Eighth Biennial Ptolemy Miniconference
Tutorial
UC Berkeley, April 15, 2009

For wireless access, get a username and password from Christopher. Then use your wireless network chooser to find the **airbears** network, then point your web browser at any web page, such as **http://ptolemy.org**. You will be prompted for a username and password.

10:00 - 12:00	Setup
12:00 - 1:00	Lunch (Provided)
1:00 - 2:30	Writing Actors (Isaac Liu and Ben Lickly)
2:30 - 2:40	Break
2:45 - 3:30	Directors (Stavros Trypakis)
3:30 - 4:15	Codegen Hello World with legacy C - EmbeddedCActor (Man-Kit (Jackie) Leung)
4:15 - 5:00	Model Transformation (Thomas Feng)

Setting up Ptolemy II and Eclipse

These instructions assume you are using Eclipse Ganymede SR2 (based on 3.4.1) under Windows or Mac OS X. Other ways of setting up and building Ptolemy II are described on the Ptolemy II install page.

Contents of this page:

Other local pages:

Install Eclipse

1. Download the latest version of Eclipse from http://www.eclipse.org.
   In April, 2009, we chose **Eclipse for RCP/Plug-in Developers**, which is 175Mb. The **Eclipse for RCP/Plug-in Developers** version includes the plug-in development environment (PDE), which is needed by the backtrack facility, which is an optional part of Ptolemy II that allows models to restore their old state. If the version of Eclipse that you install does not have the PDE, then there will be build errors, which can be fixed by excluding `ptolemy/backtrack/` from the build.
2. (*Mac OS X*): Untar the download file in `/Applications`, which will create `/Applications/eclipse/Eclipse.app`.
   (*Windows*): Unzip the download file into an appropriate place, such as `C:\Program Files`, which will create `C:\Program Files\eclipse\eclipse.exe`.
   If `eclipse\eclipse.exe` is not created, then it could be that the security policy on your machine is preventing the creation of `.exe` files. If this is the case, then try running `unzip` from the command line.
3. Finish the installation by running `/Applications/eclipse/Eclipse.app` or `eclipse/eclipse.exe`.
   The first time this is run it will complete the installation process.
   (*Windows*): If Eclipse fails to start with the message "Windows cannot access the specified device, path or file. You may not have the appropriate permisions to access the item", then it may be necessary to make `eclipse.exe` and a dll executable:

   ```
   chmod a+x eclipse.exe
   ```

If you get the message "The Eclipse executable launcher was unable to locate its companion shared library", then run the following command in the `eclipse` directory.

```
find . -name *.dll -exec chmod a+x {} \;
```

After the first run, normal start-up will occur whenever it is started.

Eclipse Preferences for Ptolemy II

The default configuration of Eclipse has some difficulties with Ptolemy II, so a few changes are necessary.

1. (*Mac OS X*): There is usuall no need to set the memory size for Eclipse under Mac OS X. By default, it is set to 512 megabytes. For details about setting the memory size, see: Eclipse Workbench User Guide/Tasks/Running Eclipse.
2. (*Windows*): There is usually no need to set the memory size for Eclipse. By default, it is set to 512 megabytes in the `eclipse.ini file.`

   In Windows, create a shortcut to `eclipse.exe` by going to the directory where Eclipse is installed, right clicking on eclipse.exe and selecting `Create Shortcut`.

   To add Eclipse to the start menu, right click on the shortcut and select `Pin to Start Menu`

3. Eclipse requires some customization to build Ptolemy II and to keep the Ptolemy II coding style. In the steps below, we outline changes to be made in the Eclipse Preferences window. Under *Windows* the Eclipse Preferences window is invoked via `Window` | `Preferences`. Under *Mac OS X* the Eclipse Preferences window is invoked via `Eclipse` | `Preferences`. For each of the changes, hit `Apply`. When all the changes are done, hit `OK`, which will close the Eclipse Preferences Window.
4. By default, Eclipse rebuilds a project when any change is made to a file. This will result in thousands of errors when Ptolemy II is first checked out, and during normal usage, can be annoying because of the pauses it creates. We suggest disabling this feature as follows:
   1. While still in the preferences window
      (*Mac OS X*: `Eclipse` | `Preferences`)
      (*Windows*: `Window` | `Preferences`)
   2. Select `General` | `Workspace`
   3. Deselect `Build automatically`.
   4. Click Apply.

5. Ptolemy II source files are worked on by many people with different editors. Unfortunately, different text editors interpret tab characters differently, so it is best to use spaces rather than tabs.
   Sadly, the Eclipse developers have chosen to use tabs as spaces, so you must adjust the Eclipse Java Formatter if you plan on contributing code to the Ptolemy II tree.
   Also, it is best if files end with new line characters, so that we can run line oriented scripts on them.
   In addition, we currently do not want the Eclipse Formatter to format comments. We hope to change this policy in the future.

We handle these changes together:

1. While still in the preferences window (`Window` | `Preferences`), expand `Java` | `Code Style` | `Formatter`
2. Under "Active profile", Click on "New..."
3. In the "New Profile" window, enter "Ptolemy II" into the "Profile Name" entry.
4. Under "Initialize settings with the following profile", select "Java Conventions [built-in]" (We use Java Conventions over the "Eclipse" setting because the Eclipse style uses tabs.)
5. Click on "OK" to close the "New Profile" window
6. A "Profile 'Ptolemy II'" window will appear. Under the "Indentation" tab, change the Tab policy to "Spaces only".
7. Under the "New Lines" tab, select "at end of file"
8. Under the "Comments" tab,
   unselect "Enable Javadoc comment formatting"
   unselect "Enable block comment formatting"
   unselect "Enable line comment formatting"
9. Click OK.
10. In the Preferences Window, click Apply.

6. Ptolemy II uses some features of Java 5.0. In particular, `ptolemy/actor/ptalon` uses generics, which require Java 5.0 or later.
   Set Eclipse to use Java 5.0 or later source code compliance.
   1. While still in the preferences window (`Window` | `Preferences`), expand `Java` | `Compiler`
   2. Make sure that Set "Compiler compliance level" is "1.5", "5.0" or later. *Note that Java 1.5 is the same as Java 5.0*. What we don't want is "1.4" or "4.0".
   3. Click Apply.

   Note the following restrictions about versions of Java and Eclipse under Mac OS X:
   - Eclipse 3.4.0 will not start up for me under Java 1.6.0_05 on the Mac. This means that if "java -version" returns 1.6.0_05, then Eclipse failed to start for me.
   - Within Eclipse 3.4.0 which was started with Java 1.5.0_13, you can compile and run your project with either Java 1.5.0_13 or Java 1.6.0_13.
   - Eclipse 3.4.0 Java Projects that include Eclipse Plug-ins don't work under either Java 1.5.0_13 or Java 1.6.0_05 on the Mac

   For details, see http://chess.eecs.berkeley.edu/ptexternal/wiki/Main/Mac.
7. Eclipse has very good compiler error/warning. One of the warnings complains if a Serializable class does not have serialVersionUID declared. Since this warning is only useful if you are tightly managing serialization, we turn it off:
   1. While still in the preferences window (`Window` | `Preferences`), Expand `Java` | `Compiler` | `Errors/Warning`
   2. Under "Potential programming problems", change "Serializable class without serialVersionUID" to "Ignore"
   3. Under "Generic Types", change "Unchecked generic type operation" to "Ignore".
   4. Under "Generic Types", change "Usage of a raw type" to "Ignore".
   5. Click Apply. If you are prompted for a full rebuild, click Yes.
8. The PtDoc Doclet in `$PTII/doc/doclets/PtDoclet.java` requires `tools.jar`, which is only in the Java Development Kit (JDK), not the Java Runtime Environment (JRE). So, be sure that a JDK is selected for building, not a JRE.
   (*Mac OS X*): select `JVM 1.5.0 (Mac OS X Default)`.

While still in the preferences window
(*Windows*: `Window` | `Preferences`)
(*Mac OS X*: `Eclipse` | `Preferences`)
   1. Select `Java` | `Installed JREs`
   2. Verify that the checked line corresponds with a JDK, not a JRE.
9. Click OK in the Preferences window to apply all of the above changes.

