ArtiSynth Installation Guide

John Lloyd
July 9, 2014

Contents

1 Introduction

This document describes how to install and run ArtiSynth on your computer system. There are typically two classes of ArtiSynth user: *end users* who mainly want to run the program with existing models, and *developers* who want to create their own models and/or modify the ArtiSynth code base. End users should read through Section 5, while developers should read further.

2 Supported Systems

ArtiSynth is currently supported for the following systems:

- Windows (32 and 64 bit)
- Linux (32 and 64 bit)
- MacOS 10.6 (i.e., Snow Leopard), on 64 bit Intel platforms

3 External Software Requirements

The following external software packages are required to run ArtiSynth:

Java JDK 1.6

This goes by various other names, including Java SE 6 and Java SDK 1.6. If necessary, it can be obtained for Windows and Linux from java.sun.com/javase/downloads, and for MacOS from www.apple.com/java. The full Java development kit (JDK) is required, which comes with the Java compiler `javac`. The run time environment (JRE) will not be sufficient. There is no need for extra bundles such at JavaFX, NetBeans, or EE.

OpenGL and JOGL (should already be installed)

OpenGL is supplied by most operating systems, so there should be no need to install it. JOGL is an interface that allows OpenGL to be called from Java. The necessary JOGL libraries are included with the ArtiSynth distribution. The version of JOGL currently used by ArtiSynth is described in the file `$ARTISYNTH_HOME/lib/JOGL_VERSION`.

4 Downloading and Installing

ArtiSynth may be obtained either by downloading one of the packaged distributions, or by checking out the most recent development version. The latter is recommended for developers working closely with the ArtiSynth team and is described in Section 6.1.

To obtain one of the packaged distributions:

1. Go to http://www.artisynth.org
2. Select Software/Downloads. This will bring up a page directing you to the ArtiSynth Download Form.
3. Select the link to the ArtiSynth Download Form.
4. Fill in and submit this form. An email containing a link and a password will automatically be sent to the e-mail address you provide in this form.
5. Open the e-mail mentioned above and click on the given link to find a password protected webpage.
6. Enter the password provided by the email. Take care that no spaces are included, as the system is sensitive to them.

7. This will lead to a different webpage, with links to various ArtiSynth distributions. Select any one to download, and click it.
8. Download the distribution, and unzip it in an appropriate location on your computer.

In this document, the location of the ArtiSynth installation directory will be referred to by the variable `ARTISYNTH_HOME`. Expansions of this will be indicated by `$ARTISYNTH_HOME`, so that if the installation directory is `/home/robert/project/artisynth_2_` then `$ARTISYNTH_HOME/lib/Linux` will denote the directory

```
/home/robert/project/artisynth_2_0/lib/Linux
```

> **Important:**
> On Windows, make sure ArtiSynth is not installed in a location for which any of the folder names contain spaces (i.e., `Program Files`). This is very important for ArtiSynth to run correctly.

5 Running ArtiSynth

This section describes how to start ArtiSynth for running existing models and demos.

5.1 Windows

The most direct way to start ArtiSynth is to run the batch file `artisynth.bat`, located in `$ARTISYNTH_HOME\bin`. This can be done by double-clicking on it in a file browser.

You can also create a shortcut to this batch file (by right clicking on it and selecting Create Shortcut), and then placing this shortcut in either the START menu or on the Desktop. However, the batch file itself must remain in `$ARTISYNTH_HOME\bin`.

Alternatively, you can open a command interpreter (`CMD`), navigate to `$ARTISYNTH_HOME`, and enter the command

```
% bin\artisynth
```

> **Note:**
> If Java is installed in a non-conventional location, the system may not be able to find it and this will cause `artisynth.bat` to fail. To fix this, locate the folder in which `java.exe` resides, and let the location of this folder be called JAVA_DIR. You should then add JAVA_DIR to your system's `PATH` environment variable (see Section 9.1).

