IRIS Explorer User's Guide
Release 3

Document Number NP2872
ISBN 1−85206−110−3

Chapter 2

Working with the Map Editor

This chapter explains, step by step, how to use IRIS Explorer tools to create and run maps. It describes how you:

- launch modules from the Module Librarian
- wire modules into a computational network, or map
- use the map to visualize numerical data
- select modules with compatible data types
- change the values of module parameters
- set parameter relationships in the Parameter Function Editor
- incorporate modules on remote systems into a map
- fix some common problems

2.1. Overview

In the Map Editor, you create an IRIS Explorer *map*. A map is a collection of *modules* of different kinds, connected in sequence, which process numerical data to accomplish a task. A module is a processing unit in a map. Each module accepts data, acts on it, and sends the result to the next module downstream. The Map Editor is the environment in which maps are created and executed, or "fired."

Maps can be used to perform a variety of tasks, for example, generating a visual image from a specific dataset. A patient may arrive at a medical clinic with an injured leg. The doctor generates a 3D CT scan, which shows a compound fracture of the femur. The CT scan data can be fed into an IRIS Explorer map and visualized as a 3D or "volume" image of the bone. The doctor may extract a 2D "slice" from the CT scan data along a plane close to the break in the bone. Then, if necessary, another slice can be taken from a slightly different angle for further elucidation. In this way, the doctor acquires a unique understanding of the fracture, which can help when deciding on an appropriate treatment.

Data is read into IRIS Explorer by the first modules in a map sequence, and the succeeding modules determine what form the results will take, and for what range of data values. For example, you can pinpoint areas where the bone is badly damaged by coloring the image according to bone density values.

The Module Librarian displays available maps and modules. You can launch single modules from the Module Librarian, arrange them in the Map Editor, and connect them to one another by their *data input* and *output ports*. These ports allow the flow of data between modules, the connections appearing as wire−like blue lines. Maps are usually arranged so that the data stream flows from left to right across the screen. You can also launch complete maps.

The Map Editor and Module Librarian menus provide you with options for manipulating the modules in a map. The modules themselves are accessible through their *control panels***.**These are rectangular panels, displayed on−screen, which contain *widgets*, the means of adjusting module parameters.

2.2. The Map Editor

The Map Editor (see **Figure 2−1**) is the work area in which you assemble modules for the purpose of organizing them into an operational map. A map is analogous to an application, in that it comprises a series of algorithms performing mathematical operations on data. Each module represents one discrete operation on the data.

2.2.1. Opening the Map Editor

To bring up the Map Editor, type **`explorer`** at the shell prompt in the shell window.

The Module Librarian and the Map Editor appear.

2.2.2. Quitting the Map Editor

To exit from the Map Editor, open the Admin menu and select *Quit.* When you do this, you automatically quit IRIS Explorer.

2.2.3. Editing Maps and Modules

You can select, copy, duplicate, delete, move, single or multiple modules by using the Map Editor. The modules may be free−standing or already wired into a map.

Use the Map Editor pull−down menus to:

- duplicate, cut, copy, paste, destroy, enable, disable, disconnect and reconnect modules (Edit menu). The operation is applied to **all** selected modules. In **Figure 2−1** only the *Render* module is highlighted as selected.

  Some of the Edit menu options have keyboard shortcuts listed next to them.
- group modules together, and open or close groups (Group menu).
- control layout within the Map Editor, such as of the connecting wires and the Log window (Layout Menu).

Alternatively you can use the drop icons at the bottom of the Map Editor window for some module operations such as destroying, disabling/enabling and disconnecting/reconnecting, see **Figure 2−1**.

Once a map is set up in the Map Editor, you can run it so that the modules fire and send data to one another. You can also incorporate several modules into a group or an application (see **Chapter 5, "Creating Groups and Applications"**).

The object visualized in **Figure 2−1** is a curved torus, generated by *GenLat*.

Figure 2−1 The Map Editor

2.3. The Module Librarian

The Module Librarian (see **Figure 2−2**) is a file browser and selector that provides access to IRIS Explorer modules and maps through its scrolling columns and Librarian shelf. Each Module Librarian has the name of its host system on the title bar.

Use the Module Librarian and its menus to:

- launch and save modules or maps in the Map Editor (File menu)
- reorganize modules into categories (Categories menu)
- store modules on the Librarian shelf (Display menu)
- add remote hosts if you are using a map distributed over several systems (Hosts menu)

Figure 2−2 The Module Librarian

2.3.1. Launching a Module

The act of bringing a module or map from the Module Librarian into the Map Editor and activating it is called *launching*. To launch a module or map, you can:

- Place the cursor on a module or map in a Module Librarian column and drag it to the Map Editor while holding the left mouse button down. Then drop it in place (the "drag and drop" technique).
- Place the cursor on the right side of the module or map in the Module Librarian and hold down the right mouse button. The Launch menu appears (see **Figure 2−3**). Select "Launch."

Figure 2−3 Module Launch Menu

- Drag and drop a module from the shelf of the Module Librarian (see **"Storing Modules on the Shelf"**).

- Use the "Open" option on the File menu. A file browser appears from which you can select a module or map (see **"File Browsers"**).

Maps in the Module Librarian are colored blue, modules are beige.

Finding a Module in the Librarian

Use the scroll bars in the Module Librarian columns to scroll up or down through the lists of available modules and maps. To scroll easily through the module categories:

- Put the cursor on the list and type the first letter or two of the module name. For example, if you want to launch the Render module, place the cursor on the list and type **`re`**. The list scrolls down to the first module name starting with **`Re`**. The Librarian is not case−sensitive, so you could also type **`RE`**.
- Place the cursor in the scrolling channel above or below the scroll bar and click on the mouse. The left mouse button moves you up or down one page at a time. The middle mouse button moves you up or down to the cursor position.

Placing the Module In the Map Editor

If you use the file browser to launch a module from a directory, the module comes up in the current launch position, which is the active point in the Map Editor. You can change the launch position by clicking the left mouse button on another spot in the Map Editor background before you launch the module. The next module is launched at that site.

To launch a module or a map from a remote system, select the remote host Librarian from the Hosts menu by clicking on the host name and then selecting the maps or module you want. You must first install the host in your host list, using the "New Host" option on the Hosts menu.

You can launch as many copies of a module as you like, although some modules might require special window server resources

Footnote: To launch many copies of modules that require a large number of color entries in the window server, such as *GenerateColormap*, **"Running the X Server "**.

. If you have more than one copy of a module in a map, the second copy will have the number **`<2>`** after its name. The third will have **`<3>`**, and so on.

2.3.2. Launching a Map

Maps, in the Module Librarian, can be identified as blue. You can launch a new map in the Map Editor by the "drag and drop" method from the Module Librarian (see **"Launching a Module"**).

If the map you want is not listed in the Module Librarian, you can select "Open" from the Module Librarian File menu. Use the file browser to find the directory and then choose the map.

The top left module in a map is always launched at the launch point.

If you want to start IRIS Explorer and launch a particular map at the same time, you can use the UNIX command line option *−map* with a filename.

Type:

```
explorer −map filename.map
```

The Map Editor opens and the map is launched and displayed automatically.

