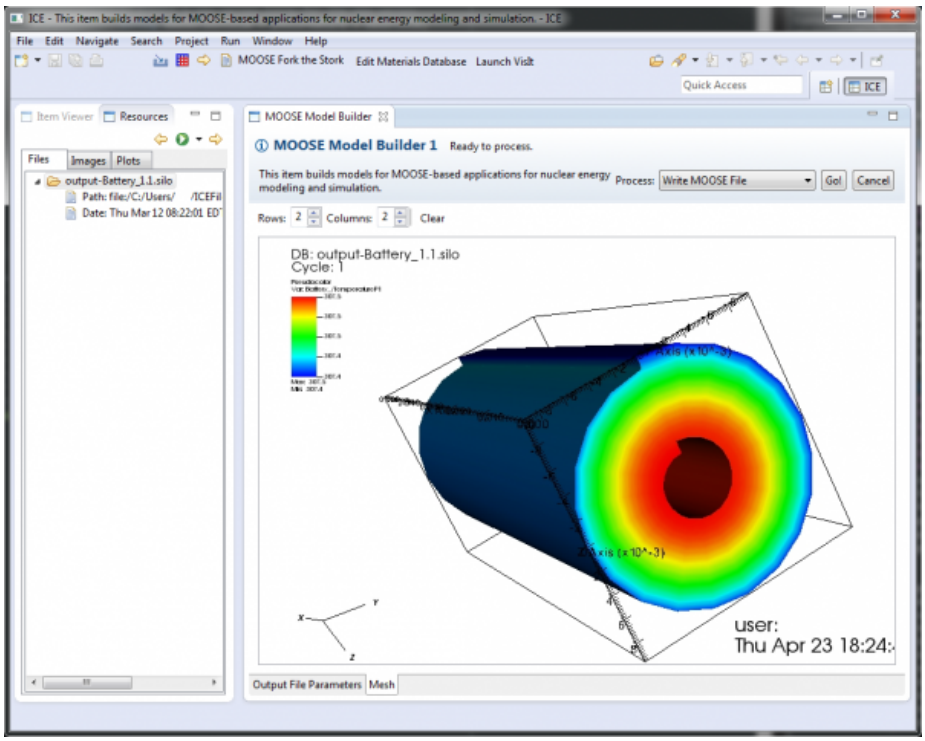


Figure 2.4: An embedded VisIt plot in the MOOSE Model Builder.

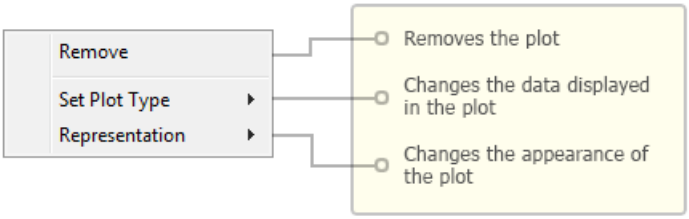


Figure 2.5: The embedded plot contains context-menu right-click functionality for modifying or deleting the plot.

  - FilledBoundary
- Meshes
  - Mesh *(default)*
- Scalars
  - Pseudocolor *(default)*
  - Contour
  - Volume
- Vectors
  - Vector *(default)*

CSV

Prerequisites The embedded visualizations for `.csv` data require no additonal software to be installed or any preference configuration.

Opening a CSV Plot The *Resources View* will pass any ICE resources pointing to `.csv` files to the *CSV Visualization Service*. If the file can be read by the *CSV Visualization Service*, then the *Resource Page* will open a plot that contains the first available series in the file. An example CSV plot embedded in a *Job Launcher* can be seen in Figure 2.6.

The embedded CSV plot includes the same toolbar described in the the standard features provided by the *CSV Plot Editor* in the *Visualization Perspective* as well as a right-click menu to modify the plot.

Right-click Menu The image below shows the context menu available when right-clicking somewhere inside the CSV plot.

By choosing a series from the *Set Plot Type* sub-menu, you can change what series is plotted. This will clear any existing series from the plot and add the selected series.

To add more series to the plot, choose a series from the *Add Series* sub-menu.

To remove a plotted series, choose one of the existing series from the *Remove Series* sub-menu. This sub-menu will be disabled if there are no series to remove.

To clear all series from the plot, click *Clear Plot* at the bottom of the menu.

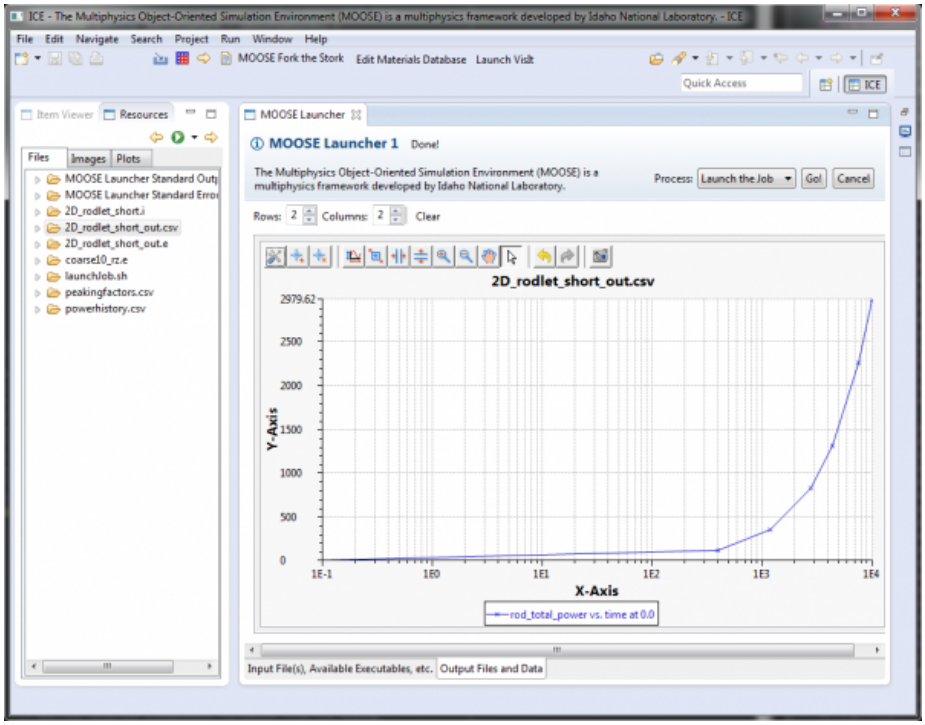


Figure 2.6: An embedded CSV plot in the MOOSE Launcher Resource Page, representing some MOOSE PostProcessor data.

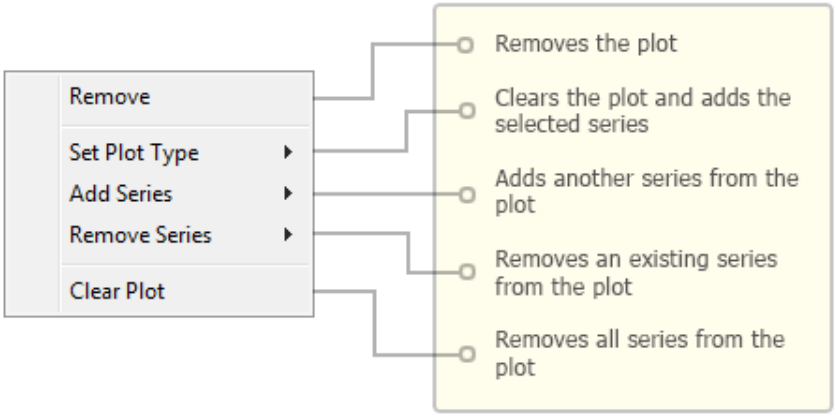


Figure 2.7: The CSV context-menu.

Chapter 3

Visualizing Output in ICE

Currently, ICE features two plugins for visualizing and plotting simulation output data:

- **VisIt Tools** - An interactive 3D visualization tool for rendering meshes, scalar plots, contour plots, and more.
- **CSV Plotting Tools** - A customizable, 2D data plotting utility for data from `.csv` files.

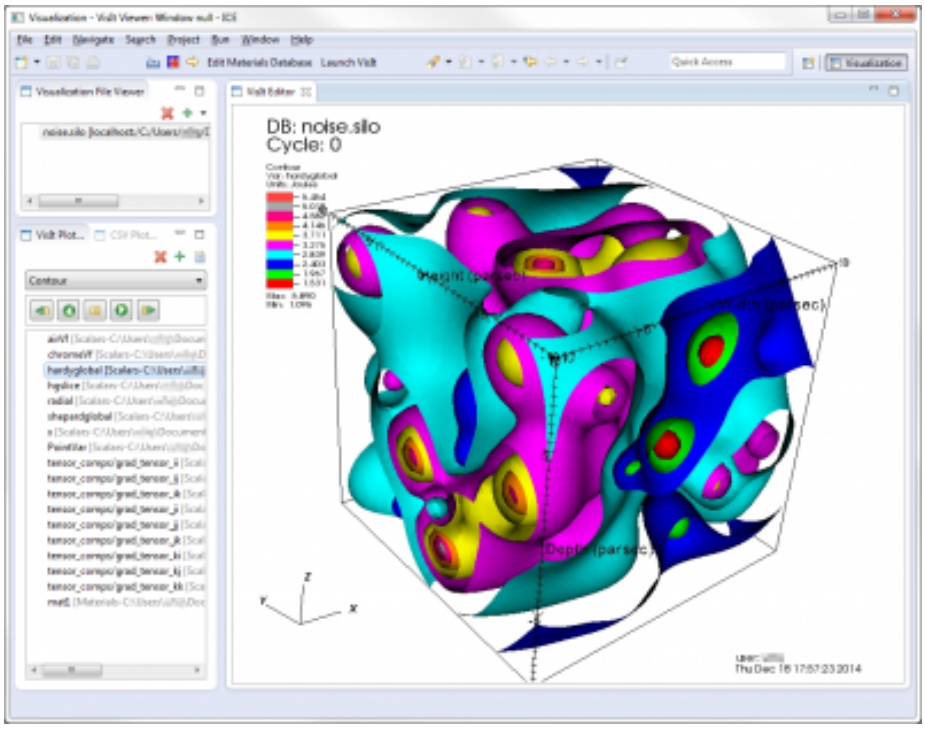


Figure 3.1: A contour plot shown in the ICE VisIt Editor.