5.2 MacOS, Linux, and Windows running Cygwin

On MacOS, Linux, and Windows systems running Cygwin (see Section 8), you can run ArtiSynth by opening a terminal window, changing to the directory `ARTISYNTH_HOME`, and running the command

```
> bin/artisynth
```

If you set your `PATH` environment variable to include `$ARTISYNTH_HOME/bin` (see Section 6.3), you can then run ArtiSynth using the command

```
> artisynth
```

from any directory.

On MacOS, you can also run ArtiSynth from a file browser by double clicking on

```
$ARTISYNTH_HOME/bin/artisynth.command
```

(`artisynth.command` is just a copy of `artisynth`; the `.command` suffix makes it recognizable to the MacOS GUI as a command). MacOS GUI).

`artisynth` can be called with a number of options; to see these, run

```
> artisynth -help
```

> The `artisynth` command (and the `artisynth.bat` command used by Windows) are actually just convenience scripts that parse options, set environment variables and invoke Java using the ArtiSynth driver class `artisynth.core.driver.Launcher`. If you have independently set the environment variables described in Section 6.3, then you can invoke ArtiSynth from Java directly using
>
> ```
> > java artisynth.core.driver.Launcher
> ```
>
> although some of the options provided by `artisynth` would be unavailable.

6 Configuring for Development Work

Users intending to use ArtiSynth to develop models of their own, or to modify or add to the ArtiSynth code base, will need to write and compile Java code. This can either be done directly, using text editors and the Java compiler `javac` to create, modify and compile the necessary code, or by using an integrated development environment (IDE), such as Eclipse. Directions on setting up the Eclipse IDE are given in Section 7.

ArtiSynth uses and references a number of environment variables, as described in Section 6.3. It is recommended that developers set these in their general system environment. Although the `artisynth` command (`artisynth.bat` on Windows) will try to set these variables automatically if they are unset, setting them in the general environment gives the developer access to a broader range of commands in `$ARTISYNTH_HOME/bin`, and permits the use of various IDEs without having to set variables in each.

For development work on Windows, it is recommended to install the Unix emulator Cygwin in order to allow use of the script-based commands in `$ARTISYNTH_HOME/bin`, as well as the `Makefile` commands. Cygwin provides both the `bash` and `tcsh` shell environments, for which environment variables can be set as described in Section 6.4 or Section 6.5. See Section 8 for instructions on installation.

6.1 Obtaining the latest development version

If you are working closely with ArtiSynth or any of its associated projects, you are encouraged to use the latest development version, instead of the packaged releases. This will provide immediate access to updates and bug fixes. In addition, if you have repository write access, you will be able to check in code so that it can be kept consistent with modifications to the ArtiSynth API.

At present, the development version is contained in a CVS repository, and checking out a version requires obtaining a guest computer account in the UBC department of Electrical and Computer Engineering (ECE). This also provides repository write access. For information on obtaining such an account, please send email to `artisynth` at `ece.ubc.ca`.

When you have an ECE account, you can obtain the latest development version by using CVS. This will require that CVS is installed on your system. It should be installed by default on MacOS, and can be obtained for Windows as part of Cygwin (see Section 8). For Ubuntu or Debian style Linux systems, it can be acquired if needed by issuing the following command from a terminal window:

```
> sudo apt-get install cvs
```

The latest development version of ArtiSynth can then be obtained by issuing the following CVS command from a terminal window:

```
> cvs -d :ext:userName@ssh.ece.ubc.ca:/ubc/ece/home/hct/other/hct/cvsroot checkout -P artisynth_2_0
```

where `userName` is the name of your ECE account. This will check out the latest version into a directory called `artisynth_2_0` located in the directory from which the `cvs` command was issued; the `artisynth_2_0` directory will now become `ARTISYNTH_HOME`.

Once you have checked out the latest version, you will also need to download the `.jar` files from the ArtiSynth server so that you can compile the code. This can be done by executing the command `updateArtisynthLibs` in the directory `$ARTISYNTH_HOME/bin`:

```
> cd $ARTISYNTH_HOME/bin
> updateArtisynthLibs
```

For more information on downloading library files, including information on how to do this on Windows, see Section 6.2.