Note: If you are not in the directory where the map resides, either change to that directory before typing the command, or use the full pathname of the map, for example, */usr/explorer/maps/cfd.map.*

2.3.3. Saving Maps and Modules

Use the Module Librarian File menu to save your maps. You must save all IRIS Explorer maps with the filename extension *.map*; otherwise, IRIS Explorer will not be able to find the file when you try to open it again.

Click on the module title bar to select a module. The title bar and access pads will be highlighted in white. (See **"Selecting a Module"**).

A file browser comes up, and you are required to enter a filename for the map, for example, *NewMap.map*.

From the File menu you can select:
- "Save All," which saves everything in the Map Editor, or
- "Save Selected," which saves only the highlighted modules into the map

Modules can be saved only as part of a map, but you can save incomplete maps and bring them up later for further work.

2.3.4. Removing Maps and Modules

You can remove some or all modules from the Map Editor at a time.

To remove the current map completely, select "Destroy All" from the Map Editor Edit Menu.

To destroy only some of the modules in the map, highlight the modules (see **"Selecting a Module"**) and select "Destroy" from the Map Editor Edit menu.

To remove a single module, select "Destroy" from the module pop–up menu (see **"Using the Pop–up Menu"**).

To bring up a new version of the module, launch the module again from the Module Librarian. The new version of the module will be numbered **<2>** (or **<3>**, if it is the third version to be launched).

Example 2–1 Launching a Map

This example illustrates how to launch an existing map from the Module Librarian and carry out some simple operations.

1. In the Module Librarian, find the map called *simple*.
2. Drag and drop the *simple* map into the top left corner of the Map Editor. The map opens to show the control panel of each module in the Map Editor.

   Figure 2–4 shows the map you should see on your screen. It visualizes the data in the file */usr/explorer/data/lattice/testVol.lat*.

Note: On workstations without two–sided lighting the isosurface in *Render* may be dark. Change the Flip Normal? option on the *IsosurfaceLat* control panel from `No` to `Yes` to display the geometry properly.

Figure 2–4 A Simple Map Example

The *simple* map is a three–module map, consisting of:
- *ReadLat*, which reads in a file containing data for a lattice (lattices are discussed in **"Defining the Data Types"**)
- *IsosurfaceLat*, which creates an isosurface, a surface of uniform value in the test volume
- *Render,* which visualizes the selected isosurface of the lattice

All the module control panels are displayed with their wiring connections to one another visible. There is also a full–scale control panel for *Render*'s volume visualization. For more information on control panels, see **"Resizing the Control Panel"**.

When you first launch a map, all the module title bars and access pads are highlighted in white, to show they are selected. You can move the entire map by dragging on one title bar.

Before you can move individual modules, you must deselect them. Do this by clicking on the background of the Map Editor. A module is reselected if you click on the title bar.

You can:

- Click on the square *Maximize* button to bring up the full−scale control panel of *IsosurfaceLat*.
- Turn the dial on the *IsosurfaceLat* module control panel to change the threshold value of the isosurface, and hence, change the shape of the object displayed in *Render*.
- Experiment with the menus on the *Render* window menu bar and the decoration icons around the border.

For more information on using *Render*, see **Chapter 3, "Using Modules"**.

2.4. Modules

Modules are the individual units in a map. They are the mathematical engines, powered by data, that make maps work. Although their functions vary, they all share a basic structure (see **Figure 2−5**). Each module has a visible form, its *control panel*, by means of which you can direct the *internal* (and invisible) engine, or *core*.

The internal portion of the module consists of:
- module wrappers, which mediate between the core and the control panel
- the module core itself, which consists of a mathematical algorithm. Each module has a different algorithm, hence it carries out a different function in IRIS Explorer.

See the *IRIS Explorer Module Writer's Guide* for more information on the module core.

Figure 2−5 General Structure of a Module

The visible portion, or control panel, which appears in the Map Editor, is a rectangular window that contains:
- *input* and *output ports*, which allow the module to accept and pass on compatible data types. See **"Understanding IRIS Explorer Data Types"** for an explanation of data types.
- *parameters*, scalar data values which can be set and altered by widgets in the control panel. See **"Setting Module Parameters"** for a full description.

The control panel has three forms, which are described in **"How Modules Work"**. You can direct the activity in the module core through the ports and widgets on the module control panel.

2.4.1. What Modules Do

Modules accept data on their input ports, and pass it along via their output ports after they have modified it. When a module acts on data, or executes, it is said to *fire*. A module may be turned off, or disabled; and it can be fired on a computer other than the one where you are running the Map Editor.

General Module Functions

IRIS Explorer modules can be grouped according to their general function (see **Table 3−1** in Chapter 3). They can:
- read in data files (input modules, such as *ReadLat*)
- generate data from other data, such as extracting a planar slice or computing a numerical result from a volumetric dataset (feature extraction and analysis modules, such as *DisplaceLat*)
- process data to create a visual form (geometric representation modules, such as *IsosurfaceLat*)
- display the geometric representations on−screen (the *Render* module)
- write data to disk files (output modules, such as *WriteLat*)

Of these, the *Render* module stands on its own. It is the means whereby the data passing through the map is visualized and is likely to be an integral component of most maps. Once the data has been visualized as an object in the *Render* window, you can use the graphic capabilities of *Render* itself to enhance certain aspects of the object and to view it from various angles.

The other modules operate on the data in various ways to produce a geometric object that makes it easier to understand, interpret, and extrapolate from the original data. *Render* and some of the more commonly used modules are described fully in Chapter 3.

Specific Module Functions

To decide which modules you want in a map, you need to have more information about what each one does, what data it accepts, and what it produces for other modules to use, as well as what parameters it can have.

You can find information in these places:

- All the available IRIS Explorer modules are listed in the Module Librarian. You can move them from one category to another in the Librarian. For details, see **"Organizing the Module Librarian"**.
- To see the modules listed by function, refer to **Table 3−1** in Chapter 3.
- To see the modules listed by the data types they accept as input and produce as output, refer to **Table 2−1** through **Table 2−3**.
- To find out in detail what the module does, see the module definitions in *IRIS Explorer Reference Pages*, or use the module Help option (see **"Getting Help"**).

2.4.2. How Modules Work

When you launch a module from the Module Librarian, the module control panel (see **Figure 2−6** through **Figure 2−8**) appears in the Map Editor. Module control panels are rectangular windows that give you access to the capabilities of the module core via two main channels:

- the input and output ports, which allow you to pass data into and out of a module, and between modules in a map (see **"Wiring Module Ports Together"**). They appear on the *micro* and *Diminutif*™ forms of the control panel .
- the widgets, including buttons, dials, and sliders, which allow you to set module parameters (see **"Setting Module Parameters"**). They appear on the Diminutif and full−scale forms of the control panel.

The control panel also has:

- a title bar, which contains the module name and access pads for the input and output ports (see **Figure 2−6**).
- sizing buttons, for enlarging or closing down the control panel.
- a pop−up menu, which provides access to the module help files, parameter function editor, and other control options (see **Figure 2−9**).

Figure 2−6 Micro Control Panel/Title Bar

Selecting a Module

Before you can move or save a module, or use any of the Edit Menu operations on it, you must select it.

To select a single module, click on the module title bar with the left mouse button. The title bar and both access pads are highlighted. You can move the module around the Map Editor by holding down the left mouse button and dragging the mouse.