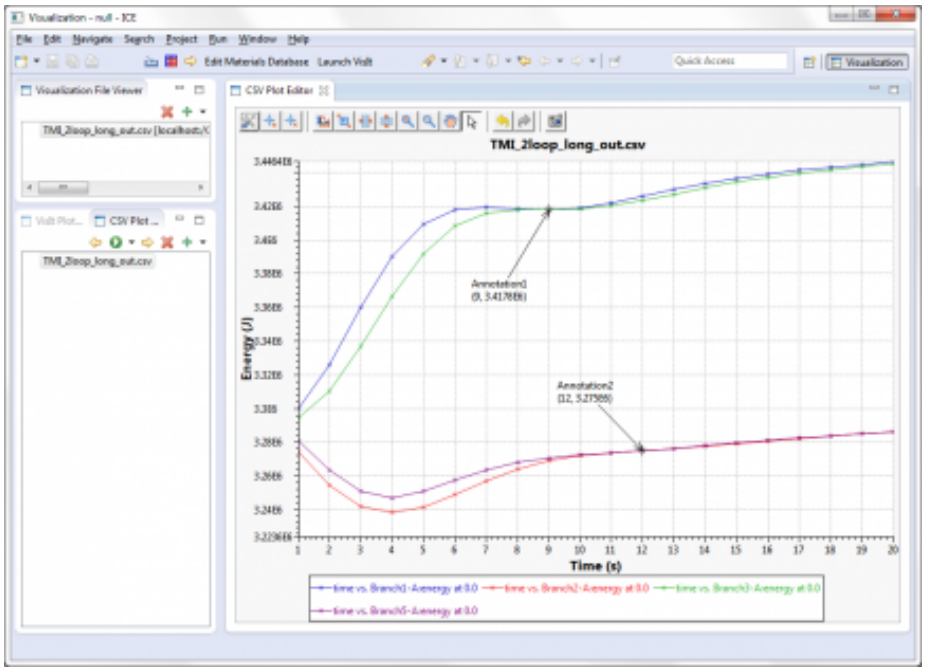


Figure 3.2: The ICE CSV Plot Editor showing MOOSE PostProcessor data.

3.1 Installation and Configuration

3.1.1 Prerequisites

To use the *VisIt Tools*, ICE requires the installation of VisIt (minimum version 2.8.2) developed by Lawrence Livermore National Laboratory, either locally or on a remote machine.

The *CSV Plotting Tools* require no additonal software to be installed.

3.1.2 Visualization Perspective

To use ICE's visualization tools, you first must switch to the *Visualization Perspective*. This perspective includes various UI components necessary for visualization that are not exposed in the default ICE perspective. To access the *Visualization Perspective*, use the the main menu bar at the top of the window and navigate to:

Window > *Open Perspective* > *Other...*

Select *Visualization* in the dialog that pops up and click *OK*. Alternatively, you can also access the same pop-up dialog by clicking the *Open Perspective* button in the main toolbar in the upper right-hand corner of the ICE workbench.

Once the *Visualization Perspective* opens, you should notice the workbench contains some new UI components. Make note of the following panels, as we

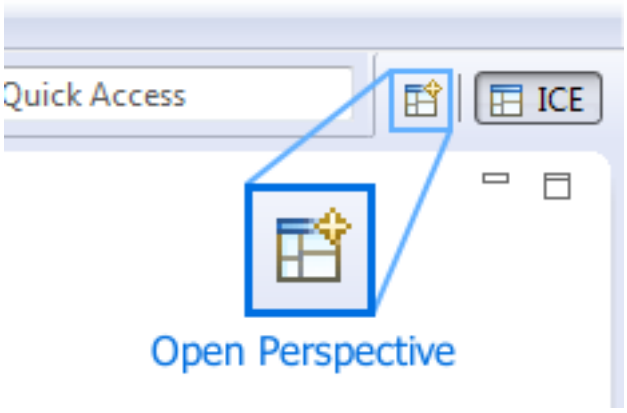


Figure 3.3: The Open Perspective button for switching to another perspective, like in this case, the Visualization Perspective.

will be referring to them in the following sections.

3.2 Visualizing Output

3.2.1 VisIt

Connecting to VisIt Once you switch to the *Visualization Perspective*, the first step necessary is to connect to your VisIt installation through ICE. To do this, click on the *Launch VisIt* button located in the ICE toolbar near the top.

A dialog will pop up offering you three options for connecting to VisIt:

1. **Launch VisIt locally**
   If you installed VisIt on your local machine, use the *Browse* button to direct ICE to your local installation directory. Using this method of connecting will launch a new VisIt session. Optionally, you can also set a port number (default 9600) and--if you want to share your VisIt session with another user--a password.
2. **Launch VisIt remotely**
   If you installed VisIt on a remote machine, specify the hostname and full path to the VisIt installation directory. Using this method of connecting will launch a new VisIt session. Optionally, you can specify a port number (default 9600) and--if you want to share your VisIt session with another user--a password. If you need or want to use an external gateway or proxy to access the remote VisIt installation, you may specify its URL and port number as well.
3. **Connect to VisIt**
   If you would like to connect to session of VisIt already running somewhere else, specify the hostname, port number, and password set on the VisIt

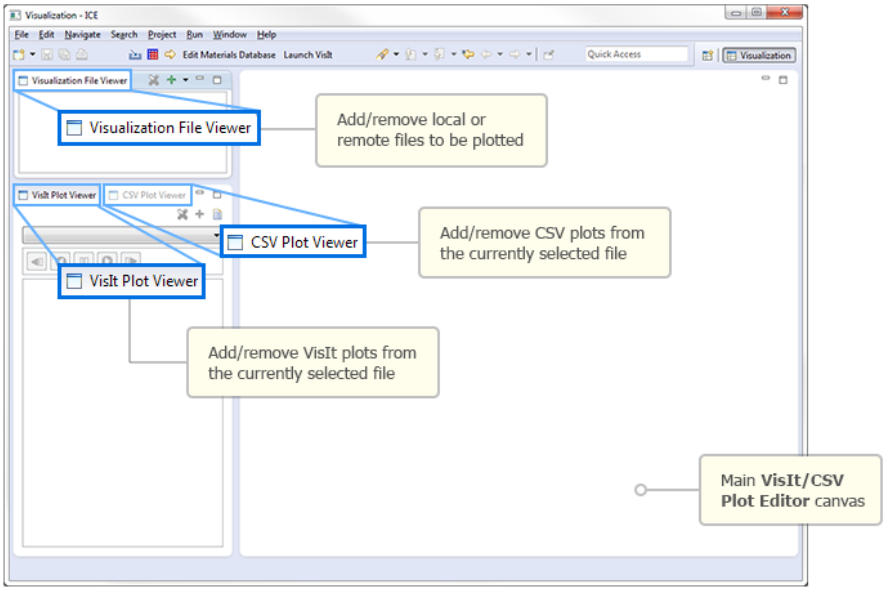


Figure 3.4: The ICE Visualization Perspective.

Figure 3.5: The Launch VisIt button in the ICE toolbar.

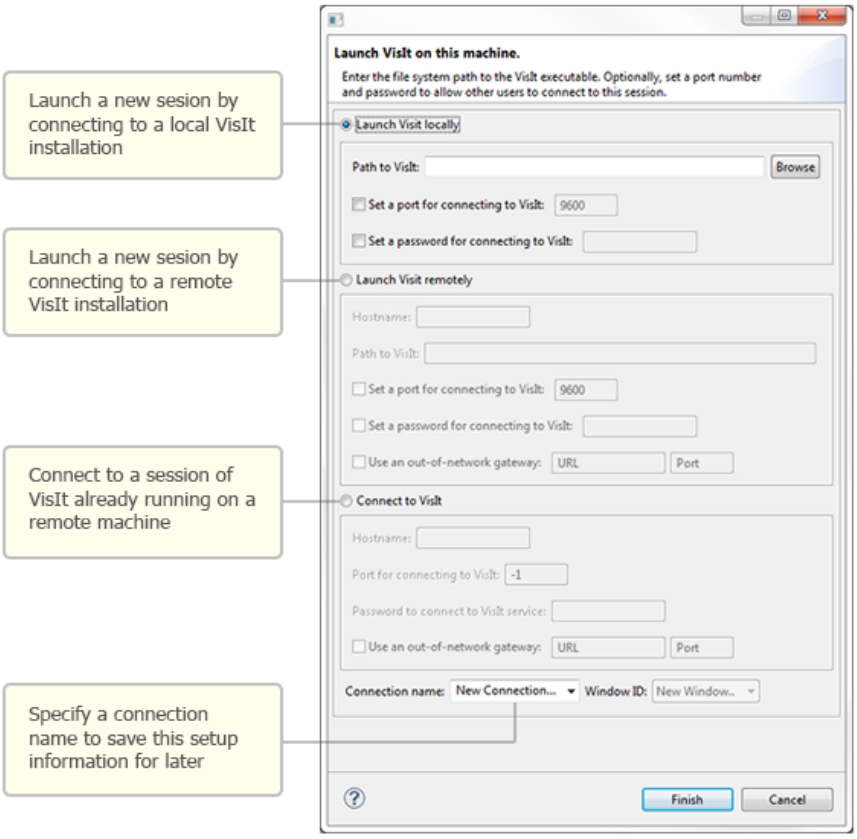


Figure 3.6: The VisIt connection wizard in ICE.

session; you will need to obtain this information from the person who initially launched the VisIt session. If you need or want to use an external gateway or proxy to access the remote VisIt installation, you may specify its URL and port number as well.