The latest development version of ArtiSynth is stored in a module that is called `artisynth_2_0` for historical reasons. However, this name does not reflect the actual version number, which is stored in the file `VERSION`. Once you have checked out `artisynth_2_0`, you can rename the top level directory to anything you like.

Given a checked out version of ArtiSynth, you can update it by issuing the command

```
> cvs update -dP
```

within the `ARTISYNTH_HOME` directory. The options `-dP` tell CVS to include new directories that have been added and ignore (prune) directories that have been removed.

It is also possible to check out ArtiSynth within the Eclipse IDE; see Section 7.

The latest development version of ArtiSynth is stored in a module that is called `artisynth_2_0` for historical reasons. However, this name does not reflect the actual version number, which is stored in the file `VERSION`. Once you have checked out `artisynth_2_0`, you can rename the top level directory to anything you like.

6.2 Downloading Libraries

Because the `.jar` files and native libraries used by ArtiSynth are rather large, they are downloaded separately from the ArtiSynth webserver. Typically, this is done automatically by the ArtiSynth application, without requiring any action by the user. If you run ArtiSynth with the `-updateLibs` command line option, the program will ensure that not only are all the required libraries present, but that they also match the latest versions on the server.

In cases where you obtain ArtiSynth by checking out a version from the source repository, or when an update adds a new `.jar` file, then it may be necessary to run a stand-alone command to initially download the libraries, because it won't be possible to compile the source without the `.jar` files.

On Linux, MacOS, and Windows systems running Cygwin, the command to download libraries is called `updateArtisynthLibs`, and is located in `$ARTISYNTH_HOME/bin`. You can run it by opening a terminal window, changing to the `ARTISYNTH_HOME` directory, and running the command

```
> bin/updateArtisynthLibs
```

If your `PATH` environment variable has been set to include `$ARTISYNTH_HOME/bin` (see Section 6.3), you can simply run the command

```
> updateArtisynthLibs
```

from any directory.

On Windows systems, the command is called `updateArtisynthLibs.bat`, and is located in `$ARTISYNTH_HOME\bin`. You can execute the command by double-clicking on it in a file-browser. Alternatively, you can open a command interpreter (`CMD`), navigate to `$ARTISYNTH_HOME`, and enter the command

```
% bin\updateArtisynthLibs
```

ArtiSynth libraries are stored under `$ARTISYNTH_HOME/lib`, with the `.jar` files being placed in the `lib` directory and the native libraries in an appropriate subdirectory (e.g., `Linux64` or `Windows`) which is created if necessary. The required libraries are listed in the file `$ARTISYNTH_HOME/lib/LIBRARIES`. This file is checked into the repository, so the system can always determine what libraries are needed for a particular checkout version.

6.3 Environment variables

The following environment variables are involved in ArtiSynth's execution:

ARTISYNTH_HOME

The path name of the ArtiSynth installation directory.

ARTISYNTH_PATH

A list of directories, separated by colons ":" (or semi-colons ";" on Windows), which ArtiSynth uses to search for configuration files such as `.artisyntInit` or `.demoModels`. A typical setting for `ARTISYNTH_PATH` consists of the current directory (indicated by "`.`"), the user's home directory, and the ArtiSynth installation directory. If `ARTISYNTH_PATH` is not defined explicitly in the user's environment, ArtiSynth assumes an implicit path consisting of the directory sequence just described.

CLASSPATH

A list of directories and/or `.jar` files, separated by colons ":" (or semi-colons ";" on Windows) which Java uses to locate its class files. This variable should be set to include `$ARTISYNTH_HOME/classes` and `$ARTISYNTH_HOME/lib/*` (the latter uses the wildcard `*` to specify all the `.jar` files in `$ARTISYNTH_HOME/lib`).

PATH

A list of directories, separated by colons ":" (or semi-colons ";" on Windows), which the operating system uses to locate executable programs and applications. Should be set to include `$ARTISYNTH_HOME/bin`. On Windows and Cygwin, `PATH` is also used to search for shared libraries, and so should also be set to include `$ARTISYNTH_HOME/lib/Windows` (on 32 bit systems) or `$ARTISYNTH_HOME/lib/Windows64` (on 64 bit systems).