To select more than one module at a time, you can "lasso" them by clicking in the background of the work area, holding down the left mouse button, and sweeping out a rectangle that encloses the desired modules. When you release the mouse button, all modules that are completely enclosed by the lasso will be selected. Alternately, you can hold down the **`<Shift>`** key and click on each module in turn.

To select all the modules currently in the Map Editor, open the Edit menu and click on "Select All."

Use the **`<Shift>`** key with the "sweep" gesture to add more modules to an already selected group.

To deselect all selected modules, move the cursor off the title bar and click any mouse button. To deselect one module, hold down the **`<Shift>`** key and click on the module.

Resizing the Control Panel

The sizing buttons let you switch between the different forms of the module control panel. When you first launch a module, it appears in the Diminutif form (see **Figure 2−7**), which shows the title bar, port access pads, and general widget layout.

Figure 2−7 Diminutif Control Panel

- Click on the *Minimize* button to fold the panel into the *micro* form (see **Figure 2−6**). This saves space in the Map Editor. You can toggle back and forth between the Diminutif and micro control panels.
- Click on the *Maximize* button to open the *full−scale*, or maxi, control panel (see **Figure 2−8**); the Diminutif panel remains in the map. You can also open the maxi control panel from the micro control panel.

Figure 2−8 Full−scale Control Panel

The full−scale panel shows details of the widgets and the parameter values. You can move it outside the Map Editor window and enlarge it as much as you wish. You can also use the *Maximize* and *Minimize* buttons to open the maxi control panel to full−screen size or to iconify it.

The maxi control panel shows the names, ranges, and current values for widgets. It also has a menu bar with a Help menu, but does not show the ports.

To close a maxi control panel, click on the left button of the title bar and select "Close" from the Window menu, or double−click quickly on the left button of the title bar. The full−scale panel disappears but the Diminutif control panel remains in the network.

If you are working with a group of modules, use the "Select All" option from the Map Editor Edit menu and then choose "Make Micro" or "Make Mini" from the Layout menu to swap between these forms. You have to open each full−scale panel individually.

Note: If the maxi control panel disappears under other windows on the screen, you can pop it up again by clicking on the *Maximize* button on the Diminutif or micro control panel.

Using the Pop−up Menu

The module pop−up menu (see **Figure 2−9**) appears when you click on the title of the module control panel with the right mouse button. You can choose any one of these options:

Fire Now	The module fires and sends processed data to the next module downstream.
Duplicate	Makes a copy of the module, which you can wire into the map.
Disable	Temporarily deactivates the module. A disabled module will not fire until you enable it again. However, you can still set its widgets.
P−Func Editor	Opens the Parameter Function Editor, which lets you establish relationships among module parameters. It is described in **"Using the Parameter Function Editor"**.
Show Log	Brings up the module Log window, which displays all the module output resulting from the current map operation (see **"Logging Module Output"**.)
Help	Explains the purpose of the module and lists its ports and parameters.
Credits	Brings up information about the module's author.
Replace	Replaces the module with a newly−launched module with the same connections and parameter settings.
Interrupt	Interrupts a running user function without destroying the module. You may get unpredictable results from a map if you halt a module when the map is still firing.
Destroy	Completely removes the module from the map and the Map Editor.

Figure 2−9 Module Pop−up Menu

Using Drag−and−Drop Operations

The Drop Icons at the bottom of the Map Editor window can also be used to perform operations on a single module or many selected modules by dragging a module onto the icon. Press the middle mouse button over the module control title bar. This will show the "drag cursor", a representation of a module. Keeping the middle mouse button pressed, drag the cursor to the required icon and release to perform the operation.

If the module dragged has been previously "marked for selection" then the operation is applied to **all** the selected modules, otherwise the operation is only applied to the dragged module.

Destroy, *Replace*, *Help*, *Show Log* and *Disable/Enable* icons provide the same functionality as above options in the module pop−up menu, but can be applied to many modules. *Disconnect* and *Reconnect* icons provide an alternative to using the options on the edit menu.

Getting Help

Each module has a Help window associated with it, which explains the purpose of the module and lists its input ports, output ports, and parameters.

To open this window, click on the title bar of the control panel with the right mouse button and select "Help" from the pop−up menu (see **Figure 2−9**).

To close the Help window, click on the left button of the title bar and select "Close" from the Window menu, or double−click quickly on the left button.

The modules are listed alphabetically and described in detail in the *IRIS Explorer Reference Pages* and by −−> −−their individual Help files and *man* pages.

Replacing Modules

Replace provides a convenient one step method of disconnecting, destroying, re−invoking and reconnecting a module! This is particularly useful when developing a module. The module can be replaced, with the new version of the executable, in one easy step. The new module is re−invoked with all the old's existing connections, P−Func and Widget values. There are two ways you can instigate a replace:

- you can drag a module to the "Replace" icon, located at the bottom of the Map Editor, to replace it or if it has been selected, replace it together with all other selected modules.
- You can also use the "Replace" option on the control panel's pop−up menu to affect a single module.

Example 2−2 Creating an Incomplete Map

This example illustrates the first steps in creating a map.

1. Launch these modules from the Module Librarian by dragging and dropping each one in turn into the Map Editor:
   - a *ReadLat*module (to read data in a lattice format)
   - a *Contour* module (to create a contour plot of the lattice data)
   - a *GenerateColormap* module (to colormap the contours according to value)
   - a *Render* module (to visualize the contoured data)

   You now have the basic components of a map in the Map Editor.
2. Save these modules as a map by selecting "Save All" from the Module Librarian File menu. This will pop up a file browser.
3. When the file browser appears, in the "Selection" type−in slot, enter the name of the file in which to save the map. You can change directory, by editting the "Filter" type−in slot.

   Note: A map name must have *.map* as a suffix for IRIS Explorer to recognize and treat it as a map, for example, *incomplete.map.*

   When you have typed in a name, click on the OK button.
4. Now destroy the modules in the Map Editor by selecting the "Destroy All" option from the Map Editor's "Edit" menu. This will allow the saved map to be loaded onto a "clean slate".
5. To load the saved map, select the "Open" option, from the "File" menu in the Module Librarian. When the file browser appears, in the "Selection" type−in slot, enter the name of the map you saved previously

   You can add the map to the list in the Module Librarian by editing your personal configuration file. This procedure is described in **Appendix A, "Configuring Your IRIS Explorer Environment"**.

To launch this map and connect the modules, go to **"Wiring Modules Together"**.

2.5. Wiring Module Ports Together

Modules communicate by passing data from one to the next through their input and output ports (see **Figure 2−10** and **Figure 2−11**). Once you have modules in the Map Editor, you can connect them together individually or severally, disconnect them, change your mind about connecting them, and alter the appearance of the connecting wires.

Figure 2−10 Module Input Ports

Figure 2−11 Module Output Ports

Each module control panel has two port access pads, one for input ports and parameters on the left of the control panel, and one for output ports and parameters on the right. A single module can have several input and/or output ports.

Data port names are listed first on the port menu, parameter port names are listed next, followed by synchronisation port(s). The port name is followed by the IRIS Explorer data type that it can accept; that is, "Lattice," "Pyramid," "Geometry," "Parameter," and "Pick." Parameter ports are associated with widgets. If the port is optional, the type is followed by "(Opt)."