Regardless of which method you choose to connect to VisIt, enter a *Connection name* at the bottom of the pop-up dialog. This will allow you to re-use this connection information in the future.

If you are connecting to an existing session, specify a *Window ID* between 1 and 16. Which *Window ID* you use depends on how you would like to connect to VisIt. If multiple users connect using the same *Window ID*, they will all see and be able to interact with the same VisIt view. However, if you would like multiple users to each have their own unique session each with its own controls, assign a unique *Window ID* to each user. The VisIt installation can support up to 16 unique window IDs at a time.

Once you are done, click the *Finish* button at the bottom, and ICE should begin connecting to VisIt.

Adding/Removing Files To open a file, find the green + icon in the *Visualization File Viewer*. Clicking directly on the green + icon will prompt a local file browser to pop up. However, if your file is located on a remote machine, or if you would like to add a file set, click on the drop-down button next to the green + icon.

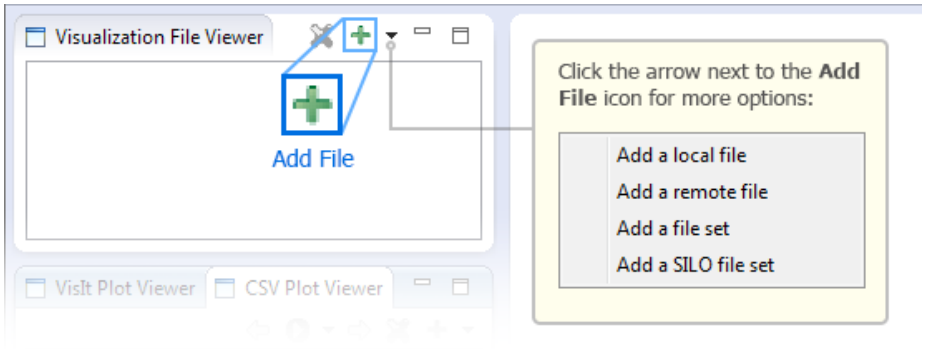


Figure 3.7: To add a new file to view in VisIt, click the green Add File button in the Visualization File View toolbar.

This will offer you four ways to open file(s):

- Open a local file
- Open a remote file
- Open a local file set
- Open a local SILO set

Once you have selected your file(s), they should appear in the *Visualization File Viewer*.

Lastly, if you would like to remove a file from the *Visualization File Viewer* list, select it, and click the red "X" button.

Adding/Removing Plots To begin adding plots, select your file in the *Visualization File Viewer* and click the green + icon in the *VisIt Plot Viewer*.

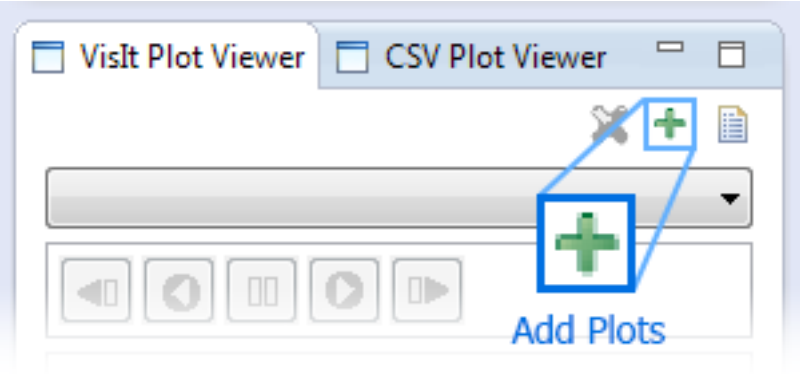


Figure 3.8: To add a new plot, click the Add Plots button in the VisIt Plot Viewer toolbar.

If there is any plottable data in your file, a dialog will pop up with a list of options to choose from. This can include mesh plots, scalar plots, vector plots, material block plots, and so forth. If there are multiple plots of each type available, you can select them all by checking off the entire category, or expand it to check off only selected plots.

When you are done selecting your plot(s), click *OK*. The selected plots should be added to the list in the *VisIt Plot Viewer*.

Lastly, if you would like to remove a plot from the *VisIt Plot Viewer* list, select it and click the red "X" button.

Rendering Plots To render a plot, *double* click it in the *VisIt Plot Viewer*, and it will appear in the main *VisIt Editor*.

The *VisIt Plot Viewer* contains a drop-down menu with a list of plotting styles available for the currently selected plot. Depending on your selected plot, this can include mesh, pseudo-color, contour, volume, and so forth. Use this drop-down menu to select the plotting style you prefer, and the *VisIt Editor* will update in real time.

The *VisIt Editor* is also interactive in that you can move your plot around by clicking and dragging the canvas or zoom by using the mouse wheel. This may not necessarily be useful for 2D plots but enables a fully rotatable look at 3D plots as in the example below.

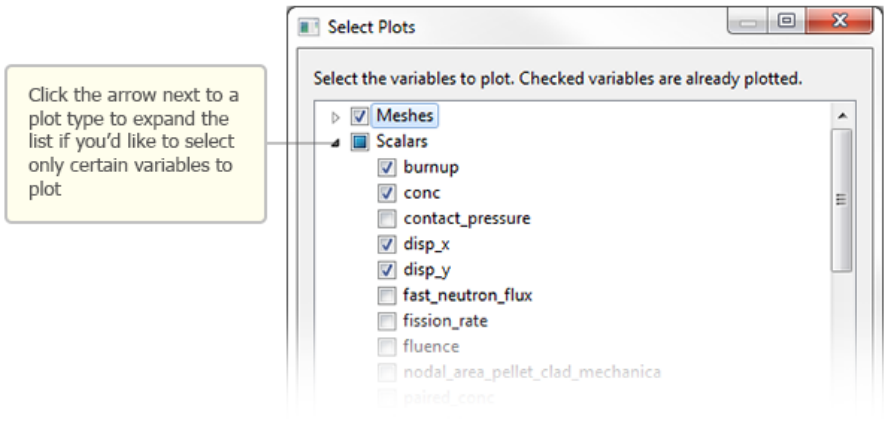


Figure 3.9: ICE prompts you with a list of available plots to select from.

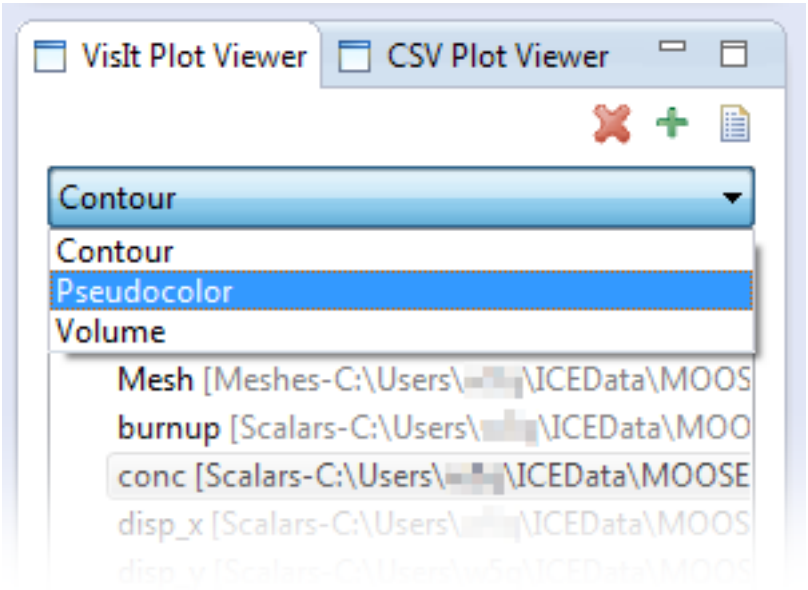


Figure 3.10: You can select the plot type in the VisIt Plot Viewer drop-down menu.

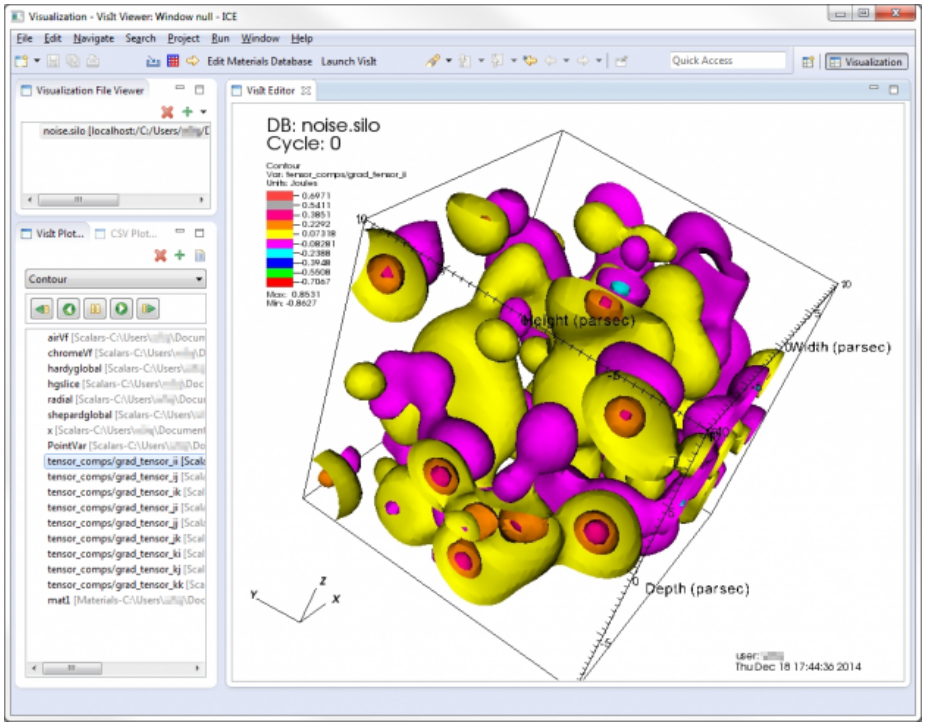


Figure 3.11: A view of a sample plot in the Visit Editor.