LD_LIBRARY_PATH

On Linux, a list of directories, separated by colons ":", which the operating system searches in order to find shared libraries. Should be set to include `$ARTISYNTH_HOME/lib/Linux` (for 32 bit systems) or `$ARTISYNTH_HOME/lib/Linux64` (for 64 bit systems).

DYLD_LIBRARY_PATH

On MacOS, the equivalent of `LD_LIBRARY_PATH`. Should be set to include `$ARTISYNTH_HOME/lib/Darwin-x86_64`.

OMP_NUM_THREADS

Specifies the maximum number of processor cores that are available for multicore execution. Setting this variable to the maximum number of cores on your machine can significantly increase performance.

Note that settings for most of the above can be derived from the value of `ARTISYNTH_HOME`.

The basic ArtiSynth execution commands described in Section 5 will automatically set these environment variables if they are not already set in the user's environment. Nevertheless, we encourage all developers to set these environment variables globally in order to provide maximum access to all the ArtiSynth development tools and scripts. Different procedures for doing this are described in the following sections.

Note:
If you are using the Eclipse IDE, and you have not set the above environment variables globally, then you will need to set some of them directly in your Eclipse launch environment, as described in Section 7.4.

6.4 Example environment setup for bash

On Linux and MacOS, if you are using `bash` as your shell, then the environment can be configured by placing a block of commands similar to the following in your `.bashrc` file, located in your home directory:

```
# set ARTISYNTH_HOME to the appropriate location ...
export ARTISYNTH_HOME=$HOME/artisynth_2_X
export ARTISYNTH_PATH=.:$HOME:$ARTISYNTH_HOME

# For 32 bit Linux:
export LD_LIBRARY_PATH=$ARTISYNTH_HOME/lib/Linux:$LD_LIBRARY_PATH
# For 64 bit Linux:
#export LD_LIBRARY_PATH=$ARTISYNTH_HOME/lib/Linux64:$LD_LIBRARY_PATH
# For MacOS:
#export DYLD_LIBRARY_PATH=$ARTISYNTH_HOME/lib/Darwin-x86_64:$DYLD_LIBRARY_PATH

export CLASSPATH=$ARTISYNTH_HOME/classes:$ARTISYNTH_HOME/lib/'*':$CLASSPATH
export PATH=$ARTISYNTH_HOME/bin:$PATH
# Set to the number of cores on your machine:
export OMP_NUM_THREADS=2
```

Be sure to set `ARTISYNTH_HOME` to the proper location of your ArtiSynth installation directory, and comment or uncomment the appropriate `LD_LIBRARY_PATH` or `DYLD_LIBRARY_PATH` section for your machine.

These environment variables will be passed on to any program which you run from the shell (such as `artisynth` or `eclipse`). However, on MacOS, they will not be passed on to programs (such as `eclipse`) which you launch from the dock.

Alternatively, you can source the script `setup.bash`, located in the installation directory:

```
> source setup.bash
```

This will determine the system type automatically and set the environment variables accordingly, with `ARTISYNTH_HOME` set to the current directory from which the script is called (however, it *won't* set `OMP_NUM_THREADS`).

6.5 Example environment setup for csh/tcsh

On Linux and MacOS, if you are using `csh` or `tcsh` as your shell, then the environment can be configured by placing a block of commands similar to the following in your `.cshrc` file, located in your home directory:

```
# set ARTISYNTH_HOME to the appropriate location ...
setenv ARTISYNTH_HOME $HOME/artisynth_2_X
setenv ARTISYNTH_PATH .":"$HOME":"$ARTISYNTH_HOME

# For 32 bit Linux:
setenv LD_LIBRARY_PATH $ARTISYNTH_HOME/lib/Linux":"$LD_LIBRARY_PATH
# For 64 bit Linux:
#setenv LD_LIBRARY_PATH $ARTISYNTH_HOME/lib/Linux64":"$LD_LIBRARY_PATH
# For MacOS:
#setenv DYLD_LIBRARY_PATH $ARTISYNTH_HOME/lib/Darwin-x86_64":"$DYLD_LIBRARY_PATH

setenv CLASSPATH "$ARTISYNTH_HOME/classes:$ARTISYNTH_HOME/lib/*:$CLASSPATH"
setenv PATH $ARTISYNTH_HOME/bin":"$PATH
# Set to the number of cores on your machine:
setenv OMP_NUM_THREADS 2
```

These environment variables will be passed on to any program which you run from the shell (such as `artisynth` or `eclipse`).