A connection is required on a module port unless the port is marked as optional (*Opt*). For example, *GenerateColormap* has an optional input port. A module will not operate properly in a map unless all its required ports are connected to compatible ports on other modules and are receiving data.

A module with parameter−based widgets always has an input and an output port for each parameter. For example, *BlendImg* has two data input ports and one parameter input port for "Blend" (see **Figure 2−10**). It has one data output port and a corresponding parameter output port for "Blend" (see **Figure 2−11**). Parameters are passed through the output ports and you can set and connect them so as to control widgets on other modules.

The synchronisation ports *Fire* and *Firing Done* are present on all modules, and as their names suggest can be used to control a firing sequence. See **Using the Synchronization Ports**. A *Firing Done* output port can be connected to a *Fire* input port. Loop controller modules have an additional output port *Loop Ended*. See **Constructing Loops**.

2.5.1. Making a Compatible Connection

To select an input or output port, click on the associated access pad using the right mouse button. A list of the input or output ports and parameters associated with that module appears (see **Figure 2−10** and **Figure 2−11**).

IRIS Explorer makes use of five data types: Lattice, Pyramid, Geometry, Parameter, and Pick. They are explained in **"Understanding IRIS Explorer Data Types"**. You can connect only input and output ports that accept the same data type.

For example, you can connect an output port called "Colormap − − Lattice" to the input port called "Input − − Lattice" but not to one called "Input − − Pyr." Only an output port of type "Pyramid" is compatible with the latter.

When you click on an output port to select it, all the compatible input ports on other modules in the Map Editor are highlighted in green. Likewise, when you select an input port, all compatible output access pads light up in a different shade of green. Incompatible ports are grayed out, so they cannot be selected.

Since a lattice can assume many forms, the Map Editor does type−checking for types of lattices and lets you connect only those modules that accept and produce the same type of lattice. For more information on lattices, see Chapter 3, "Using the Lattice Data Type" in the *IRIS Explorer Module Writer's Guide.*

2.5.2. Wiring Modules Together

Wiring modules together is a simple process. You can wire input to output port or output to input port. **Figure 2−12** shows a connection from the output port of *Contour* to an input port of *Render.*

To make the connections, follow these steps:

1. Click on the access pad of the first module and select a port from the cascading menu. The access pads on other modules in the Map Editor that have compatible ports are highlighted and you are in wiring mode.
2. Click on one of the highlighted pads and select a matching port from the port menu.

   Once you click on the second port, the highlight disappears and a blue wire is shown running between the ports, signifying that a connection has been made.

To cancel an incomplete connection, when only one port has been selected, click on the background of the Map Editor.

You exit the wiring mode automatically once a connection has been made.

Figure 2–12 Making Connections between Modules

2.5.3. Connecting Several Modules

You can wire more than one input port to a given output port, and vice versa, provided that the data types on the ports are compatible. This is called "fanning" connections out and in.

Figure 2–13 shows a *ReadLat* module with two connections on its output port, one to the input port of *Contour* and one to the input port of *Isosurface.* To make these connections:

1. Select the output port of the module you want to connect (*ReadLat* in **Figure 2–13**).
2. Select the input port of the second module (*Isosurface* in this case). A connection is made between the first and second modules *(ReadLat* and *Isosurface*).
3. Select the input port of the third module (*Contour* in this case).
4. Then select the output port of the first module (*ReadLat*) again. This port now has one other connection to it.
5. The cascaded port menu shows the names of the output module and port already connected, below a <*Connect*> bar. Click on the <*Connect*> bar to add the new output port to the list (in this case, *DiffLat*). The highlighting disappears, indicating that the connection has been made (see **Figure 2–13**). The number following the data type on the port menu indicates how many connections the port has.

Figure 2–13 Making Multiple Connections

To connect an input port to the output ports on several different modules, follow the same procedure.

2.5.4. Disconnecting and Reconnecting Modules

You may wish to disconnect a module or a number of modules completely from the map, or you may wish to simply break a single connection, between two modules. To break a single connection between two modules, follow these steps:

1. Choose either of the modules in the linkage and click on the access pad of the wired–up port.

   The cascaded port menu shows the name of the other module port connected to it (see **Figure 2–13**).
2. Select the connected port by clicking on its name (not on the <*Connect*> bar).

The connection between the modules is broken and the blue wire disappears. This method is useful if you wish to break only one connection.

If you wish to break all a module's connections then it is easier to "disconnect" the module, rather than breaking each connection in turn. There are two ways to disconnect a module:

- you can use the "Disconnect" sub–menu from the Map Editor "Edit" menu to break the connections of a selected

module or a number of selected modules. There are three options to the "Disconnect" sub−menu:
  - *All* – breaks all connections
  - *Input* – breaks input connections only
  - *Output* – breaks output connections only
- you can drag a module to the "Disconnect" icon. This operation breaks both input and output connections. As for all the drop icons, if the module has been selected, the operation is applied to all selected modules.

When you use "Disconnect", IRIS Explorer will "remember" a module's connections so that they can be re−wired later if desired using "Reconnect". If another disconnect operation is applied to it, only the connections most recently broken will be "remembered".

The "Reconnect" operation, using either the Map Editor "Edit" menu, or the "Reconnect" drop icon can be applied to a module after a "Disconnect". This operation will attempt to restore its connections to the state prior to the last "Disconnect". If an individual connection to another module cannot be re−instated, for example the other module has been destroyed, then that connection will be skipped. If a connection has already been re−instated, for example the user has already manually wired it up, then again the connection will be skipped. The operation will also work on a module that has been destroyed and re−launched.

2.5.5. Wire Routing

You can change the direction, or slant, of the wires that connect the modules in a map using "Wires" on the Map Editor Layout menu. Each option to "Wires" changes the alignment of the wires between modules, as their titles indicate.

To change the slant of the connecting wires, choose one of the three options, "Pt to Pt," "Right Angle," or "Diagonal." Click on each one in turn to see its effect.

It is an art to keep the wires in a map distinct and disentangled. However, no damage results from their being crossed or threaded underneath a module. The map merely becomes a little harder to read.

You can create more aesthetic maps by collapsing several modules into a group module. In this case all the selected modules disappear and are replaced by one control panel of your own design. For more details, refer to **Chapter 5, "Creating Groups and Applications"**.

Example 2−3 Wiring Modules Together

This example illustrates how to connect modules to make a functioning map (see **Figure 2−14**). In the previous example, you saved a set of four modules.

Figure 2−14 Example of Map Connections

Launch the map by dragging and dropping it in the Map Editor. Then wire the modules together and set widgets as follows:

1. Connect the Lattice output port of *ReadLat* to the "Input −− Lattice" port of the *Contour* module.
2. Type a filename into the file browser of the *ReadLat* module. A good lattice data file is in */usr/explorer/data/lattice/testVol.lat*.
3. Open the full−scale control panel for *Contour* by clicking on the square *Maximize* button (for more information, see **"Resizing the Control Panel"**).
4. Set the minimum and maximum values for *Contour* by turning the dials to 1.0 and 2.0.
5. Connect the Lattice output port of *GenerateColormap* to the "Colormap − − Lattice (Opt)" port of *Contour*.

