# SCIRun User Guide

## SCIRun 4.7 Documentation

Center for Integrative Biomedical Computing
Scientific Computing & Imaging Institute
University of Utah

SCIRun software download:

**http://software.sci.utah.edu**

Center for Integrative Biomedical Computing:

**http://www.sci.utah.edu/cibc**

This project was supported by grants from the National Center for Research Resources **(5P41RR012553-14)** and the National Institute of General Medical Sciences **(8 P41 GM103545-14)** from the National Institutes of Health.

Author(s):
SCIRun Development Team

# Contents

**Chapter 1**

# SCIRun Introduction

**SCIRun is a *modular dataflow programming* Problem Solving Environment (PSE). SCIRun has a set of Modules that perform specific functions on a data stream. Each module reads data from its input ports, calculates the data, and sends new data from output ports.**

In SCIRun, a module is represented by a rectangular box on the Network Editor canvas. Data flowing between modules is represented by pipes connecting the modules. A group of connected modules is called a Dataflow Network, or net (see figure 1). An infinite number of nets can be created, each solving a separate problem. Chapter 1 of this tutorial demonstrates the use of SCIRun to visualize a tetrahedral mesh.

This chapter demonstrates the construction of a network comprised of three standard modules: ReadField, ShowField, and ViewScene. This chapter also instructs the user on reading Field data from a file, setting rendering properties for the nodes, edges, and faces (the nodes are rendered as blue spheres), and rendering geometry to the screen in an interactive ViewWindow.

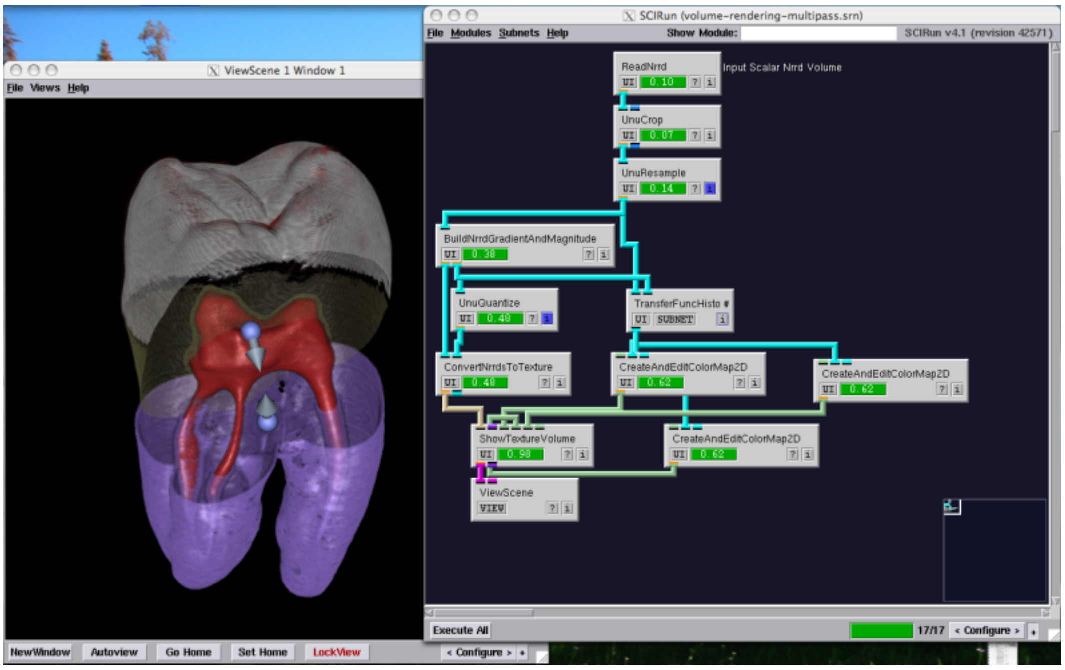


Figure 1.1. SCIRun Dataflow Network

## 1.1 Starting SCIRun

### 1.1.1 SCIRun usage

Use the –**help** flag for SCIRun usage and a full list of command line flags.

```
Usage: scirun [args] [net_file] [session_file]
    [-]-d[atadir]              : scirun data directory
    [-]-r[egression]           : regression test a network
    [-]-R[egressionimagedir]   : output directory for regression tests
    [-]-s[erver] [PORT]        : start a TCL server on port number PORT
    [-]-e[xecute]              : executes the given network on startup
    [-]-E[xecute]              : executes the given network on startup and
                                 quits when done
    [-]-c[onvert]              : converts a .net to a .srn network and exits
    [-]-v[ersion]              : prints out version information
    [-]-h[elp]                 : prints usage information
    [-]-p[ort] [PORT]          : start remote services port on port number PORT
    [-]-l[ogfile] file         : add output messages to a logfile
    [-]-t[imeout] N            : kill scirun after N seconds
    [-]-I[mage] file           : Save resulting images in file
    [--nosplash]               : disable the splash screen
    net_file                   : SCIRun Network Input File
    session_file               : PowerApp Session File
```

### 1.1.2 Preparation for Unix, Mac, Linux, etc.

Preparations must be made before starting SCIRun and its PowerApps. Commands below are typed in a terminal emulation application (e.g. xterm). See SCIRun's initialization file *$HOME/.scirunrc* for additional information. Change directory to SCIRun's build directory and start SCIRun from the command line:

