Appendix A. Installation and Troubleshooting

Authors: *Joseph T. Buck*
Christopher Hylands
Alan Kamas

Other Contributors: *Stephen Edwards*
Edward A. Lee
Kennard White

A.1 Introduction

This appendix consists of the following sections:

- "Obtaining Ptolemy" on page A-1 discusses how to obtain Ptolemy.
- "Ptolemy mailing lists and the Ptolemy newsgroup" on page A-2, discusses various forums for discussion about Ptolemy.
- "Installation" on page A-3 discusses how to install Ptolemy.
- "Troubleshooting" on page A-15 discusses how to find and fix problems in Ptolemy.
- "Known bugs" on page A-32 lists known problems in Ptolemy.
- "Additional resources" on page A-39 discusses other resources.

A.2 Obtaining Ptolemy

Ptolemy binaries are currently available for the following architectures: HP running HPUX10.20, Sun 4 (Sparc) running Solaris2.5.1,. In addition, Ptolemy has been compiled on SunOS4.x HPUX9.x, Linux RedHat 5.0, NetBSD_i386, IBM RS/6000 AIX4.x, SGI Irix6.x and the DEC Alpha Digital Unix 4 platforms. These additional platforms are not supported in-house and thus these ports are not tested and may not be complete.

Installing the full system requires about 150 Megabytes of disk space (more if you optionally remake). The demonstration version of Ptolemy, "Ptiny," only requires 12 Megabytes of disk space. All versions requires at least 8 Megabytes of physical memory.

For the latest information on Ptolemy, get the Frequently Asked Questions list. Send electronic mail to `ptolemy-hackers-request@ptolemy.eecs.berkeley.edu` with the message: `get ptolemy-hackers.FAQ` in the body (not the subject) of the message.

You may also send any questions on obtaining Ptolemy to the email address: `ptolemy@eecs.berkeley.edu`. If you have questions about using or enhancing Ptolemy, you should use the ptolemy-hackers mailing list. See "Ptolemy mailing lists and the Ptolemy newsgroup" on page A-2 for details.

A.2.1 Access via the Internet

Ptolemy is available *without support* via Internet FTP. Source code, executables, and documentation are all available on the FTP site at
`ftp://ptolemy.eecs.berkeley.edu`.

This site contains the latest release of Ptolemy, patches to the current release, a Postscript version of the Ptolemy manual, the demonstration version of Ptolemy, Ptolemy papers and journal articles, as well as the archives for the mailing list.

To obtain Ptolemy via FTP:

a. FTP to Internet host `ptolemy.eecs.berkeley.edu`.

b. Login as `anonymous`; give your full email address as the password.

c. Change to the `/pub` directory: `cd pub`

d. Follow the directions in the `README` file there.

There is an FTP mirror in Japan:
`ftp://ftp.iij.ad.jp/pub/misc/ptolemy`
The notation above means log into `ftp.iij.ad.jp` and then cd to */pub/misc/ptolemy*.

There is also an FTP mirror in France at:
`ftp://chinon.thomson-csf.fr/mirrors/ptolemy.eecs.berkeley.edu/`

A.2.2 Access via the World Wide Web

There is a World Wide Web (WWW) page for Ptolemy:
`http://ptolemy.eecs.berkeley.edu/`
The Ptolemy WWW page contains information about Ptolemy, demonstrations of Ptolemy programs, access to the Ptolemy FTP site, and a hypertext version of the Ptolemy Documentation.

There is a WWW mirror of our site in France at:
`http://chinon.thomson-csf.fr/ptolemy`

A.2.3 Obtaining documentation only

All of the manuals as well as a number of Ptolemy-related papers and journal articles are available from the WWW site and from the FTP site in the `/pub/ptolemy/papers` directory.

A.3 Ptolemy mailing lists and the Ptolemy newsgroup

There are two publicly available mailing lists and a newsgroup that discuss Ptolemy.

A.3.1 Ptolemy mailing lists

The Ptolemy mailing lists are run by the Majordomo mailing list server. This server can automatically subscribe you to mailing lists and it can send you monthly archive files for each of the lists. To find out more about our Majordomo server, send an email letter to:
`majordomo@ptolemy.eecs.berkeley.edu` with the word `help` in the body of the letter.

There are two mailing lists for Ptolemy users:

`ptolemy-hackers@ptolemy.eecs.berkeley.edu`

`ptolemy-interest@ptolemy.eecs.berkeley.edu`

`ptolemy-hackers`

A forum for Ptolemy questions and issues. Users of the current release who have a Ptolemy question, comment, or think they've found a bug should send mail to `ptolemy-hackers`. Ptolemy users in companies and universities around the world who are interested in discussing Ptolemy post and read from the `ptolemy-hackers` group.

To subscribe to the `ptolemy-hackers` mailing list, send mail to:
`ptolemy-hackers-request@ptolemy.eecs.berkeley.edu`
with the word `subscribe` in the body of the message. To leave the mailing list put the word `unsubscribe` in the body of your message.

`ptolemy-interest`

A mailing list for Ptolemy announcements. Messages about new releases, bug fixes, and other information of interest to all Ptolemy users is posted here. Note: This is not a discussion group, and you cannot post to this mailing list. Message volume is very light. All users of Ptolemy should belong to this group in order to stay current with work being done on Ptolemy. Announcements sent to `ptolemy-interest` are also sent to `ptolemy-hackers`, so you need not subscribe to both lists.

To subscribe to the ptolemy-interest mailing list, send mail to:
`ptolemy-interest-request@ptolemy.eecs.berkeley.edu`
with the word `subscribe` in the body (not the subject) of the message.

A.3.2 Ptolemy Newsgroup

There is a Ptolemy newsgroup: `comp.soft-sys.ptolemy`. Just like the `ptolemy-hackers` mailing list, the `comp.soft-sys.ptolemy` newsgroup is a forum of the discussion of Ptolemy questions, bug reports, additions, and applications. Note that all mail sent to the `ptolemy-hackers` mailing list is automatically posted to the `comp.soft-sys.ptolemy` newsgroup as well. The name is chosen to correspond to similar newsgroups for the Khoros and Matlab systems, which are also under `comp.soft-sys`.

A.4 Installation

Ptolemy is a large software system that relies on a properly configured software environment. This section will take you step by step through the installation of Ptolemy, including all of the basic information required to get from an FTP archive to being able to run the system. Note that this information may have changed after the manual was printed. Thus, the most up-to-date instructions are included in `$PTOLEMY/doc/html/install`.

Ptolemy is distributed in several forms:

- The Ptolemy group provides Ptolemy and Gnu tools binaries for Solaris2.5.1 and HPUX10.20. We use these two platforms for in-house development and testing, so they are more stable than other platforms.
- We also make binaries for other architectures available. These binaries have been contributed by users and have not been tested in-house. You can find these binaries at `ftp://ptolemy.eecs.berkeley.edu/pub/ptolemy/contrib`. The list of contributed binaries varies, but usually includes Linux, AIX and DEC Alpha.
- If your machine is not one of the two primary platforms, and we do not have contributed binaries for it, then you will have to do a compile from scratch. For more information see “Rebuilding Ptolemy From Source” on page A-10.

A.4.1 Location of the Ptolemy installation

The Ptolemy system uses the environment variable `$PTOLEMY` to locate the Ptolemy distribution. If you are rebuilding Ptolemy from sources without using any prebuilt Ptolemy binaries, then you can set `$PTOLEMY` to any location. If you are using prebuilt binaries, then there are a few issues concerning exactly to what value `$PTOLEMY` is set. The issues occur because certain facilities, such as shared libraries, the Gnu compiler and Tcl/Tk save certain pathnames at build time. If at run time, certain files cannot be found that were present at build time, then there will be various failures.

The Ptolemy binaries that we distribute were built with `$PTOLEMY` equal to `/users/ptolemy`. If you can create a symbolic link from `/users/ptolemy` to your Ptolemy distribution, then you may find using the prebuilt binaries easier. If you cannot create a link and are using prebuilt binaries, then be aware of the following points:

- To use Ptolemy, you will need to set your `PTOLEMY` environment variable to the location of the distribution. For example, if your Ptolemy distribution was in `/usr/local/ptolemy`, then under C shell (`/bin/csh`), you would type
  `setenv PTOLEMY /usr/local/ptolemy`.
- If you are using prebuilt binaries that were built with `$PTOLEMY` equal to `/users/ptolemy`, but your Ptolemy distribution is not at `/users/ptolemy`, then you may need to set `LD_LIBRARY_PATH` or `SHLIB_PATH` so that the binaries can find the appropriate shared libraries at run time. The `pigi` script will attempt to properly set `LD_LIBRARY_PATH` if it is not already set, but the `pigi` script is easily fooled. See `$PTOLEMY/.cshrc` for sample C shell commands that set the appropriate variable. See also “pigi fails to start up, giving shared library messages” on page A-17, and “Shared Libraries” on page D-1.
- If you are compiling Ptolemy using the prebuilt Gnu binaries that we provide, and the Ptolemy distribution is not at `/users/ptolemy`, then you will need to set an additional environment variable, see “Gnu Installation” on page A-7 for more information.
- If you are using prebuilt binaries, and the Ptolemy distribution is not at `/users/ptolemy`, then the `tycho` command may fail for you. For a workaround, see “tycho fails to start up, giving `TCL_LIBRARY` messages” on page A-18.

A.4.2 Basic Ptolemy installation

First note the approximate disk space requirements for Ptolemy. The numbers below are for the Solaris version of Ptolemy, but other distributions are similar:

- Ptolemy from `pt-0.7.1.src.tar.gz` and `pt-0.7.1.sol2.5.tar.gz`: 186 Megabytes
- Gnu binaries for Solaris: 25 Megabytes
- Ptolemy from `pt-0.7.1.src.tar.gz` and `pt-0.7.1.sol2.5.tar.gz` if you remake (optional): approximately 200 Megabytes
- Documentation Postscript Files (optional): approximately 30 Megabytes
- Gnu Binary and Sources (optional): 70 Megabytes
- Additional Sources (needed to port Ptolemy) (optional): 32 Megabytes
- Complete rebuild from `pt-0.7.1.src.tar.gz`, `pt-0.7.1.other.src.tar.gz` and `pt-0.7.1.gnu.tar.gz`: under Solaris 2.5.1: 500 Megabytes

Important Note: Although we distribute binaries, you must also install the source tree in `pt-0.7.1.src.tar.gz`. *Installing this src directory is not optional*. The `src` directory contains the Ptolemy source code, but it also contains the icons and interpreted Tcl code used in the user interface. If you absolutely must discard the source code, you can remove all files under `src` with extensions `.cc`, `.h`, or `.c`. For more information, see “Freeing up Disk Space” on page A-14.

A.4.3 The ptolemy user

The preferred way to install Ptolemy involves creating a fictitious user with the userid `ptolemy`, together with a home directory for the `ptolemy` user. Using the binaries is easier if the `ptolemy` user’s home directory is `/users/ptolemy`. Once the `ptolemy` user account has been created, log in or `su` to user `ptolemy`. If you do not wish to create a user called `ptolemy`, see below for an alternative.

A.4.4 Installation without creating a ptolemy user

The preferred installation technique, as indicated above, is to create a user called `ptolemy`. The reason for this is that running Ptolemy requires an appropriate user configuration. At minimum, the user’s path must be set up properly. The `ptolemy` user is also configured to run an X window manager (`twm`) with suitable X resources that are known to work well. In troubleshooting an installation, having the `ptolemy` user properly configured can be very valuable.

Ptolemy can be installed without creating a ptolemy user. If you do this, *every* Ptolemy user must set a `PTOLEMY` environment variable to point to the root directory of Ptolemy. The installation is the same as below, except that `~ptolemy` is replaced with `$PTOLEMY`.

A.4.5 Obtaining Ptolemy

Ptolemy is available via the Ptolemy Web site at `http://ptolemy.eecs.berkeley.edu`. The Ptolemy releases can be found under the `Downloads` link. The Ptolemy release can also be found on the Ptolemy ftp site at `ftp://ptolemy.eecs.berke-`

`ley.edu/` follow the instructions in the `README` file, and be sure to download in binary mode.