Setting up Eclipse to manage your Ptolemy II development environment

Eclipse will manage your ptII code tree as a *project* called the ptII project. There several ways to set up the ptII project,

1. Eclipse is unaware of the version control aspects of the project.
   If you are downloading the source from major Ptolemy release, and don't want to update regularly, then use this choice
   (No Eclipse Subversion plug-in (Subversive), No Subversion (`svn`) from the command line)
2. Eclipse is unaware of the version control aspects of the project, use the shell version of Subversion to update the tree.
   (No Eclipse Subversion plug-in (Subversive), use Subversion (`svn`) from the command line)
3. Eclipse is aware of the Subversion aspects of the project. You can use both Eclipse and shell version of Subversion to update the tree.
   (Use both the Eclipse Subversion plug-in (Subversive) and the shell `svn` command)

Without Subversive, you'll need to manage the Subversion aspects in the usual way, i.e. with Subversion commands being submitted to a bash shell. With Subversive, Eclipse will do all sorts of things to "help" you. Most of these things are good, but there is a learning curve.

Please choose **one** of the methods below to set up Eclipse for Ptolemy II.

1. Eclipse is unaware of the version control aspects of the project. (No Subversion)

If you are downloading the source from major Ptolemy release, and don't want to update regularly

1. Download a ptII source tree from `http://ptolemy.eecs.berkeley.edu/ptolemyII/ptIIlatest`
2. Configure the .classpath and .project files. See `.classpath.default`. below.
3. Create the ptII project
   - In `File` | `New` | `Project`, select 'Java project'. In the 'Contents' box, click on 'Create project from existing source' and browse to the PTII directory. Finish.
   - When asked if you want to shift to the Java perspective, click on Yes.

OR

2. Eclipse is unaware of the version control aspects of the project, use the shell version of Subversion to update the tree.

1. Install Subversion:
   - Tortoise SVN - Windows GUI Client
     If you are not installing Subversive, then choose TortoiseSVN
   - Cygwin - Windows command line client

If you are more comfortable with a command line, then choose Cygwin. Make sure that you also install Cygwin's Subversion package.

- To check out the ptII svn repository with **read-only** access:

```
svn co svn://source.eecs.berkeley.edu/chess/ptII/trunk ptII
```

Note: Most users will select `svn://` because they will be accessing the tree with **read-only** Subversion access.
If you have an ssh account on `source.eecs.berkeley.edu`, then check out ptII with read/write access:

```
svn co svn+ssh://source.eecs.berkeley.edu/chess/ptII/trunk ptII
```

Or, if your user id on source.eecs is different than your current user id on your machine:

```
svn co svn+ssh://yourID@source.eecs.berkeley.edu/chess/ptII/trunk
```

- Configure the .classpath and .project files. See `.classpath.default`. below.
- Create the ptII project
  - In `File` | `New` | `Project`, select 'Java project'. In the 'Contents' box, click on 'Create project from existing source' and browse to the PTII directory. Finish.
  - When asked if you want to shift to the Java perspective, click on Yes.

OR

3. Setting up Eclipse for use with Subversion: Installing Subversive

There are at least two Subversion plugins for Eclipse, Subversive and Subclipse. Either will work, for a brief comparison, see Subclipse vs. Subversion. The instructions below are for Subversive. To use Subclipse, see Installing Subversion (SVN) into Eclipse.

The Subversive Eclipse plug-in adds Subversion to Eclipse. After installation, you can use either Eclipse to manage your project, or the `svn` from the command line.

Follow the Subversive installation instructions or these instructions below

1. To download Subversive, in Eclipse, follow these menus
   Mac OS X: `Help | Software Updates | Available Software`
   Windows: `Help | Software Updates | Available Software.`
2. Click on "Add Site" and enter

   ```
   http://download.eclipse.org/technology/subversive/0.7/update-site/
   ```

   Find

   ```
   Subversive SVN Team Provider (Incubation)
   ```

   and click on the box next to it.
3. Then click `Install` and click through the license agreement.

4. When you are prompted to restart, click `No`, we will restart later
5. It is **required** that a second package, the `Subversive SVN Connectors plug-in` be installed. To do this, go back to the "Software Updates and Add-ons" window and select `Available Software` | `Add Site` and enter

   `http://www.polarion.org/projects/subversive/download/eclipse/2.0/ganyı`

6. *Mac OS X*: There is no need to install JavaHL, in fact JavaHL will likely cause problems
   Select this package:

   ```
   Subversive SVN Connectors
   SVNKit 1.2.0 Implementation (Optional)
   ```

   Windows: **Note: select only one `JavaHL` Win32 binary**. If both JavaHL 1.4.5 and JavaHL 1.5.0 are installed, then Eclipse will have problems. The version of JavaHL is dependent on the version of command line `svn` that is installed on your machine.
   If `svn -version` returns 1.4, then install JavaHL 1.4.5.
   If `svn -version` returns 1.5, then install JavaHL 1.5.
   Select these packages:

   ```
   JavaHL 1.4.5 Win32 Binaries(Optional)
   Native JavaHL 1.4 Implementation (Optional)
   Subversive SVN Connectors
   ```

7. Click on `Install` and click through the rest of the installation.
8. When prompted to restart Eclipse, do so.

Setup the Subversion Configuration to Sane setting

Sadly, Subversion is by default configured with nonsense initial default values for commit times and end of line settings.

Tell Subversion to use commit times

In the default settings, if you check out files with subversion, then they will have their modification times set to the current time. This is incorrect:

1. The ptII tree includes derived files that are generated by Autoconf, JavaCC and Antlr. If the commit times are not preserved, then the source file might have a mod time later than the derived file which means that either the tool must be run or else the build system must touch the derived file.
2. If the commit times are preserved, the `ls -ltr` will quickly show which files were changed when

To change the Subversion configuration, hunt for the `.subversion/config` file.

Under Windows with Cygwin, you might find it as `$HOME/.subversion/config`. Another place to look is `c:/Documents and Settings/`*`yourlogin`*`/Application Data/Subversion /config.`

On the Mac, try `/Users/`*`yourlogin`*`/.subversion`.

If the directory does not exist, try running `svn --version` to create the directory.

Make this change to `.subversion/config`

```
### Set use-commit-times to make checkout/update/switch/revert
### put last-committed timestamps on every file touched.
use-commit-times = yes
```

Tell Subversion to automatically set the eol style

With the default settings, if you use Subversion to add a makefile, then the file will be checked in with binary settings. When the file is checked on out non-Windows platforms, then the file will have `\r\n` characters, which will break Solaris `/usr/ccs/bin/make` and cause other problems.

In addition, we would like keyword strings like `$Id: eclipse.htm 53064 2009-04-12 00:50:32Z cxh $` to be automatically expanded and updated.

Find the `.subversion/config` file and uncomment the `enable-auto-props` line so that it looks like:

```
enable-auto-props = yes
```