   Notice that *Contour* has two connections on its input pad, but they go to two separate input ports.
6. Connect the "Contours −− Geometry" output port of the *Contour* module to the "Input −− Geometry" input port of the *Render* module. An object appears in the *Render* window representing contour curves of equal value through the data in the lattice file.
7. Move the slider on *Contour* to create more sets of contour lines.

8. Break the connection to *Render*. Notice that the contour lines disappear. Connect it up again and they reappear.
9. Disconnect *GenerateColormap* from *Contour* and see what happens.

If you want to keep the finished map, remember to save it again.

2.6. Firing Modules

When a module is activated, it fires and the user function operates on the data it has received from its input ports. As the module fires, its title bar turns yellow, and stays yellow until the module has completed execution. The color change is called *execution highlighting*. (This highlighting can be turned off by toggling the "Exec Hilite" option on the Layout menu). This might take a fraction of a second or several minutes, depending on the complexity of the user function and the amount of data it is processing.

Firing can occur at several different stages of the map−building process. A module fires when:

- it is launched in the Map Editor, provided it has all the data it needs on its input ports
- modules upstream of it in the map fire, sending data to it
- a module upstream of it in the map fires, that has a connection to its *Fire* port.
- one of its parameter values is changed
- you choose "Fire Now" from the module pop−up menu
- you connect it to an upstream module which has already fired and produced data on its output port
- you connect its *Fire* port to an upstream module which has already fired
- it is triggered by an external event; for example, data arrives on a port from an external file or process

2.6.1. Forcing a Firing

You can force a module to fire by:

- Selecting "Fire Now" from the module pop−up menu.

  This is useful for modules that have no parameters, for example, many of the image−processing modules, or if you do not want to change a parameter value.
- Changing the value of a module parameter.

  You change the value of a parameter by moving the widget that controls the parameter of interest. You can turn a dial, move a slider up or down, or switch a button off or on (see **"Setting Module Parameters"**).

Each time you alter a parameter or change a connection in a map, in effect feeding in more or altered data values, the affected modules automatically fire again. This can cause a cascade of firings downstream of the module.

For an in−depth explanation of the mechanism behind module firing, see Appendix B of the *IRIS Explorer Module Writer's Guide*.

2.6.2. Using the Synchronization Ports

When a module fires successfully a connection from its *Firing Done* port to a downstream module's *Fire* port can cause the downstream module to fire in exactly the same way that sending new data to the module would cause it to fire.

You can therefore use these synchronization ports to

- control a sequence of module firings rather than allowing the module firing mechanism to determine the ordering. Normally this is controlled by the data flow, therefore these ports are useful when there is no need for any data to be transferred between modules. For example wiring modules as shown in **Figure 2−15** will cause module firings to be sequential in the order D,C, B, when A is fired. Omitting the *Firing Done* to *Fire* port connections would cause the modules B, C and D to attempt to fire simultaneously.
- control successive firings of a module, for example a module that sends out a sequence of data on one of its output ports. The *AnimateCamera* module sends out a series of cameras, usually to *Render*. A new camera will be sent each time an upstream module that is connected to its *Fire* port, fires. Modules that are designed to fire repeatedly

are usually wired into a loop, rather than a map with a linear dataflow. See **Constructing Loops** for further discussion.

Figure 2–15 Controlling Module Firing with Fire Ports

2.6.3. Temporarily Disabling a Module

There are a number of ways in which module(s) can be disabled:

- you can use the "Disable" option from the Map Editor "Edit" menu to disable a selected module or a number of selected modules temporarily. The "Enable" option reactivates it or them again.
- you can drag a module to the "Disable" icon, located at the bottom of the Map Editor, to disable it or if it has been selected, disable it together with all other selected modules. The "Enable" icon can be used to reactivate it or them again.
- You can also use the "Disable/Enable" toggle option on the control panel's pop–up menu to affect a single module.

Disabling a module is useful if:

- You want to modify several module parameters without having the module fire every time you make a change.
- Your map contains a module that takes several minutes to finish firing.

Each time you made a connection to the module or altered a parameter, you would have to wait several minutes while it fired before you could continue. You can avoid the wait by disabling the module while you complete your operations, then enable it again when you are ready to fire the complete map.

If you disable a module that is connected in sequence with other modules, it will not send new data to the modules that follow it. If those modules rely entirely on the disabled module for data, they also cease to fire. The flow of data will be blocked, along this branch at least, until you reenable the disabled module.

2.7. Logging Module Output

When modules fire in the Map Editor, they often generate informative text messages. For example, *PrintLat*, *PrintPyr*, and *PrintPick* all output text strings describing their data. Other modules may also produce messages in the course of a map cycle, all of which are logged by IRIS Explorer. You may be interested in seeing all generated messages, or only those put out by a particular module.

2.7.1. Logging Map Data

The Map Editor Log window captures all the messages generated by modules in the Map Editor. It is a read–only scrolling text window that typically appears below the Map Editor (see **Figure 2–16**). Messages in the window scroll upward, with new lines inserted until the size limit of the Log is reached. After that, lines are deleted. The Log lists the output from each module as it occurs, with the name of the generating module as a prefix to each line.

Figure 2–16 The Map Log Window

You can hide or display the Map Editor Log by toggling the "Log Window" option on the Map Editor Layout menu.

2.7.2. Logging Module Data

To see text messages for an individual module without having to reviewing the entire Map Editor Log, you can open and examine a Module Log (see **Figure 2–17**). Each Module Log displays messages generated by that module alone.

Open the Module Log by selecting "Show Log" from the module pop–up menu (see **Figure 2–9**).

Figure 2–17 Module Log Window

2.7.3. Reviewing Log Messages

You can cut and paste the information displayed in the Log windows into other windows. By selecting the "Editable Log" option from the Log window pop−up menu (see **Figure 2−18**) you can type further text into the window, if, for example, you wish to annotate the output. The menu appears when you click the right mouse button in the Log window. To change the Log back to a read−only window (the default), select the "Un−Editable Log" option. The other two options let you clear the window and save its contents.

Figure 2−18 Log Window Pop−up Menu

You can also send the log output to the UNIX shell from which you launched IRIS Explorer. To have text messages sent to `standard output`, start IRIS Explorer with this command:

```
explorer −debug printAlert
```

If you use this option, you can still use the Log windows for reviewing text messages.

2.8. Understanding IRIS Explorer Data Types

In IRIS Explorer, the data that modules accept and pass on is characterized by the *data type.* The data type is the form that data takes in IRIS Explorer. There are five IRIS Explorer data types: lattice, pyramid, geometry, parameter, and pick.

All data is passed through the module input and output ports, each of which accepts or outputs only one data type. Modules may, however, have several ports with a different data type on each one.

2.8.1. Defining the Data Types

The IRIS Explorer data types are:

Lattice	A series of arrays, in one or more dimensions, that hold dimension variables, nodal data values, and coordinate values. For example, data that defines a volume, such as a 3D CT scan of a bone, is in lattice format. This is the most commonly used data type.
Pyramid	A hierarchy of lattices, with information on how the lattices are connected. The pyramid data type is used to manipulate finite element data and molecular data. For example, data describing a complex organic molecule and a finite−element mesh of a car body are both in pyramid format.
Geometry	A collection of geometric primitives, including polygons, lines, and points, used to build a visual representation of a dataset. For example, a contour surface from a CT scan is a collection of polygons. The vertices of each polygon can have associated color and surface normals. Data is displayed in the *Render* module in geometry format.
Parameter	A parameter holds a scalar value: an integer, real number, or string. The value of a parameter input to a module is controlled by a widget on the module control panel . You can also perform arithmetic functions on them using the Parameter Function Editor (see **Chapter 4**). Parameters are used to set values for slicing planes, error tolerances, quantities, and filenames.
Pick	A specialized data type that lets you select an area of an object in the display window of a render or image−display module and collect data about the selected area. This information can be fed to another module such as a query module.