Untarring the distribution

Go to the directory where you either saved the downloaded `*.tar.gz` files. These files have been compressed with the Gnu `gzip` program, a compression program from the Free Software Foundation. In order to uncompress the files, you need the program `gzcat`.The `gzcat` program is available via anonymous FTP from `ftp://ptolemy.eecs.berkeley.edu/pub/gnu`. Now proceed as follows:

a. `gzcat pt-0.7.1.doc.tar.gz | ( cd ~ptolemy/..; tar xf - )`

This uncompresses the documentation, changes directory to the parent of the `ptolemy` user, and then creates all of the documentation files.

b. `gzcat pt-0.7.1.src.tar.gz | ( cd ~ptolemy/..; tar xf - )`

This uncompresses the `src` tar file and creates the source files. You must not skip this step. Ptolemy depends on these files being present. Note that you may get a few warning messages during this and the following step about the tar program not being able to create some directories because they already exist. This is expected (the same directory is mentioned in several of the tar files), so you need not worry.

c. If you are running the HPUX 10.20 version:
```
gzcat pt-0.7.1.hppa.tar.gz | ( cd ~ptolemy/..; tar xf - )
```
If you are running the Solaris2.5.1 version:
```
gzcat pt-0.7.1.sol2.5.tar.gz | ( cd ~ptolemy/..; tar xf - )
```
If you are running another platform for which a binary has been provided:
```
gzcat pt-0.7.1.platform.tar.gz | \
        ( cd ~ptolemy/..; tar xf - )
```
This uncompresses the binaries and creates the executable files. Note that is possible to install binaries for multiple platforms on the same file system because different directories are used for each set of binaries. Just execute whichever of the above commands apply.

d. (Optional) If you are planning on porting Ptolemy, you will need the other sources file:
```
gzcat pt-0.7.1.other.src.tar.gz| \
                  (cd ~ptolemy/..; tar xf - )
```
The `other.src` tar file includes sources to a subset of `Octtools` (including `vem`), `Tcl/Tk` and `xv`. If you wish to rebuild Ptolemy completely from source, then you will need this tar file. Alternatively, you may be able to use the Octtools libraries from a binary tar file.
If you are planning on recompiling Ptolemy, but are really short on disk space, and you already have Tcl/Tk (comes with most Linux installations) then you can download `pt-0.7.1.oct.src.tar.gz` instead of `pt-0.7.1.other.src.tar.gz`. Note that this file has not been extensively tested. For more information, see “Ptolemy and Tcl/Tk” on page A-13

e. (Optional) You no longer need the `*.tar.gz` files that you downloaded from the Web or FTP site. Remember to delete these files to free up disk space.

f. (Optional) The X11 program `xv` is used by some of the image processing demos. If you already have a version of `xv`, or you don't plan on doing any image processing, you can also remove the `xv` binary in `$PTOLEMY/bin.$PTARCH/xv`.

A.4.6 Special considerations for use under OpenWindows

Ptolemy was developed using the X11R4 through X11R6 distributions from MIT. Although Ptolemy runs fine under OpenWindows 2, there are problems with running the Ptolemy graphical interface with OpenWindows version 3.0 under SunOS4.x. Some users have had no problems at all, but others have had intermittent problems such as "bad match" errors. We believe this may be a problem with the X-server supplied with the OpenWindows 3.0, but the error is elusive and we have not yet tracked it down. These problems seem not to occur in OpenWindows 3.3 and later (OW3.3 was distributed with Solaris2.3).

In order for all utilities included with this distribution to work under OpenWindows 2, you must install the shared libraries for the Athena widgets (the freely redistributable widget set from the MIT X11 distribution), which are provided with this distribution under the `$PTOLEMY/athena.sun4` directory. To install them, become root and copy all files in that directory into `/usr/openwin/lib` (or, if you have installed OpenWindows in a non-standard place, into `$OPENWINHOME/lib`). If you do not wish to do this, you could leave them in place and have every Ptolemy user change their `LD_LIBRARY_PATH` environment variable to search `~ptolemy/athena.sun4` before `/usr/openwin/lib`. Consult the Unix manual entry for the `ld` program to learn more about `LD_LIBRARY_PATH`.

After installation, the `$PTOLEMY` directory will contain several scripts for starting up X11R6 (Xrun), OpenWindows with *olwm* (Xrun.ow), or OpenWindows with *twm* (Xrun.ow.twm).

If only have OpenWindows, and not X11R5 or X11R6 and you plan on rebuilding from source, you may need the libraries in `athena.sun4`. `athena.sun4` is part of the `pt-0.7.1.sun4.tar.gz` tar file.

A.4.7 Gnu Installation

If you are planning on extending Ptolemy by writing your own stars and you are using prebuilt binaries, you will need to install the same compiler that was used to build the prebuilt binaries. In particular, Ptolemy supports dynamic linking of newly defined stars. *Dynamic linking will not work, however, if the new stars are compiled with a different version of the compiler than that used for the rest of the system*. Thus, you must either use the same compiler that we used for this distribution of Ptolemy or you must recompile the entire Ptolemy system. Note that this is not a complete set of Gnu software.

It is also possible to build Ptolemy with other, non-Gnu C++ compilers, such as Sun-Soft's C++ compiler. We have built this release of Ptolemy with the following compilers:

`sol2.5.cfront`	Sun CC, version 4.1(native) for Solaris2.5.1
`hppa.cfront`	HPUX CC version A.10.34 for HPUX 10.20[1]

We do not distribute these non-Gnu C++ binaries. If you choose to use a non-Gnu C++ compiler, you must completely rebuild Ptolemy. The libraries in the tar files were produced by the Gnu C++ compiler and are not interoperable with code from other compilers. When you completely rebuild Ptolemy, you must be sure to first remove all previously compiled object files. Note that to compile Ptolemy with a non-Gnu C++ compiler, you will still need to use Gnu `make`.

The Ptolemy 0.7.1 binaries we ship were built with egcs-1.0.2. Ptolemy 0.7.1 was also built with gcc-2.7.2.2 and libg++-2.7.2. Note that Ptolemy uses the gcc and libg++ shared libraries, if you are using your own version of gcc and libg++, it must have been configured with --enable-shared. For further information, see "Can I use my own version of gcc and libg++?" on page A-28.

Gnu Compiler Environment Variables

The Gnu compiler is dependent upon where it was built. The executable that we supply assumes that the compiler is installed in a directory called `/users/ptolemy`. If you do not wish to rebuild the compiler, then you must either install Ptolemy in this directory, or create a symbolic link from this directory to the actual directory in which Ptolemy is installed. If you cannot install such a link, then you will still be able to run Ptolemy, *but you may not be able to dynamically link new stars or recompile Ptolemy.*

Even if your Ptolemy installation in not in `/users/ptolemy` and even if you cannot create a link from `/users/ptolemy` to the actual location of Ptolemy, you may still be able to use this compiler by setting two environment variables before using the compiler. We provide a C-shell script in `$PTOLEMY/bin/g++-setup` that sets the variables. If you are running the C-shell, type: `source $PTOLEMY/bin/g++-setup` to set the variables.

If you will always be using the prebuilt Gnu compiler shipped with Ptolemy with these variables, you may want to add the above command to your ~/`.cshrc` file.

The script sets the following two variables:

```
setenv C_INCLUDE_PATH \
       $PTOLEMY/gnu/$PTARCH/lib/gcc-lib/$PTARCH/egcs-2.90.27/include
setenv GCC_EXEC_PREFIX \
       $PTOLEMY/gnu/$PTARCH/lib/gcc-lib/$PTARCH/egcs-2.90.27/
```

The above assumes that the environment variable `PTOLEMY` is set to the name of the actual installation directory of Ptolemy, and `PTARCH` is set to the type of workstation (such as `sol2.5`, `hppa`, etc.).

You should not set the Gnu environment variables if you are using a version of the Gnu compiler that is different from the version that we are shipping. If you are using a different version of the Gnu compiler, then you will probably need to do a complete rebuild for dynamic linking to work.

See the `gcc` info format file for a complete list of environment variables. Note that `GCC_EXEC_PREFIX` must have a trailing slash, "`/`". The above variable work for Solaris2.x, if you are running SunOS4.1.3, see "Sun OS4 specific bugs" on page A-38. One symptom of having improperly set environment variables is if you see messages about:

1. On the HP, type "what /usr/bin/CC" to see what version you are running.

```
ptolemy/src/kernel/isa.h:44: conflicts with new declaration
with C linkage
```

Another symptom is if you have missing `strcmp()` symbols at link time.

If, under gcc-2.7.2.2, you get warnings about ‘conflict with built in declaration’, and your compiler is not installed where it was built, you may need to create a link in your gcc-lib. We have also seen problems with functions that have variable numbers of arguments. If you compile the file with the `-v` option, you can see what directories `gcc` is including. You could try creating a link:

```
(cd $PTOLEMY/gnu/$PTARCH/lib/gcc-lib/$PTARCH/2.7.2.2; \
     ln -s .. $PTARCH)
```

Untarring

To install the Ptolemy Gnu subset, proceed as follows:

a. Change to the directory that contains the files you downloaded via FTP or the Web.

You should now have a `pt-0.7.1.gnu.`*`xxx`*`.tar.gz` file (where *`xxx`* is an architecture supported by Ptolemy such as `hppa` or `sol2.5`) in your current directory.

b. If you are running on an HP workstation under HPUX10.20 then:
```
gzcat pt-0.7.1.gnu.hppa.tar.gz | \
            (cd ~ptolemy/..; tar xf - )
```
If you are running the Solaris2.5.1 version:
```
gzcat pt-0.7.1.gnu.sol2.5.tar.gz | \
            ( cd ~ptolemy/..; tar xf - )
```
If you are running another platform for which we provide a tar file:
```
gzcat pt-0.7.1.gnu.platform.tar.gz | \
             ( cd ~ptolemy/..; tar xf - )
```
Note that these are single commands split over two lines for readability. Do not type a space after the backslash at the end of the first line, just press `Return`.

c. There is also a Gnu tar file which contains the Gnu source code. You will save disk space and Ptolemy will still run if you do not untar the Gnu source code. However, if you plan to redistribute the Gnu tools (give them to anyone else) you must include sources, according to the Gnu Public License. Therefore, it may be a good idea to keep these source files around.
If you want to untar the Gnu source code:
```
gzcat pt-0.7.1.gnu.tar.gz | ( cd ~ptolemy/..; tar xf - )
```

d. You no longer need the `*gnu*.tar.gz` files that your got from the FTP site or the tape. You may delete these files to free up disk space.

A.4.8 Testing the Installation

Note that the following tests assume that you have created a `ptolemy` user and installed the system there. One advantage of such an installation, is that the `ptolemy` user already has a working `.cshrc` and `.login` file to make start-up easier.

To test Ptolemy, assuming you have set up a `ptolemy` user:

a. `login` as ptolemy

If the X server is not already running, the `.login` script will attempt to start it. If your installation is different from ours, you may need to modify `.login` to work at your site (in particular, you may need a different *path* variable).

b. `cd demo`

c. `pigi`

Follow instructions in “Starting Ptolemy” on page 2-2.

If you have not set up a `ptolemy` user, then set your `PTOLEMY` environment variable to point to the installation directory. If your Ptolemy distribution is at `/users/myptolemy`, under the C shell, you would type:

```
setenv PTOLEMY /users/myptolemy
```

If you use a shell other than the C-shell consult your documentation on how to set environment variables. The next steps are to change to the Ptolemy directory and to start up Ptolemy:

```
cd $PTOLEMY
bin/pigi
```

Note that the ptolemy user provides a model of a user properly configured to run Ptolemy. All the dot files (`.cshrc`, `.login` etc.) in the home directory are set up according to the tastes of the Ptolemy authors and according the standard use of windowing software in the Ptolemy development group.

A.4.9 Rebuilding Ptolemy From Source

If you wish to rebuild Ptolemy from source (this step is recommended if you plan to do major development work, such as adding a new domain), it is simply a matter of editing the appropriate configuration file and typing `make`. This is explained in a bit more detail below. Note that to rebuild completely from source, you need the `pt-0.7.1.other.srcs.tar.gz` tar overlay, as well as the `pt-0.7.1.src.tar.gz` tar overlay. If you are having problems rebuilding, you may want to look over “Ptolemy will not recompile” on page A-27.

To do a build of all of Ptolemy using the Gnu compiler from the distribution, first make sure that either Ptolemy is installed in `/users/ptolemy`, or that there is a symbolic link from `/users/ptolemy` to the installation directory. Alternatively, you can try setting the environment variables described in “Gnu Compiler Environment Variables” on page A-8.