Lastly, if there is any time series data associated to your plot, you can manually walk through the time steps, or play them continuously as a short video, using the playback buttons located in the *VisIt Plot Viewer*.

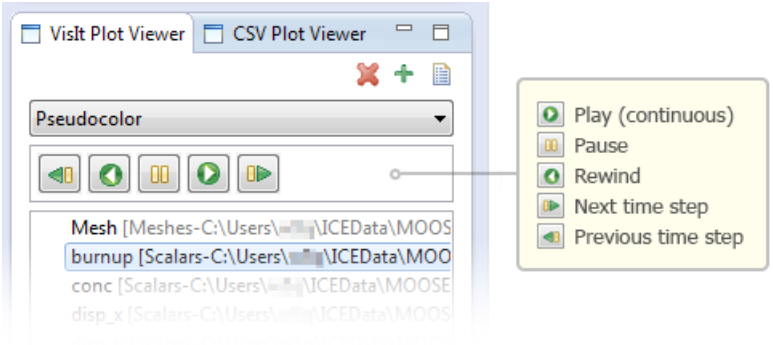


Figure 3.12: For time-series data, you can play through all time steps with the playback buttons in the Visit Plot Viewer.

Executing Python Commands While many of VisIt's features are already accessible in ICE, work to enable a more robust feature set is on-going. In the meantime, features not yet integrated into ICE can still be accessed via Python commands by clicking the Python script button located in the *VisIt Plot Viewer*.

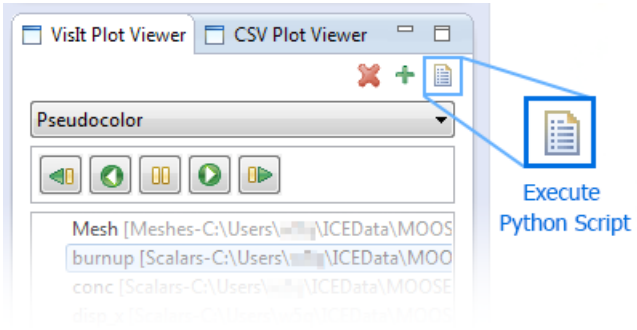


Figure 3.13: You can manipulate a VisIt plot with Python commands using the Execute Python Script button in the VisIt Plot Viewer.

Writing Python scripts for VisIt is beyond the scope of this tutorial. However, you are welcome to refer to the VisIt Python Interface Manual provided by the VisIt development team at Lawrence Livermore National Laboratory.

CSV Plot Viewer

ICE includes, out of the box, basic CSV data plotting utilities for fast and easy x/y graph visualizations. This section describes how to open your CSV data using the *CSV Plot Viewer* in the *Visualization Perspective*.

Adding/Removing Files To open a file, find the green + icon in the *Visualization File Viewer*. Clicking directly on the green + icon will prompt a local file browser to pop up. You can also access this option by clicking on the drop-down button next to the green + icon.

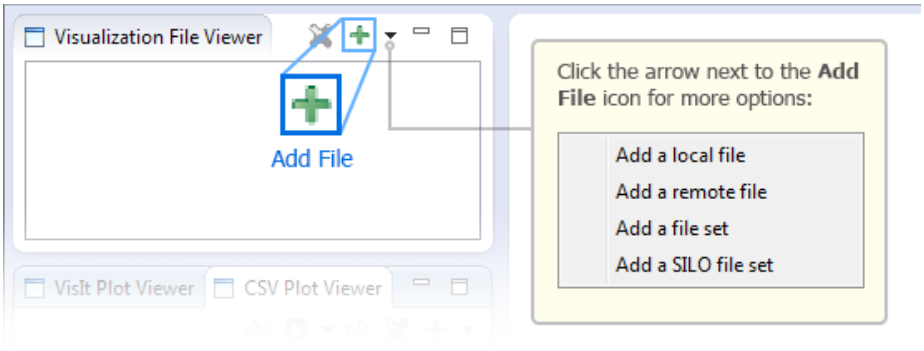


Figure 3.14: The VisIt Add File sub-menu.

This will offer you four ways to open file(s):

- Open a local file
- Open a remote file
- Open a local file set
- Open a local SILO set

Once you have selected your file(s), they should appear in the *Visualization File Viewer*.

Lastly, if you would like to remove a file from the *Visualization File Viewer* list, select it, and click the red "X" button.

Adding/Removing Plots

Adding Plots To begin adding plots, select your file in the *Visualization File Viewer* and click the green + icon in the *CSV Plot Viewer*.

Selecting Initial Plot Data If the data in your CSV file is properly formatted, then a dialog will appear. This dialog gives you a list of variables from your data file.

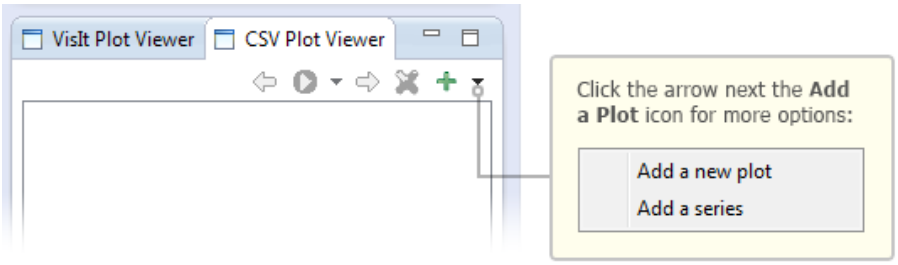


Figure 3.15: The CSV Add Plot button.

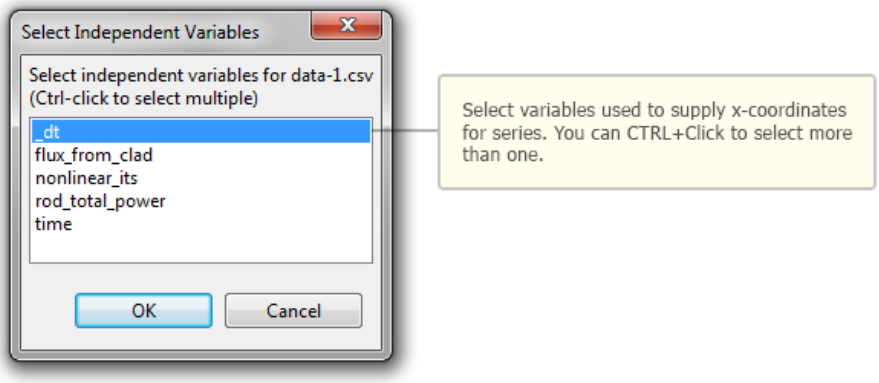


Figure 3.16: The CSV Add Plot independent variables dialog.

In this first dialog, you select independent variables from your CSV file. Independent variables are those whose values determine the x-coordinates of plotted series. You can select multiple data as independent variables by holding the *CTRL* key while clicking values in the dialog's list. When you are finished selecting independent variables from the list, click *OK* or press *Enter*.

A second dialog allowing you to select the plot type will appear. The default plot type is *Line*, which when used means the xy-coordinates of added series will be connected by a line. The plot type selected from this dialog will be used for all series generated from this sequence of dialogs.

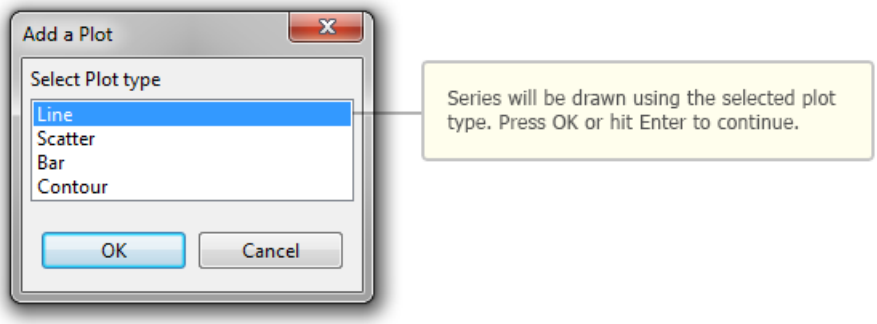


Figure 3.17: The CSV plot type dialog.

Once you have chosen your desired plot type, click *OK* or press *Enter*.

A third dialog allowing you to select the features that will actually be plotted will appear.

The list on the left includes the *independent variables* selected in one of the previous dialogs. You must select at least one of these independent variables, as they provide the x-coordinates of series generated from the dialog.

The list on the right includes all *features* available in the file. You must select at least one of these features, as they provide the y-coordinates of series generated from the dialog.

You can also select multiple variables from either list by holding the *CTRL* key while clicking variables, although you should note that every combination of selected independent (x) and feature (y) variables will be plotted.

Once you have selected your desired x- and y-axis variables, click *OK* or press *Enter*. A new plot will be added to the list in the *CSV Plot Viewer*. To open this plot, simply click it, and it will open in a new *CSV Plot Editor*.

Showing/Moving Plots To re-open an existing plot, click its item in the *CSV Plot Viewer*, which is usually located on the left. The associated *CSV Plot Editor* in the main workbench space will be brought to the top or activated. You can also click on the associated *CSV Plot Editor*'s tab to open it, or you can click and drag its tab to move it to another spot in your workbench.