Alternatively, you can source the script `setup.csh`, located in the installation directory:

```
> source setup.csh
```

This will determine the system type automatically and set the environment variables accordingly, with `ARTISYNTH_HOME` set to the current directory from which the script is called (however, it *won't* set `OMP_NUM_THREADS`).

6.6 Example environment setup on Windows

On Windows, it is recommended to set the environment variables in your global user settings, as described in Section 9. A full set of variable settings would look something like this:

```
ARTISYNTH_HOME c:\users\joe\artisynth_2_X
ARTISYNTH_PATH .;c:\users\joe;%ARTISYNTH_HOME%
CLASSPATH %ARTISYNTH_HOME%\classes;%ARTISYNTH_HOME%\lib\*
PATH %ARTISYNTH_HOME%\bin;%ARTISYNTH_HOME%\lib\Windows;%PATH%
OMP_NUM_THREADS 2
```

Of course, instead of `\users\joe` and `\users\joe\artisynth_2_X` you should use your home directory and the ArtiSynth install directory, and `OMP_NUM_THREADS` should be set to the maximum number of cores available on your machine.

6.7 Example environment setup on Cygwin

If you are using Windows Cygwin, then you can also set the environment variables in `.bashrc` for use in the bash shell. This is done similarly to Section 6.4, but with some differences: the shared library directory is placed in `PATH`, and `ARTISYNTH_HOME`, `ARTISYNTH_PATH`, and `CLASSPATH` must be set to use the Windows path style:

```
# set AH to the location of the ArtiSynth install directory
AH=$HOME/artisynth_2_X

# For 32 bit Windows:
export PATH=$AH/bin:$AH/lib/Windows:$PATH
# For 64 bit Windows:
#export PATH=$AH/bin:$AH/lib/Windows64:$PATH

# Use Windows path style:
export ARTISYNTH_HOME=`cygpath -w $AH`
export ARTISYNTH_PATH=".;`cygpath -w $HOME`;$ARTISYNTH_HOME"
export CLASSPATH="$ARTISYNTH_HOME\classes;$ARTISYNTH_HOME\lib"'\*;'"$CLASSPATH"
# Set to the number of cores on your machine:
export OMP_NUM_THREADS=2
```

The scripts `setup.bash` and `setup.csh`, described above, are configured to work properly for Cygwin.

> **Note:**
> For Windows, it is still preferable to set environment variables globally, as described in Section 6.6, because setting them in the Cygwin bash shell will not allow them to be seen when programs (such as Eclipse) are launched directly from the desktop.

7 Using the Eclipse IDE

Many ArtiSynth users employ the Eclipse IDE for both code development and running ArtiSynth. This section describes how to import ArtiSynth into Eclipse, and set the relevant environment variables. It is assumed that the user is familiar with both Eclipse and Java programming; an introduction to these topics is beyond the scope of this document.

Eclipse can be obtained from www.eclipse.org/downloads. The version that you want is (at the time of this writing) Eclipse IDE for Java Developers. Also, make sure that Eclipse is configured to run Java 6 or newer; for details, see Section 7.6.

If you are using a CVS check out of the development version (Section 6.1), then you should first read Section 7.1.

If you already have a version of ArtiSynth installed (either from a distribution or CVS checkout), you can import this into Eclipse as a project; see Section 7.2). Otherwise, you can obtain a CVS checkout directly from within Eclipse; see Section 7.3.

> For historical reasons, the default Eclipse settings that are bundled with ArtiSynth use a project name of `artisynth_2_0`, even though this does not reflect the current distribution number. If you wish, it is possible to rename your project (to the current distribution number or something else) after you have completed the installation procedures described in either Section 7.2 or Section 7.3. Simply select the project in the Package Explorer window, and then choose Refactor > Rename from the context menu.