Also, uncomment the `[auto-props]` and the section below that sets the properties for the files. Also, set props for at least `makefile`, `*.tcl` and `*.mk` files. See $PTII/doc/coding/svn-config-auto-props.txt for a more complete set of properties. The bottom of the `.subversion/config` file should look like:

```
###
### Settings for your ~/.subversion/config file.  To use this:
### 1) Edit your ~/.subversion/config file
### 2) Insert the contents of this file into your
###    ~/.subversion/config file
### 3) You must set "enable-auto-props" to yes
###    BEFORE the [auto-props] line.
###    _BE_SURE_ to check that [auto-props] does NOT appear twice
###    If you append the contents of this file at the end
###    Of your config file, you will need to comment out
###    the preexisting [auto-props] line.

### Set enable-auto-props to 'yes' to enable automatic properties
### for 'svn add' and 'svn import', it defaults to 'no'.
### Automatic properties are defined in the section 'auto-props'.
### For this to work, the "enable-auto-props = yes" line should
### be before the "[auto-props]" line.
enable-auto-props        = yes

[auto-props]
### Section for configuring automatic properties.
### The format of the entries is:
###   file-name-pattern = propname[=value][;propname[=value]...]
### The file-name-pattern can contain wildcards (such as '*' and
### '?').  All entries which match will be applied to the file.
```

```
### Note that auto-props functionality must be enabled, which
### is typically done by setting the 'enable-auto-props' option.
[auto-props]
*.MF = svn:eol-style=native;svn:keywords=Author Date Id Revision
*.aart = svn:eol-style=native;svn:keywords=Author Date Id Revision
*.ac = svn:eol-style=native;svn:keywords=Author Date Id Revision
*.am = svn:eol-style=native;svn:keywords=Author Date Id Revision
*.apt = svn:eol-style=native;svn:keywords=Author Date Id Revision
*.avi = svn:mime-type=video/avi
*.bat = svn:eol-style=native;svn:keywords=Author Date Id Revision
*.bsh = svn:eol-style=native;svn:keywords=Author Date Id Revision
*.c = svn:eol-style=native;svn:keywords=Author Date Id Revision
*.cat = svn:eol-style=native;svn:keywords=Author Date Id Revision
*.cgi = svn:eol-style=native;svn:keywords=Author Date Id Revision
*.classpath = svn:eol-style=native;svn:keywords=Author Date Id Revision
*.cmd = svn:eol-style=native;svn:keywords=Author Date Id Revision
*.cnd = svn:eol-style=native;svn:keywords=Author Date Id Revision
*.conf = svn:eol-style=native;svn:keywords=Author Date Id Revision
*.config = svn:eol-style=native;svn:keywords=Author Date Id Revision
*.cpp = svn:eol-style=native;svn:keywords=Author Date Id Revision
*.css = svn:eol-style=native;svn:keywords=Author Date Id Revision
*.cwiki = svn:eol-style=native;svn:keywords=Author Date Id Revision
*.data = svn:eol-style=native;svn:keywords=Author Date Id Revision
*.dcl = svn:eol-style=native;svn:keywords=Author Date Id Revision
*.doc = svn:mime-type=application/msword
*.dsp = svn:eol-style=CRLF
*.dsw = svn:eol-style=CRLF
*.dtd = svn:eol-style=native;svn:keywords=Author Date Id Revision
*.egrm = svn:eol-style=native;svn:keywords=Author Date Id Revision
*.ent = svn:eol-style=native;svn:keywords=Author Date Id Revision
*.fn = svn:eol-style=native;svn:keywords=Author Date Id Revision
*.ft = svn:eol-style=native;svn:keywords=Author Date Id Revision
*.fv = svn:eol-style=native;svn:keywords=Author Date Id Revision
*.g = svn:eol-style=native;svn:keywords=Author Date Id Revision
*.gif = svn:mime-type=image/gif
*.grm = svn:eol-style=native;svn:keywords=Author Date Id Revision
*.gz = svn:mime-type=application/x-gzip
*.h = svn:eol-style=native;svn:keywords=Author Date Id Revision
*.handlers = svn:eol-style=native;svn:keywords=Author Date Id Revision
*.htc = svn:eol-style=native;svn:keywords=Author Date Id Revision
*.html = svn:eol-style=native;svn:keywords=Author Date Id Revision
*.ico = svn:mime-type=image/x-icon
*.ihtml = svn:eol-style=native;svn:keywords=Author Date Id Revision
*.in = svn:eol-style=native;svn:keywords=Author Date Id Revision
*.java = svn:eol-style=native;svn:keywords=Author Date Id Revision
*.jmx = svn:eol-style=LF
*.jpg = svn:mime-type=image/jpeg
*.jpg= svn:mime-type=image/jpeg
*.js = svn:eol-style=native;svn:keywords=Author Date Id Revision
*.jsp = svn:eol-style=native;svn:keywords=Author Date Id Revision
*.junit = svn:eol-style=native;svn:keywords=Author Date Id Revision
*.jx = svn:eol-style=native;svn:keywords=Author Date Id Revision
*.m = svn:eol-style=native;svn:keywords=Author Date Id Revision
```

```
*.m4 = svn:eol-style=native;svn:keywords=Author Date Id Revision
*.manifest = svn:eol-style=native;svn:keywords=Author Date Id Revision
*.mdo = svn:eol-style=native;svn:keywords=Author Date Id Revision
*.meta = svn:eol-style=native;svn:keywords=Author Date Id Revision
*.mf = svn:eol-style=native;svn:keywords=Author Date Id Revision
*.mk = svn:eol-style=native;svn:keywords=Author Date Id Revision
*.mod = svn:eol-style=native;svn:keywords=Author Date Id Revision
*.mov = svn:mime-type=video/quicktime
*.mpg = svn:mime-type=video/mpeg
*.ms = svn:eol-style=native;svn:keywords=Author Date Id Revision
*.n3 = svn:eol-style=native;svn:keywords=Author Date Id Revision
*.nroff = svn:eol-style=native;svn:keywords=Author Date Id Revision
*.patch = svn:eol-style=native;svn:keywords=Author Date Id Revision
*.pdf = svn:mime-type=application/pdf
*.pen = svn:eol-style=native;svn:keywords=Author Date Id Revision
*.php = svn:eol-style=native;svn:keywords=Author Date Id Revision
*.pl = svn:eol-style=native;svn:keywords=Author Date Id Revision
*.pm = svn:eol-style=native;svn:keywords=Author Date Id Revision
*.png = svn:mime-type=image/png
*.pod = svn:eol-style=native;svn:keywords=Author Date Id Revision
*.pom = svn:eol-style=native;svn:keywords=Author Date Id Revision
*.ppt = svn:mime-type=application/powerpoint
*.project = svn:eol-style=native;svn:keywords=Author Date Id Revision
*.properties = svn:eol-style=native;svn:keywords=Author Date Id Revision
*.py = svn:eol-style=native;svn:keywords=Author Date Id Revision
*.rb = svn:eol-style=native;svn:keywords=Author Date Id Revision
*.rdf = svn:eol-style=native;svn:keywords=Author Date Id Revision
*.rnc = svn:eol-style=native;svn:keywords=Author Date Id Revision
*.rng = svn:eol-style=native;svn:keywords=Author Date Id Revision
*.rnx = svn:eol-style=native;svn:keywords=Author Date Id Revision
*.roles = svn:eol-style=native;svn:keywords=Author Date Id Revision
*.rss = svn:eol-style=native;svn:keywords=Author Date Id Revision
*.schemas = svn:eol-style=native;svn:keywords=Author Date Id Revision
*.sh = svn:eol-style=native;svn:keywords=Author Date Id Revision
*.sh = svn:eol-style=native;svn:executable
*.sql = svn:eol-style=native;svn:keywords=Author Date Id Revision
*.svg = svn:eol-style=native;svn:keywords=Author Date Id Revision
*.tcl = svn:eol-style=native;svn:keywords=Author Date Id Revision
*.tgz = svn:mime-type=application/x-compressed
*.tif = svn:mime-type=image/tiff
*.tiff = svn:mime-type=image/tiff
*.tld = svn:eol-style=native;svn:keywords=Author Date Id Revision
*.txt = svn:eol-style=native;svn:keywords=Author Date Id Revision
*.types = svn:eol-style=native;svn:keywords=Author Date Id Revision
*.vm = svn:eol-style=native;svn:keywords=Author Date Id Revision
*.vsl = svn:eol-style=native;svn:keywords=Author Date Id Revision
*.wsdd = svn:eol-style=native;svn:keywords=Author Date Id Revision
*.wsdl = svn:eol-style=native;svn:keywords=Author Date Id Revision
*.xargs = svn:eol-style=native;svn:keywords=Author Date Id Revision
*.xcat = svn:eol-style=native;svn:keywords=Author Date Id Revision
*.xconf = svn:eol-style=native;svn:keywords=Author Date Id Revision
*.xegrm = svn:eol-style=native;svn:keywords=Author Date Id Revision
*.xgrm = svn:eol-style=native;svn:keywords=Author Date Id Revision
```

```
*.xhtml = svn:eol-style=native;svn:keywords=Author Date Id Revision
*.xhtml2 = svn:eol-style=native;svn:keywords=Author Date Id Revision
*.xlex = svn:eol-style=native;svn:keywords=Author Date Id Revision
*.xlog = svn:eol-style=native;svn:keywords=Author Date Id Revision
*.xls = svn:mime-type=application/excel
*.xmap = svn:eol-style=native;svn:keywords=Author Date Id Revision
*.xml = svn:eol-style=native;svn:keywords=Author Date Id Revision
*.xroles = svn:eol-style=native;svn:keywords=Author Date Id Revision
*.xsamples = svn:eol-style=native;svn:keywords=Author Date Id Revision
*.xsd = svn:eol-style=native;svn:keywords=Author Date Id Revision
*.xsl = svn:eol-style=native;svn:keywords=Author Date Id Revision
*.xslt = svn:eol-style=native;svn:keywords=Author Date Id Revision
*.xsp = svn:eol-style=native;svn:keywords=Author Date Id Revision
*.xtest = svn:eol-style=native;svn:keywords=Author Date Id Revision
*.xul = svn:eol-style=native;svn:keywords=Author Date Id Revision
*.xweb = svn:eol-style=native;svn:keywords=Author Date Id Revision
*.xwelcome = svn:eol-style=native;svn:keywords=Author Date Id Revision
*.zip = svn:mime-type=application/zip
.htaccess = svn:eol-style=native;svn:keywords=Author Date Id Revision
INSTALL = svn:eol-style=native;svn:keywords=Author Date Id Revision
KEYS = svn:eol-style=native;svn:keywords=Author Date Id Revision
LICENSE = svn:eol-style=native;svn:keywords=Author Date Id Revision
Makefile = svn:eol-style=native;svn:keywords=Author Date Id Revision
NOTICE = svn:eol-style=native;svn:keywords=Author Date Id Revision
README = svn:eol-style=native;svn:keywords=Author Date Id Revision
abs-linkmap = svn:eol-style=native;svn:keywords=Author Date Id Revision
abs-menulinks = svn:eol-style=native;svn:keywords=Author Date Id Revision
makefile = svn:eol-style=native;svn:keywords=Author Date Id Revision
```

See Subversion Users: Re: Setting svn:eol-style and `http://maven.apache.org/developers/svn-eol-style.txt`

BTW - To set the style on already checked in file, run:

```
svn propset svn:eol-style native filename
```

To set the keywords on an alredy checked in file, run: svn propset svn:keywords "Author Date Id Revision" *filename* Then run `svn commit` to commit the change.