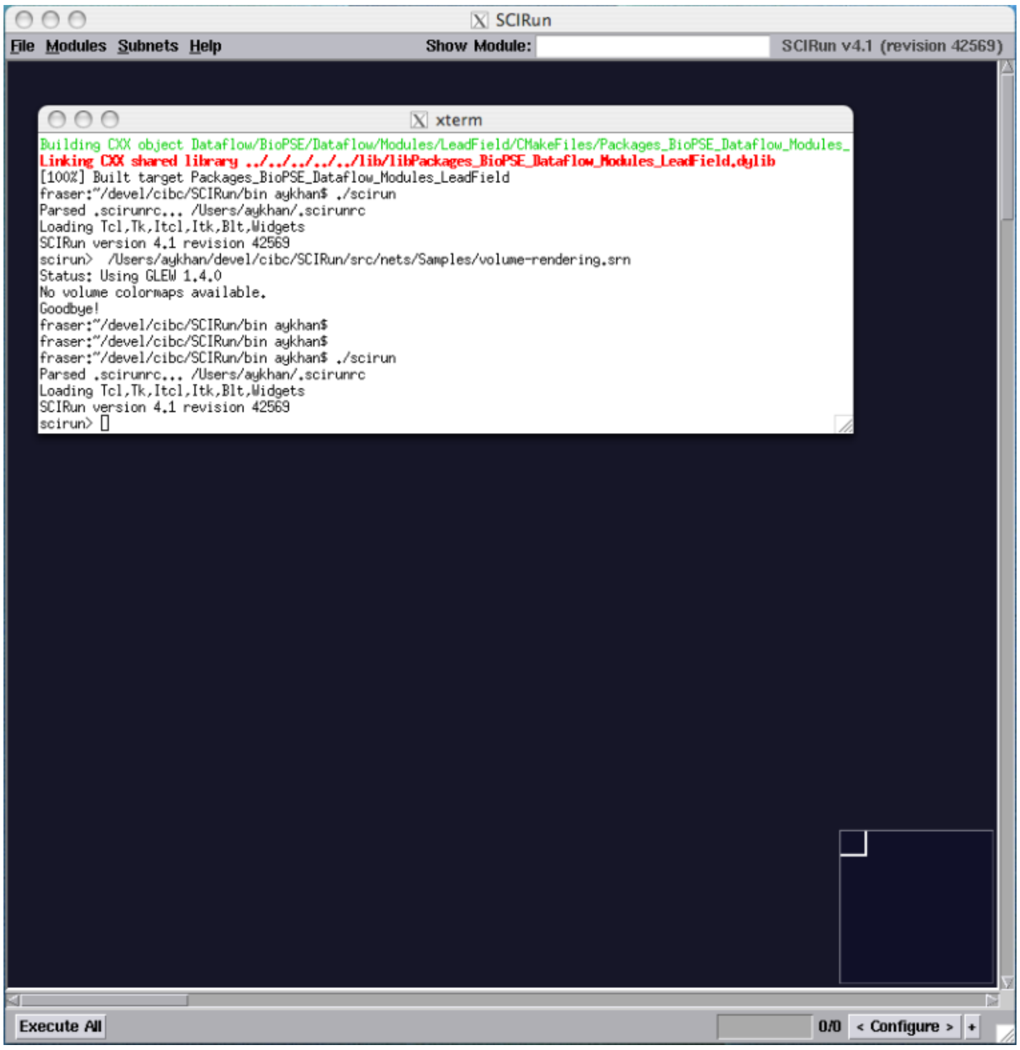
```
cd SCIRun/bin
./scirun [--execute] [network_file]
```

The **scirun** command may take the name of a SCIRun network file (network files have a **.srn** extension) for SCIRun to load. The **–execute** flag tells SCIRun to execute the network after it is loaded. Network files are discussed in a later section.

**Do not start SCIRun in the background, i.e. do not type: scirun &.**

#### Mac OS X 10.4

Run SCIRun in the X11 Terminal. If the binary package has been installed, the SCIRun application can be launched from Finder.



```
Building CXX object Dataflow/BioPSE/Dataflow/Modules/LeadField/CMakeFiles/Packages_BioPSE_Dataflow_Modules_
Linking CXX shared library ../../../../../lib/libPackages_BioPSE_Dataflow_Modules_LeadField.dylib
[100%] Built target Packages_BioPSE_Dataflow_Modules_LeadField
fraser:~/devel/cibc/SCIRun/bin aykhan$ ./scirun
Parsed .scirunrc... /Users/aykhan/.scirunrc
Loading Tcl,Tk,Itcl,Itk,Blt,Widgets
SCIRun version 4.1 revision 42569
scirun>  /Users/aykhan/devel/cibc/SCIRun/src/nets/Samples/volume-rendering.srn
Status: Using GLEW 1.4.0
No volume colormaps available.
Goodbye!
fraser:~/devel/cibc/SCIRun/bin aykhan$
fraser:~/devel/cibc/SCIRun/bin aykhan$
fraser:~/devel/cibc/SCIRun/bin aykhan$ ./scirun
Parsed .scirunrc... /Users/aykhan/.scirunrc
Loading Tcl,Tk,Itcl,Itk,Blt,Widgets
SCIRun version 4.1 revision 42569
scirun> 
```

Figure 1.2. SCIRun on OS X 10.4

### 1.1.3 Preparation for Windows

In Windows, SCIRun can be stared from Explorer, or launched from a “Windows Command Prompt” (cmd.exe). Alternatively, you can start it up by browsing My Computer or Windows Explorer to the directory that contains scirun.exe and double-clicking on it. The SCIRun initialization file *.scirunrc* can be found in the user applicaton data folder. Command line flags are the same as for Unix platforms.

```
cd ‘‘c:\Program Files\SCIRun 4.0\bin’’
scirun.exe [--execute] [network_file]
```

If you install the Windows binary version of SCIRun or associate .srn files with SCIRun, you can also double-click on any scirun network file (.srn) and it will run SCIRun with that network file. To associate srn files with SCIRun, do the following:

1. Open a Windows Explorer window, click the Tools menu and select Options
2. Click the File Types tab, and select New
3. In the Dialog that appears, type **srn** and click OK
4. In the Details for SRN extension frame, click **Change** and navigate to scirun.exe

### 1.1.4 Post startup

If the name of a network file is not provided, SCIRun starts with an empty window as shown below.