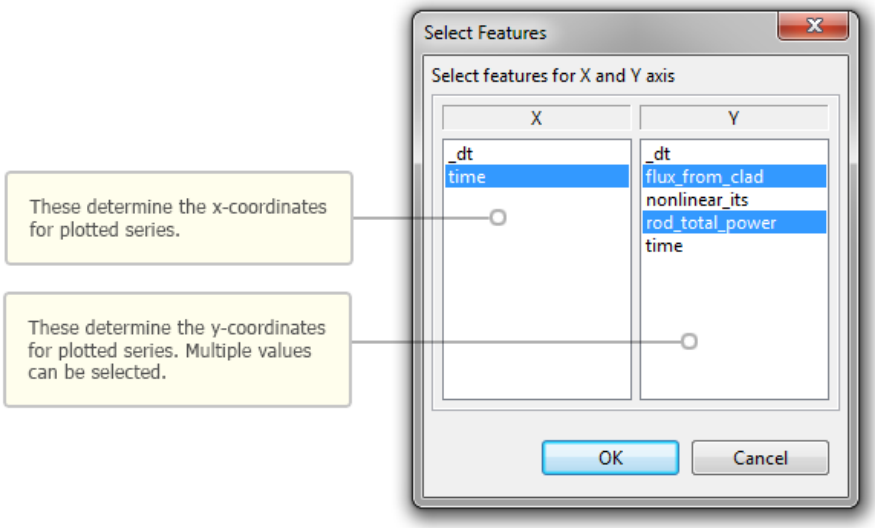


Figure 3.18: The CSV Plot features dialog.

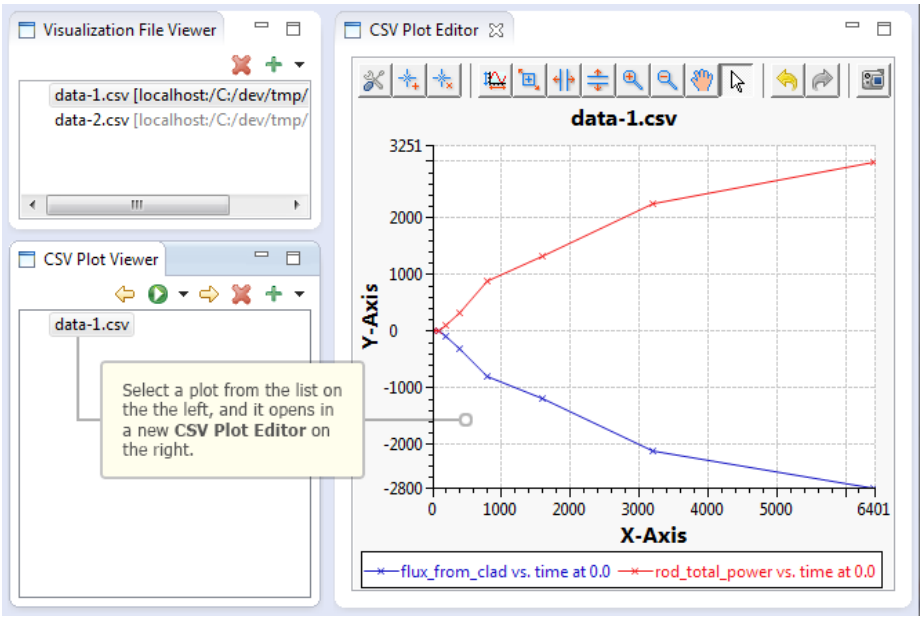


Figure 3.19: The ICE CSV Plot Editor.

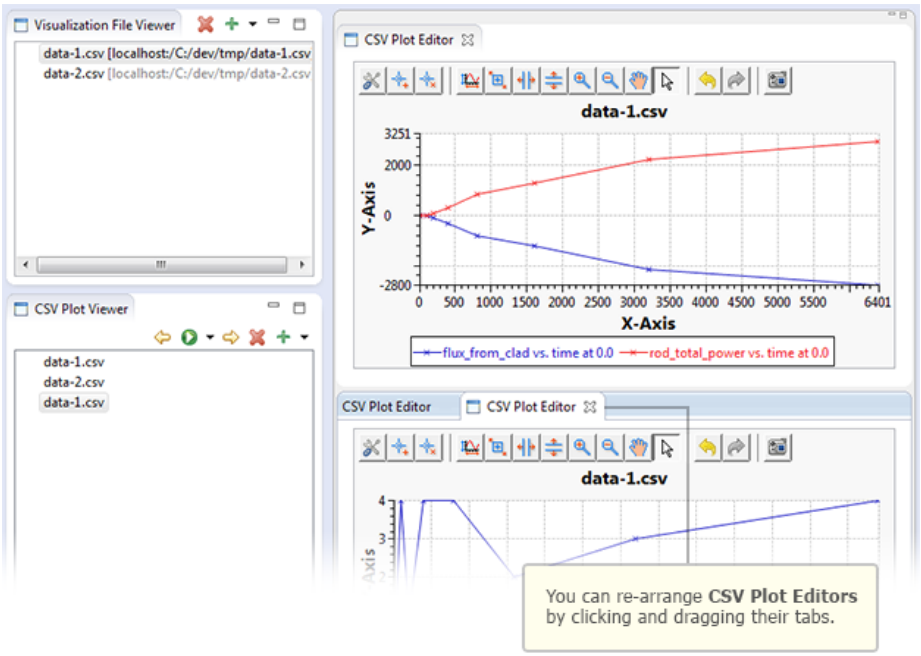


Figure 3.20: You can move multiple plots around through mouse-click and drag.

Removing Plots Lastly, if you would like to remove a plot from the *CSV Plot Viewer* list, select it and click the red "X" button. To permanently remove it from view, you will also need to close the *CSV Plot Editor* in the main workbench space.

Adding Series to a Plot To add more series to an existing *CSV Plot Editor*, you must first select the desired plot in the *CSV Plot Viewer*. Locate the green + button in the *CSV Plot Viewer* and click on the drop-down button next to it. Select *Add a series*.

You will then be prompted with the same sequence of dialogs as in the section on Selecting Initial Plot Data.

When you have finished selecting plot data as described in that section, the new data will be added to your selected plot as new series.

Plot Toolbar The plotting widget used by ICE's *CSV Plot Editor* includes a toolbar with helpful utilities for navigating your plotted data or customizing the plot's appearance. You can hover the mouse cursor over each button to view a tool tip describing what the button does.

Clicking the first button will open a dialog that allows you to customize the appearance of the plot or individual series on the plot, including titles, scales, grids, colors, and fonts. The last button allows you to save the current appearance of the plot to a `.png` image file. Feel free to try out the different utilities available in this toolbar.

Chapter 4

Developing MOOSE Applications with ICE

4.1 Introduction

This article is designed to walk MOOSE developers through a typical workflow for developing MOOSE-based applications in ICE. Since ICE is built on top of the Eclipse platform, a large variety of sophisticated software development tools and technologies for developing scientific software can be integrated into the ICE platform. Version control, code editing, code completion, code building, and code generation are just a few of the various technologies now available to MOOSE-based application developers using ICE. Additionally, after developing your custom MOOSE application, the usual MOOSELauncher and MOOSE-Model Items and the ICE Visualization perspective are still at your disposal for constructing input files, launching jobs, and visualizing results.

4.2 Cloning MOOSE

To clone MOOSE, simply switch to the Git Perspective in the top right corner of ICE. You will be presented with the following view.

Now click the 'Clone a Git Repository' button in the toolbar of the 'Git Repository' view (or the hyperlink in the middle of the view if you have not repositories). You will be presented with the following wizard.

Enter `https://github.com/idaholab/moose` into the URI entry and select next. This will present you with the branch selection wizard page. Select which branches you'd like to import in this clone and click Next. The last page will let you specify the clone location on your local filesystem. If you'd like this to be in your local ICE workspace entry /home/username/ICEFiles in the entry and click Finish.

To import MOOSE into your ICE Project Explorer, simply right click the

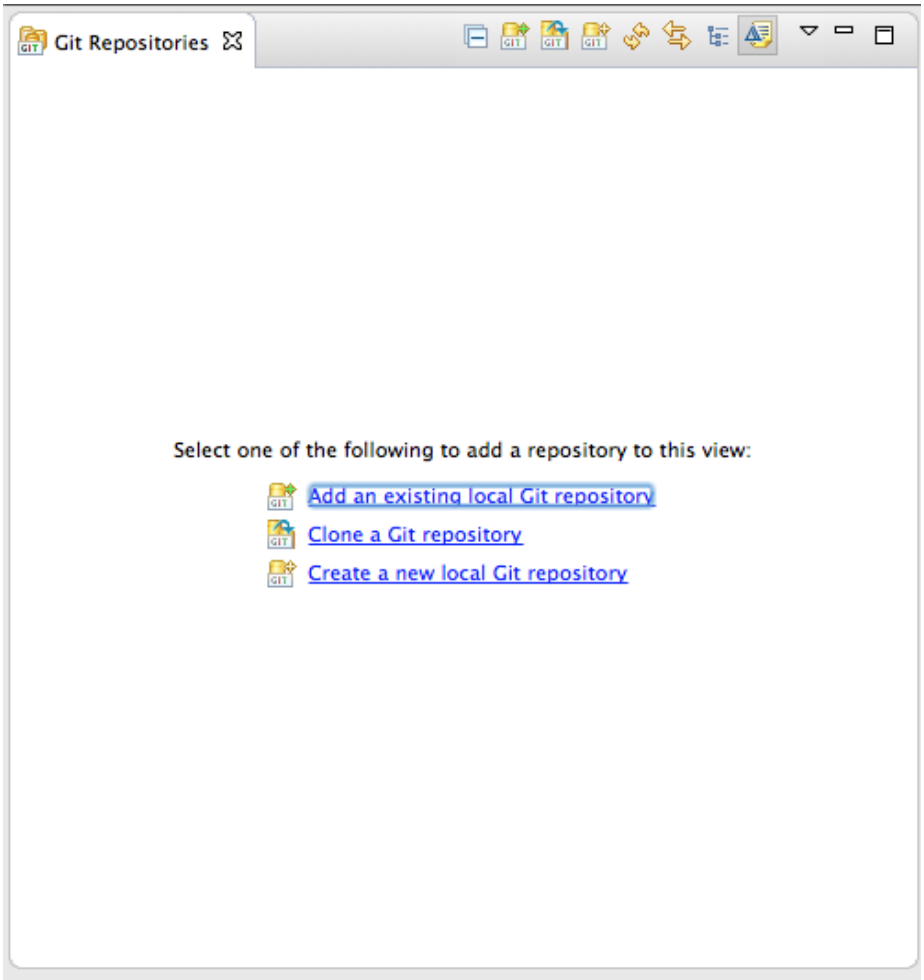


Figure 4.1: A view of the ICE Git Perspective.

