Installing ArtiSynth from GitHub into Eclipse (MacOS)

1 Overview

This document describes how to install ArtiSynth directly from GitHub and import it into the Eclipse IDE (Integrated Development Environment), within a MacOS environment. While for demonstration purposes it is faster to install ArtiSynth using a precompiled release, installing from GitHub provides access to the most recent updates, and the Eclipse IDE is a convenient platform for developing and modifying the Java code used to create ArtiSynth models.

There are three parts to the installation process:

1. Installing Java
2. Installing Eclipse
3. Installing ArtiSynth from GitHub

Your host machine should be a 64 bit system that is based on the Intel processor. ArtiSynth may also run on ARM-based systems that provide an Intel compatibility layer.

> The new Apple ARM-based systems provide an Intel compatibility layer, named Rosetta, that appears to allow ArtiSynth to run as is.

We recommend a minimum of 8 Gbytes of memory, and 16 Gbytes or more is better if you are doing FEM work with larger numbers of elements.

This document is intentionally brief and describes only installation from Git into an Eclipse IDE. More complete instructions, including how to create and add models, integrate external models, and update ArtiSynth, are given in the General Installation Guide. Instructions for visualizing, navigating, editing and simulating models are given in the ArtiSynth User Interface Guide. Instructions on how to *build* a model using Java are given in the ArtiSynth Modeling Guide.

2 Installing Java

ArtiSynth requires that you have a full Java development kit (JDK) installed, which comes with a Java compiler; a simple run time environment (JRE) will not be sufficient. By default, ArtiSynth is compiled to be compliant with Java 8. While ArtiSynth will generally work under later Java versions, use of both the MATLAB and Jython interfaces generally requires Java 8. Therefore we currently recommend using Java 8; this also provides maximum compatibility with MATLAB. We specifically recommend the Java SE Development Kit 8u*XXX* (where *XXX* is the latest revision number), which can be obtained from Oracle (registration required). At the time of this writing, the download page is located at

www.oracle.com/java/technologies/javase/javase-jdk8-downloads.html

and the latest release is `8u281`. This page provides downloads for all systems; be sure to choose the download link appropriate to yours. For MacOS, this will be `jdk-8u281-macosx-x64.dmg` After the file downloads, open it, and follow the installation instructions.

The `java` and `javac` commands associated with your Java 8 JDK must be visible from a terminal window. To verify this, open a terminal window, run the command `javac -version`, and check that the command is found and that the version matches the JDK. If it does not, follow the instructions in Section 6.

3 Installing Eclipse

Eclipse is an integrated development environment (IDE) that is commonly used for writing and maintaining Java programs. While other IDEs exist for Java, such as NetBeans and IntelliJ, Eclipse is currently the most commonly used amongst ArtiSynth users. It should be noted, however, that an IDE is not *required* for Java programming; simpler programs, in particular, can often be created by editing the `.java` source files using a text editor and then compiling them with command line tools, including the `compile` utility supplied by ArtiSynth.

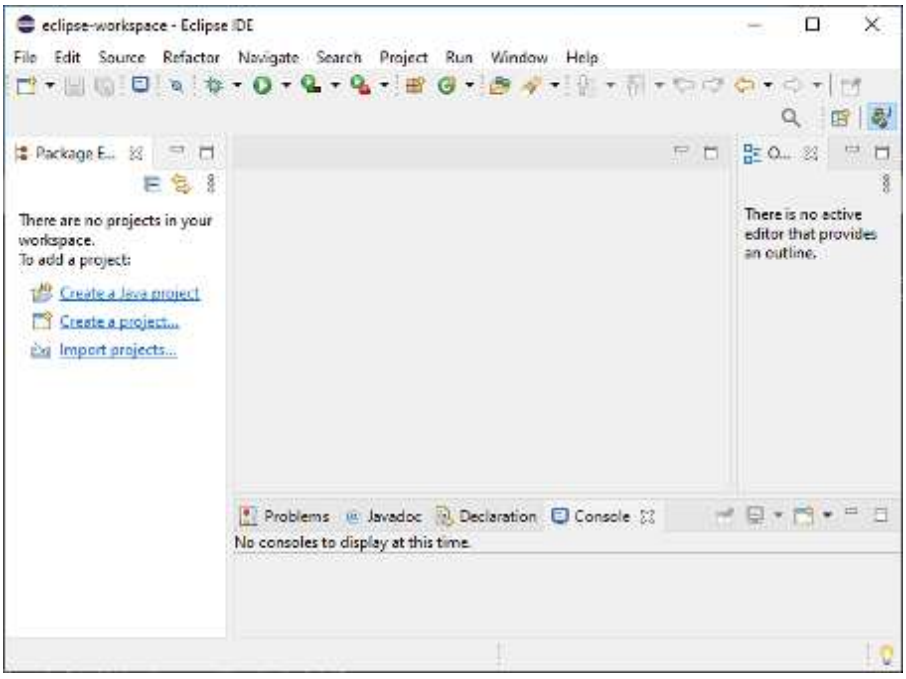
This document describes specifically the installation of Eclipse 2020-12; other versions should be similar.