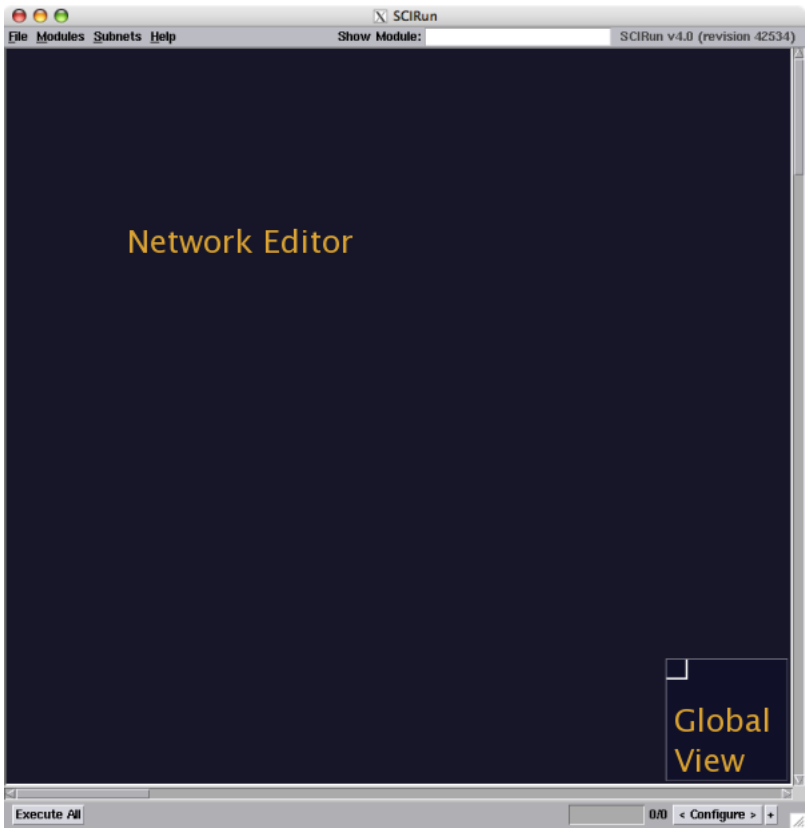


Figure 1.3. SCIRun initial NetworkEditor window

SCIRun may encounter errors during start up. Errors, warnings, and other messages are displayed in SCIRun's log console. Errors should be reported to the SCIRun development team at scirun-develop@sci.utah.edu. Developers with SCI or SCI collaborator accounts can report bugs and request features through the tracker on the SCI GForge portal.

### Remote Display

Normally, SCIRun's main window is displayed on a console that is part of the computer on which SCIRun runs. It is possible, however, to display SCIRun on a remote console. In the discussion that follows, the term local refers to the machine running SCIRun and the term remote refers to the machine displaying SCIRun.

*Advanced GL remote rendering is problematic.*

SCIRun is compiled against the GL driver on the local machine, but when displayed remotely, uses the remote machine's GL driver. Many times the local and remote drivers are not compatible and will result in SCIRun crashing. For this reason, if possible, try not to run SCIRun in this manner.

To display SCIRun remotely, the value of the **DISPLAY** environment variable must be set correctly on the local machine. Also, the local machine must be allowed to send display commands to the remote machine.

Normally, the remote machine makes a connection to the local machine via the ssh command. In this case, ssh sets the value of DISPLAY in such a way that the local machine has permission to send display commands to the remote machine. However, ssh connections result in poor display performance because of encryption activity on the connection.

To increase performance, the value of DISPLAY is set after establishing the ssh connection.

For a sh-style shell:

```
export DISPLAY=remote-machine-name:0.0
```

For a csh-style shell:

```
setenv DISPLAY remote-machine-name:0.0
```

Note that this technique defeats the encryption protection on the connection.

After overriding the value of DISPLAY set by ssh, the local machine will lack permission to send display commands to the remote machine. Use the xhost command on the remote machine to give permission to the local machine:

```
xhost +local-machine-name
```

## 1.1.5 Anatomy of the Main Window

The SCIRun main window consists of the Menu Bar, the Global View frame, the Message frame, and the Net Edit frame.

**Menu Bar:** The menu bar is used to load networks, save networks, quit SCIRun, create network modules, and perform other tasks. The menu bar consists of the following menu items:

**File:** The File menu contains the following items:

**Load:** Loads a network from a file

**Insert:** Adds a network to the NetEdit frame without overlap.

**Save:** Saves a network to a file

**Save As...:** Saves a network to a new file

**Clear Network:** Removes all modules and connections from the NetEdit frame

**Select All:** Selects all modules.

**Execute All:** Executes all modules.

**Create Module Skeleton...:** Contains the Create Module Skeleton option which guides a user through the creation of a basic module.

**Quit:** Quits SCIRun. Pressing key combination Control-q also quits. *Do not press Control-c to exit SCIRun: doing this will drop SCIRun into a debugger.*

**Modules:** This menu is composed of sub-menus. Each sub-menu corresponds to a category within the SCIRun package database. A category is a group of related modules. Each menu item in a category sub-menu creates a specific module and places it in the NetEdit frame. The NetEdit frame pop-up menu also provides access to the SCIRun and BioPSE (and possibly other) package menus. Activate the NetEdit frame pop-up menu by clicking the right mouse button (Btn3).

**SCIRun:** The SCIRun menu is used to create modules (from the SCIRun package) for use in the Net Edit frame.

**BioPSE:** The BioPSE menu creates modules (from the BioPSEpackage) for use in the NetEdit frame. It consists of category sub-menus and module menu items.

**Other Package Menus:** There may be other package menus if other packages have been installed. They also have category sub-menus and module menu items.