SSH Configuration.

If you will be accessing the ptII repository with **read/write** access, then you will need to do a little more configuration concerning passwords.

Note: Most users will **not** need to do this configuration because they will be accessing the tree with **read-only** Subversion access. Read-only users can skip to Check Out Ptolemy II from the Subversion repository

To access the ptII repository with read/write access, you will need configure Subversive to use your password. If don't do this step and you leave the default settings, and have Cygwin ssh installed, then `c:/cygwin/bin/ssh.exe` window will pop up each time ssh is called.

The fix is to set Subversive to use SVNKit `Window` | `Preferences` | `Team` | `SVN` | `SVN`

`interface | SVNKit (Pure Java) | OK`

Check Out Ptolemy II from the Subversion repository

Below we describe how to set up Eclipse so Eclipse manages the Subversion interaction.

Eclipse will manage your ptII code tree as a *project* called the ptII project. Assuming Eclipse is running:

1. Install Subversive by following the instructions above.
2. In Eclipse, add the ptII SVN Repository: `Window` | `Show View` | `Other` | `SVN` | `SVN Repositories`
3. In the SVN Repository view, right click, select `New` and set the url.
   To check out the ptII svn repository with **read-only** access:

   `https://source.eecs.berkeley.edu/svn/chess/ptII`

   or

   `svn://source.eecs.berkeley.edu/chess/ptII`

   Note: Most users will select `https://` or `svn://` because they will be accessing the tree with **read-only** Subversion access.
   If you have an ssh account on `source.eecs.berkeley.edu`, then check out ptII with read/write access:

   `svn+ssh://source.eecs.berkeley.edu/home/svn/chess/ptII`

   Or, if your user id on source.eecs is different than your current user id on your machine:

   `svn co svn+ssh://`*`yourID`*`@source.eecs.berkeley.edu/chess/ptII`

   Read/write users will be prompted for their source.eecs.berkeley.edu username and password. This typically happens more than once since in case the svn+ssh protocol is being used passwords are not cached.
4. If you get a "Certificate Problem" window, then select `Trust` or `Trust Always`. For details and a workaround, see Certificate is not issued by a trusted authority.
5. In the SVN Repository view, right click on `trunk` and select `Find/Check out As`

Be sure to check out the **`trunk`** not the entire tree, which include all the branches and the trunk.

   Follow the wizard, creating selections as follows:
   - In the "Check Out As" window, use the default, `Check out as a project configured using the New Project Wizard`, then hit `Finish`.
   - In the "Select a wizard" window, select `Java` | `Java Project`, then hit `Next`.
   - In the "Create a Java Project" window,
     set "Project Name" to: `ptII`
     set "Project Layout" to `Use project folder as root for sources and class files`. This is done so that `$PTII/bin` will contain the Ptolemy II command

line scripts instead of the `*.class` files created by Eclipse.
    - Click Finish.
    - If you see a Confirm Overwrite Window that says:

```
The project you created, ptII, contains resources other
than the standard .project file. If any of those
resources have the same name and relative location of a
resource you are checking out, the local resource will
be replaced with the resource from the repository
```

      then you probably did not change "Project Layout" as per above. Click cancel and recreate the Java Project.
    - In the "Open Associated Perspective?" window, click "Yes".

6. Wait for the checkout to finish. *Go get coffee.*
   Note: If you did not disable build automatically, as suggested above, then Eclipse will build the project and produce **thousands of errors**. This is not really a problem. You will fix the errors below with the `.classpath.default` file.

7. You will want to view the project in the Java perspective. If you aren't already in the Java perspective, select `Window` | `Open Perspective` | `Java` in the menu. By default, Eclipse offers a "Package Explorer" as the main navigation mechanism for the source files. Many developers much prefer to use the "Navigator." To get the Navigator, select `Window` | `Show View` | `Navigator` in the menu. You can now browse the source files.

8. Find the file in the ptII home directory called `.classpath.default` and copy it into a new file called `.classpath`.
   Note that you can use the Eclipse Navigator browse to the `.classpath.default` file, open the file, copy the contents, open `.classpath`, paste the contents and save `.classpath`.
   Or, you can use Windows Explorer or the shell.
9. In Eclipse, perform a Refresh on the ptII project by `Window` | `Open Perspective` | `Java`
   Go to the Package Explorer or the Navigator.
   Right mouse click on the ptII icon and select Refresh.
   Note: If you do not do copy `.classpath.default` and select "Refresh", then Eclipse will produce **thousands of errors**. This is not really a problem. It reflects that you probably don't have installed many optional packages that are needed to build subsets of Ptolemy II. You can ignore the errors (not advised, since they will mask errors you make), or you can copy the `.classpath.default` file and rebuild. Alternatively, you can install the optional packages and run configure.

10. Build Ptolemy II by selecting `Project` | Build All. (*If Build All is greyed out, then perhaps "Build Automatically" was not disabled and Ptolemy has been built. See above for more about Build Automatically*.) You will get many warnings that you can ignore, but hopefully no errors.

Running Ptolemy II

1. In the Run menu, select "Run Configurations...".

2. In the resulting dialog, select "Java Application" and click "New".

3. In the dialog, fill in the boxes as follows:
   - Name: `Vergil`
   - Project: `ptII`
   - Main class: `ptolemy.vergil.VergilApplication`

4. Press the Run button.

The Ptolemy II welcome window should appear.

You may now wish to read the Using Vergil tutorial.

Simple Debugging Session

1. Locate ptolemy/vergil/VergilApplication in the Explorer and double click. Place a breakpoint on the first line of main() by using `Run` | `Add/Remove Breakpoint`
2. Tell Eclipse which class to run with `Run` | `Run`. On the Main tab, select the Ptolemy II package and enter ptolemy.vergil.VergilApplication as Main class.
3. Press the Run button
4. To debug, quit Vergil, and place a breakpoint in, say, the fire() method of ptolemy.domains.ct.kernel.CTBaseIntegrator. Then `Run` | `Debug`, and as above. Open the Lorenz CT demo from the Quick Tour and run it.

Optional Extensions

Ptolemy II includes a number of packages that rely on software that you may or may not have installed, such as MATLAB, the Java comm package (for serial port connections), joystick support, Java Advanced Imaging (JAI), the Java Media Framework (JMF), and Java 3D. If you wish to use or extend these features, you will need to perform a few extra steps. These steps require execution of a script called "configure" in the Ptolemy II home directory, which in turn requires (on Windows) installation of Cygwin, a package that offers Unix-like facilities within Windows. You can find instructions for installing Cygwin at http://ptolemy.eecs.berkeley.edu/ptolemyII/ptIIlatest/cygwin.htm.

The procedure below will modify the .classpath file that is provided in the version control repository to customize it for the software that you have installed. The procedure is as follows:

1. Start up Cygwin bash shell.
2. Set the PTII variable and export it
   Windows:

   ```
   export PTII=c:/Documents\ and\ Settings/yourLogin/workspace/ptII
   ```

   Mac:

   ```
   export PTII=c:/Users/yourLogin/workspace/ptII
   ```

3. cd to the PTII directory

```
cd "$PTII"
```

Note that we use double quotes around `PTII` because the value of `PTII` has spaces in it.