`$PTOLEMY` should point to the location of the installation so that the toplevel makefile is at `$PTOLEMY/makefile`. `$PTARCH` should be set to the name of the architecture you are running. `$PTARCH` is used to select a makefile from `$PTOLEMY/mk/config-$PTARCH.mk`. The script `$PTOLEMY/bin/ptarch` will return the architecture of the machine on which it is run.

Next make sure that `$PTOLEMY/bin.$PTARCH` and `$PTOLEMY/bin` are both in your path (`$PTOLEMY/bin.$PTARCH` is where the compiler is installed. `$PTOLEMY/bin` is where certain scripts used to build star lists are located).

Then proceed as follows:

a. `setenv PTOLEMY /users/ptolemy`

b. `setenv PTARCH ‘$PTOLEMY/bin/ptarch‘`

c. `set path = ($PTOLEMY/bin $PTOLEMY/bin.$PTARCH $path)`

d. `cd $PTOLEMY` (or `cd /users/ptolemy`)

The toplevel makefile at `$PTOLEMY/makefile` can rebuild some or all of Ptolemy.

- To rebuild the Gnu tools, Tcl/Tk, `xv`, Octtools and Ptolemy, type `make bootstrap`. This target requires the following files: `pt-0.7.1.gnu.tar.gz`, `pt-0.7.1.other.src.tar.gz` and `pt-0.7.1.src.tar.gz`.
- To rebuild Tcl/Tk, `xv`, Octtools and Ptolemy, type `make everything`. This target requires the following files: `pt-0.7.1.other.src.tar.gz` and `pt-0.7.1.src.tar.gz`.
- To rebuild just Octtools and Ptolemy, type `make install_octtools install`.This target requires the following files: `pt-0.7.1.other.src.tar.gz` and `pt-0.7.1.src.tar.gz`. (If you are short on disk space, you may be able to download `pt-0.7.1.octtools.tar.gz` instead of `pt-0.7.1.other.src.tar.gz`).
- To rebuild just Ptolemy, type `make install`.This target requires the following files: `pt-0.7.1.src.tar.gz`.

An easier approach is to log in as “ptolemy” (assuming you created such a user). The path and environment variables are already set up for this user. Yet a third approach is to make use of the setup for this user, as follows:

a. `cd $PTOLEMY` (or `cd /users/ptolemy`)

b. edit `$PTOLEMY/.cshrc` to define `PTOLEMY` correctly

c. `source $PTOLEMY/.cshrc`

d. `make install`

If you wish to customize your installation, you may have to edit the configuration files. These files are in `$PTOLEMY/mk`. The configuration files are all named `config-$PTARCH.mk` where the `$PTARCH` is something like `sol2.5` for a Sun Sparc system running Solaris2.5.1 or `hppa` for an HP Precision Architecture machine. They are included by other makefiles and define symbols specifying compiler flags, the directory where X include files are located, etc.

If you wish to rebuild using a non-Gnu C++ compiler rather than `g++`, use `config-sol2.5.cfront.mk` as a starting point to produce your configuration file. This has been tested with Sun CC version 4.1 with Solaris2.5.1. For other platforms, you may need to do some tweaking. See `config-hppa.cfront.mk` for using HP CC version 4, and `config-irix5.cfront.mk` for using SGI Irix CC. Note that the term “cfront” is historical, and that not all of these compilers are actually cfront based. We use the term cfront to refer to non-Gnu C++ compilers.

To rebuild the system, first adjust the configuration parameters in the appropriate configuration file. For example, if you are using the Gnu tools on a Sun Sparc running

Solaris2.5.1, then you will need to adjust the `config-sol2.5.mk` file.

Next, run `make`. The Ptolemy source files include extensions found only in Gnu `make`, which is included in the Gnu subset of the Ptolemy distribution. (Make sure that the Gnu tools are installed correctly.) Sun `make` will fail on certain makefiles that have Gnu `make` extensions. See `$PTOLEMY/src/gnu/README` for a discussion of Gnu `make` compatibility.

You will get some warnings from the compiler, but the following warnings can safely be ignored:

- any warning about `file_id` defined but not used.
- any warning about `SccsId` defined but not used
- variable might be clobbered by `longjmp` or `vfor`'
  This warning may be a problem under heavy optimization. We will work to remove these from future releases.
- many warnings about `cast discards const`.

Details of how Ptolemy is built and the general layout

Below we describe some of the details of how Ptolemy is built from sources.

To build Ptolemy, you must have your `PTOLEMY` and `PTARCH` environment variables set. `PTOLEMY` is set to the location of the Ptolemy tree, `PTARCH` is set to the name of the machine architecture (the script `$PTOLEMY/bin/ptarch` will return the architecture of the machine on which it is run). The directory `$PTOLEMY/mk` contains master makefiles that are included by other makefiles (The makefile `include` directive does this for us). `$PTOLEMY/mk/config-$PTARCH.mk` refers to the makefile for the architecture `$PTARCH`. For instance, `$PTOLEMY/mk/config-sol2.5.1.mk` is the makefile that contains the Solaris2.5.1 specific details.

When you change to the `$PTOLEMY` directory and type `make`, `$PTOLEMY/makefile` contains a rule that checks to see if the directory `$PTOLEMY/obj.$PTARCH` exists. If this directory does not exist, then make runs the command `sh MAKEARCH`, where `MAKEARCH` is a Bourne shell script at `$PTOLEMY/MAKEARCH`. `MAKEARCH` will create the necessary subdirectories under `$PTOLEMY/obj.$PTARCH` for `$PTARCH` if they do not exist.

We split up the sources and the object files into separate directories in part to make it easier to support multiple architectures from one source tree. The directory `$PTOLEMY/obj.$PTARCH` contains the platform-dependent object files for a particular architecture. The platform-dependent binaries are installed into `$PTOLEMY/bin.$PTARCH`, and the libraries go into `$PTOLEMY/lib.$PTARCH`. Octtools, Tcl/Tk, and Gnu tools have their own set of architecture-dependent directories. The Ptolemy Programmer's Manual describes the directory tree structure in more detail.

We are able to have separate object and source directories by using the `make` program's `VPATH` facility. Briefly, `VPATH` is a way of telling `make` to look in another directory for a file if that file is not present in the current directory. For more information, see the Gnu `make` documentation, in Gnu Info format files in `$PTOLEMY/gnu/common/info/make-*`.

Ptolemy and Tcl/Tk

Ptolemy0.7.1 uses `itcl2.2`, which is an object-oriented extension to Tcl (Tool Command Language) and Tk. `itcl2.2` includes a modified version of `tcl7.6` and `tk4.2`. If you have `itcl2.2` already installed, you may use your installed version. You will need to either edit
`$PTOLEMY/mk/config-default.mk` or create the proper links in `$PTOLEMY/tcltk`.

In theory, it is possible to build Ptolemy0.7.1 without `itcl2.2`. However, without `itcl2.2`, Tycho, the syntax manager and the Gantt chart facilities will not work. There may be other features that fail to operate. We strongly encourage you to build with `itcl2.2`.

In previous releases, the layout of the `$PTOLEMY/tcltk` directory was such that Tcl and Tk were in separate directories so that upgrading Tcl and Tk could be done separately if necessary. In Ptolemy 0.7.1, we are using `itcl2.2`, the Tcl and Tk are under the `$PTOLEMY/tcltk/itcl*` directories. We have left separate `$PTOLEMY/tcltk/tcl*` and `$PTOLEMY/tcltk/tk*` directories so that we can easily support separate Tcl and Tk releases in the future.

- `$PTOLEMY/tcltk/itcl` contains the architecture-independent Itcl directories `include`, `lib` and `man`. For instance, `tcl.h` might be at `~ptolemy/tcltk/itcl/include/tcl.h`.
- `$PTOLEMY/tcltk/itcl.$PTARCH` contains the architecture-dependent Itcl directories `bin` and `lib`. For instance, on the sun4, `libtcl.a` might be at `~ptolemy/tcltk/itcl.sun4/lib/libtcl.a`.

Notes for building on the sol2.5 platform (Sun4s running Solaris2.5.1)

Solaris2.5.1 is not shipped with a C compiler, and the Gnu tar file we ship does not include absolutely everything necessary to build on a compilerless Solaris2.5.1 machine. Notably, `bison` may be necessary to compile the Gnu C compiler and the Ptolemy program `ptlang`. If, when you are building the Gnu C compiler, `bison` is called and you do not have it, try using the Unix `touch` command on the offending .c file. The tar files we distribute include `ptlang.c`, which is generated from `ptlang.y`, so you should not need `bison` to compile Ptolemy. You can get a fairly complete set of Gnu tools via anonymous FTP from `ftp://ftp.uu.net/systems/gnu/solaris2.3/`.

- If you choose to compile Gnu `gcc` with SunSoft's `cc`, be sure that you are not using `/usr/ucb/cc`. Otherwise, you may see errors while compiling `gcc/protoize.c`:

  ```
  "/usr/include/sys/ucontext.h", line 25: syntax error before or
  at: stack_t
  ```

  The solution is to place `/opt/SUNWspro/bin` in your path before `/usr/ucb.`
- You will need `/usr/ccs/bin` in your path to pick up `ar`, `lex` and `yacc`.
- If you are building gcc-2.7.2 under Solaris, you must have `/bin` in your path before `/usr/ucb`. See `$PTOLEMY/src/gnu/README` for more information.
- Under `sol2.5.cfront`, `$OPENWINHOME` must be defined to build `xv` since `xmkmf` relies on `$OPENWINHOME`. In the C shell, one would type:

```
setenv OPENWINHOME /usr/openwin
```

A.4.10 Freeing up Disk Space

If you are short on disk space, you can consider removing the following files

- `$PTOLEMY/src/domains/sdf/demo/ppimage` (~1020 Kb). This file is used by some of the image processing demos.
- If you rebuild and reinstall Ptolemy from source, you may remove the `$PTOLEMY/obj.$PTARCH` directory once you are done installing.
- If you do not plan to rebuild Ptolemy from source, you can remove all the `.cc`, `.c` and `.h` files in the `$PTOLEMY/src` directory. Note that the `$PTOLEMY/src` directory contains the icons and interpreted Tcl code used in the user interface, so removing the `src` directory completely will prevent pigi from running.
- If you are very short on space, you may want to run Ptiny, a version of Ptolemy that only contains most of the SDF and DE domains. Ptiny is available via anonymous FTP from `ptolemy.eecs.berkeley.edu`. Note that Ptiny might be from an older release than the current release of the main Ptolemy distribution.

A.4.11 Other useful software packages

Some parts of Ptolemy use other software packages. The packages below are all available on the internet. If you have internet FTP access, you can find lots of FAQs (Frequently Asked Questions) via anonymous FTP at `pit-manager.mit.edu` in `/pub/usenet/news.answers`.

- The Utah Raster Toolkit (URT) is used by some of the Multimedia demos. Most of the URT utilities work with images in RLE (run-length encoded) format, so most of its utilities begin with the prefix `rle` such as is `rletoppm`. The original URT is available via anonymous FTP from `ftp.cs.utah.edu` in `/pub/urt-*.` Note that the original URT does not include configuration files for many modern platforms, including Solaris2.5.1. We have made URT sources and binaries for Solaris2.5.1 available via anonymous ftp from `ptolemy.eecs.berkeley.edu` in `pub/misc/urt`.
- The following Gnu utilities are available via anonymous FTP from `prep.ai.mit.edu`. See the file `/pub/gnu/GETTING.GNU.SOFTWARE` on that site. For further information, write to:

  Free Software Foundation
  675 Mass Ave
  Cambridge, MA 02139
  USA

Ptolemy uses the following Gnu software:

`ghostscript -` PostScript previewer. Note that `oct2ps` can use `ghostscript` to generate Encapsulated PostScript (EPS).

`gdb` - needed for `pigi -debug`

`gzip` - Used to compress and uncompress files.

A.5 Troubleshooting

This section lists common difficulties encountered when installing and running Ptolemy. This list is, of course, by no means complete. If you do not find your particular problem here, refer to the section “Additional resources” on page A-39.

The most recent version of this section can be found on the bottom of the Ptolemy home page at `http://ptolemy.eecs.berkeley.edu/papers/almagest/appendixA.html`. The same file should be available via anonymous ftp from `ptolemy.eecs.berkeley.edu` as `pub/ptolemy/ptolemy0.7.1/TROUBLE_SHOOTING_0.7.1`.