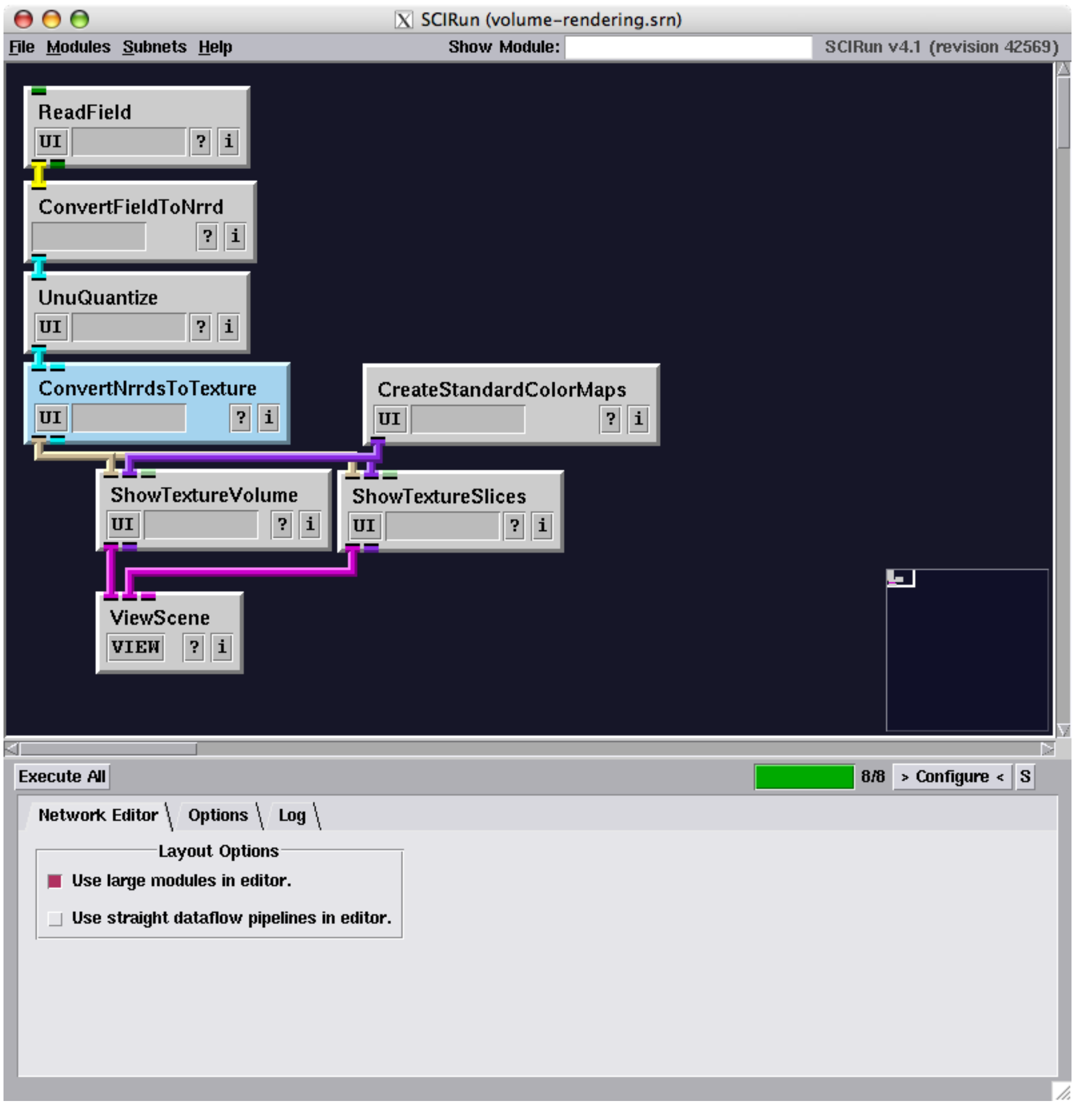
**Subnets:**

**Global View Frame:** The Global View Frame is located in the lower right corner of the main window. It is used to navigate complex networks.

**NetEdit Frame:** The NetEdit Frame occupies the main window. It is used to build and execute networks.

**Show Module:**

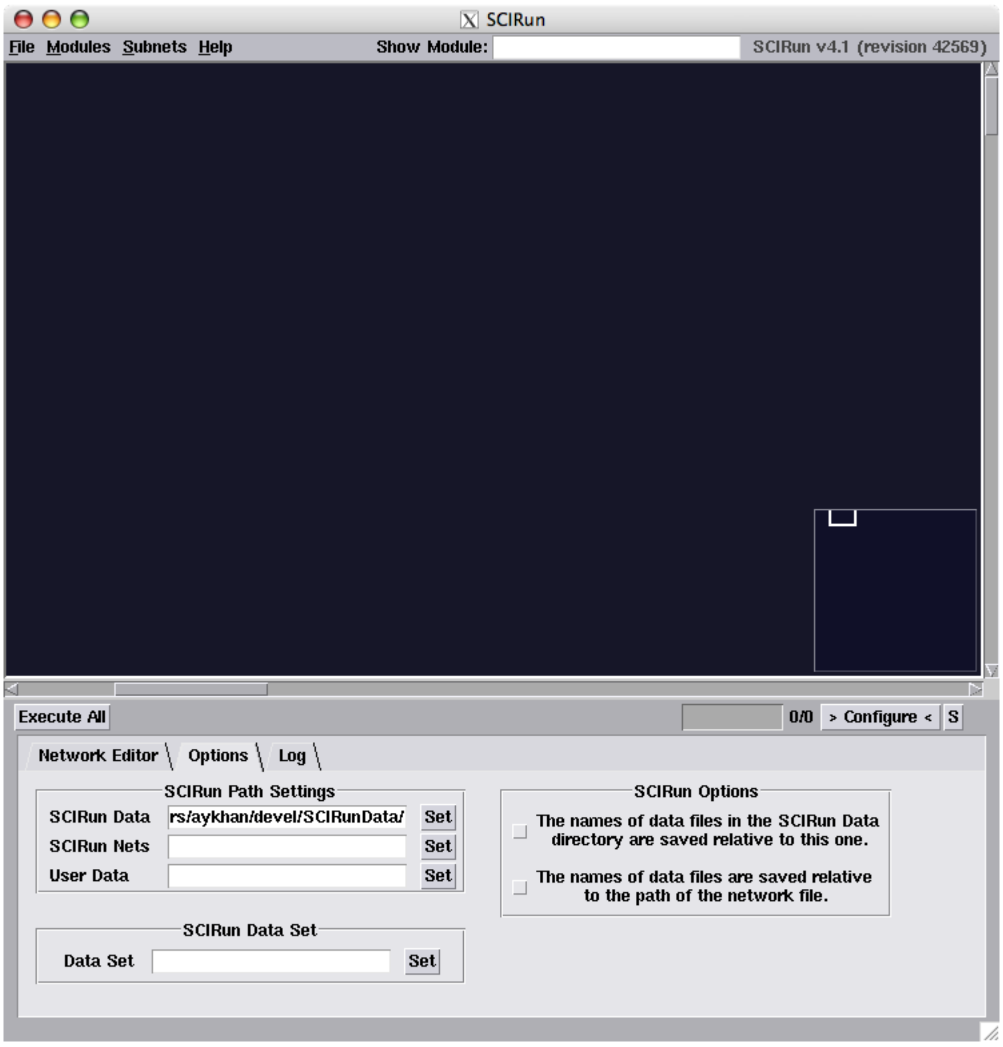
**Configure Frame:** This dockable frame contains three tabs. Toggle expanding the Configure frame by pressing the **Configure** button. Toggle the frame's docking mode by pressing the button to the right of the Configure button.



- [✓] Use large modules in editor.
- [ ] Use straight dataflow pipelines in editor.