4. Run configure:

```
./configure
```

This will create `$PTII/.classpath`

5. If you plan on running the Ptolemy II startup scripts in `$PTII/bin` on the command line, you will probably want to run the following commands within Cygwin bash:

```
cd $PTII/bin
make
```

6. In Eclipse, perform a Refresh on the ptII project by `Window`Open Perspective | `Java`
   Go to the Package Explorer or the Navigator.
   Right mouse click on the ptII icon and select Refresh.
   This will cause Eclipse to see the new `ptII/.classpath` and to build the project (or you may have to manually rebuild if you turned off automatic rebuild).

Synchronizing with the repository

1. In the Navigator view scroll/open until you see the resource (directory or file) to be synchronized.
2. Right click that resource and select Team->Synchronize With Repository...
3. The Synchronize view should then be visible.
4. Select the Incoming, Outgoing, or Incoming/Outgoing menu bar icon as appropriate

Restoring a file from the repository

1. In the Navigator view scroll/open until you see the resource (directory or file) to be restored.
2. Right click that resource and select Replace With->Latest From Head.

Copying Files

Usually, it is better to extend Java classes than to copy them, but sometimes copying files under Subversion is useful. To copy files, use the `svn cp` command. Do not drag and drop directories as the directories contain a `.svn` directory that will have settings for the source location. Instead, use `svn cp`. See also `svn mv`.

Debugging Tcl Tests

1. Go to Windows -> Open Perspective -> Java.
2. In Package Explorer, locate ptjacl.jar -> tcl.lang -> Shell.class -> Shell and highlight it.

> For some reason Eclipse sometimes hides certain jar files. In this case you have to add new a Debug Configuration (in the menu *Run -> Debug Configurations...*) in Eclipse similar as the decription in section Running Ptolemy II, then choose the ptII project as Project and use *tcl.lang.Shell* as Main class. The remaining steps are described below.

3. Go to Run -> Debug...
4. Select Arguments tag.
5. In Program arguments, put alljtests.tcl or any individual test tcl file.
   (E.g. SimpleDelay.tcl)
6. In VM arguments, put `-Dptolemy.ptII.dir=`*`your PtII directory`*
   (E.g. `-Dptolemy.ptII.dir=c:\hyzheng\ptII`).
   In case your directory path contains spaces, you need to use quotes. (E.g. `-Dptolemy.ptII.dir="c:\my workspace\ptII"`).
7. Select Local directory, browse to the directory containing the tcl tests.
   (E.g. `C:\hyzheng\ptII\ptolemy\domains\de\lib\test`)
8. Select Debug.

The nice thing of using Eclipse is that you can very easily locate where the exception is thrown by clicking the classes listed in the stack trace. You may further register a breakpoint to do more diagnosis.

Troubleshooting

Preferences

If you have already used Eclipse and you would like to start over with new projects and preferences, remove the `workspace` directory in the Eclipse directory. The `workspace` directory will only appear if you have already run Eclipse. **Note that removing the `workspace` directory will cause Eclipse to 'forget' about any projects that you may have set up**

Rebuilding Briefly flashes a window

If you have problems where clicking on build briefly flashes up a window, look in `$PTII/.classpath` for and empty exclusion that looks like | |

If you have problems with the classpath, look in the `workspace/.metadata/log` file that is in the directory where eclipse is installed. For more information about the `.metadata` directory, see below.

Eclipse takes a long time to start up

If Eclipse takes a long time to start up, then the problem could be a problem in your .metadata file.

Basically, when eclipse starts up, it might try to update `H:/workspace/.metadata` . The solution is covered in http://www.eclipse.org/documentation/html/plugins/org.eclipse.platform.doc.user/doc/tasks/running_eclipse.htm: The way I figured this out was by running Norton Antivirus and doing View -> File System Realtime Scan Statistics and then I noticed that my machine was updating H:/workspace/.metadata

I think I introduced the problem by clicking on the Eclipse.exe binary and selecting Pin to Start Menu. My solution was to remove the Eclipse bogus entry in the start menu and then create a shortcut, change Start in property and then pin that shortcut to my start menu.

Writing Actors

Ben Lickly
Isaac Liu

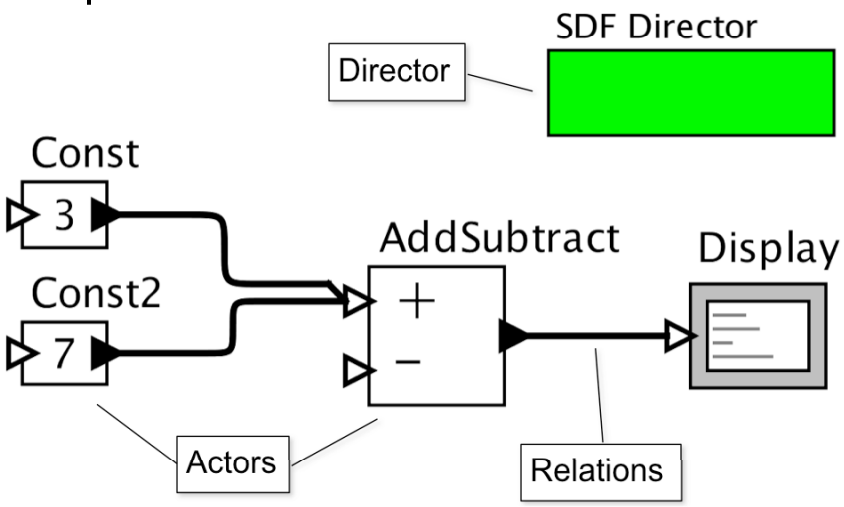
Ptolemy Ptutorial

Berkeley, CA
April 16, 2009

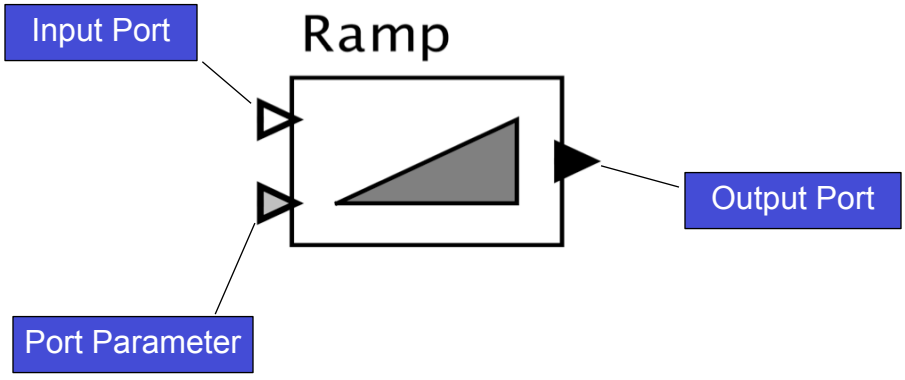
Basic Terminology

- A model is a a set of interconnected *actors* and one *director*
- Actor
  - Input & output *ports* connected by *relations*
  - Communicates using *tokens*
  - When it is *fired* it produces and consumes tokens
- Director
  - Implementation of semantics for component interaction

Simple Ptolemy II Model

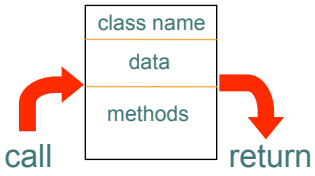


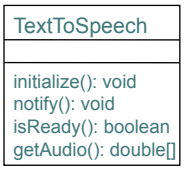
Anatomy of an Actor



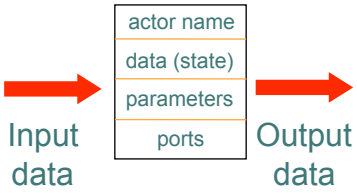
Object- vs. Actor-Oriented Design

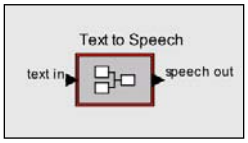
Object-orientation: Things happen to objects





Actor-orientation: Actors make things happen





Example by Edward Lee

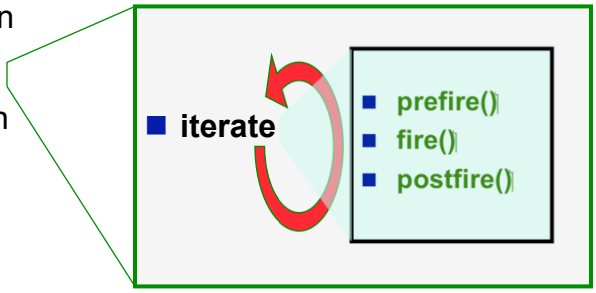
Abstract Semantics

- Flow of control
  - Initialization
  - Execution
  - Finalization

- **preinitialize()**
  - **declare port types**
  - **set scheduling information**
  - **change model structure**
- **initialize()**
  - **initialize local variables**