It is possible to define new data types in IRIS Explorer, and you may find that the input and output port menus of some modules list data types in addition to those described here. Those are user−defined data types. They work exactly as IRIS Explorer data types do, handling data in a specific format.

The structure of the lattice data type is discussed in more detail in **Chapter 7, "Using the DataScribe"**. All the data types are described fully in the *IRIS Explorer Module Writer's Guide.*

2.8.2. Using the Data Types

The IRIS Explorer data types represent groups of data. A given instance of a data type on a port can be generic or highly specific. For example, a module may have an input port of Lattice type defined so that it will accept a wide variety of lattices of different sizes, or so that it will accept only 2D scalar lattices of real numbers with uniform coordinate mapping.

For example, the *Contour* module accepts a Lattice data type on its input port, but the lattice must have two or three dimensions. If you try to pass it data in the form of a 1D lattice, you get an error message.

The module definitions give the data type specifications for the ports on existing modules. When you build a module, you can define each port as you like. The more general the specification for a port, the more modules you can connect to it (but the harder the module is to write).

If you have any doubts about port compatibility when you connect modules into a map, you can check the data types acceptable to each module by:

- calling up the Help window from the module pop–up menu
- reading the module definitions in the *IRIS Explorer Reference Pages*

2.8.3. Looking at Module Data Types

Certain modules can accept a pyramid or lattice data type and output a lattice or geometry data type after performing a data conversion. These include *PyrToLat, LatToGeom*, and *Contour*. Modules with several input ports may be able to accept two or more different data types, one on each port. For example, *Streakline* has input ports for a lattice and a pick and an output port for geometry. *Render* has input ports for geometry and parameter data and output ports for lattice, geometry, and pick data.

Mixed Data types

Table 2–1 lists some of the IRIS Explorer modules according to the data type (Lattice, Pyramid, Geometry or Pick) they can accept on an input port.

Some modules that accept lattices are further defined by whether they can accept 1D, 2D, or 3D lattices, or a combination. For details, refer to the input port specifications of each module in the *IRIS Explorer Reference Pages.*

Input Port Data	Modules
Lattice	Ball, BallStick, BoundBox, ChannelSelect, ColorXform, Contour, CropLat2D, CropLat3D, DiffLat, DisplaceLat, GenerateColormap, Gradient, Graph, Histogram, InterpLat, Interpolate, IsosurfaceLat, LatFunction, LatToGeom, LatToPyr, Legend, MagnitudeLat, Mixer, MultiSlice, OrthoSlice, PrintLat, ProbeLat, PyrToGeom, SampleCrop, ScaleLatNode, Shell, Slice, SubSample, Transform, Triangulate2D, Triangulate3D, VectorGen, VolumeToGeom, WireFrame, WriteLat
Pyramid	AtomicSurf, BallStick, BoundBoxPyr, CullPyr, CropPyr, IsosurfacePyr, PrintPyr, PyrToGeom, PyrToLat, Transform, WritePyr
Geometry	Render, WriteGeom
Pick	Annotation, Ball, CropLat2D, CropLat3D, InterpLat

Table 2–1 Input Data Accepted by Modules

Table 2–2 lists some IRIS Explorer modules according to the data type (Lattice, Pyramid, Geometry or Pick) they can produce on an output port.

Output Port Data	Modules
Lattice	AtomicSurf, ChannelSelect, ColorXform, CropLat2D, CropLat3D, DisplaceLat, DisplayImg, GenerateColormap, GenLat, Gradient, Histogram, Interpolate, LatFunction, MagnitudeLat, Mixer, OrthoSlice, PyrToLat, ReadLat, ReadPlot3D, Render, SampleCrop, ScaleLatNode, Shell, Slice, SubSample, Transform,

	TransformGen, WaveFormColormap
Pyramid	CullPyr, CropPyr, LatToPyr, ProbeLat, ReadPyr, ReadPDB, Transform, Triangulate2D, Triangulate3D
Geometry	Annotation, Ball, BallStick, BoundBox, BoundBoxPyr, Contour, CropLat2D, CropLat3D, DrawText, IsosurfaceLat, IsosurfacePyr, LatToGeom, Legend, MultiSlice, PyrToGeom, ReadGeom, VectorGen, VolumeToGeom, WireFrame
Pick	Render

Table 2–2 Output Data Produced by Modules

Note: All modules that have input ports for parameters also have output ports for them.

Image Processing Modules

The image processing modules, listed in **Table 2–3**, manipulate lattices. They accept only the lattice data type on their input ports and/or produce it on their output ports.

Modules		
AbsImg	AddImg	AndImg
BlendAlphaImg	BlendImg	BlurImg
CompassAngleImg	CompassDirImg	CompassImg
ComplementImg	DisplayImg	DivImg
ExpImg	ForwardFFTImg	FourierConjgImg
FourierCrossCorrImg	FourierDivImg	FourierExpFltImg
FourierGaussFltImg	FourierMagnitudeImg	FourierMergeImg
FourierMultImg	FourierPhaseImg	FourierPwrImg
GaussBlurImg	GrayScaleImg	HistEqualImg
HistNormImg	HistScaleImg	InverseFFTImg
LaplaceEdgeImg	LogImg	MaxFltImg
MaxImg	MedianFltImg	MinFltImg
MinImg	MultImg	NegImg
OrImg	PowerImg	RankFltImg
ReadImg	RobertsEdgeImg	RotZoomImg
SGIPaletteImg	SharpenImg	SobelEdgeImg
SqRootImg	SquareImg	SubtractImg
ThreshImg	WriteImg	XorImg
ZoomImg		

Table 2–3 Image Processing Modules

2.9. Setting Module Parameters

Module parameters allow you to set and change scalar values for each module. For example, a parameter called "Min Range" that accepts integers may let you set the minimum value for a range of temperature values for the module.

Parameters are controlled through *widgets* on the module control panel. Widgets are mechanisms for regulating a module's parameter values. The parameter names are given on the module's input port menu and on the full–scale control panel. A single control panel may contain a number of widgets, each controlling a different parameter.

Widgets include buttons, sliders, dials, text slots, scrolled lists, radio buttons, option menus, and file browsers, as well as drawing areas.

2.9.1. Buttons

Buttons are rectangular or diamond−shaped widgets that let you select a value or switch a specific parameter on or off. They come in two styles (see **Figure 2−19**):

- The one−shot, or single, button can be on or off. It is usually oblong and contains a label on the button itself.
- Diamond−shaped radio buttons offer a set of choices, only one of which is valid at a given time. They have a colored center when on, and are convex when off.

Use the left mouse button to click the buttons on and off.

Figure 2−19 Button Widgets

2.9.2. Sliders

Sliders (see **Figure 2−20**) are either horizontial or vertical bars that let you increase or decrease the value of a parameter in discrete steps. Sliders generally have integer values.