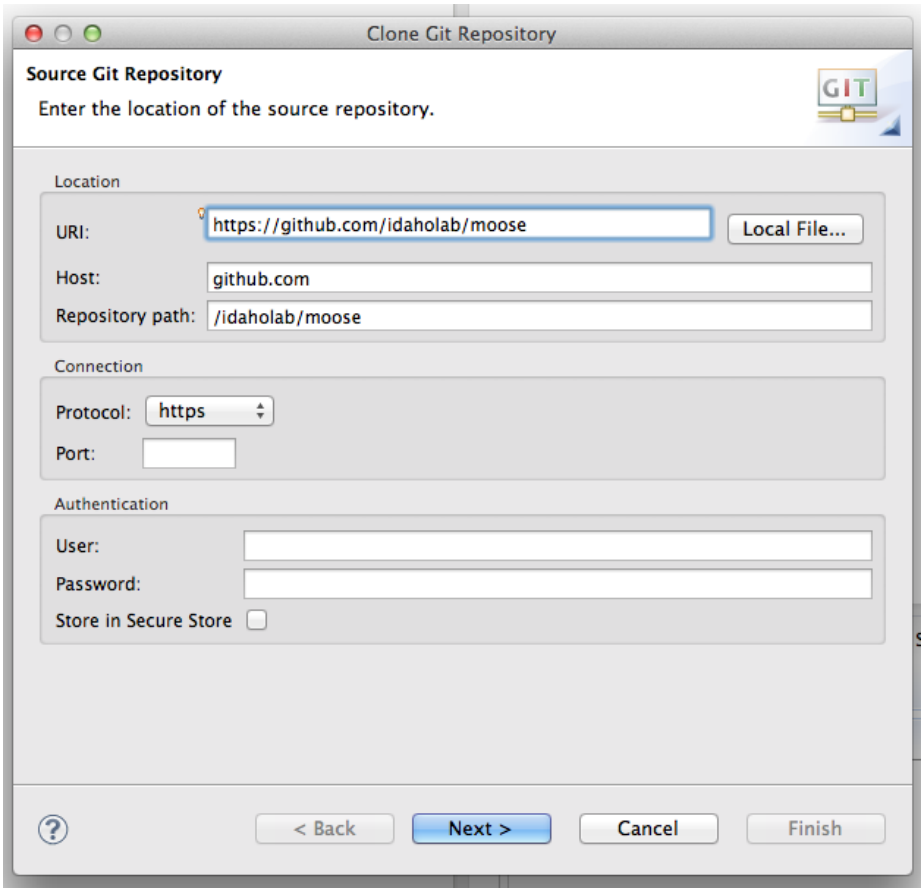


Figure 4.2: The first page of the Clone Repository Wizard requesting information about the remote Git repository URL.

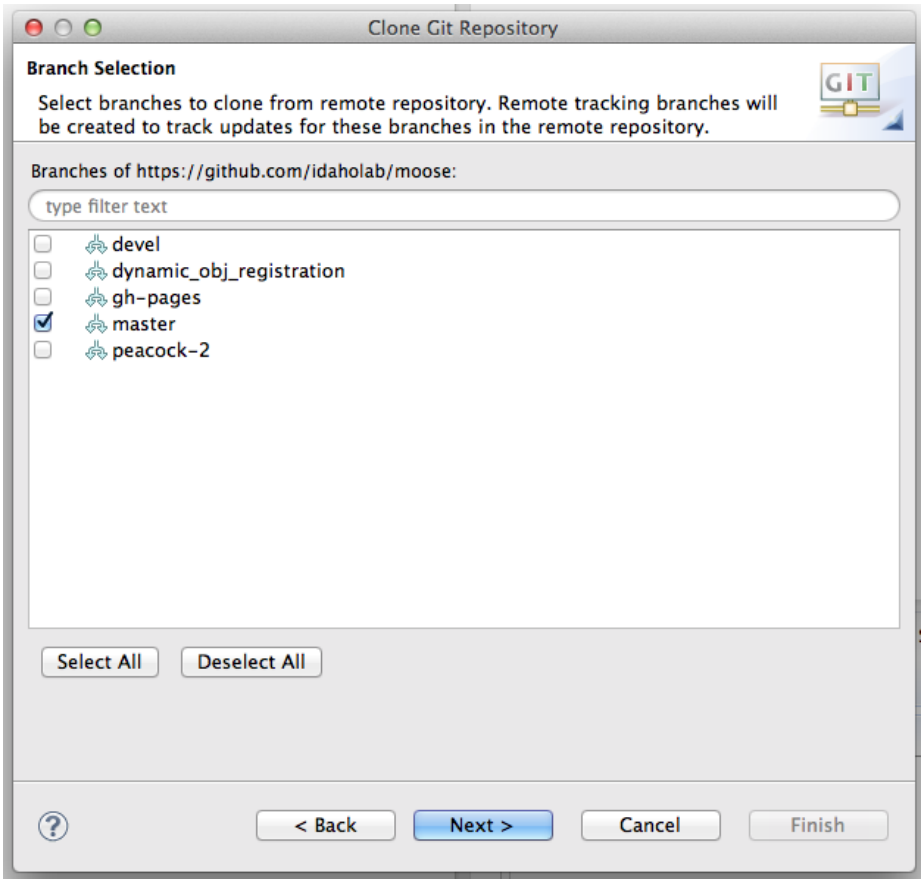


Figure 4.3: The second page of the Clone Repository Wizard requesting information about which branches to pull down.

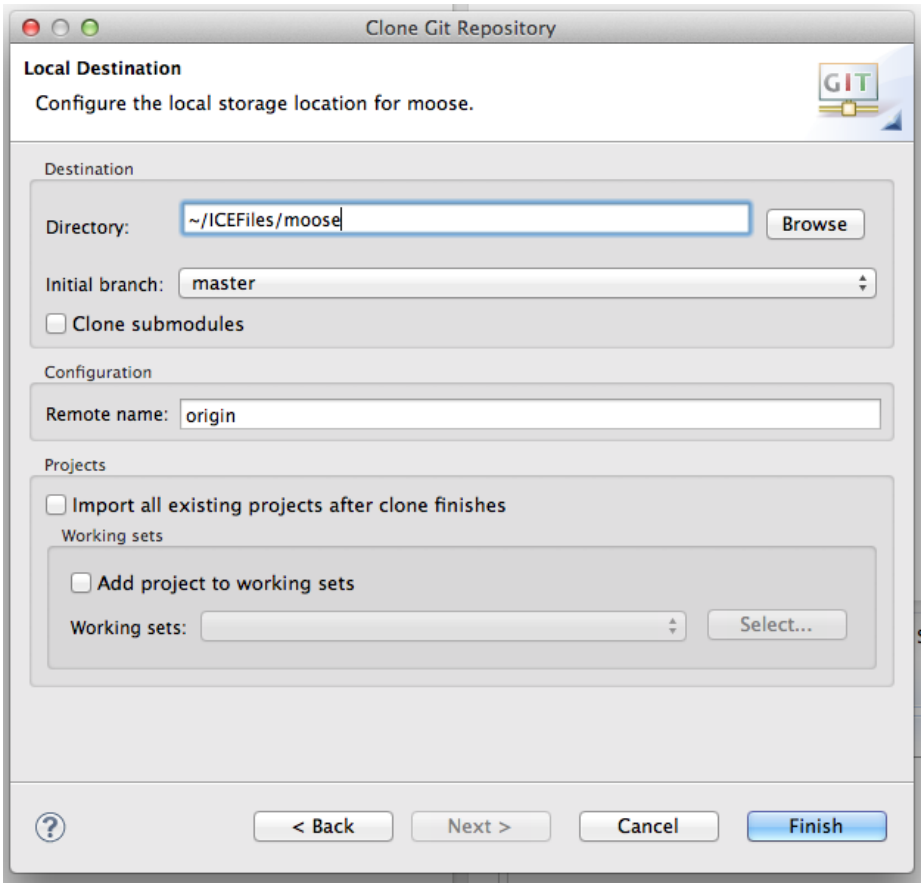


Figure 4.4: The final page of the Clone Repository Wizard requesting information about the local location of the repository.

created moose repository in the Git Repository view and select 'Import Projects'. On the first wizard page, select 'Import as New Project' and click finish. This will present you with the ICE New Project wizard. In this wizard, open the C/C++ tree node and select 'Makefile project with Existing Code'. Provide a valid project name and toolchain and click finish. You should see MOOSE in your Project Explorer.