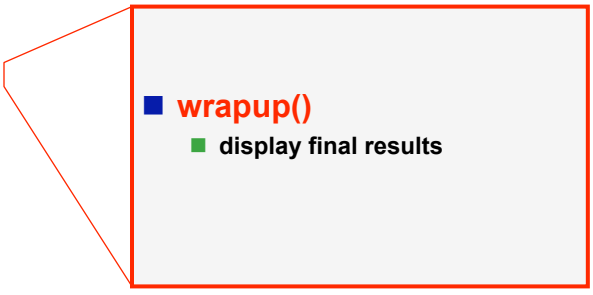
Abstract Semantics

- Flow of control
  - Initialization
  - Execution
  - Finalization



Abstract Semantics

- Flow of control
  - Initialization
  - Execution
  - Finalization

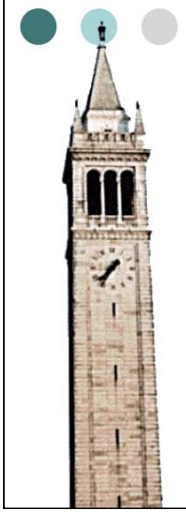


Writing an example actor

- Let's try creating an actor in Java
- Ptolemnizer actor:
  - Replaces “t” in input string with “pt”

Caveats

- In order to ensure domain-polymorphism, actors should not update state in their *fire* method.
- To be copied correctly, actors should implement the *clone* method.
  - Must include variable initializations and relative type constraints.



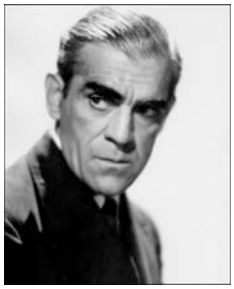
Directors

Stavros Tripakis

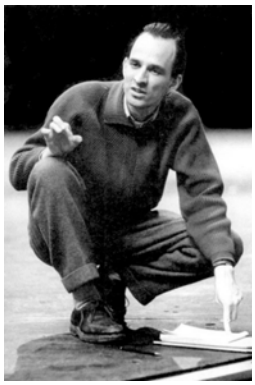
Researcher, UC Berkeley

Ptolemy Ptutorial, Apr 15, 2009

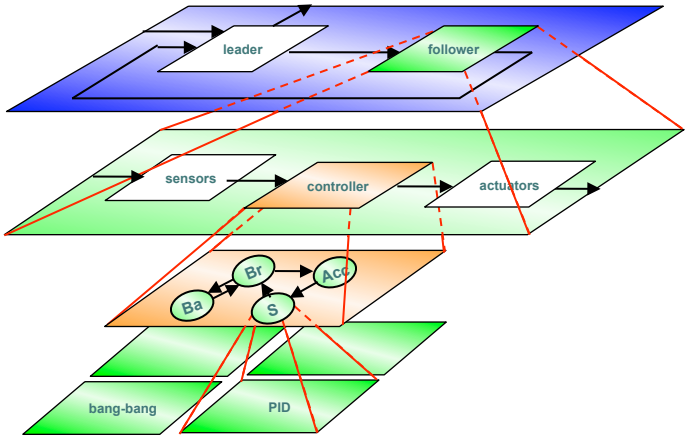
Actors



Directors

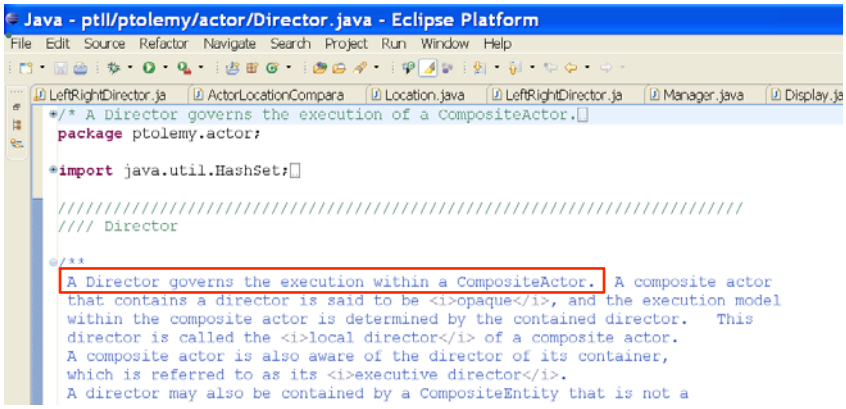


Hierarchy



Tripakis: 4

Directors in Ptolemy



Tripakis: 5

Example execution sequence

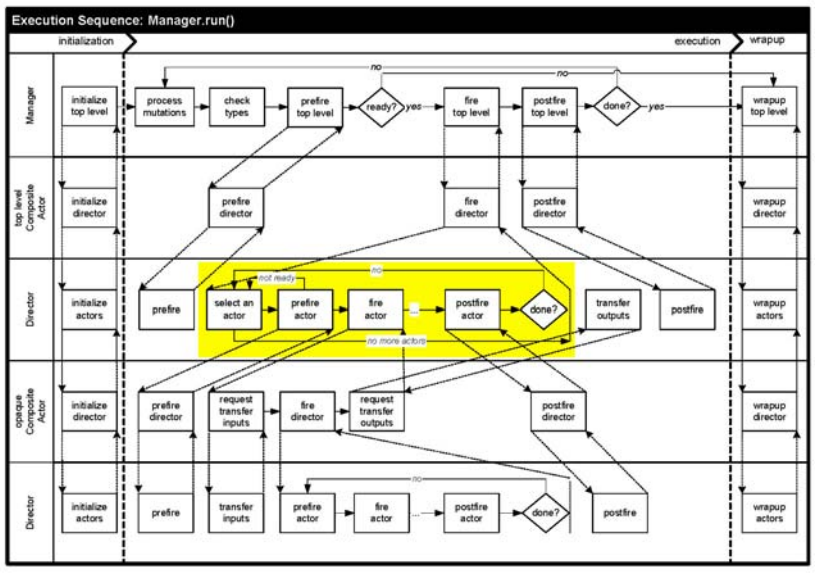
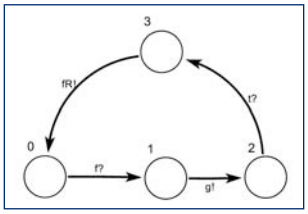


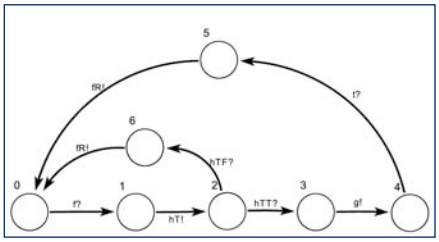
FIGURE 2.14. Example execution sequence implemented by run() method of the Director class.

Tripakis: 6

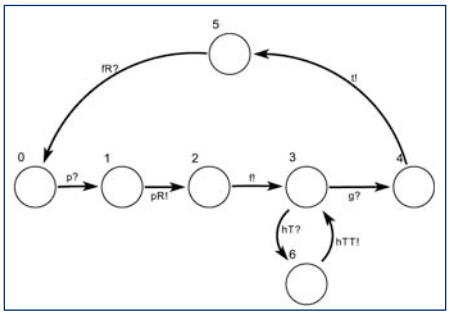
Actor and Director Protocols



SDF actor



Domain-polymorphic actor



SDF director

Tripakis: 7

Extension Exercise

Build a director that fires actors in left-to-right order, as they are laid out on the screen.