Figure 1.4. SCIRun NetworkEditor configure options

**Network Editor:**



SCIRun Path Settings

| | | |
|---|---|---|
| SCIRun Data | rs/aykhan/devel/SCIRunData/ | Set |
| SCIRun Nets | | Set |
| User Data | | Set |

SCIRun Options

- ☐ The names of data files in the SCIRun Data directory are saved relative to this one.
- ☐ The names of data files are saved relative to the path of the network file.

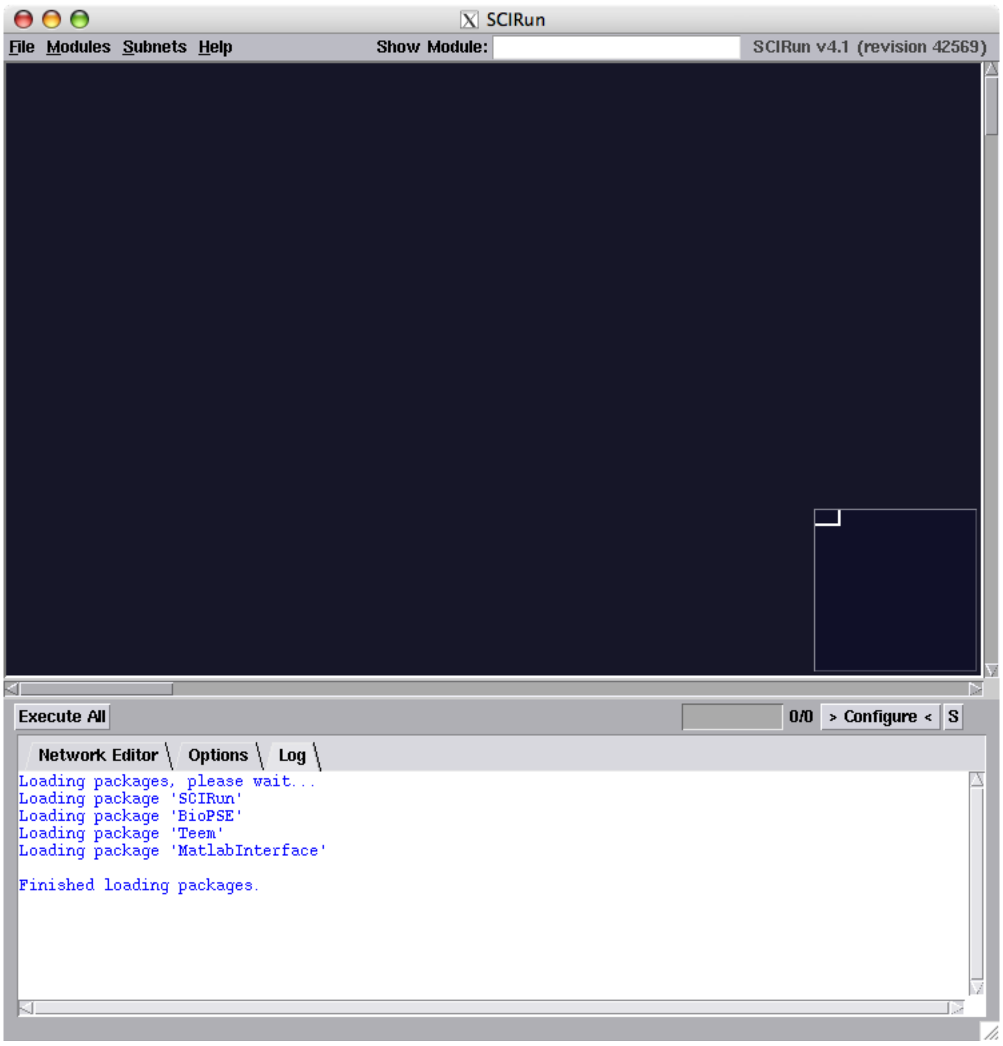
SCIRun Data Set

Data Set [ ] Set

Figure 1.5. SCIRun configure options

**Options:**

**Log:** Messages during program startup are displayed in the Message frame. The Message frame reports errors, warnings, and important information. Errors on startup may mean SCIRun has been installed incorrectly or has been installed from a buggy distribution; please report these errors.



```
Loading packages, please wait...
Loading package 'SCIRun'
Loading package 'BioPSE'
Loading package 'Teem'
Loading package 'MatlabInterface'

Finished loading packages.
```

Figure 1.6. SCIRun log messages

### 1.1.6 The Terminal Window

After starting, SCIRun runs a shell-like application in the terminal window called the SCIRun shell. The SCIRun shell displays the prompt `scirun>`. This program is a modified Tool Command Language (TCL) shell program. It is possible to type TCL'ish SCIRun commands at the prompt. A later section describes use of the SCIRun shell.

#### SCIRun's Environment

SCIRun responds to a number of variables to alter its default behavior. They can be in two places: the .scirunrc file, or in environment variables.

When started by a user the first time, SCIRun creates a default version of .scirunrc in the user's home directory (for unix-based environments) or in the directory that you compiled or installed scirun.exe in (for windows). Users may modify their default .scirunrc file.

SCIRun then processes the .scirunrc file on each subsequent startup. The .scirunrc file contains assignments to environment variables that affect SCIRun's behavior. Lines beginning with '#' are ignored.

Users may also set SCIRun related environment variables in their shell. Values of variables set in the shell override values set in .scirunrc.

See the content of .scirunrc for a complete list of SCIRun related environment variables. The following is a partial list of variables understood by SCIRun.