A.5.1 Problems with tar files

EOF messages while using tar on Suns

There is a bug in the SunOS 4.1.3 version of `/bin/tar`. Sometimes a command such as:

```
gzcat foo.tar.gz | ( cd ~ptolemy/..; tar xf - )
```

may produce error message such as:

```
tar: read error: unexpected EOF
```

when reading from a pipe if the `tar` in the command is Sun’s `/bin/tar`. One workaround is to use GNU tar, another is to use `gzcat` and `dd`:

```
gzcat foo.tar.gz | dd conv=sync,block |
                 ( cd ~ptolemy/..; tar xf - )
```

Another workaround is to uncompress the file first, and then run `/bin/tar`:

```
gzcat foo.tar.gz > foo.tar
cat foo.tar | (cd ~ptolemy/..; /bin/tar xf -)
```

A.5.2 Problems starting pigi

pigi: Command not found

Running the `pigi` command is the most common way of starting up Ptolemy. If you get a message like:

```
ptolemy@kahn 2% pigi
pigi: Command not found
ptolemy@kahn 3%
```

then try the following:

- The `pigi` script is located at `$PTOLEMY/bin/pigi`. If that file is not present, then you need to download the Ptolemy src file, `pt0.7.1.src.tar.gz`. This file is not optional, it contains the `pigi` script and other files necessary to run Ptolemy.
- Be sure that your path includes `$PTOLEMY/bin`. Under `csh`, do:

```
set path = ($PTOLEMY/bin $PTOLEMY/bin.$PTARCH $path)
```

Mr. Ptolemy window does not come up

Ptolemy consists of two processes, `vem` and `pigiRpc`, that communicate via Remote Procedure Calls (RPC). `Vem` is the first process that starts up, and it produces a vem console window in the upper left corner of the screen and a green demo window just below it. When `pigiRpc` starts up, you should see a window in the middle of your screen that has the Mr. Ptolemy bitmap and a brief description of the binary you are running.

If the `pigiRpc` process fails to connect to the `vem` process, you won't see the Mr. Ptolemy bitmap, and the shift-middle-button menus will not be active. This problem seems to be most common on Linux machines, in part because they are often not on a network. If you are running under Linux and your installation is configured to use the network, then you may need to rebuild Ptolemy from source. See "Linux specific bugs" on page A-37 for more information.

If you are running on a machine that is not connected to a network, you will need to provide some network support for `pigi` to start up. `vem` and `pigiRpc` communicate with each other via RPCs, which require some intra-machine network support. One quick test is that you should be able to ping yourself:

```
/usr/etc/ping ‘hostname‘
```

There are several workarounds to this. One is to add the name of your host to the loopback line in `/etc/hosts` (here we add the name *myhostname*):

```
127.0.0.1  localhost myhostname
```

Under FreeBSD, you might have to add a fully qualified domain name. If you do not have a fully qualified domain name, sendmail might have problems. An example `/etc/hosts` entry would be:

```
127.0.0.1  localhost myhostname myhostname.mydomain
```

Another solution is to use `route` to route packets to your host through the loopback interface. As `root`, type:

```
route add ‘hostname‘ localhost 0
```

See the `ping`, `netstat` and `route` commands for more information about networking.

If the Mr. Ptolemy image fails to come up, another thing to check is that Tycho has the proper `tclIndex` files. If you try to open a facet by typing 'F', and you get a message like:

```
invalid command name "::tycho::Oct::openFacet"
    while executing
"::tycho::Oct::openFacet"
```

then check to see if the `TYCHO` environment variable is mis-set. You should be able to run Ptolemy with out setting `$TYCHO`, so if it is set, try `unsetenv TYCHO` and then restarting. Also, check to see that `$PTOLEMY/tycho/typt/kernel/tclIndex` exists. This file is used by `pigi` to find the `::tycho::Oct::openFacet` command at runtime. If it does not exist, create it by running `make sources` in that directory.

pigi fails to start when put in the background

A common problem occurs when `pigi` is started in the background and the user has the line

```
stty tostop
```

in their `.login` or `.cshrc` file. This command configures the terminal to halt any process that is running in the background when it tries to write to the terminal. One fix is to run `pigi` in the foreground. Another fix is to eliminate this command from your login files.

pigi fails to start up, giving shared library messages

On most platforms, Ptolemy is built using shared libraries. In Ptolemy 0.7.1, the Solaris2.x, HP and possibly Linux platforms use shared libraries, SunOS4.x does not. See the "Shared Library" appendix for more information about shared libraries.

At run time, if shared libraries cannot be found, you may see a message under HPUX-10.x like:

```
/usr/lib/dld.sl: Can't find path for shared library: libuprintf.sl
```

Under Solaris 2.x, you might see:

```
ld.so.1: /users/cxh/pt/bin.sol2/vem: fatal: librpcserver.so:
can't open file: errno=2
```

The message that you see may vary but the problem is that the binary cannot find the shared libraries to which it was linked. There are a few reasons this could be happening:

- The shared libraries are not where the binary expects them to be. If you are running from prebuilt binaries and your Ptolemy tree is not at `/users/ptolemy`, then this may be the problem.

  If you are running from prebuilt binaries and your Ptolemy distribution is not at `/users/ptolemy`, then you may need to set an environment variable to indicate what path should be searched for shared libraries. Under HPUX, the environment variable is `SHLIB_PATH`; under Solaris, the environment variable is `LD_LIBRARY_PATH`. The file `$PTOLEMY/.cshrc` should contain the proper commands to set the appropriate environment variable, though you may need to uncomment some lines. For HPUX, you could type the following command. (Do not type a space after the backslashes at the end of lines, just press `Return`):

```
setenv SHLIB_PATH {$PTOLEMY}/lib.{$PTARCH}:\
      {$PTOLEMY}/octtools/lib.{$PTARCH}:\
      {$PTOLEMY}/gnu/{$PTARCH}:\
      {$PTOLEMY}/tcltk/itcl.{$PTARCH}/lib/itcl
```

For Solaris, you could type the following command (all on one line):

```
setenv LD_LIBRARY_PATH {$PTOLEMY}/lib.{$PTARCH}:\
      {$PTOLEMY}/octtools/lib.{$PTARCH}:\
      {$PTOLEMY}/gnu/{$PTARCH}:\
      {$PTOLEMY}/tcltk/itcl.{$PTARCH}/lib/itcl
```

It is best to place these commands in your `~/.cshrc` file. It is possible that you might have to add other directories to the shared library path. For example, the Solaris2.x Ptolemy binaries are compiled with `/usr/openwin/lib` as the location of the X windows libraries. If your X windows libraries are in another directory, then you will need to add that directory to the shared library path. See “Window system problems” on page A-20.

- You do not have a library with that exact name. You may have an earlier or later version. Recompiling Ptolemy from scratch is one solution. It may also be possible to set up symbolic links to the proper libraries from a directory on the shared library path.

tycho fails to start up, giving `TCL_LIBRARY` messages

There are several ways to start up `tycho`, the Ptolemy syntax manager. `$PTOLEMY/bin/tycho` is a link to a script that processes command line arguments and starts up the appropriate binary. If you type `tycho -pigi`, `tycho` starts up with a binary that includes the Ptolemy system. If you type just `tycho`, then `tycho` starts up with the generic `itkwish` binary that is built from the `itcl` sources, which does not include any of the Ptolemy system.

If your Ptolemy distribution is not at `/users/ptolemy`, and you are running from prebuilt binaries, then if you run `tycho` with the prebuilt generic `itkwish` binary, you may see messages about:

```
application-specific initialization failed: can’t find /users/
ptolemy/tcltk/itcl/lib/tcl7.4/init.tcl; perhaps you need to install
Tcl or set your TCL_LIBRARY environment variable?
```

What’s happening here is the `itkwish` binary we ship has the `/users/ptolemy` path hard-coded into it and the binary is not finding the libraries it needs. The reason this happens is that we want `tycho` to be able to run outside of Ptolemy on machines that have only generic `itkwish` installed from the `itcl` distribution. There are a few workarounds:

- Create a link from `/users/ptolemy` to the Ptolemy distribution.
- `$PTOLEMY/bin/itkwish` is a link to a that will attempt to set your environment properly and then start the real `itkwish`. If you place `$PTOLEMY/bin` in your path before `$PTOLEMY/bin.$PTARCH`, then you will be running the `itkwish` script which might work for you.
- Run `tycho -ptiny` instead. The `tycho` script assumes that the Tcl installation is located in `$PTOLEMY/tcltk` if it is called with the `-ptiny`, `-ptrim` or `-pigi` arguments, so the script will do the right thing.
- Set environment variables in a script and then call `itkwish`:

```
#!/bin/sh
TCL_LIBRARY=$PTOLEMY/tcltk/itcl/lib/tcl
TK_LIBRARY=$PTOLEMY/tcltk/itcl/lib/tk
ITCL_LIBRARY=$PTOLEMY/tcltk/itcl/lib/itcl
ITK_LIBRARY=$PTOLEMY/tcltk/itcl/lib/itk
IWIDGETS_LIBRARY=$PTOLEMY/tcltk/itcl/lib/iwidgets2.0
export TCL_LIBRARY TK_LIBRARY ITCL_LIBRARY
export ITK_LIBRARY IWIDGETS_LIBRARY
exec $PTOLEMY/bin/tycho $*
```

For further information about troubleshooting Tycho, see the `$PTOLEMY/tycho/doc/troubleshooting.html`.

A.5.3 Common problems while running pigi

X11 version of pxgraph fails to come up or displays a blank window

If the X11 version of `pxgraph` program is given exceptional numbers, such as the IEEE floating-point `Inf`, `-Inf`, or `NaN` ("not a number"), then it will issue a cryptic error message "problems with input data" and will fail to display a plot. The stars that use `pxgraph` are supposed to intercept this and pop up an error message in a window. However, as of this writing, this does not work on all platforms. On such platforms, the error message, unfortunately, goes to the standard output, which may be buried several layers deep in your windowing system. It is also possible for the standard output to be lost, so that no error message appears. Thus, if you get such a `pxgraph` failure, look for instabilities in your Ptolemy schematic that would cause it to produce such exceptional numbers.

Old flowgraphs do not work (facets are inconsistent)

A `pigi` schematic contains references to icons. These icons are referenced by their location in the file system, typically using either an absolute path, a path relative to user's home directory, or a path relative to the environment variable `PTOLEMY`. If the master for any of these icons is not in the expected place, `vem` will issue a warning, telling you the facet is inconsistent, and there will be blank space in place of the icon in the schematic. To find out what icon masters are missing, run the program `masters`. Instructions for doing this are given in "Copying and moving designs" on page 2-50. Invalid masters will be labeled "`INVALID`". You must replace the invalid reference with a reference to a valid master. The Tcl script `$PTOLEMY/bin/ptfixtree` can be useful for changing large numbers of facets. `$PTOLEMY/bin/ptfixtree.tcl` file contains limited instructions on how to use it.

One typical scenario for users upgrading from an earlier version of Ptolemy is that they will have references to `~ptolemy` in the directory tree. But the newer version may be installed somewhere else. One solution is to use the masters program to replace references to `~ptolemy` with `$PTOLEMY`.

Ptolemy simulations do not stop

In the SDF domain, it is possible to have multirate systems where a single iteration fires a very large number of stars. This happens when the number of samples produced or consumed by various connected stars in the system are mutually prime, or they have very large least common multiples. If a simulation is taking an unreasonable amount of time, then look for such mutually prime numbers (e.g, rates such as 53:97). Sometimes, in such circumstances, it can take a long time for the simulation to respond to pushing the "stop" button. It should, however, eventually respond.

Multi-porthole galaxies fail

If a galaxy contains a input or output multi-porthole, and the icon of the galaxy is named `Foo.input=2`, `Foo.output=2`, etc., the galaxy will fail to compile. This is because Ptolemy behaves as if anything ending in `input=`*X* or `output=`*X* must be a star. Avoid using names like `Foo.input=3` for galaxies.

Star is a compiled-in star and cannot be dynamically loaded

When you create a new universe (schematic), the domain assigned to the universe by default is SDF. If you create stars in the palette that are from another domain and then try to run the universe, you may get the error message

```
star 'Poisson' is a compiled-in star of domain DE. Cannot dynamically
load a compiled-in star class.
```

The solution is to change the domain of the universe by choosing edit-domain from the pigi menu (the keyboard short cut is 'd').

A.5.4 Window system problems

Below we discuss various problems we've seen between Ptolemy and the X window system.