Tripakis: 8

The LeftRightDirector

```
public class LR extends Director {

    public Parameter iterations;

    public LR(CompositeEntity container, String name)
            throws IllegalActionException, NameDuplicationException {
        super(container, name);

        iterations = new Parameter(this, "iterations");
        iterations.setExpression("10");
        iterations.setTypeEquals(BaseType.INT);

        _itercnt = 0;
    }

    private int _itercnt;

    public void initialize() throws IllegalActionException {
        super.initialize();
        _itercnt = 0;

        Nameable container = getContainer();
        if (container instanceof CompositeActor) {
            java.util.List<Actor> myactors = ((CompositeEntity) container).entityList();

            ActorXCoordinateComparator comp = new ActorXCoordinateComparator();
            java.util.Collections.sort(myactors, comp);

            Iterator actorIter = myactors.iterator();
            while (actorIter.hasNext()) {
                Actor a = (Actor) actorIter.next();
                _debug("----------------------------------------");
                _debug(a.getDisplayName());
                _debug("----------------------------------------");

            }

        }
    }
...
```

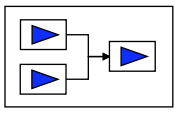
Tripakis: 9

The LeftRightDirector

```
...
    public boolean postfire() throws IllegalActionException {
        super.postfire();
        int iters = ((IntToken)iterations.getToken()).intValue();
        if (_itercnt < iters) {
            _itercnt++;
            return true;
        }
        return false;
    }

    private class ActorXCoordinateComparator implements Comparator<Actor> {

        public int compare(Actor o1, Actor o2) {
            double[] location1 = { Double.NEGATIVE_INFINITY,
                    Double.NEGATIVE_INFINITY };
            double[] location2 = { Double.NEGATIVE_INFINITY,
                    Double.NEGATIVE_INFINITY };

            List<?> locations = ((Entity) o1).attributeList(Locatable.class);
            if (locations.size() > 0) {
                location1 = ((Locatable) locations.get(0)).getLocation();
            }
            locations = ((Entity) o2).attributeList(Locatable.class);
            if (locations.size() > 0) {
                location2 = ((Locatable) locations.get(0)).getLocation();
            }

            if (location1[0] < location2[0]) {
                return -1;
            } else if (location1[0] > location2[0]) {
                return 1;
            } else {
                return 0;
            }
        }
    }
}
```

Tripakis: 10

Model-to-Code Transformation



```
Var v1, v2,
Var vv1, vv2,
Var v11, v12,
Function foo() {
…
…
}
Function bar () {
…
…
}
Program:…
```

Ptolemy Tutorial 2009

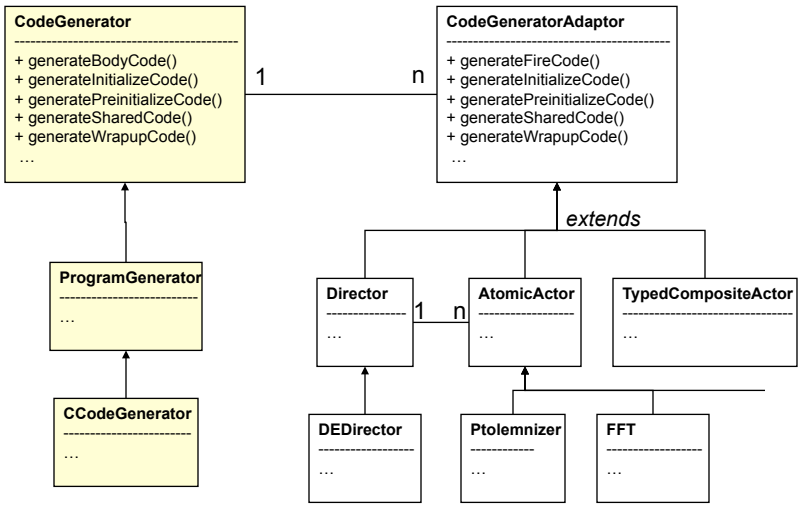
Man-Kit Leung
Berkeley, CA
April 15, 2009

Goal

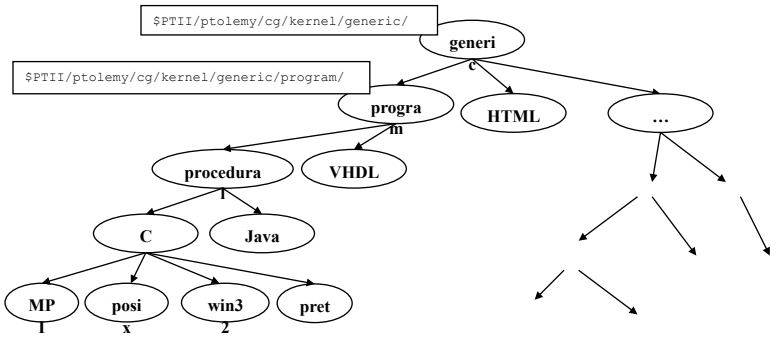
Transform Ptolemy models to any text or languages

How: *Learn how to customize the Ptolemy code generator.*

Some Basics before we start…



Target Hierarchy

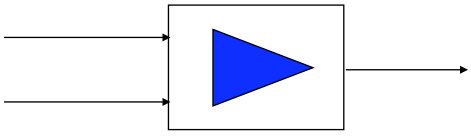


Sections of the Generated Content:

```
IncludingFiles
--------------------------------
SharedCode
--------------------------------
VariableDeclaration
--------------------------------
PreinitizeCode
--------------------------------
InitializeCode
--------------------------------
BodyCode
--------------------------------
WrapupCode
```


Non-trivial Components

If we need to generate complex code for an atomic component (e.g. FFT) that is highly parameterizable…



Template

We can use our template facility.

```
Var $v1;
Var $v2;
Var $v3;

Function $foo () {
  loop i = 1 to $bound:
    bar(i);
  end loop
}
```

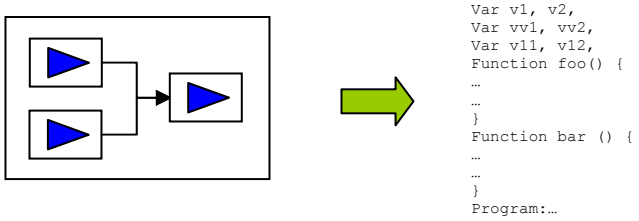
Static Text

Holes
(parameterized text)

Ptolemy Code Generation Tutorial 2009
Presenter: Man-Kit Leung (mankit@eecs.berkeley.edu)
4/15/2009

Model-to-Code Transformation

Customizing the Code Generator for Different Purposes



Goal#1: Transform Ptolemy models to *any* languages or text
Goal#2: Learn how to *customize* the Ptolemy code generator.

Example#1: Generating a simple HTML page that describes the model

---

Desired code:

```
<html>
    <head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
    <title> modelName </title>
    </head>

    <body>
    CompositeActorName contains
    <ul>
    <li> componentName1 </li>
    <li> componentName2 </li>
    <li> componentName3 </li>
    </ul>
    </body>
</html>
```

---

1. Create a new **CodeGenerator** subclass
    a. Create a new package/folder, named `html`, under `$PTII/ptolemy/cg/kernel/generic/`. For convenience, I already did this and created a template file (`HTMLCodeGenerator.java`) so we can fill in together.
    b. Override the `_generateBodyCode()` and `comment()` methods.
2. Create new **Adaptors** classes
    a. Under `$PTII/ptolemy/cg/adaptors/generic/html/adaptors/ptolemy/actor/lib/`, `Director.java`, `TypeCompositeActor.java`, and `AtomicActor.java` are the three adaptor files we need to fill in.
    b. Override the `_generateFireCode()` method for each class.
3. Generate code!!
    a. Start up Vergil (Main Class: `ptolemy.vergil.VergilApplication`)
    b. Open or Create a Ptolemy model (e.g. `File -> Open/New`).
    c. Under `Graph -> Instantiate Attribute`, type in the class of the new code generator (e.g. `ptolemy.cg.kernel.generic.html.HTMLCodeGenerator`).
    d. Double click on the new code generator (the blue box) to generate code.