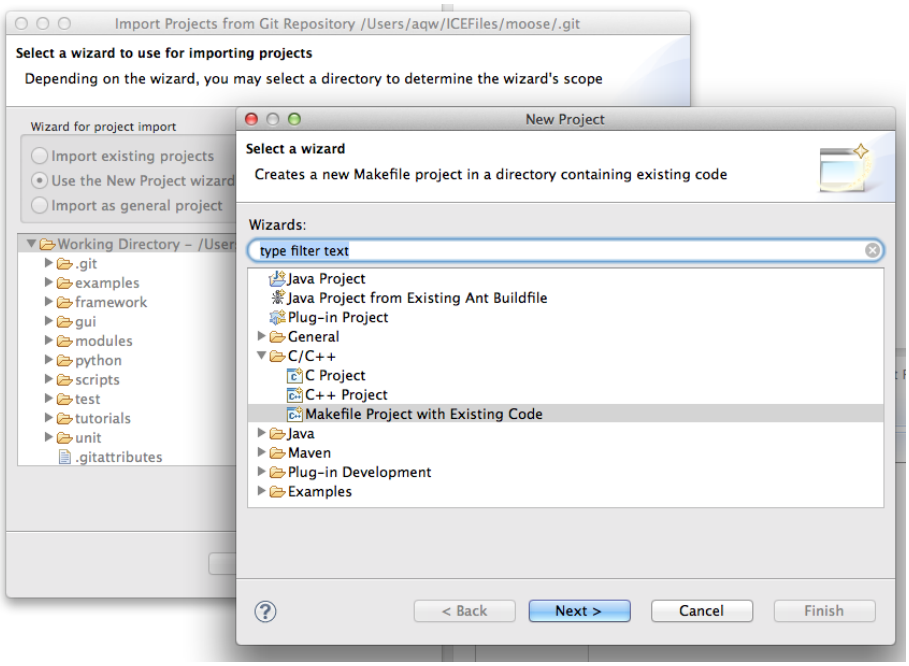


Figure 4.5: The import projects wizard for the ICE Git repositories view.

4.3 Building MOOSE

To build MOOSE/Libmesh within ICE, open the Make Target view by going to Window > Show View > Other and search and select Make Target. With MOOSE imported into your Project Explorer, you should see the MOOSE project in the Make Target view. Right click on that project and select New. A dialog will pop up prompting you for the Make Target name, target name, and build command. Set the name as 'Build Libmesh', uncheck 'Same as target name' and leave the Make target blank, uncheck 'Use builder settings' and set the command as 'sh scripts/update_and_rebuild_libmesh.sh', then click 'Ok'. Now you should see a 'Build Libmesh' target, which upon double-clicking will execute the update_and_rebuild_libmesh.sh script with the output streaming in the Console view.

Once that is done, you can create another Make Target in the same manner, this time setting the target as all, but setting the build command (assuming you have CMake installed on your system) as 'cmake -E chdir framework make' (feel free to add -j N to this command, where N is the number of make threads). If you do not have CMake installed, you can right click on the MOOSE project in the Project Explorer and select Properties. In this Properties dialog, select C/C++ Build and append to the Build directory entry 'framework'. Now, double-clicking this make target will execute the MOOSE build, and you should see the output streaming in the Console.

4.4 Forking the Stork

The internal MOOSE development team provides another GitHub repository called stork at https://github.com/idaholab/stork that represents the base structure needed to create a new MOOSE application. So 'Forking the Stork' implies forking this repository, changing its name to whatever you've decided to call your MOOSE application, and cloning that locally to begin work. The MOOSE team calls this 'Forking the Stork' and provides a link to the repository at mooseframework.org/create-an-app.

ICE now provides this functionality in an easy-to-use toolbar button using the tools provided by the Eclipse EGit plugins. To 'Fork the Stork' in ICE, simply click the 'MOOSE Fork the Stork' button in the toolbar.

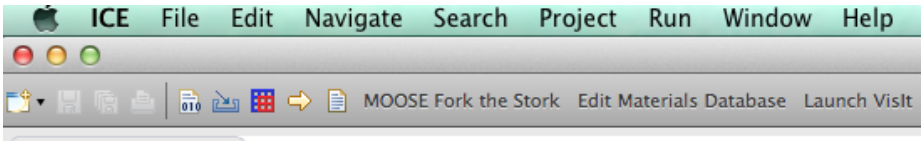


Figure 4.6: The ICE Fork the Stork button in the toolbar.

This will present a new dialog asking for the name of your new MOOSE application, as well as your GitHub username and password. Upon providing this information and clicking 'Ok', ICE will fork the https://github.com/idaholab/stork repository for you, rename it to your provided application name, clone it to ~/ICEFiles, and import it into ICE as a new C++ project in the C/C++ perspective's Project Explorer view.

Additionally, the import generates a fully configured Make Target in the Make Target view, and sets up the C++ Indexer to point to your ICE MOOSE project's include files. This is essential for providing code completion and MOOSE code search while your developing your MOOSE application. To look at a MOOSE class that you've referenced in one of your application's source files, simply click the class name or the header file and click F3. ICE will take you directly to the declaration for that MOOSE class so that you can peruse and look up its method definitions.

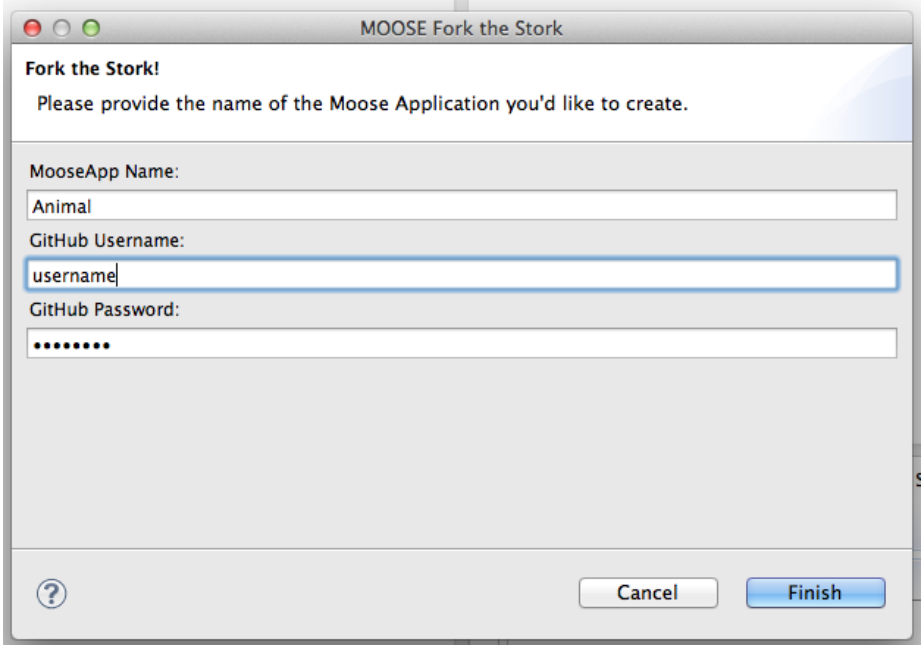


Figure 4.7: The Fork the Stork Wizard for entering your new MOOSE application name and your GitHub credentials.

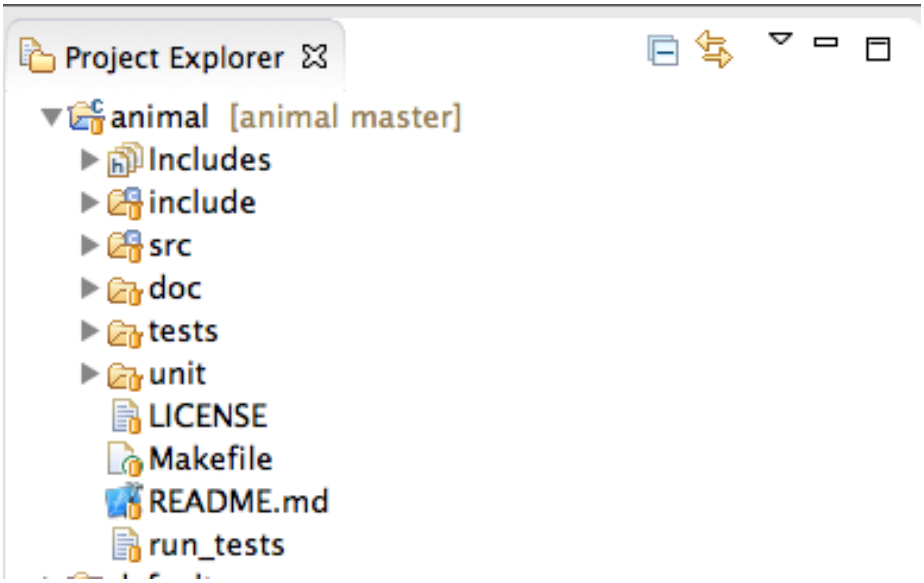


Figure 4.8: ICE creates the newly forked MOOSE application as a C/C++ project in the Project Explorer.

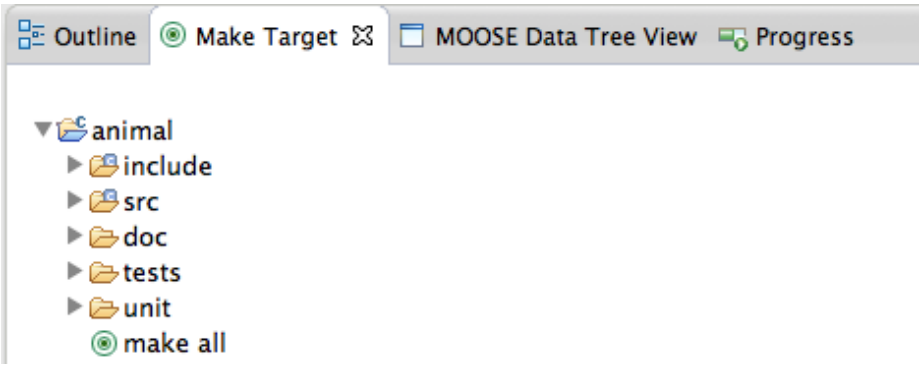


Figure 4.9: ICE adds a new Make Target to the Make Targets View for building your new application.