Each slider has:

- a title giving the name of the parameter
- a "thumb" that moves from one end of the bar to the other allowing the sliders value to be changed
- text slots showing the minimum, maximum, and current values of the parameter.

Figure 2−20 Slider Widgets

You can change any one of these values by clicking on the text slot to highlight it, typing in a new value, and pressing **`<Enter>`**.

To change a parameter value, place the mouse cursor on the "thumb" and hold the left mouse button down while you push or pull the slider along its bar. The current value changes as the thumb moves. When you release the mouse button, the new value is sent to the module.

To move the bar to the cursor position, place the cursor in the slider slot and click on the middle mouse button.

2.9.3. Dials

Dials are circular widgets that allow the continuous adjustment of a parameter value between set limits (see **Figure 2−21**). Like sliders, each dial has a title and slots showing minimum, maximum, and current values. You can type new values into any slot (remember to press **`<Enter>`**), and you can also change the current value by turning the dial pointer. Dials generally have float values.

Figure 2−21 Dial Widgets

The dials have a vernier action, in that you can use the pointer for coarse settings. Place the cursor on the pointer and push it clockwise or counterclockwise while holding the left mouse button down. The values change linearly from minimum to maximum through a single revolution of the pointer.

To make a fine adjustment to the parameter value, place the cursor on the dial center and push it clockwise or counterclockwise. The values increment or decrement in minute steps, and the inner dial revolves 50 times for one revolution of the pointer. This is useful for fine−tuning adjustments to a parameter that spans a large range of values, or one that increments its value in minute steps.

2.9.4. Text Type−in Slots

Text type−in slots (see **Figure 2−22**) accept alphanumeric characters.

Figure 2−22 Text Type−in Slots

To type in data values or filenames, click once or twice in the slot until the upright text cursor appears. If necessary, the slot scrolls to the left to give you enough room for your text.

Text type–in slots appear in some input modules as part of a file browser widget (see below).

2.9.5. File Browsers

The file browser widget appears in modules that read or write files from or to the file system. To use the file browser, you must have the full–scale control panel of the module open (see **Figure 2–23**).

Figure 2–23 A File Browser

The file browser has two windows:

- The top one, Contents, displays the contents of the current directory, including files and subdirectories.
- The bottom one, Selection, shows the current selection (file or subdirectory).

You can list the files in any subdirectory on your system. Rather than having to type a filename into a text slot on the module, you can simply select the file from the file browser. You can move up a directory hierarchy by selecting the "../" entry (the first one in the list).

To select a file or directory from the Contents list, double–click on the name.

If your selection is a directory, the file browser displays its contents in the Contents window and its path and name in the Selection window. If it is a file, it sends the filename to the module. The filename appears in the Diminutif control panel text slot and the module fires.

To open a file, you can also click once on the filename to select it, and then click on the *OK* button. The filename appears in the text type–in slot of the Diminutif control panel, and the module fires.

You can use the file browser to traverse directories in a completely different directory tree, in two ways:

- Simply type the directory or filename into the text slot. This takes you directly to your destination.
- Use the Current Directory option menu (see **Figure 2–23**). Each directory entry in this hierarchy contains a cascading menu for all its contents.

  Click on the Current Directory option menu, choose the item you want from each menu, and click on it to get to the next menu level.

2.9.6. Option Menus and Scrolled Lists

Option menus and scrolled lists let you select one or more items from a group of options.

Option menu buttons present a menu from which you can choose one option at a time. The option may be a value, a range of values, or a property, such as uniform or curvilinear (see **Figure 2–24**). The option menu buttons are rectangular, with a small raised bar on the right.

Figure 2–24 Option Menu Widgets

Click anywhere on the button to display the menu and select the item you want.

A scrolled list widget lets you select one or more items from a list of options that may extend beyond the current list window. You may be able to choose one option or several options at a time, depending on how the properties of the particular list were defined when the module was created.

Click on an item to select it. If the scrolled list allows you to select more than one item, click on all the items you want to select.

Figure 2–25 Scrolled List Widget

2.9.7. Drawing Areas

A drawing area is a blank rectangle on the control panel in which an image or visual object can be displayed (see **Figure 2–26**). For example, the modules *DisplayIng* and *Render* have drawing areas on their control panels.

Figure 2–26 A Drawing Area Widget

You can control the size, appearance, and orientation of the subject in the drawing area by using the widgets on the control panel. These vary from module to module, depending on the module's function.

The window may take up varying amounts of space on the control panel. When you enlarge the control panel, the drawing area may be enlarged proportionally. Click on the *Maximize* button of the full–scale control panel to open up the control panel to full–screen size.

Example 2–4 Experimenting with Widgets

This example illustrates how you can affect an object in *Render* by using widgets to change parameter values. Launch the map named *widgets* (see **Figure 2–27**) by dragging and dropping it in the Map Editor. It contains five modules:

- *ReadLat* reads in a lattice data file
- *Contour* generates a contour surface of the volume
- *IsosurfaceLat* generates an isosurface of the volume
- *WireFrame* generates a box that defines the boundaries of the volume
- *Render* visualizes the objects in its window

Note: On workstations without two–sided lighting the isosurface in *Render* may be dark. Change the Flip Normal? option on the *IsosurfaceLat* control panel from `No` to `Yes` to display the geometry properly.

1. Open the full–scale control panels for *Contour*, *IsosurfaceLat*, and *Render* by clicking on the *Maximize* button.
2. Change the dial setting on *IsosurfaceLat*. Notice what happens in the *Render* window.
3. Change the slider position on *Contour* module. Note the change in the *Render* window.

Figure 2–27 Experimenting with Widgets

2.10. Organizing the Module Librarian

Modules and maps are organized into categories in the Module Librarian. **Figure 2–28** shows one category, called "Modules." You can reorganize the contents of these categories, displayed in scrollable columns, as described below.

Figure 2–28 Module Category

2.10.1. Creating New Categories

You can organize modules differently for each system on which you run IRIS Explorer. For example, you may create a category for each remote host, or for all the complete maps on your system.

To create new categories in the Module Librarian, select " New" from the Categories menu. When the pop–up menu appears, enter the name you want for the new category. This adds the name to the Categories menu. Select the name from the menu and an empty column of that name appears in the Librarian.

To hide categories in the Librarian, click on the category title with the right mouse button and select "Hide" when it appears.

To redisplay the category, select the category name from the Categories menu.

To update a category, select "Update" from the Display menu after you have added new modules or maps.

Note: You cannot delete categories, only hide them.

When you choose "Update" from the Display menu, new categories that have not been saved already are deleted. To save new Librarian categories, see **"Saving the Librarian Configuration"**.

2.10.2. Moving Modules Around

You can move maps or modules from an existing category to a new one by dragging and dropping them in place. Items in a category can be ordered by user preference, which is not necessarily alphabetical. You may want to put frequently launched modules at the top of the list and leave the others lower down.

Plan the order of your modules before you move them. New modules are added at the end of the list, and once they are in place, they cannot be moved again within the category.

2.10.3. Storing Modules on the Shelf

The Librarian shelf (see **Figure 2−2**) is a storage area for modules. It is like a bookshelf where you place books to which you often refer. For example, you might stack modules that you use a lot, such as *Render*, *DisplayImg*, or modules of your own creation, on the shelf. Shelved modules are still available from the Module Librarian as well.