Solution:

HTMLCodeGenerator.java:

```
    public String comment(String comment) {
        return "<!-- " + comment + " -->" + _eol;
    }

    public String generateCode(int section) throws IllegalActionException {
        StringBuffer code = new StringBuffer();
        code.append("<html>" + _eol +
                    "<head>" + _eol +
                    "<meta http-equiv=\"Content-Type\" content=\"text/html; charset=utf-8\" />" + _eol +
                    "<title>" + getContainer().getName() + "</title>" + _eol +
                    "</head>" + _eol + _eol +

                    "<body>" + _eol +
                    getAdapter(getContainer()).generateFireCode() +
                    "</body>" + _eol +
                    "</html>" + _eol);
        return code.toString();
    }
```

AtomicActor.java:

```
    public String generateCode(int section) throws IllegalActionException {
        return "<li>" + getComponent().getName() + "</li>" + _eol;
    }
```

TypedCompositeActor.java:

```
    public String generateCode(int section) throws IllegalActionException {
        StringBuffer code = new StringBuffer();

        Director directorAdapter = (Director) getCodeGenerator()
        .getAdapter(((ptolemy.actor.CompositeActor) getComponent()).getDirector());

        code.append(getComponent().getName() + " contains: " +
                    "<ul>" + _eol +
                    directorAdapter.generateFireCode() +
                    "</ul>" + _eol);

        return code.toString();
    }
```

Director.java:

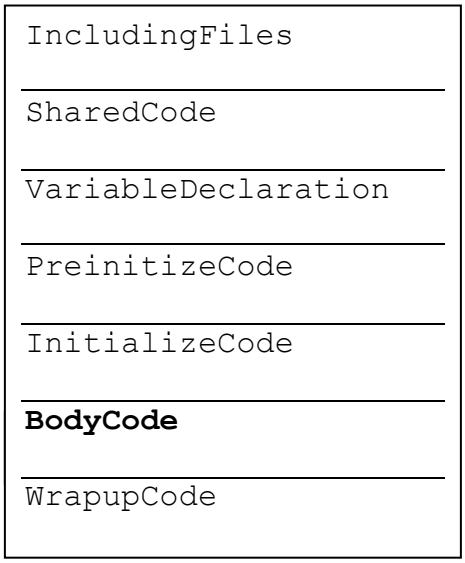
```
    public String generateCode(int section) throws IllegalActionException {
        StringBuffer code = new StringBuffer();
        code.append("<li>" + _director.getName() + "</li>" + _eol);

        Iterator actors = ((CompositeActor) _director.getContainer()).deepEntityList().iterator();
        while (actors.hasNext()) {
            NamedObj actor = (NamedObj) actors.next();
            CodeGeneratorAdapter adapter = getCodeGenerator().getAdapter(actor);
            code.append(adapter.generateFireCode());
        }
        return code.toString();
    }
```

General Procedure:

1. Update the ptII tree to get the most recent code and files (Eclipse users: `Team->Update`.)
2. Create a new package under both `$PTII/ptolemy/cg/kernel/ and $PTII/ptolemy/cg/adaptors/` hierarchies of targets (e.g. `generic -> program -> html`).
3. Extend the CodeGenerator and Adaptors (Director, TypedCompositeActor, and AtomicActor) classes, as we did above.
4. Override the necessary methods (e.g. `comment(), generateCode(int section), and etc.`).

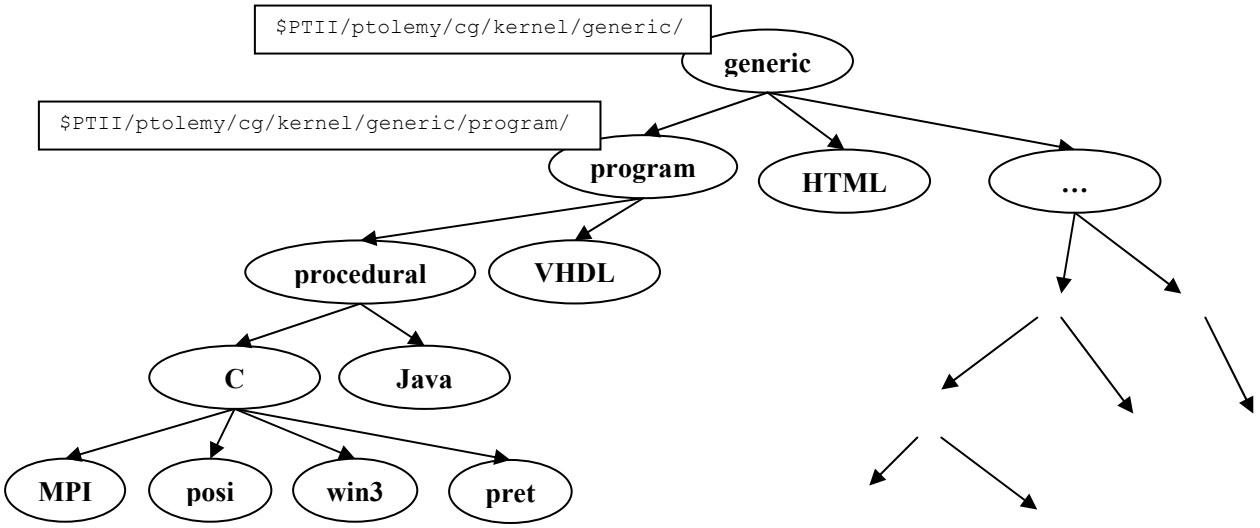
II. General Info:

A. Sections of the Generated Code:



You can generate code into different sections of the program by overriding the corresponding generate*() methods. These sections allow you to make logical separation within a program. By default, all the sections are empty. Our previous example overrides the _generateBodyCode() method, so our generated program has a non-empty BodyCode section.

B. Target Hierarchy:



The above diagram illustrates the overall picture of the target hierarchy within the code generation package. When adding a new target, you want to think about where to place this new target within this hierarchy. Often, it allows code and functionality reuse if this is done properly (it means there are less stuffs you need to override or implement on your own).

For further references:

A simple tutorial to write an adaptor with template
http://ptolemy.berkeley.edu/conferences/07/leung_ptolemy_tutorial_codegen.doc

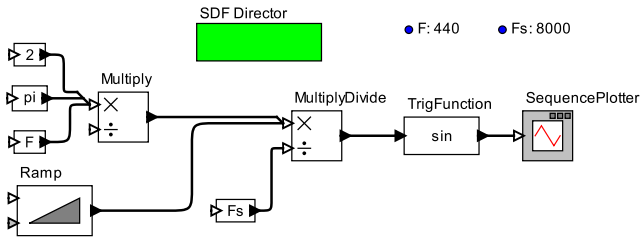
Detailed documentation of the template and the macro language.
http://chess.eecs.berkeley.edu/pubs/251.html

Generating code for different models of computation.
http://chess.eecs.berkeley.edu/pubs/401.html

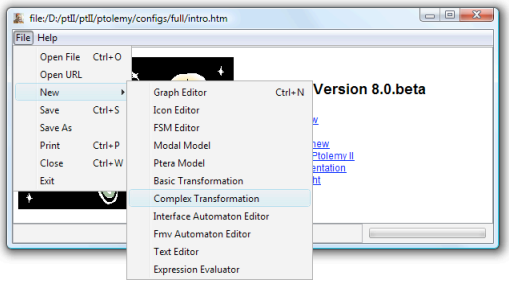
Tutorial: HelloWorld for Model-Based Transformation

Thomas Huining Feng
CHESS, EECS, UC Berkeley

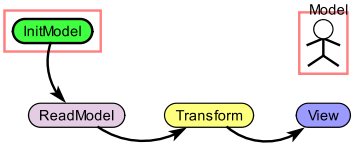
April 15, 2009



1. Create a complex transformation (File→New→Complex Transformation).



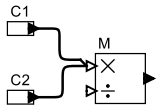
2. Create the following design.



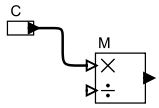
InitModel and the Model parameter are already created automatically. ReadModel is under IO in the library. Create edges by holding down the Ctrl key and dragging the mouse from the start to the end. Double-click ReadModel and set modelFile as:
`$PTII/ptolemy/actor/gt/demo/SinewaveOptimization/Sinewave.xml`

3. Run the workflow. A window pops up with the original model in it. Close the window.
4. Right-click Transform and select "Look Inside" to edit the transformation rule.
5. Add two AtomicActorMatcher's and one MultiplyDivide (in Actors/Math). Configure each AtomicActorMatcher (double-click):
   (a) Add a SubclassCriterion with superclass "ptolemy.actor.lib.Const".
   (b) Add a PortCriterion. Check the boxes to the left of output and multi, and that below output.

   Connect the three actors and rename them to C1, C2 and M as below.



6. Select everything, copy (Ctrl-C), change to the Replacement tab, and paste (Ctrl-V). Delete C2.



7. Double-click C and add an AttributeCriterion. The name is "value" and the value is "$(C1.value * C2.value)".
8. To try it out, select the second-to-last button on the toolbar.

   Enter the same file name as we entered for ReadModel:
   `$PTII/ptolemy/actor/gt/demo/SinewaveOptimization/Sinewave.xml`
9. In the popup window, a match to the pattern is highlighted. The last 5 buttons from left to right are:

   (a) Highlight previous match.
   (b) Highlight next match.
   (c) Replace the highlighted match.
   (d) Replace all matches (whether highlighted or not).
   (e) Close this window, and open the current model in an ordinary editor.

   Note: Although there seem to be only 3 matches to the pattern, the status bar shows 6. (Why?)
10. Close the transformation rule window and run the workflow again.