## The Initialization File

The contents of **.scirunrc** look typically like:

```
#
#  For more information, please see: http://software.sci.utah.edu
#
#  The MIT License
#
#  Copyright (c) 2009 Scientific Computing and Imaging Institute,
#  University of Utah.
#
#
#  Permission is hereby granted, free of charge, to any person obtaining a
#  copy of this software and associated documentation files (the "Software"),
#  to deal in the Software without restriction, including without limitation
#  the rights to use, copy, modify, merge, publish, distribute, sublicense,
#  and/or sell copies of the Software, and to permit persons to whom the
#  Software is furnished to do so, subject to the following conditions:
#
#  The above copyright notice and this permission notice shall be included
#  in all copies or substantial portions of the Software.
#
#  THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
#  OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
#  FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL
#  THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
#  LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
#  FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER
#  DEALINGS IN THE SOFTWARE.
#

#
# Sample .scirunrc file.
#
```

```
# This file is copied from the SCIRun src tree file:
#   ./SCIRun/src/scirunrc
#
# Lines beginning with '#' are commented out.
#
# All variables in this file have been commented out or set
# to their default values.
#
# This file contains a list of environment variables that can be either set
# in your shell or here. If you set these variables in your shell, they will
# override the values in this file.
#
# All variable declarations must use the syntax:
#    Environment_Variable_Name = Value
#
# Boolean variable declarations use this syntax:
#    Environment_Variable_Name = <true,1,on,yes|false,0,off,no>
#

#####################################
### ENVIRONMENT VARIABLES SECTION ###
#####################################

# REMEMBER, if these variables are set in your shell environment, they will
# override the valuse in this file.

### SCIRUN_ACCEPT_LICENSE
# Records whether license was accepted
# SCIRUN_ACCEPT_LICENSE = true

### SCIRUN_LOAD_PACKAGE = [comma-seperated list]
#   The format of SCIRUN_LOAD_PACKAGE is a comma seperated list of
#   package names.  This list specifies the package libraries to load
#   when SCIRun is first brought up.
# SCIRUN_LOAD_PACKAGE = BioPSE,Uintah,Teem


### SCIRUN_SHOW_HIDDEN_MODULES = [boolean]
#   Show modules that have been set to be hidden. These hidden modules include
#   obsolete or unfinished modules. Use extra care when using these modules
#   as they may be deleted in a future version or may cause scirun to crash
# SCIRUN_SHOW_HIDDEN_MODULES = true

### SCIRUN_DATA = [path]
```

```
#   Directory location that 'Reader' modules will default to.
# SCIRUN_DATA = /usr/sci/data/SCIRunData

### SCIRUN_DATASET = [name]
#   Name of data set in SCIRUN_DATA to default to.
# SCIRUN_DATASET = sphere


### SCIRUN_NET_SUBSTITUTE_DATADIR = [boolean]
#   If true, module filename variables will be automatically substituted for
#   'SCIRUN_DATADIR' and 'SCIRUN_DATASET' in saved .net files.
# SCIRUN_NET_SUBSTITUTE_DATADIR = false

### SCIRUN_NET_RELATIVE_FILENAMES = [boolean]
#   If true, SCIRun will save all filenames relative to the location of the
#   network file in the .srn files.
# SCIRUN_NET_RELATIVE_FILENAMES = true


### SCIRUN_MYDATA_DIR = [path]
#   Directory, or list of directories separated by colons,
#   that data 'Reader' and 'Writer' modules will list as
#   optional paths in the directories drop down menu.
# SCIRUN_MYDATA_DIR = $HOME/data:/scratch/data

### SCIRUN_NET_DIR = [path]
#   Directory, or list of directories separated by colons,
#   that the network editor will list as
#   optional paths in the directories drop down menu.
# SCIRUN_NET_DIR = [path]

### SCIRUN_TMP_DIR = [path]
#   Directory for writting temporary files - must be world read/writeable
# SCIRUN_TMP_DIR = /tmp

### SCIRUN_SERV_TMP_DIR = [path]
#   Directory for writting temporary files - must be world read/writeable
#   This is one is used by the external application interface. Both scirun
#   and scirunremote use this one to exchange files.
# SCIRUN_SERV_TMP_DIR = /tmp

### SCIRUN_CONFIRM_OVERWRITE = [boolean]
#   If false, SCIRun writer will not prompt the user before overwriting a file
# SCIRUN_CONFIRM_OVERWRITE = true
```

```
### SCIRUN_INSERT_NET_COPYRIGHT = [boolean]
#   If true, adds SCI copyright to any SCI .net files that are created.
# SCIRUN_INSERT_NET_COPYRIGHT = false

### SCIRUN_NOSPLASH = [boolean]
#   If true, SCIRun will not display the splash screen at startup.
# SCIRUN_NOSPLASH = false


### SCIRUN_HIDE_PROGRESS = [boolean]
#   If true, SCIRun will not display progress bars during startup and loading
# SCIRUN_HIDE_PROGRESS = false


### SCIRUN_FAST_QUIT = [boolean]
#   If true, SCIRun will not ask the user to save a network before quitting
# SCIRUN_FAST_QUIT = false

### SCI_REGRESSION_TESTING = [boolean]
#   If True, regression testing mode will turn on.  Execution starts after a
#   network is loaded and SCIRun will exit immediately after creating an image.
# SCI_REGRESSION_TESTING = false


### SCIRUN_REGRESSION_TESTING_TIMEOUT = [integer]
#   If set, and if SCI_REGRESSION_TESTING is True, SCIRun will automatically
#   kill itself after specified seconds.
# SCI_REGRESSION_TESTING_TIMEOUT = 300


### THREAD_NO_CATCH_SIGNALS = [boolean]
#   Turns off the SCI Signal handlers.
# THREAD_NO_CATCH_SIGNALS = true


### SCIRUN_DRAWARRAYS_DISABLE = [boolean]
#   Turns off use of glDrawArrays.  This should be set to true if you wish
#   to avoid drawing artifacts on _ATI_ Radeon cards under Linux or to fix
#   cross-platform remote display problems.
# SCIRUN_DRAWARRAYS_DISABLE = false
```

```
### SCIRUN_MPEG_LICENSE_ACCEPT = [boolean]
#   License information describing the mpeg_encode software can be found
#   in SCIRun's Thirdparty directory, in the mpeg_encode/README file:
#   This software is freely distributed.  That means, you may use it
#   for any non-commercial purpose.  However, patents are held by
#   several companies on various aspects of the MPEG video standard.
#   Companies or individuals who want to develop commercial products
#   that include this code must acquire licenses from these companies.
#   For information on licensing, see Appendix F in the standard.
#   For more information, please see the mpeg_encode README file.
#   If you are allowed to use the MPEG functionality based on the above
#   license, you may
#   enable MPEG movie recording in SCIRun (accessible via the SCIRun
#   Viewer's "File->Record Movie" menu) by setting the value of
#   SCIRUN_MPEG_LICENSE_ACCEPT to "true".
# SCIRUN_MPEG_LICENSE_ACCEPT = false

### SCIRUN_USE_DEFAULT_SETTINGS = [boolean]
#   Force the sizes for ShowField to calculate defaults when loading a new
#   Field.  Also Forces the Viewer to autoview whenever a new netork is
#   loaded.
#SCIRUN_USE_DEFAULT_SETTINGS = true

### DISABLE_MATLAB_PROCESS
# Disable forking of SCIRun on start of SCIRun to support Matlab running in
# separate process
# DISABLE_MATLAB_PROCESS = true

############################
# GUI PREFERENCES SECTION ##
############################

### SCIRUN_GUI_UseGuiFetch = [boolean]
#   Allows the user to tell a module to fetch the GUI from wherever it
#   happens to be on the screen and bring it to the mouse, and then
#   return it to its previous location.
SCIRUN_GUI_UseGuiFetch = on

### SCIRUN_GUI_MoveGuiToMouse = [boolean]
#   Makes GUIs appear near the mouse on GUI creation.
SCIRUN_GUI_MoveGuiToMouse = on


### SCIRUN_STRAIGHT_CONNECTIONS = [boolean]
```

```
#   Use straight lines to represent connections between modules in the
#   network editor.
# SCIRUN_STRAIGHT_CONNECTIONS = off

### SCIRUN_LARGE_MODULES = [boolean]
#   Use a large module layout instead of a smaller one
# SCIRUN_LARGE_MODULES = off


#################
# UNDOCUMENTED ##
#################
# LOGNAME
# USER
# SCI_DEBUG
```