Error: ld.so: libXext.so.4: not found

You have not installed the shared library needed by Ptolemy when it is used under OpenWindows. See "Special considerations for use under OpenWindows" on page A-7.

pigi fails to start and gives a message about not finding fonts

The default fonts for `vem` are specified in the file `$PTOLEMY/lib/pigiXRes9`, and also `pigiXRes9.bw` and `pigiXRes9.cp`. These files define a set of X window resources. The `pigiXRes9.bw` file is used if `pigi` is started with the `-bw` option. The `.pigiXRes9.cp` file is used in `pigi` is started with the `-cp` option. The definitions in these files can be overridden by the user. For example, a user who prefers to use microscopic fonts could set the X resource as follows:

```
Vem*font:  *-times-medium-r-normal--*-120-*
```

If, however, the fonts defined in these files on not available on the system, then the Ptolemy installer should change them in the files `$PTOLEMY/lib/pigiXRes9*`.

The fonts for Tk (and hence, the fonts for most of the dialog boxes) are specified in `$PTOLEMY/lib/tcl/ptkOptions.tcl`. These may similarly require modifications at some sites. In the worst case, if many standard fonts are not available, it may be necessary to redefine the default fonts built into the Tk source code, and recompile Tk. You may find the X11 program `xlsfonts` useful.

Ptolemy startup window only has an OK button

If the Ptolemy startup window does not have the Mr. Ptolemy bitmap and the copyright button, but instead the startup window is very small and has only an OK button, then you probably have font problems, see the section above for details about fonts.

Emacs confuses .pl files with Perl or Prolog

The `.pl` extension used to define Ptolemy stars is the same extension used for the Perl and Prolog languages. Some text editors, such as Emacs, have special modes for editing Perl and Prolog files. These modes are inappropriate for editing Ptolemy files. You can add the following line to your `.emacs` file in your home directory:

```
(setq auto-mode-alist (cons '("\\.pl$" . c++-mode) auto-mode-alist))
```

Problems with the colormap

Some applications, for example FrameMaker 5, allocate as many colors as they can from the colormap when they start up. This may force applications that are started later (such as `pigi`) to have access to a very restricted set of colors. If when you start `pigi`, the welcome window appears in black and white, then you may have such a situation. If the situation is worse, and there are not enough colors in the colormap for `vem` to start, then you may not even get this far. One solution is simply to exit the offending application (e.g., FrameMaker or Netscape), and restart `pigi`. A better solution is to configure the offending application to use fewer slots in the colormap. We have found that for FrameMaker 5, the following X resources (placed in your `.Xdefaults` file) usually solve the problem:

```
        Maker.targetExactColors: 2
        Maker.minimumExactColors: 0
        Maker.targetColorCube: 4
        Maker.minimumColorCube: 1
```

This still leaves FrameMaker with a very rich set of colors to use. You may need to replace the 2 or 4 with smaller numbers if you have other color-intensive applications running (such as root window pictures of beaches in Tahiti).

The HP window system, VUE, may not display the correct colors when running Ptolemy. If the Vem window appears with white text on a tan background, or if the Ptolemy run window appears blue instead of tan, then VUE is getting the Ptolemy colors wrong.

The solution here is force VUE to use the regular Ptolemy X resources. Before starting `pigi`, in an xterm, do the following line:

```
xrdb -load $PTOLEMY/.Xresources
```

Then run `pigi`.

The window manager crashes

The window manager `twm` sometimes crashes when you are running Ptolemy. We do not know why. It seems to be an interaction with Tk. Our solution is to simply restart it. You may wish to make sure your configuration does not log you out when the window manager exits.

Problems with Sun Sparc5s with 24 bit TCX framebuffers

Some Sun Sparc5 machines have a 24 bit framebuffer called a TCX framebuffer. On these machines, `vem` will fail to start with a message like:

```
A Serious X Error has occurred:
         BadValue (integer parameter out of range for operation)
         Request X_QueryColors (minor code 0)
Type ‘y’ to continue, ‘n’ to exit, ‘a’ to abort:
```

The problem here is that the root window is a `TrueColor` window, which causes problems with `vem`. As a workaround, Arnaud LaPrevote suggests starting OpenWindows with

```
openwin -server Xsun -dev /dev/fb defclass PseudoColor
```

It seems you can safely ignore the warning message about specified class or parameter not available that occurs during startup. This bug does not occur on the 24 bit UltraSparc framebuffer, so it seems that the `vem` bug is only tickled by the TCX frame buffer. For more information, see `http://ptolemy.eecs.berkeley.edu/ptolemy0.7.1/html/sparc5tcx.html`.

Problems with Mac X and Ptolemy

Some people have had difficulties running Ptolemy with Mac X. Tze-Wo Leung of Bell Northern Research suggests the following setup for Mac X when using Ptolemy:

- My hardware setup is: Mac IIci w/ 20 MBytes DRAM, 210 MB HD and 21” two-page display/21gs Radius B&W monitor. The UNIX server is a HP 715 workstation.
- The ‘Display:’ option in the ‘Edit Remote Command’ MUST be set to ‘(2) Color Rootless’ mode.
- Increasing the memory allocation of Mac X to 4 MBytes also helps. The memory size can be increased by first exiting Mac X, then using the ‘Get Info’ option under the File menu to pop-up the file info window where one can change the memory allocation.
- In the control panel, the characteristics of the monitor are set to ‘Colors’ and ‘256’.

Problems with Exceed and Ptolemy

Under Hummingbird Exceed5.0, you may need to start up another X client before starting `pigi`. If you get an error message like:

```
Error: UGetFullTechDir: cannot read .Xdefaults ‘vem.technology’:
UserMain: OpenPaletteInit() failed
```

then try starting up another X client such as `xclock` before starting up `pigi`.

Problems with XFree86

XFree86 3.1 has problems with the `vem` Edit-Label widget. If you see messages like

```
A Fatal Xt Toolkit Error has occurred:
Attempt to unmanage a child when parent is not Composite
Type ‘y’ to continue, ‘n’ to exit, ‘a’ to abort
(continuing may have unpredictable consequences):
```

then the solution is to upgrade to XFree86 3.2.

A.5.5 Problems with the compiler

The first thing to try is compiling a ‘hello world’ program in C or C++. In C++, you should probably try using the stream functions, below is a sample file:

```
#include <stream.h>
main() { cout << "Hello, Ptolemy.\n"; }
```

Try compiling the file with `g++ -v` and `-H` flags turned on. `-v` tells you what steps the compiler is running, `-H` tells you what include files are being read in.

```
g++ -v -H hello.cc
```

Look at each step of the compilation, and pay particular attention to the assembler and loader steps. You can use the `-save-temps` gcc option to save any temporary files created in each step. Then, if necessary, you can try running each step by hand.

as vs. gas

`gcc` can use the native assembler or the GNU assembler. Often the GNU assembler is installed as ‘`as`’. Check your path to see which version you are getting. `gcc` can often be configured at compiler build time to use the native assembler or the GNU assembler (`gas`), but once the compiler is built, you are stuck with one or the other assemblers. The Ptolemy project makes GNU binaries available. Most of the GNU binaries that we distribute use the native assembler, that is, they don’t use `gas`. However, the hppa GNU `egcs-1.0.2` binaries use `gas`. We distribute a `gas` binary with the hppa GNU `egcs-1.0.2` binaries.

Collect

To pick up C++ constructors and destructors, `g++` can use the native loader or a program called ‘`collect`’ (for more information, see Joe Buck’s g++ Frequently Asked Questions [FAQ] described below). We usually use `collect`, because it works with the Pure Inc. tools. The collector is usually located at `gcc-lib/$PTARCH/COMPILER_VERSION/ld`, e.g. the gcc-2.7.2.2 sun4 collector might be at `$PTOLEMY/gnu/sun4/lib/gcc-lib/sun4/2.7.2.2/ld`. Note that `g++` under Solaris2.x does not use `collect`.

You can pass the collector arguments so that it will print out more information. Try

```
g++ -v -Wl,-debug hello.cc
```

or, if you are within Ptolemy:

```
    make LINKER="g++ -v -Wl,-debug"
```

If you pass `collect` the `-debug` flag, you will get a lot of output. Part of the output will include what binaries and paths collect is using. Below is part of the output the `collect -debug` generated by a working installation.

```
    ld_file_name        = /usr/bin/ld
    c_file_name         = /users/ptolemy/bin.sun4/gcc
    nm_file_name        = /usr/sww/bin/gnm
    strip_file_name     = /usr/tools/gnu/bin/gstrip
    c_file              = /usr/tmp/cca01064.c
    o_file              = /usr/tmp/cca01064.o
    COLLECT_NAMES       = /users/ptolemy/gnu/sun4/lib/gcc-lib/
    sun4/2.7.2.2/ld
    COLLECT_GCC_OPTIONS = -v -L../../lib.sun4 -static -L../../
```

```
        octtools/lib.sun4 -L../../tcltk/tk.sun4/lib -L../../tcltk/
        tcl.sun4/lib -L/usr/X11/lib -o pigiRpc
        COLLECT_GCC         = gcc
```

If you need to change the `*_file_name` values, try modifying your `$path` so that the new program is in front of the program listed. For instance, if, under `csh`, one wanted to use `/usr/local/bin/nm` instead of `/usr/sww/bin/gnm` in `nm_file_name` above, one would type:

```
     set  path=($PTOLEMY/bin  $PTOLEMY/bin.$PTARCH  /usr/local/bin
$path)
```

The collector will also respond to certain environment variables, see the source in the `gnu` tar overlay at `$PTOLEMY/src/gnu/src/gcc/collect2.c`.

The collector creates a temporary file that has the constructors and destructors in it. To get collect to save the temporary file, set the following environment variable:

```
        setenv COLLECT_GCC_OPTIONS -save-temps
```

If the collector is getting an old version of GNU `nm`, then you could have problems. Passing the collector the `-debug` flag might help here.

Error: Linker: no constructors in linked-in code!

If you see the above message while linking a new star, then you might be having `nm` problems. The above message seems to occur under Linux. Joe Buck says that the thing to do is:

The incremental linker is searching the object file for a global symbol that has the form of a constructor for a static or global object. It then calls that constructor. Ptolemy stars use these constructors to “register” themselves on the list of known stars. You can then create instances of your new star by using its `clone()` method. If it can’t find the symbols, then the new star’s code isn’t accessible.

The “`nm`” program is used to find the constructor symbols. Perhaps you have an older version of `nm` on your system? Find the `.o` file corresponding to your star and execute

```
        /usr/bin/nm -g --no-cplus mystar.o | grep GLOBAL
```

(`--no-cplus` tells nm not to demangle the symbols). You should get some constructor and destructor symbols.

Environment variables

Shell environment variables control where the Gnu compiler looks for subprograms and include files. For more information, see“Gnu Compiler Environment Variables” on page A-8.

Using trace

The SunOS4.1 `trace` command can be invaluable in determining what a program is doing at run time. If you compile with `gcc -v -save-temps` then you can try running `trace` on the various steps, and see each system call. Unfortunately, the filenames are truncated, but often this is enough to see what is going on. Solaris has a similar `truss` command.

A.5.6 Problems compiling files

There are several ways to handle problems while compiling files. These problems are often caused by strange interactions between `.h` files, and they occur while compiling a particular file or a set of files. Note that these problems are different than problems that occur during link time, which we discuss in "Missing symbols while linking pigiRpc" on page A-29.

Using cpp to diagnose .h file problems

If you are having problems with include files, try modifying a hello world program (see above) to include those files. Note that you could be getting unexpected substitutions from the C preprocessor `cpp`, so looking at the `cpp` output can be useful in solving compiler installations problems and include file problems

The gcc `-E` and `-P` options are very useful in wading through include file problems. `-E` stops compilation after the C preprocessor runs, and outputs the resulting file. `-P` strips off the line numbers from the output.

Try using the `-E` option, and look at the output file. Sometimes the problem will be obvious. Note that if your compile arguments include `-o` *`filename.o`*, then *`filename.o`* will have `cpp` text output, not the usual object file. Note further that in some compilers, the `-c` option (create a `.o` file) will override the `-E` option. If `-c` does override `-E`, you will have to grab the output of the make command and place it in a temporary file, say `/tmp/doit`, edit `/tmp/doit` and remove the `-c` option and then type `sh /tmp/doit`. If `-c` does not override `-E`, and you are within Ptolemy, you can try using the `OPTIMIZER` makefile flag to pass arguments to the compile. For instance:

```
make OPTIMIZER=-E Linker.o > Linker.e
```

Another approach is to run `cpp` and then re-run the compiler on the `cpp` output. In `gcc,` the `-P` option strips out the `cpp #line` comments. You can use `-E -P` to generate a new file that has all the cpp substitutions in it, and then try compiling the new file:

```
make OPTIMIZER=”-E -P” Linker.o > tst.cc
```

Edit `tst.cc` and remove the first line, which will have the gcc command in it. Make `tst.o`:

```
make OPTIMIZER=”-v -H” tst.o
```

Using the `gcc` arguments `-E -dM` will tell you what symbols are defined by `cpp` at the end of the compile. See the `gcc` man page or the `gcc` info format file for more information.