Eclipse can be downloaded from www.eclipse.org/downloads/packages. From this page, choose "Eclipse IDE for Java Developers", macOS x86_64, which will download the file

```
eclipse-java-2020-12-R-macosx-cocoa-x86_64.dmg
```

Open this file, click on the "Eclipse Installer", select "Eclipse IDE for Java Developers", and follow the install instructions. When the install is complete, click the Launch button. The following dialog will appear, asking you to select a workspace directory:

This is where Eclipse settings and project information will be stored. Unless you have a another Eclipse already installed, the default location should be fine. (Remember also to check the "Use this as the default ..." box so that this query won't appear every time you open Eclipse.) Next, click Launch. A welcome page will appear; close this. A "donate" panel may also appear; close this too. You should then see an empty Eclipse display, similar to this:



> Before closing Eclipse, you may want to pin its icon to your dock to make it easier to restart.

Newer versions of Eclipse come with their own version of Java. However, since we want to use the Java 8 JDK, as discussed above, we need to configure Eclipse to use that instead. This can be done as follows:

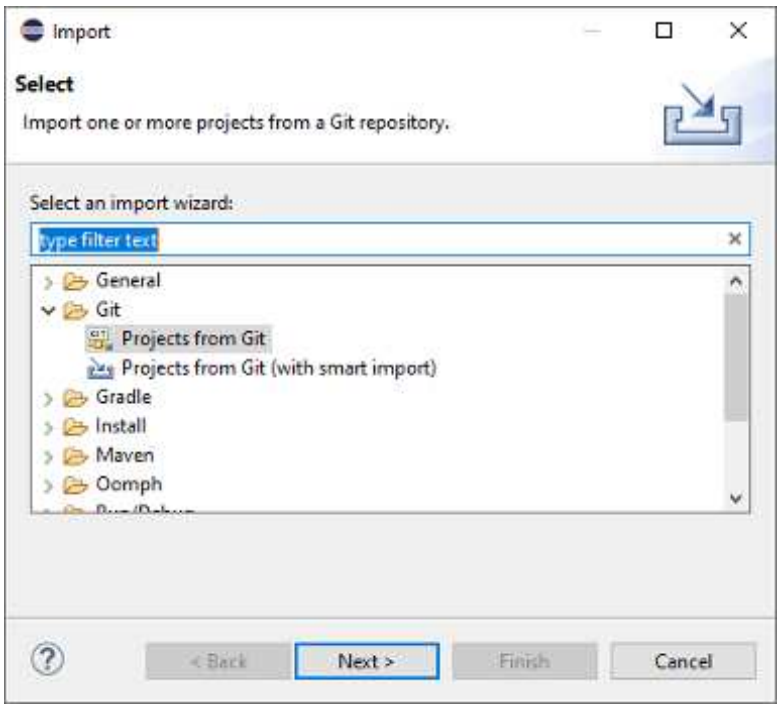
1. From the main menu, select "Eclipse > Preferences...".
2. A Preferences dialog will open. In the left panel, select "Java > Installed JREs", which will open an Installed JREs panel. On MacOS, all installed JDKs *should* appear in the panel including the Eclipse default and the JDK 8 which was just installed. Click on the box to the left of the JDK 8 entry to select it; ignore warnings about Java 15 compatibility.