7.1 Using Eclipse and CVS

For general information on using CVS within eclipse, see http://wiki.eclipse.org/CVS_FAQ.

It is recommended that you ensure Eclipse works properly with external CVS commands (this will be particularly necessary if you used the cvs command directly to obtain the checkout). To do this:

1. From within Eclipse, choose Window > Preferences (or Eclipse > Preferences)
2. Navigate to Team > CVS > Ext Connection Method.
3. Select Use another connection method type to connect, and make sure this is set to `extssh`.
4. Click OK.

> `extssh` is a connection protocol that is built in to eclipse, but is not recognized by external CVS commands; it essentially means "use the `ext` protocal in conjunction with `ssh` (secure shell)".

7.2 Importing ArtiSynth into Eclipse

1. From **outside** Eclipse, install the ArtiSynth Eclipse settings by unzipping `$ARTISYNTH_HOME/support/eclipse/eclipse-Settings.zip` into `$ARTISYNTH_HOME`. This will create the files `.project`, `.classpath`, and `ArtiSynth.launch`, along with the directory `.settings`, in `$ARTISYNTH_HOME`.

> **Attention MacOS users:**
>
> The default zip utility on MacOS will create a new sub-folder called `eclipseSettings` and will extract the files there. *You do not want this!!* Some of the files are then labelled as “hidden” by MacOS, which will prevent you from moving them to the correct place manually. Either extract the file directly to the `$ARTISYNTH_HOME` directory with a more standard application like 7-Zip (7zX for OSX), or use the `unzip` utility from the command-line.

2. From within Eclipse, choose File > Import ....
3. In the Import window, select General > Existing Projects into Workspace and click Next.
4. In the field Select root directory, enter (or browse to) `$ARTISYNTH_HOME` and then click Finish.

If Eclipse complains that "No projects are found to import", that most likely means that `eclipseSettings.zip` was not properly unzipped into `$ARTISYNTH_HOME`.

7.3 Importing ArtiSynth directly from CVS

It is also possible to import the ArtiSynth development version directly from the CVS repository:

1. Choose File > Import from the main menu.
2. Select CVS > Projects from CVS and click Next.
3. Click on Create a new repository location and click Next.
4. Fill in the repository information:
   - Host: ssh.ece.ubc.ca
   - Repository Path: /ubc/ece/home/hct/other/hct/cvsroot
   - User: *your UBC ECE account name*
   - Password: *your UBC ECE account password*
   - Connection Type: Select `extssh` if you don't intend to access the project using external CVS commands. If you do wish to use external CVS commands, select `ext` instead; however, this may not always work and may result in errors at the end of step 6.
5. Click Next.
6. In the box next to Use specified module name, type `artisynth_2_0` and click Next
7. Select Check out as a project in the workspace (with the project name `artisynth_2_0`) and click Next
8. Specify the location for the check out; this will become the ArtiSynth install directory `$ARTISYNTH_HOME`. If you leave Use default workspace location selected, this will be `workspace/artisynth_2_0`, where `workspace` is the Eclipse workspace directory. Otherwise, you can specify an explicit checkout location (which does not have to be located in the Eclipse workspace).
9. Click Finish.
10. If necessary, open a Java perspective by choosing Window > Open Perspective > Java. The project `artisynth_2_0` should appear in the Package Explorer window.
11. From **outside** Eclipse, install the ArtiSynth Eclipse settings by unzipping `$ARTISYNTH_HOME/support/eclipse/eclipseSettings.zip` into `$ARTISYNTH_HOME`. (This will create the files `.classpath` and `ArtiSynth.launch`, along with the directory `.settings`, in `$ARTISYNTH_HOME`, and will overwrite the existing `.project` file).