Narrowing the problem down.

If you are having strange problems compiling one file, you might want to try to find the smallest file that causes the problem. This method can take time, but it is sometimes the only way to find a solution. One way is to wrap code in `#ifdef NEVER` ... `#endif` and narrow the bug down to one function. Changing the `#include` declarations at the top, and following the include file change can also help here.

Using c++filt to demangle symbols

When a C++ file is compiled, the symbol names found inside a `.o` file or a library file have been specially processed by the compiler. This special processing is called mangling.

The symbol names may look unusual, for example, `makeNew__10KnownBlockPCcT1` is the mangled version of `KnownBlock::makeNew(char const *, char const *)`. You may find it useful to be able to convert the mangled symbol names back to the human readable C++ symbol name. Under gcc-2.7.2.2, you can use the `c++filt` program to do the conversion:

```
cxh@brahe 9% echo “makeNew__10KnownBlockPCcT1” | c++filt
KnownBlock::makeNew(char const *, char const *)
cxh@brahe 10%
```

On platforms where we distribute the GNU compiler, `c++filt` can be found at `$PTOLEMY/bin.$PTARCH/c++filt`.

Sources of information for compiler problems

`$PTOLEMY/gnu/common/man/man1/gcc.1` contains the gcc man page. This file is shipped with the prebuilt GNU binaries. You can try placing `$PTOLEMY/gnu/common/man` in your `MANPATH` environment variable:

```
setenv MANPATH $PTOLEMY/gnu/common/man:$MANPATH
```

`$PTOLEMY/gnu/common/info/gcc*` contains the GNU Info format documentation. Use Emacs (M-x info) or a program such as `tkinfo` to view the info pages (`tkinfo` is available via anonymous FTP from `ptolemy.eecs.berkeley.edu` in `pub/misc`).

`$PTOLEMY/src/gnu/g++FAQ.txt` is Joe Buck’s g++ Frequently Asked Questions document in text format, (g++FAQ and other FAQs are available via anonymous FTP from `rtfm.mit.edu` in `pub/usenet/news.answers`).

The following FAQs might also help: c.faq hpux.faq solaris2.faq solaris2_porting.faq sun_sysadmin.faq.

A.5.7 Generated code in CGC fails to compile

The `Makefile_C` target uses the Ptolemy makefile structure to determine platform dependencies in the C language Code Generation (CGC) domain compile command. If you are having problems with platform dependencies, you may want to use the `Makefile_C` target. Some demos, such as the CGC `commandLine` demo use the `Default-CGC` target. The `Default-CGC` target has the compiler name set as a target parameter, which is usually either `gcc` or `cc`. Unfortunately, not all machines are shipped with a working `cc` binary and not all machines have `gcc`, so we cannot choose a default that will work in all circumstances. If you do not have the compiler that is listed in the target parameter, you can do any of the following:

- Type a ‘T’ while in the facet to bring up the Target Parameters window and change the value of `compilerCommand` to a compiler that you have.
- Create a link in `$PTOLEMY/bin.$PTARCH` for the compiler you don’t have. For example, if you don’t have `gcc`, and your `cc` is at `/usr/ccs/bin/cc`, you could do

  ```
  cd $PTOLEMY/bin.$PTARCH; ln -s /usr/ccs/bin/cc gcc
  ```
- Use the `Makefile_C` target instead of the `Default-CGC` target. (Some demos still use the `Default-CGC` target so that we can continue to test that target).

The targets in CGC are configured by default with reasonable guesses about the compile and link options that are required to compile the code. However, the actual options required depend on your system configuration. For instance, your default `cc` compiler may not have been configured to automatically find the X11 include files. You might, therefore, get an error message the `Xlib.h` cannot be found. You should find out where on your system `Xlib.h` is installed, use the 'T edit-target' command to add a compile option of the form `-L`*path_name* where *path_name* is the full path of the directory containing the file. 'T edit-Target' is on the shift-middle-button menu.

Certain compilers will change their behavior depending on the values of certain environment variables:

- The hppa `cc` compiler will use the `CCOPTS` environment variable. If your X11 libraries were at `/usr/sww/X11R5/lib`, then one could exit `pigi`, set the variable `setenv CCOPTS -L/usr/sww/X11R5/lib` and restart `pigi`. Then when you compile CGC demos with hppa `cc`, you should be able to find the proper libraries. See the hppa `cc` man page for more information. Note that under HPUX10.x, the bundled C compiler is not ANSI compliant so it may fail to compile some CGC Universes.
- GNU `gcc` will also use certain environment variables. The `LIBRARY_PATH` variable may help:
  ```
  setenv LIBRARY_PATH /usr/sww/X11R5/lib
  ```
  See the `gcc` documentation for more information. See also "Gnu Compiler Environment Variables" on page A-8.

A.5.8 Ptolemy will not recompile

If Ptolemy fails to recompile, you may be using a substantially different version of the GNU compiler. The system is most likely to build if you use the same tools that we used originally. The GNU tools we used are supplied with the Ptolemy distribution. We discuss common Ptolemy compilation problems below. For further information about recompiling Ptolemy, see "Rebuilding Ptolemy From Source" on page A-10 and see Volume 3 of the Ptolemy Almagest, "The Ptolemy Programmer's Manual".

Messages about "unexpected end of line seen" while running make

If you are running a version of make other than GNU make, you may see messages like:

```
make: Fatal error in reader: ../../mk/stars.mk, line 52: Unex-
pected end of line seen
```

Ptolemy contains GNU make extensions, you must run GNU make to build Ptolemy, even if you are not using the GNU compiler. GNU make binaries are available via anonymous FTP in `ptolemy.eecs.berkeley.edu`. The Ptolemy binary tar files for hppa, sol2 and sun4 contain GNU make binaries. You can get just the GNU make binary in `pub/gnu/$PTARCH/make.gz`, where `PTARCH` is one of hppa, sol2 or sun4, or you can get the GNU make binary, along with the GNU compiler and other binaries in `pub/gnu/ptolemy0.7.1`.

Apparently, older versions of GNU make, such as 3.71, can fail with a message like:

```
../../mk/stars.mk:113: *** commands commence before first target.
Stop.
```

If you get such a message, type `make -v` to see what version of GNU make you are running.

Can I use my own version of Tcl/Tk?

Ptolemy 0.7.1 uses `itcl2.2`, which is an extension to Tcl/Tk. If you have `itcl2.2` already installed, you may use your installed version. See "Ptolemy and Tcl/Tk" on page A-13. Tycho will not work with `itcl2.1`, you must use `itcl2.2`. Tycho is necessary for viewing the contents of stars and other important features.

Can I use my own version of gcc and libg++?

Ptolemy 0.7.1 uses egcs-1.0.2, but `gcc-2.7.2.2` and `libg++-2.7.2` should also work, see "Dynamic linking fails" on page A-29 for information about Gnu versions and Dynamic linking.

To determine what version of gcc you are running, type `gcc -v`. To determine what version of libg++ you are running look at the libg++ filename.

```
cxh@kahn 32% gcc -v
Reading specs from /users/ptolemy/gnu/sol2.5/lib/gcc-lib/
sparc-sun-solaris2.5.1/2.7.2.2/specs
gcc version 2.7.2.2
cxh@kahn 33% ls /users/ptolemy/gnu/sol2.5/lib/libg++.so*
/users/ptolemy/gnu/sol2.5/lib/libg++.so@
/users/ptolemy/gnu/sol2.5/lib/libg++.so.2.7.2*
```

Ptolemy is configured to use shared library versions of `libg++` and `libstdc++` if they are supported on your platform. To compile Ptolemy from scratch, you should be sure that you have these libraries. Under Solaris, the libraries are named `libg++.so` and `libstdc++.so`. Under HPUX10.x, these libraries are named `libg++.sl` and `libstdc++.sl`. A common problem is that the proper version of gcc is installed, but only the static libraries were built.

Under Solaris, you might see error messages like:

```
ld: fatal: relocations remain against allocatable but non-writable
```

The fix is to configure `gcc` and `libg++` with `--enable-shared` and build the shared libraries or to download the prebuilt Gnu binaries. See `ftp://ptolemy.eecs.berkeley.edu/pub/ptolemy/ptolemy0.7/html/g++shared.txt` for more information.

Can't find genStarList or genStarTable during recompilation

The solution is to include `$PTOLEMY/bin` in your path. We don't include certain files that are derived from other files. In the star directories, `.cc` and `.h` files are derived from `.pl` files, and the *domainname*`stars.cc` file is generated from `$PTOLEMY/bin/genStarTable`.

"CGCMakefileTarget.h: No such file or directory" while linking pigiRpc

If you are in `$PTOLEMY/obj.$PTARCH/pigiRpc`, and you type `make`, and `$PTOLEMY/obj.$PTARCH/domains/cgc/targets/main/CGCMakefileTarget.o` does not exist, then `make` will try to create it. Unfortunately, `make` does not have the include files right, so `CGCMakefileTarget.h` is not found. There is nothing particularly special about `CGCMakefileTarget.o.` It is just first in the list of files on which `pigiRpc` depends.

The workaround is to run `make` from `$PTOLEMY`, rather than `$PTOLEMY/obj.$PTARCH.pigiRpc`.

Eventually, we would like to fix this so that it is not necessary to build in other directories before building in `$PTOLEMY/obj.$PTARCH/pigiRpc`. The solution here would be to move more `.o` files into `lib.$PTARCH`.

Missing symbols while linking pigiRpc

On the sun4 with libg++-2.5.2, if you compile pigiRpc with the g++ `-O` option, then you may have missing symbols while linking `pigiRpc`. The workaround is to upgrade to a newer version of libg++. If you are using prebuilt Gnu binaries, then you may need to set an environment variable (see “Gnu Compiler Environment Variables” on page A-8).

If, at link time, you see messages about undefined symbols, and the undefined symbols begin with `_vt`, then you probably have compiler version incompatibilities. In earlier releases of `libg++`, we have seen cases where the symbol `vt$7istream$3ios` is undefined. This symbol should be present in `libg++.a`, but it seems that if the compiler is not built with `-O2`, then this symbol will not be present in `libg++.a`. The workaround is to rebuild the library by hand, with something like:

```
cd obj.$PTARCH/gnu
make CC=/users/ptolemy/gnu/sun4/bin/gcc CXX=/users/ptolemy/gnu/sun4/bin/g++ CFLAGS=”-g -O2” CXXFLAGS=”-g -O2”
```

Note that you can find out what files have the undefined symbols by using the Unix `nm` command. For example on Suns, the command `nm -o $PTOLEMY/lib.sun4/* | grep MySymbol` will find all the files that have symbols that contain the string `MySymbol`. See “Using c++filt to demangle symbols” on page A-25 for information about how to interpret symbol names in a library.

If, at link time, you see messages about undefined `ifstream` symbols in `libptolemy.a`, then the problem could be that you are using `gcc-2.7.2`, but linking against `gcc-2.7.2.2` libraries. Brian Evans pointed out that fix is to remove the `libg++` and `libstdc++` libraries in `$PTOLEMY/gnu/$PTARCH/lib` and create symbolic links to your local Gnu installation.

A.5.9 Dynamic linking fails

Ptolemy has the ability to load stars dynamically during run time. The stars are compiled into `.o` files and loaded with the Unix loader or with the `dlopen()` function. Dynamic linking is tricky and dependent on the Unix loader. There are several reasons dynamic linking can fail:

- If you are upgrading from an earlier release be sure to remove all the .o files in the directory where the source files for you star is located.
- If you are using prebuilt Ptolemy binaries, be sure that you are using the prebuilt Gnu compiler that is also available with the Ptolemy binaries (Ptolemy0.7.1 was built with `egcs-1.0.2`) The alternative is to rebuild Ptolemy with your local Gnu compiler, but be aware that versions earlier than `gcc-2.5.6` and `libg++-2.5.3` have bugs. `gcc-`

`2.4.x` and `libg++-2.4.x` and earlier are known to have serious bugs, so you may want to upgrade. For more information, see “Gnu Installation” on page A-7.