## 1.2 SCIRun Data Variables

For convenience, the SCIRUN_DATA environment variable (and optionally SCIRUN_DATASET) can be set before starting SCIRun. Alternatively, use the –**datadir** flag when starting SCIRun.

**SCIRUN_DATA:** Points to the location of the SCIRunData directory on the system. Note: it is assumed the SCIRunData directory has already been downloaded (this is a separate download from the SCIRun build).

**SCIRUN_DATASET:** Indicates which dataset to load. For example, in SCIRunData, SCIRUN_DATASET can be aneurysm, sphere, utahtorso etc.

**SCIRUN_DATAFILE:**

### Setting environment variables

For unix environments, environment variables may be set as shown above in the SCIRUN_DATA section.

For windows environments, follow these directions: Click the Start button, select Control Panel, and then System. In the Advanced tab, click the Environment Variables button. Select New and enter the name and value of the environment variable. If it already exists, click on it and select Edit to modify it.

Chapter 2

# SCIRun Networks

## 2.1 SCIRun Network Building Blocks

### 2.1.1 Modules

A module is a single-purpose unit that functions within a dataflow environment. Modules have at least one input port for receiving data, located at the top of the module, or one output port for sending data, located at the bottom of the module.

For a more complete description of modules, consult the SCIRun Modules, Networks, and Sub-Networks Section of the User Guide.

All modules have an indicator that alerts the user to messages that exist in a module's log. Different colors represent different types of messages. Gray means no message, blue represents a Remark, yellow a Warning, and red an Error. To read messages, click the module's indicator button to open the log window.

- module categories

### 2.1.2 Connections

Data is transfered from one module to another using dataflow connections,commonly refered to as **pipes**. Each dataflow pipe transfers a specific datatype in SCIRun, denoted by a unique color. Pipes run from the output **port** of one module to the input port(s) of one or more other modules. Ports of the same color correspond to the same datatype and can be connected. These colors are described in detail in the Anatomy of a Module Section of the User Guide.

## 2.2 Loading Data

Load data using reader or importer modules in the DataIO module category. Supported file formats can be found in chapter 4.

Chapter 3

# SCIRun Objects

## 3.1 Mesh

SCIRun meshes are classified as unstructured, structured, or regular. A mesh consists of nodes and implicit or explicit node connectivity information.

Node locations and connectivities for unstructured meshes are specified explicitly. Node locations are specified explicitly, and connectivities are known implicitly for structured meshes. Node locations and connectivities are known implicitly for regular meshes.

A regular mesh is more constrained than structured and unstructured meshes. Likewise, a structured mesh is more constrained that a unstructured mesh. A more constrained mesh can be trivially converted to a less constrained mesh by explicitly enumerating implicit properties. For instance, a LatVol mesh is converted to a StructHexVol by enumerating a LatVol mesh's node coordinates (connectivities are still implied). Also, a LatVol mesh is converted to a HexVol mesh by enumerating a LatVol's node locations and connectivities. Module ToStructured performs this type of conversion.

It is not meaningful to convert a less constrained mesh to a more constrained mesh. It is possible, however, to sample an unstructured field at regular intervals to create a regular field. For example, a HexVolField is interpolated onto a LatVol mesh using modules SampleLattice and DirectInterpolate.

### 3.1.1 Unstructured Meshes

#### CurveMesh

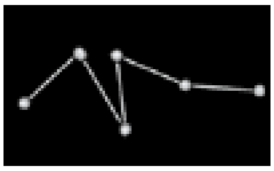


Figure 3.1. Curve Mesh

A curve mesh is a segmented curve.

#### HexVolMesh

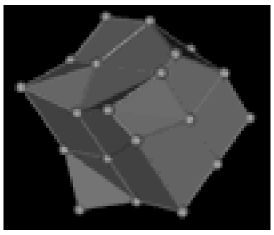


Figure 3.2. HexVol Mesh

A hex volume mesh is a subdivision of space into hexagonal elements.

#### PointCloudMesh

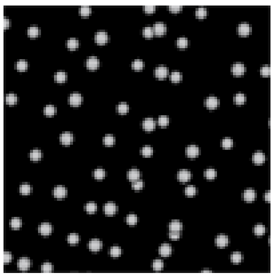


Figure 3.3. Point Cloud Mesh

A point cloud mesh is a set of unconnected points.

### PrismVolMesh

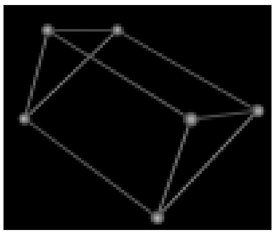


Figure 3.4. PrismVol Mesh

A Prism Volume mesh is a subdivision of space into prism elements. Prism elements have five faces, two trangular faces connected by three quadrilateral faces.