> **Attention MacOS users:**
>
> This is where most installs fail.
>
> The default zip utility on MacOS will create a new sub-folder called `eclipseSettings` and will extract the files there. *You do not want this!!* Some of the files are then labelled as "hidden" by MacOS, which will prevent you from moving them to the correct place manually. Either extract the file directly to the `$ARTISYNTH_HOME` directory with a more standard application like 7-Zip (7zX for OSX), or use the `unzip` utility from the command-line. For the latter, open a terminal window, change to the ArtiSynth install directory, enter the command
>
> ```
> unzip support/eclipse/eclipseSettings.zip
> ```
>
> and respond with `y` when asked if it is OK to replace `.project`.

12. From **outside** Eclipse, download the required `.jar` files and native libraries using the `updateArtisynthLibs` command as described in Section 6.2.
13. Finally, load the new settings into the project by selecting the project `artisynth_2_0` in the Package Explore window and selecting Refresh from the context menu.

7.4 Configuring environment variables

It may be necessary to set certain environment variables directly in your Eclipse launch configuration (as described below) so that ArtiSynth can locate configuration files and native library support.

If you are running on Windows, and you have set the environment variables as described in Section 6.6, then you can skip this section. If you are running on Linux or MacOS, and you are starting eclipse directly from a terminal window, you can also skip this section. Otherwise, you will need to set certain environment variables directly in your Eclipse launch configuration (as described below) so that ArtiSynth can locate configuration files and native library support.

> **Note:** At present, eclipse does not expand environment variables. In all the variable settings described below, references to `$ARTISYNTH_HOME` should be expanded (manually) to the path of the ArtiSynth install directory.

- On all systems, you will need to set `ARTISYNTH_HOME` to the path of the ArtiSynth installation directory. You may also optionally set `ARTISYNTH_PATH` as described in Section 6.3, and you should set `OMP_NUM_THREADS` to the maximum number of cores on your machine if you want the system to take advantage of multiple cores.
- On Windows, you will need to set `PATH` to `$ARTISYNTH_HOME\lib\Windows` (32 bit systems) or `$ARTISYNTH_HOME\lib\Windows64` (64 bit systems).
- On Linux, you will need to set `LD_LIBRARY_PATH` to `$ARTISYNTH_HOME/lib/Linux` (32 bit systems) or `$ARTISYNTH_HOME/lib/Linux64` (64 bit systems).
- On MacOS, you will need to set `DYLD_LIBRARY_PATH` to `$ARTISYNTH_HOME/lib/Darwin-x86_64`.

> **Note:** There is a bug in Eclipse 4.0+ on Windows where it will replace the system's native `%PATH%` variable rather than appending to it. To correct this behaviour, define `PATH` as `${env_var:path};$ARTISYNTH_HOME\lib\Windows64`

To set environment variables within Eclipse:

1. Open a java perspective if necessary by choosing Window > Open Perspective > Java.
2. Select the artisynth project in the Package Explorer form.
3. Choose Run > Run Configurations... to open the Run Configurations window.
4. In the left panel, under Java Application, select ArtiSynth.
5. In the right panel, select the environment tab.
6. To create a new environment variable, click the New button and enter the name and value in the dialog box.
7. When finished, make sure that Append environment to native environment is selected, and click Apply.

7.5 Building and Running ArtiSynth

Once ArtiSynth has been installed, you should be able to build and run the system. If necessary, first open a Java perspective by choosing Window > Open Perspective > Java. The project `artisynth_2_0` (or whatever you have renamed it to) should appear in the Package Explorer window.

To build the system, select the project `artisynth_2_0` in the Package Explorer window, and then choose Project > Build Project (it may be necessary to deselect Build Automatically in order to enable Build Project). To run the system, choose Run > Run.

If you are using a CVS checkout of the development version, then you can obtain the latest updates from the repository by selecting the project in the Package Explorer and selecting Team > Update from the context menu.

7.6 Checking that Eclipse uses Java 6 or newer

If you are unsure that Eclipse is using Java 6 or newer, then:

1. Choose Window > Preferences (or Eclipse > Preferences).
2. Select Java > Installed JREs.
3. Check the installed JREs in the table, under the Name field. Make sure that Java 6 is present (this will likely be indicated by a name containing `jdk-1.6.0.xx`, where `xx` is an update number, or, on Windows, `jre6`).
4. If Java 6 is not present, first make sure that it is installed on your system (see Section 3), and then add it to the list of installed JREs by clicking Search in the Preferences window, navigating to the Java 6 install directory, and clicking OK.
5. Make sure that Java 6 is selected from the list of installed JREs (via the checkbox on the left).