- If you are using prebuilt Ptolemy binaries and have the prebuilt Gnu compiler, be sure that either the Ptolemy distribution is available as `/users/ptolemy`, or you are setting the Gnu environment variables in `$PTOLEMY/bin/g++-setup`. Again, “Gnu Compiler Environment Variables” on page A-8.
- If, at link time, you see messages about undefined symbols, then see the section above, “Missing symbols while linking pigiRpc”.
- Dynamic linking may not work on machines using a different release of the operating system than that used to build the Ptolemy binaries. The solution is to rebuild Ptolemy from source.
- If you are having problems compiling the star from Ptolemy, try running the compile by hand from the shell. Unfortunately, part of the compile command is not always visible in the vem window. To see the all of the compile command, try running a few `vem` commands, such as ‘i’ (look-inside) to flush the vem buffer. Once you have produced a `.o` file, you can load the `.o` file into Ptolemy with the load-star command.
- If you cannot compile your star from the shell, try compiling a simple c++ program to verify that the compiler is working. Place the code below in a file called `hello.cc` and if you are using the Gnu compiler, try compiling it with `g++ -v`. The `-v` option will show the compiler steps.

  ```
  #include <stream.h>
  main(){ cout << "Hello, Ptolemy.\n";}
  ```
- If you are having problems with undefined symbols at load time, try compiling your star with the same level of optimization as the binary was built with. We ship `pigiRpc` and `ptcl` binaries that have been compiled with `-O2`, so you may want to compile your star with `-O2`.
- If you are running under HPUX9.x, and you see messages like:

  ```
  collect2: ld returned 1 exit status
  /bin/ld: Invalid loader fixup needed
  ```

  then you probably need to create a `make.template` file to load your stars. The problem here is that Ptolemy attempts to compile your star with default arguments, however, since HPUX9.x uses `shl_load()` style linking, you need special compiler arguments, so you need a makefile See `$PTOLEMY/mk/userstars.mk` for more information.

  You may also need to upgrade your version of the Gnu assembler named `gas`. Version 2.5.2 has been reported to have the problem, while version 2.6 seems to work fine. Note that `gas` is often named `as`. If you compile your star with the `g++ -v` option, then you will see which assembler the compiler is using. You can then call the assembler with the `--version` flag to see what version the assembler is.

A.5.10 Dynamic linking and makefiles

You may find it easier to use a `makefile` to build `.o` files for incremental linking. As part of the incremental linking process, `pigi` checks for the existence of a `Makefile` or `makefile` in the directory where the star resides. If a `Makefile` or `makefile` exists, then `make` *`XXXStarName`*`.o` is run, where *`XXXStarName`*`.o` is the name of the `.o` file to be incrementally loaded.

Another approach is to create a `make.template` file in the directory that contains rules to convert the `.pl` file to a `.o` file. If the `make.template` file includes `$PTOLEMY/mk/userstars.mk`, then most of the configuration is done. For example, to include a star `SDFSensorExcitation`, with optimization set at `-O2`, the `make.template` would contain:

```
ROOT = $(PTOLEMY)
VPATH = .
OPTIMIZATION=-O2
include $(ROOT)/mk/config-$(PTARCH).mk
INCL = -I$(ROOT)/src/domains/sdf/kernel -I$(KERNDIR)
PL_SRCS = SDFSensorExcitation.pl
DOMAIN = SDF
include $(ROOT)/mk/userstars.mk
```

Then, from a shell, the command to execute would be `make -f make.template depend` and then, from within `pigi`, it would be possible to link in the star. See the contents of `userstars.mk` for complete instructions.

If you have a star that requires multiple `.o` files, Tom Parks points out that `ld -r` might help. For example if `SDFWirelessChannel.o` uses functions from `Wireless.o`, the following commands might help:

```
ld -r SDFWirelessChannel.o Wireless.o
mv a.out SDFWirelessChannel.o
```

To load in multiple stars, you may find the `ptcl multilink` command useful.

A.5.11 Path and/or environment variables not set in "debug" pigi

When running Ptolemy's interactive graphical interface with the debug option

```
pigi -debug
```

the path may not be set correctly, or environment variables are not at their normal values. This is caused by the Gnu debugger, `gdb`, overwriting values set by the `pigi` start-up script. From the `gdb` manual:

```
*Warning:* GDB runs your program using the shell indicated by your
`SHELL' environment variable if it exists (or `/bin/sh' if not). If
your `SHELL' variable names a shell that runs an initialization file-
-such as `.cshrc' for C-shell, or `.bashrc' for BASH--any variables
you set in that file affect your program. You may wish to move setting
of environment variables to files that are only run when you sign on,
such as `.login' or `.profile'.
```

If your `.cshrc` file specifies a value for a variable, it will override anything in the pigi start-up script. If this is the case, perhaps setting the `SHELL` environment variable to `/bin/sh` before firing off the debugger will fix the problem.

1.5.12 DE Performance Issues

DE Performance can be an issue with large, long-running universes. Below we discuss a few potential solutions.

Tom Lane pointed out that the Calendar Queue scheduler can be slower than the old DE scheduler if your time stamps span a wide range. This is because the Calendar Queue Scheduler tries to set up too many bins spanning the range. The old DE scheduler may work faster, as it keeps a queue of current-scheduled events, which is often fairly short.

Tom Lane also pointed out:

> If you have both a wide range of timestamps and a lot of future events in the queue at once, you might find it would help to improve the `PriorityQueue` code to provide a genuine priority queue (i.e., a heap, with O(log N) performance) rather than a simple list like it is now. But you ought to profile first to see if that's really a time sink.
>
> Also, you have to keep in mind that the overhead for selecting a next event and firing a star is not trivial. It helps if your stars do a reasonable amount of useful work per firing.

A few other points that may help you:

DE simulation can have certain inherently high cost, so using SDF or DE with SDF functionality inside wormholes can greatly improve performance.

- If you are running a long simulation, you should be sure that your machine is not paging or worse yet swapping, you should have plenty of memory. Usually 64Mb is enough, though 128Mb can help. Depending on what platform you are on, you may be able to use the program `top` (ftp://eecs.nwu.edu/pub/top). You might also find it useful to use `iostat` to see if you are paging or swapping.
- One way to gain a slight amount of speed is to avoid the GUI interface entirely by using `ptcl`, which does not have Tk stars. See "Some hints on advanced uses of ptcl with pigi" on page 3-19 for details.

A.6 Known bugs

There are a number of known bugs that we have not had a chance to fix. You may find useful information about architecture dependencies in `$PTOLEMY/mk/config-$PTARCH.mk` for the particular architecture you are running under.

A.6.1 Bugs in vem

- Deleting wires that are too long (i.e., end of wire is past the terminal, but near it) can result in `vem` dumping core.
- Labels that end with a carriage return are neither displayed nor printed correctly.
- Editing icons stimulates a memory leak in `vem` that can make the process grow quite big. After editing a few dozen icons, you may wish to exit `vem` and restart.
- The "layer" command only works when editing icons, not when editing schematics.

- The “copy-objects” command will copy from one window to another only when editing icons, not when editing schematics.
- `vem` or `pigiRpc` may fail to start up on 4 bit (16 color) screens. At UC Berkeley, we use 8 bit and 24 bit screens. Black and white (1 bit) support in vem is a little weak. If you are on a black and white screen, you may need to modify `$PTOLEMY/lib/pigiXRes9`. (Note that `$PTOLEMY/lib/pigiXRes9.bw` is read when the `-bw pigi` option is used to create black and white screen dumps from a color monitor. `pigiXRes9.bw` has very little to do with running on a black and white screen).
- If the `VEMBINARY` environment variable is set, then `bin/pigiEnv.csh` will use the contents of that variable as the `vem` binary. However, the binary must be named `vem`, or there will be minor problems with X resources, such as small fonts and incorrect snap. Most users will never use the `VEMBINARY` environment variable. It is primarily used to debug `vem`.
- If your X server runs out of colors, `vem` may crash. One workaround is to not run color hogs like `Netscape`, or to limit the number of colors `Netscape` can use.
- Bug fixes to `vem` now mean that `vem` is more picky about text label heights. Currently, the range for text label heights is 0 to 80. If you select text and then use E, to edit the label, you may see an error if your text height is greater than 80. One workaround is to delete the text and reenter it with a height that is within the proper range. Another workaround is to use an old `vem` binary to edit the height and set it to within the range.
- If you re-read a facet that has an open run window, you won’t be able to dismiss the run window. The workaround is to open a new run window.

A.6.2 Bugs in pigi

- It is possible to make more than one icon to represent a given star. For instance, the icon `And` and `And.input=2` refer to the same star, but the former can have any number of inputs, while the latter has exactly two inputs. As of this writing, this facility does not work for galaxies. You should create only one icon for each galaxy.
- `$PTOLEMY/bin/pigi` is a link to `$PTOLEMY/bin/pigiEnv.sh` which may add to your path. Under certain circumstances, you may get the message ‘Warning: ridiculously long PATH truncated’. To work around this problem, try shortening your path. One trick is to use shorter pathnames in your path, perhaps using symbolic links.
- If your X server runs out of colors, `pigi` may crash. One workaround is to not run color hogs like `Netscape`, or to limit the number of colors `Netscape` can use.
- The compile-SDF target fails if a galaxy has incrementally linked in stars. The problem is that the `.h` file for the incrementally linked in star cannot be found by compile-SDF. As a workaround try editing the makefile in `~/PTOLEMY_SYSTEMS` and adding the directory where the incrementally linked stars are. Then build the binary by hand.
- The compile-SDF target cannot handle star parameters that use the Tcl Interpreter to evaluate Tcl commands.
- On some machines, the SDF Matlab demonstrations will appear to hang Ptolemy. This

is a known bug in the Matlab external interface library that requires that the Matlab process be attached to a terminal. There are two workarounds: (1) run Ptolemy in the foreground, or (2) manually control the Ptolemy interface to Matlab by using the `matlab` command in `ptcl` (see 3.9.10 on page 3-16) via the Ptolemy console. In both workarounds, the Matlab process will be attached to a terminal.

- The Networks Of Workstations active messages (`NOWam`) CGC target is slow, but it does run under Solaris. However these demos will not work in an environment that requires Kerberos for `rsh`-ing jobs.
- Mixed CGC/VHDL/TclTk demos leave `ptvhdlsim` running after exiting.
- Tom Lane pointed out the following problem in 0.7:

  > An initializable delay attached to any multiporthole output will fail. For example "Fork -> Printer", if you use the double-arrowed form of the Fork icon and put an initializable delay on the arc.

- The ACYLOOP Scheduler can fail in various ways for both simulation and code generation. In particular, Compile-SDF should use the DEF scheduler, not ACYLOOP.

A.6.3 Bugs in tycho

- If you start up tycho with Ptolemy in the background (i.e, `tycho -ptrim &`), then exiting from the Matlab console may hang. The workaround is to place your `tycho` process into the foreground. This is a known bug with Matlab.
- See `$PTOLEMY/tycho/doc/bugs.html` for a more complete list of Tycho bugs.

A.6.4 Code generation bugs

- The CGC multirate `filterbank` and CGC `chaoticBits` demos bring up messages about initializable delays.
- If you try to run the CGC demos that generate sound, and you do not have a properly configured audio device, or you do not have write permission to `/dev/audio`, then you get a cryptic and uninformative error message, rather than something useful. The CGC `tremolo` demo will probably only work on a Sun SPARCstation running SunOS4.1.3 or Solaris2.x. The CGC `alive` demo will only work on an SGI.
- The use of initializable delays and variable delays in the code generation domains may result in incorrect code being generated. Variable delays are delays which depend upon the value of a galaxy or universe parameter. The workaround is to use regular delays of a constant value (i.e. "4") in the code generation domains. Note that initializable and variable delays work fine in the simulation domains.

  For example, the `CGC:multirate:filterbank` demo produces initializable delay warnings. The output from the `CGC:multirate:filterbank` demo is different than the output from the `SDF:multirate:filterbank` demo in that the original and reconstructed signals don't overlap as much in the CGC version.

A.6.5 Bugs in the X11 version of `pxgraph`

- If the X11 version of `pxgraph` is given exceptional numeric input, such as the IEEE floating point `Inf`,
  `-Inf`, or `NaN`, then it displays a blank window only.
- If `pxgraph` is given an empty file to plot, all it does is send a message “problems with input data” to the standard output.
- There is no way to produce hardcopy without running `pxgraph` interactively.
- Specifying the colors for data sets using X resources does not work.
- See `$PTOLEMY/src/pxgraph/README.txt` for information about the Java version of `pxgraph`.