### QuadSurfMesh

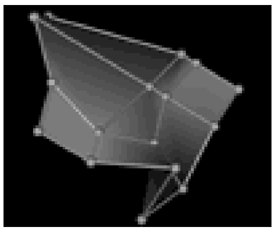


Figure 3.5. QuadSurf Mesh

A surface made of connected quadrilaterals.

### TetVolMesh

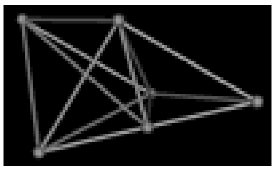


Figure 3.6. TetVol Mesh

A tet volume mesh is a subdivision of space into tetrahedral elements.

#### TriSurfMesh

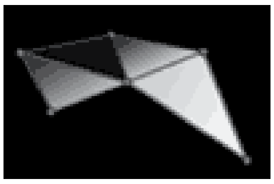


Figure 3.7. TriSurf Mesh

A tri surface mesh is a surface made of connected triangles.

### 3.1.2 Structured Meshes

#### StructCurveMesh

#### StructHexVolMesh

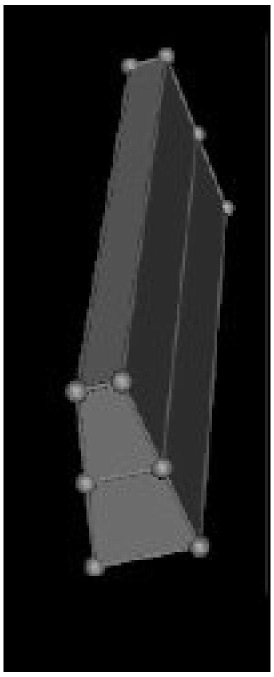


Figure 3.8. StructHexVol Mesh

A subdivision of space into structured hexagonal elements.

### StructQuadSurfMesh

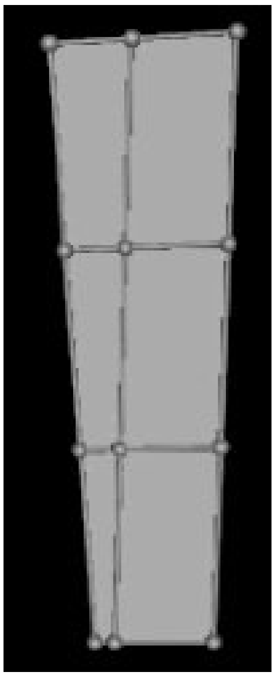


Figure 3.9. StructQuadSurf Mesh

A surface made of connected quadrilaterals on a structured grid.

## 3.1.3 Regular Meshes

### ImageMesh

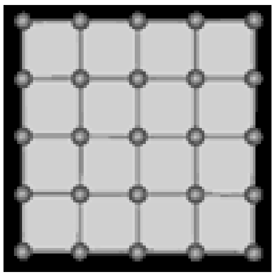


Figure 3.10. Image Mesh

An Image mesh is a regular 2D grid. Note that an Image mesh is not used for image processing.

#### LatVolMesh

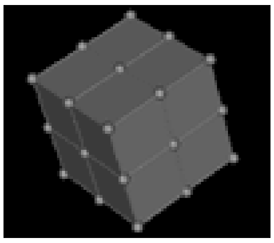


Figure 3.11. LatVol Mesh

A lattice volume mesh is a regular 3D grid.

#### ScanlineMesh

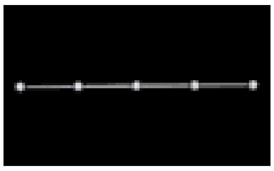


Figure 3.12. Scanline Mesh

A scanline mesh is a regularly segmented straight line (a regular 1D grid).

## 3.2 Field

A Field contains a geometric mesh, and a collection of data values mapped on to the mesh. Data can be stored at the nodes, edges, faces, and/or cells of the mesh.

## 3.3 Storing types in SCIRun Field

The dimensionality of the mesh type determines the available storage locations. For example, a TriSurfMesh has nodes, edges, and planar faces, but not cells, which are assumed to be three-dimensional elements. As a result, a TriSurf cannot store data in cells, but can store data in edges or faces.

Data types can be tensor, vector, double precision, floating point, integer, short integer, char, unsigned integer, unsigned short integer, unsigned char.

## 3.4 Matrix

SCIRun matrices are implemented as DenseMatrix, DenseColMajMatrix, ColumnMatrix and SparseRowMatrix subtypes.

**DenseMatrix** An MxN matrix using MxN storage units.

**DenseMatrix** An MxN column-major matrix using MxN storage units.

**ColumnMatrix** An Mx1 matrix using M storage units.

**SparseRowMatrix** A MxN matrix where most elements are zero and no storage is allocated for zero valued elements.

## 3.5 ColorMap

SCIRun ColorMap type is a mapping of color values to data values.

Chapter 4

# Supported File Formats

SCIRun supports the SCIRun Field format, the Teem nrrd format, Matlab MAT-files, and a number of text-based file formats such as JHU, OBJ, BYU and DIF. The full set of supported formats are listed in SCIRun reader and writer module user interfaces.

SCIRun can read files containing the following data:

- Node coordinates
- Node connectivities
- Structured mesh parameters
- Column matrices
- Dense matrices
- Sparse row matrices
- ColorMap parameters

SCIRun unstructured field types are stored in two or three text files. Curve, hex volume, quad surface, tet volume, and tri surface fields each require at least two files: a node coordinate, a connectivity. A matrix file containing data may also be read. Point cloud fields require no connectivity file.

Note that field readers read node coordinate and connectivity files only. Therefore, to construct a complete field from text-based files a ReadField module is used to read node coordinate and connectivity data and a ReadMatrix module is used to read field data from a matrix file. Outputs from modules ReadField and ReadField are sent to module SwapFieldDataWithMatrixEntries, which combines their outputs to create a complete field object.

When writing a field, use module SwapFieldDataWithMatrixEntries to split a field into a geometry stream and a data stream. Send the geometry stream to module WriteField, which writes node coordinate and connectivity files, and send the data stream to module WriteMatrix, which writes field data to a matrix file.

The structured (curve, quad surface, and hex volume) field types each require only a node coordinate file. SCIRun column matrix, dense matrix, and sparse row matrix data are stored in column matrix, dense matrix and sparse matrix files respectively. A Colormap is stored in a colormap file.