7.7 Preventing excessive resource copying

By default, ArtiSynth classes are built in a directory tree (`$ARTISYNTH_HOME/classes`) that is separate from the source tree (`$ARTISYNTH_HOME/src`). That means that Eclipse will try to copy all non-Java files and directories from the source tree into the build tree. For ArtiSynth, this is excessive, and results in many files being copied that don't need to be, since ArtiSynth looks for resources in the source tree anyway.

On MacOS and Linux, you can nhibit most of this copying:

1. Choose Window > Preferences (or Eclipse > Preferences).
2. Select Java > Compiler > Building.
3. Open Output folder, and in the box entitled Filter resources, enter the string:

```
Makefile,*.l*,*.?,*.??,*.???,*.????,???,????,?????
```

That should filter out the copying of most non-java files.

On Windows, Eclipse doesn't support `?` as a filter character, so you need to enter something like this instead:

```
Makefile,*.a*,*.b*,*.c,*.cc,*.cxx,*.e*,*.g*,*.h*,*.i*,*.m*,*.n*,*.o*,*.r*,*.s*,*.t*
```

Or, to prevent copying any resource, simply enter:

```
*
```

8 Installing Cygwin

Cygwin provides Windows users with a Linux-like, shell-based command environment. Its provides useful tools and enables ArtiSynth users to employ all the script-based commands in `$ARTISYNTH_HOME/bin`, as well as various `Makefile` commands.

Cygwin can be downloaded from www.cygwin.com. Run the executable that was just downloaded (setup.exe) to begin installation. After selecting a download source, an install directory, a package directory, an internet connection, and a download site, the installer will display a list of packages which the user can select for download. Normally, a default set of packages is already selected for installation. It is advisable to install these packages to ensure that Cygwin retains its basic capabilities. It is also recommended that you select the following additional packages:

- `archive`
- `asciidoc`, `cvs`, `make` (located under `devel`)

- `python` (located under `interpreters`).
- `openssh` (located under `net`)

If you are planning to compile C/C++ code, you may also want to install the various `gcc` and `gdb` packages (located under `devel`).

9 Setting Windows Environment Variables

A user can view, set, or change environment variables on Windows via the following steps:

1. Right-click My Computer, and then click Properties.
2. Click the Advanced tab.
3. Click Environment variables.
4. Choose one of the following options:
   - Click New to add a new variable name and value.
   - Click an existing variable, and then Edit to change its name or value.
   - Click an existing variable, and then Delete to remove it.

Variable settings can reference other environment variables, by surrounding them with percent signs, as in `%VARIABLE_NAME%`. For example, suppose you already have an environment variable `HOME` that gives the location of your home directory, and your ArtiSynth distribution is located in `packages\artisynth_2_X` relative to your home directory. Then the environment variable `ARTISYNTH_HOME` can be specified as

```
%HOME%\packages\artisynth_2_X
```

9.1 The Windows PATH variable

The special environment variable `PATH` tells the system where to find applications and programs. It consists of a list of directory names separated by semicolons ";". If you have applications or programs that reside in non-standard locations, you can enable the system to find them by adding their containing directories to the existing `PATH` value. For example, the programs associated with ArtiSynth reside in `$ARTISYNTH_HOME\bin`. If `ARTISYNTH_HOME` has the value `"c:\packages\artisynth_2_X"`, then the ArtiSynth programs can be made visible to the system by setting `PATH` to

```
%PATH%;c:\packages\artisynth_2_X\bin
```

Alternatively, if the ArtiSynth home location is described by the environment variable `ARTISYNTH_HOME`, then the above can be expressed as

```
%PATH%;%ARTISYNTH_HOME%\bin
```

Note that most programs and applications need to be restarted in order to get them to notice changes to the `PATH`.