A.6.6 HPPA specific bugs

- If you are rebuilding Ptolemy on the hppa, you must have X11 installed. Apparently, HP ships machines without most of the X11 include files. The prebuilt binaries also use libraries from X11R6. See `ftp://ptolemy.eecs.berkeley.edu/pub/ptolemy/contrib/hpux`.
- If you are building the Gnu tools under HPUX10.20, you will also need Gnu `sed`, which is downloadable from the Ptolemy ftp site in `ftp://ptolemy.eecs.berkeley.edu/pub/gnu/hppa` and you may need to patch HPUX. See `$PTOLEMY/src/gnu/README` for details.
- If you are using the prebuilt binaries under HPUX10.20, then you will need to install X11R6 shared libraries. See `ftp://ptolemy.eecs.berkeley.edu/pub/ptolemy/ptolemy0.7.1/README.hpux`
- The CGC demo `animatedLMS` demo dumps core upon start-up.
- On the hppa, under the HP C++ compiler, `Xhistogram` may have rounding problems.
- On the hppa, the CGC `animatedLMS` demo requires editing of Target Parameters to compile.
- On the hppa, the CGC `animatedLMS` demo gets an Arithmetic Exception and core dumps.
- On the hppa, the SDF `animatedLMS` demo generates a “`non-numeric value`” error in Tcl if the step size is so large that the system is made to go unstable.
- Compilation under older versions of `hppa.cfront` may fail. The problem seems to be with the `+A ld` command line argument. You must use the `+A ld` argument when compiling `pigiRpc` and `ptcl`, or incremental linking of new stars will fail. However, use of the `+A` argument produces the warning:

  ```
  CC: error: could not find __head symbol. You must use CC to
  link. If your main is not in C++, a call to _main() is
  required.(740)
  ```

Even if this warning is printed, a viable binary is still produced. The +A `ld` argument is not necessary for other binaries, such as `vem` and `pxgraph`.

- If you are running an older version of the HP Cfront compiler, you may need to add `-DPOSTFIX_OPT=` to your c++ command line. See `config-hppa.cfront.mk`.
- If you are having problems incrementally linking in stars, and you are getting messages like:
  `/bin/ld: (Warning) Inter-quadrant branch in XXX`
  where `XXX` is the name of the `.o` file you are trying to link in,
  then you may need to be compiling your stars with the proper level of optimization. See "Dynamic linking and makefiles" on page A-31 for more information.
- If you have problems with `ld` under HPUX, you should try patching your operating system with a patch from HP. Try `http://europe-support.external.hp.com` if you are in Europe, `http://us-support.external.hp.com` if you are anywhere else.
- If, under HPUX9.x, you get a message like:
  `ld: fatal: relocations remain against allocatable but non-writable sections`
  Then you may need to apply a patch to `ld`.
- To incrementally link new stars under HPUX9.x, you will probably need to supply a `make.template` file. See "Dynamic linking fails" on page A-29.
- The `multilink` command seems to trigger `Inter-quadrant branch messages`, and then hang pigi. We are not sure why.
- HPUX10.x will build a `pigiRpc` with the Process Network (PN) domain, but the event loop is broken. The PN domain is not part of the default hppa build, so this problem will not affect most users.
- If you are under HPUX10, then building the PN domain requires Distributed Computing Environment (DCE) threads. You will need to install the DCE development set of the OS CDs. If you don't have a `/usr/include/pthread.h`, then you probably don't have the DCE development set installed.
- The Java version of `pxgraph` is slow under HPUX. See `$PTOLEMY/src/pxgraph/README.txt` for information on using the X11 version of `pxgraph`.

A.6.7 IBM AIX specific bugs

Xavier Warzee suggest the following for gcc and AIX.

To generate `gcc-2.7.2` under AIX3.2.5 with `cc`, you need the PTF U436313. The PTF U432238 is suggested in the README.RS6000 file from the `gcc-2.7.2` distribution, but this PTF is superseded by the PTF U436313. This PTF allows you to upgrade the cc IBM compiler from the release 1.3.0.0 (delivered with AIX 3.2.5) to the release 1.3.0.33 which compiles gcc-2.3.6.

A.6.8 Silicon Graphics IRIX5 specific bugs

The Silicon Graphics port is not one of our main ports, so there are several serious bugs:

- Linking of a full size `pigiRpc` binary can fail with a ‘GOT Overflow’ message. The problem is that `pigiRpc` has too many symbols. The Irix `dso` man page suggests using shared libraries, but `gcc-2.7.2` may not support C++ shared libraries under Irix. The workaround is to run `ptrimRpc` rather than `pigiRpc`.
- Installation of the `xv` man pages will fail. Irix does not have `nroff` and `tbl`.
- To build the CGC demos, you may need to exit `pigi`, set the environment variable `SGI_CC` to `-cckr`:
  ```
  setenv SGI_CC -cckr
  ```
  and restart `pigi`. See the sgi `cc` man page for more information.
- `sdf:image:motionCompensation` demo has some odd looking blocks in the McompOutput.1 image. Some of the images from the `irix5` run of this demo do not look like images in the sun4 version of this demo.

A.6.9 Linux specific bugs

We do not have access to a Linux system onsite at UC Berkeley, so our support of Linux is very limited. If you are having problems with prebuilt Linux binaries, you may want to try rebuilding Ptolemy from scratch. You may also want to check the ptolemy-hackers archives, located at the bottom of the Ptolemy home page at `http://ptolemy.eecs.berkeley.edu`.

- Linux uses Gnu `ar`, which seems to have problems if it is run on two files whose names are not unique in the first 13 characters. We have renamed a number of stars to workaround this problem. The `make checkjunk` command will report the names of files that are not unique in the first 13 characters. You should only have problems with this if you are creating stars of your own and placing them in libraries.
- If you are running the Linux version on a standalone machine, then please refer to section “Mr. Ptolemy window does not come up” on page A-16.

A.6.10 Sun Solaris 2.x specific bugs

- If you are building gcc under Solaris, you must have `/bin` in your path before `/usr/ucb` see `$PTOLEMY/src/gnu/README`.
- If, while linking, you get a message like
  ```
  unable to locate archive symbol table: Format error: archive fmag
  ```
  then you have probably run out of swap space. See the Solaris `swap` command for how to add more swap with `mkfile`.
- The Gnu make binary built for Solaris2.5 will not work with Solaris2.4. If you try to run the Solaris2.5 Gnu make binary under Solaris2.4, you will get an error message like:
  ```
  ld.so.1: make: fatal: relocation error: symbol not found: setlinebuf: referenced in make
  ```

A.6.11 Sun OS4 specific bugs

- If you are using the prebuilt Gnu binaries, then you may have problems if your Ptolemy distribution is not at `/users/ptolemy`, and you try setting the Gnu environment variables with the `$PTOLEMY/bin/g++-setup` script. We are working towards a solution, but you might find the following variables will work for you:
  ```
  unsetenv GCC_EXEC_PREFIX
  unsetenv C_INCLUDE_PATH
  unsetenv CPLUS_INCLUDE_PATH
  setenv LIBRARY_PATH $PTOLEMY/gnu/$PTARCH/lib
  setenv COMPILER_PATH $PTOLEMY/gnu/$PTARCH/lib/gcc-lib/$PTARCH/
  2.7.2.2
  setenv GCC_INCLUDE_DIR $PTOLEMY/gnu/$PTARCH/lib/gcc-lib
  ```
  We are working on a solution for this.
- Under SunOS4.x, we use the older BSD `ld` style incremental linking because of problems with g++ shared libraries. This means that the SunOS4.x binaries must be linked statically instead of dynamically, so the binaries are much larger than on platforms that use `dlopen()` style incremental linking. It also means that only the symbols that are used by stars present at link time are actually present in the binary. This can be a problem if your incrementally linked star uses a symbol from `libg++.a` that is not used by a star present at link time. See Appendix D, “Shared Libraries” for more information.

A.6.12 DEC Alpha specific bugs

- The DEC Alpha port is the first true 64 bit port of Ptolemy, so there are bound to be 64 bit vs. 32 bit bugs
- The CGC fixed-point demos give incorrect results. This is most likely due to the fact that the DEC Alpha is 64 bits.
- SDF/DDF Wormholes cause `SIGFPE` signals, which crash `pigi`. The DDF `ifThenElse` demo uses such SDF/DDF Wormhole.
- The full pigiRpc may fail to start with messages like:
  ```
  /sbin/loader: Fatal Error: lazy_text_resolve: symbol malloc
  should not have any relocation entry
  ```
  The workaround is to do:
  ```
  setenv LD_BIND_NOW yes
  ```
  See the DEC Unix `loader` man page for more information about `LD_BIND_NOW`. You may see similar error messages when you run the CGC fixed point demos.
- Vem produces lots of `Unaligned Access` messages. The `pigi` script does
  ```
  uac p noprint
  ```
  to turn them off .

A.6.13 Gnu compiler bugs

- If you write your own stars, then be aware that `gcc-2.7.2` may not choose the expected cast. In particular, `g++` has been known to cast everything to `Fix` if the explicit cast is omitted. The arithmetic is then performed using fixed-point computations. This will be dramatically slower than double or integer arithmetic, and may yield unexpected results. It is best to explicitly cast states to the desired form. For more information, see the ptlang chapter in the Ptolemy Programmers Manual.

- If you already have Gnu make installed, and the binary is called `gmake`, then the installation of the Ptolemy Gnu tools may fail. The workaround is to create a link in your path from `make` to `gmake`.

A.7 Additional resources

The best, most complete source for information on Ptolemy is to be found in the Ptolemy manual, *The Almagest*. The manual is included in every distribution and is available in our WWW and FTP sites.

A second source is the `ptolemy-hackers@ptolemy.eecs.berkeley.edu` mailing list. This list provides a forum for Ptolemy questions and issues. Users of the current release who have a Ptolemy question, comment, or think they've found a bug should send mail to `ptolemy-hackers`. Archives of the mailing list are also available. See the "Ptolemy mailing lists and the Ptolemy newsgroup" on page A-2.

A third source is the Ptolemy Usenet news group `comp.soft-sys.ptolemy`.

A fourth source on the latest information about Ptolemy is the Ptolemy World Wide Web (WWW) server accessible by programs such as Netscape. The Universal Resource Locator (URL) for the Ptolemy WWW server is `http://ptolemy.eecs.berkeley.edu`. The WWW pages contain on-line access to a quick tour of Ptolemy, a list of publications, a collection of technical papers, information on the members of the development team, and links to WWW information on related tools. The WWW interface is a superset of our FTP server discussed below.

Another source is the Ptolemy FTP site. To access this, use anonymous FTP to `ptolemy.eecs.berkeley.edu.` The directory `pub/ptolemy/papers` contains some of the latest Ptolemy papers and articles.

A.8 Submitting a bug report

Ptolemy is offered without support, but we are nonetheless interested in hearing your feedback with regard to bugs. A good bug report consists of the following elements:

- What OS you are running under (e.g. SunOS4.1.3, Solaris2.5.1, HPUX-10.01).
- What version of Ptolemy you are running. If you are not running the latest version, you may want to upgrade and see if your bug has been fixed already.
- Whether you are using prebuilt binaries, or if you compiled your binaries yourself.
- If the problem is with the compiler, and you are using the Gnu compiler, you should also state whether you are using the prebuilt compiler we provide, or your own version. You should also try running `gcc -v` to see what version of the compiler you are running.
- It is best for us if you can reproduce the bug in a small test case and send us your `~/pigiLog.pt` file, along with a detailed description of what you did and what hap-

pened.

- The best place to mail a bug report is to send it to the Ptolemy-hackers mailing list at `ptolemy-hackers@ptolemy.eecs.berkeley.edu`.
- If the bug consists of a problem with facets, you might want to uuencoded these facets and include them in your mail message. Note that there is a 40K size limit to mail to Ptolemy-hackers, so if your mail message is large, then you may want to send it only to `ptolemy@ptolemy.eecs.berkeley.edu`. To uuencode a facet and a galaxy:

```
tar -cf /tmp/facets.tar yourfacet yourgalaxy
uuencode /tmp/facets.tar facets.tar > facets.uu
```

- Then include the `facets.uu` file in a mail message. You may get some leverage out of compressing the `facets.tar` file before uuencoding it.