To place a module on the shelf, use the "drag and drop" technique. Once a module is on the shelf, you can launch it immediately or leave it there to be launched later. You can launch it repeatedly from the shelf.

2.10.4. Removing Modules from the Librarian

Use the "Delete" option from the Module Launch menu (see **Figure 2−3**) to remove a module or map from a category or from the Librarian shelf.

You can restore a module you accidentally removed by selecting "Update" from the Display menu. It restores the original setting, so you also lose any unsaved changes you have made to the Librarian.

2.10.5. Saving the Librarian Configuration

Once you have arranged the IRIS Explorer modules and maps in categories and on the Librarian shelf to your satisfaction, you can save the new organization for future operations.

1. Select "Save Librarian Config" from the Module Librarian File menu.
2. When the "Save Configuration" file browser appears, save the configuration with a new filename.
3. Use a text editor to incorporate the configuration changes in the new file into the *.explorerrc* file in your home directory. If you do not currently have one, copy /usr/explorer/Explorer.config to this file. The *Category* command in this file governs the layout of categories.

   However, the *.explorerrc* file contains more information than IRIS Explorer provides when you save a new configuration, so you should edit the file rather than overwrite it. Add lines of the form

   ```
   category name module...
   ```

   for each new category or

   ```
   category shelf module..
   ```

   for each module on the Librarian shelf.

To set the module path, that specifies the directories to search for modules, and for further configuration commands see **"The .explorerrc Configuration File"** in Appendix A.

2.11. Common Problems

When all the launched modules have been connected and your map is operational, the modules fire and an object becomes visible in the *Render* or *DisplayImg* window—in theory, perhaps. In practice, this does not always happen. Here are a few common problems and ways to fix them.

Note: If a module port is "required," then it must be receiving valid data from an upstream module in order to work properly. All ports are required unless marked as "Opt(ional)."

2.11.1. You Need a Larger Memory Arena than Your System Can Give You

A message to this effect may appear when IRIS Explorer starts up. It means that you do not have enough disk space to run IRIS Explorer properly. You can:

1. Clean off your disk to make more disk space for the shared memory arena.
2. Create a temporary directory (*tempdir*) on another disk on your machine if you have one (for details, see **"The .explorerrc Configuration File"** in Appendix A).
3. Reduce the size of your arena, which is the amount of disk space necessary to hold all the IRIS Explorer files, before you run IRIS Explorer (see the reference in step 2).

2.11.2. No Object Is Displayed in the Render Window

If nothing appears in your Render window, try the following procedure:

1. Make sure all the required input ports on each module are connected to upstream modules that can send them valid data.
2. *Render* might be looking in the wrong direction. Click the right mouse button in the *Render* window (which brings up a pop−up menu) and select "Functions" then "View All."

   This brings an object that may be out of the viewing area into the center of the window.
3. The module might not be reading in the data from disk. Make sure that modules that need data from files, such as *ReadImg*, have been given access to those files.
4. The module might be disabled. Enable the module by bringing up the pop−up menu from the control panel and selecting "Enable." You can also select all the modules by using the Edit menu in the Map Editor, then choose "Enable."
5. The widgets might not be set to values which can produce data. Check the settings: turn the dial or move the slider if you have an active threshold parameter.

2.11.3. No Change in Object with Change in Parameters

If you have changed object parameters, but the changes do not appear in the object, try the following procedure:

1. Enable the module by bringing up the pop−up menu from the control panel and selecting "Enable." You can also select all the modules by using the Edit menu in the Map Editor, and choose "Enable" from the Edit menu.
2. Make sure all the required ports have data by connecting them to valid ports in other modules.
3. Some modules, such as *GenerateColormap*, have "Build/Run" toggle buttons. If the module is in "Build" mode, it will not fire and send data downstream. Change the mode to "Run."

2.11.4. Viewing Lattice Data in the Render Module

If the lattice is 1D or 2D, such as an image:

1. Connect the input port of a *LatToGeom* module to the "Output – – Lattice" port of the module that is reading in or generating lattice data.
2. Connect the *LatToGeom* module's "Output – – Geometry" port to *Render*.

If the lattice is 3D, such as a volumetric dataset:

1. Select a module that will provide the visual representation you want, for example, *IsosurfaceLat*, *Contour*, *Slice*, *OrthoSlice*, *WireFrame*, or *VolumeToGeom*.
2. All these modules except *Slice* and *OrthoSlice* produce geometry. They can be connected to *Render* by wiring their "Output –– Geometry" ports to *Render*. The two slicers produce lattices, which must be passed through a *LatToGeom* module first.

   See **"Slicing Volumetric Data"** in Chapter 3 for more information about using slicers.

2.11.5. The Module Librarian Does Not Display a Saved Map

If you want a map to be displayed in the Module Librarian after you use "Update," the map must be in a directory that the module Librarian recognizes.

To ensure this:

1. Check that the directory in which you saved the map is in the module path. You can check this and/or add subdirectories to the module path by editing the *.explorerrc* file. See **Appendix A** for details.

   The default directory is */usr/explorer/maps/**.
2. Check that you used the correct pathname when you saved the map. It might be in another directory that is not in the module path.

2.11.6. A Saved Map is Empty When Reopened

You need to save the map again. To prevent this from happening a second time:

1. Make sure you use "Save All" from the Module Librarian File menu to save modules that are not highlighted.
2. If you save certain modules only with "Save Selected," the title bars of those modules must be highlighted, otherwise the map is saved without any modules in it.

2.11.7. When You Try to Destroy a Module, Nothing Happens

When you select "Destroy" from the pop–up menu, IRIS Explorer sends the module a message to exit. If the module is busy computing, it may not get the message for a while. If you want it to quit immediately, select "Destroy" again from the module pop–up menu. This sends a "kill" message to the module's controller, which then shuts down the module without waiting for a response from the module itself.

2.11.8. You Cannot Launch a Module Librarian Remotely

If you get messages from the Local Controller (LC) or Global Controller (GC) when you are trying to launch a Module Librarian for a remote system, there is probably a command generating output from your *.cshrc* file. You will see the output from the command in the error message. For example, some people print a "message of the day" or a witty saying from the *.cshrc* file.

This output interferes with the communication between the Local Controller, which is needed by the remote Module Librarian, and the Global Controller, which starts up the Local Controller on the remote system.

The offending command and its output must be removed before you can run the Module Librarian on that remote system.

Note: The best place for such output is the *.login* file. Other programs besides IRIS Explorer get confused and do not work properly when there is output from the *.cshrc* file.

2.11.9. You Get a Shared Library Error When You Launch Remote Module

Error messages that mention a shared library indicate that some part of IRIS Explorer is not properly installed on the remote system. For more information, see **"Remote Execution of Modules"**.

2.11.10. You Cannot Connect Two Modules, or a Module Produces Unexpected Data

You can check this in two ways:

1. The module may be connected to an incompatible port and is therefore receiving the wrong data type. Bring up the input and output port pads and check the port connections.
2. Verify that the module is producing the desired data type by connecting it to *PrintLat* or *PrintPyr* and printing the output. IRIS Explorer produces an ASCII file showing the structure and contents of the data, which you can look at to track down any errors.