4.5 Adding a New Kernel

Once you've cloned and built MOOSE, and Forked the Stork to produce a new MOOSE application ready for development, you can easily create custom Kernels with ICE. To create a new Kernel, right click on your new MOOSE-based application project and select New > MOOSE Object > Kernel.

This action will display an input prompt asking for the name of your new Kernel subclass. Simply enter the name and push 'Ok'. Then ICE will automatically generate a new include and source file in include/kernel and source/kernel, respectively. The new files are the stubbed out, base implementation of a subclassed Kernel that you can then add to and modify.

4.6 Building your MOOSE App

Building your MOOSE application is simple because the 'Fork the Stork' action produced a Make Target for you. Simply double-click that make target and you application will build, producing the application executable.

4.7 Pushing Changes Back to GitHub

To push changes to the remote GitHub repository at `https://github.com/username/animal`, switch back to the Git perspective and click your applications git repository in the Git Repositories view. On the bottom right of the screen, you should see another set of tabbed views, one of them being the Git Staging view.

Click the Git Staging View and drag any Unstaged Changes to the Staged Changes section. Now provide a brief commit message and click 'Commit and Push', enter your GitHub credentials, and watch as your files are committed to the remote repository!

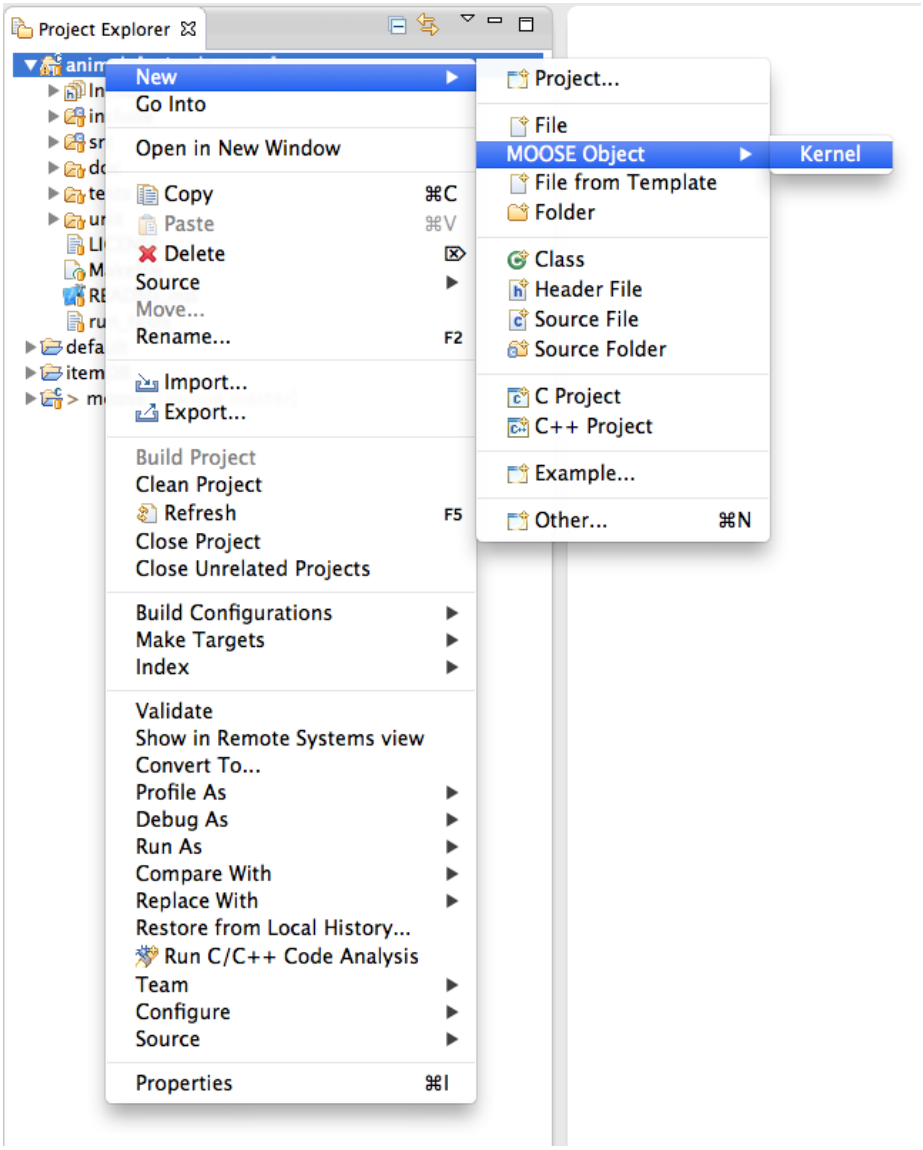


Figure 4.10: Creating a new Kernel in ICE is easy, just right click on your project and select New - MOOSE Object - Kernel.

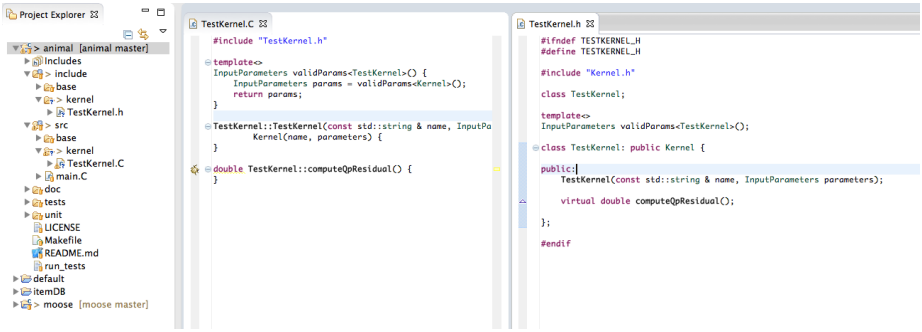


Figure 4.11: The created source code for the Add Kernel context menu action.

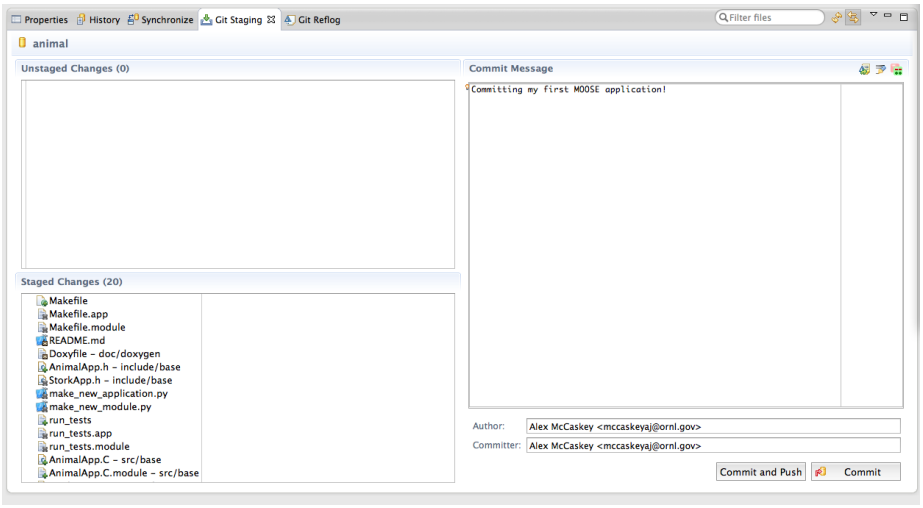


Figure 4.12: Committing your new MOOSE application is easy, just stage it in the Git Staging view of the Git Repositories Perspective.

4.8 Executing Built MOOSE Application

Now that you've developed a new MOOSE application you need to develop input files for it and execute it to see your desired results. This is simple with ICE: just use the built in MOOSE Model Builder and MOOSE Launcher Items. Detailed instructions can be found at Using MOOSE with ICE.

Online Resources

[1] Developing MOOSE Applications with ICE. https://wiki.eclipse.org/Developing_MOOSE_Applications_with_ICE.

[2] Getting ICE. https://wiki.eclipse.org/Getting_ICE.

[3] ICE Bugs. https://bugs.eclipse.org/bugs/describecomponents.cgi?product=Ice.

[4] ICE Build Instructions. https://wiki.eclipse.org/ICE_Build_Instructions.

[5] ICE Development Team. https://wiki.eclipse.org/About_ICE.

[6] ICE Eclipse Page. http://www.eclipse.org/ice.

[7] ICE FAQ. https://wiki.eclipse.org/ICE_FAQ.

[8] ICE Project Page. https://projects.eclipse.org/projects/technology.ice.

[9] ICE Source Code. https://github.com/eclipse/ice.

[10] ICE SourceForge Wiki. http://niceproject.sourceforge.net.

[11] ICE Youtube Channel. http://www.youtube.com/jayjaybillings.

[12] MOOSE. http://mooseframework.org.

[13] The ICE Wiki. https://wiki.eclipse.org/ICE.

[14] Using MOOSE with ICE. https://wiki.eclipse.org/Using_MOOSE_with_ICE.

[15] VisIt. https://wci.llnl.gov/simulation/computer-codes/visit/.

[16] Jay Jay Billings, Jordan H. Deyton, S. Forest Hull, Eric J. Lingerfelt, and Anna Wojtowicz. A domain-specific analysis system for examining nuclear reactor simulation data for light-water and sodium-cooled fast reactors. *Submitted to the Annals of Nuclear Energy*, abs/1407.2795, 2014.