3. Finish by clicking the "Apply and Close" button at the bottom of the Preferences dialog.

4 Installing ArtiSynth

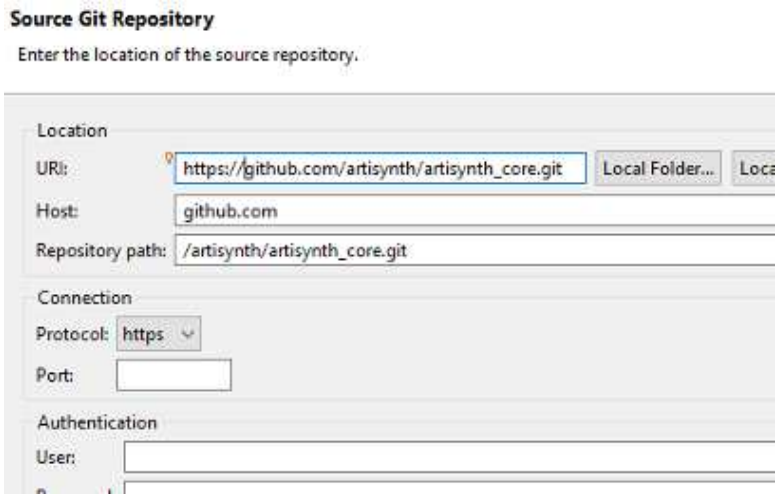
You can now install ArtiSynth from GitHub.

1. From the panel at the left of the main Eclipse frame, choose "Import projects...". This will cause an Import dialog to appear, as shown below. Open "Git > Projects from Git", and then click Next.



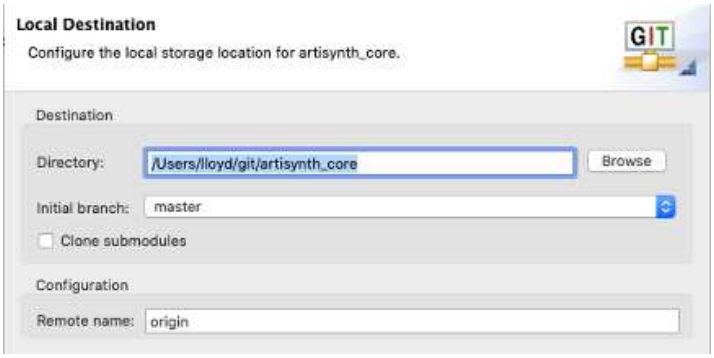
2. In the next dialog, choose Clone URI, and click Next.

3. A Source Git Repository dialog will appear, as shown below. In the URI field at the top, enter `https://github.com/artisynth/artisynth_core.git`
   This will automatically fill the Host and Repository path fields. Click Next.



4. A Branch Selection dialog will appear; uncheck `svn`, so that only `master` is selected. Click Next.

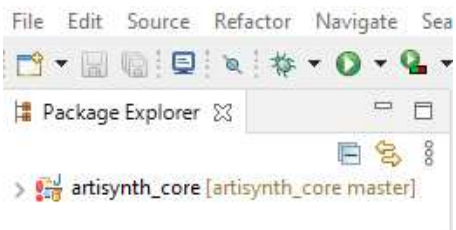
5. A Local Destination dialog will appear, as shown below, indicating the directoryinto which ArtiSynth will be placed locally. Use the default location, or edit it to some other desired location. This will be your *ArtiSynth installation directory*. Click Next.



6. ArtiSynth will now be downloaded; this may take a few minutes, depending on your network connection speed. Another dialog will appear, asking you to select to project import wizard. Leave the default ("Import existing Eclipse projects") selected, and click Next.

7. An Import Projects dialog will appear, confirming that you want to import `artisynth_core`. Leave everything as is, and click Finish.

`artisynth_core` has now been imported into Eclipse as a project. However, we are not quite done. Eclipse will try to compile `artisynth_core`, but will fail because some Java and native libraries are missing. (These libraries are not included in the GitHub repository because they are quite large.) The compile failure will be indicated by a red exclamation mark next to the `artisynth_core` project entry in the Package Explorer:



The Java and native libraries must be downloaded separately, *outside* of Eclipse. Open a terminal window, change directories to the ArtiSynth installation folder (e.g., `/home/roger/artisynth_core`), and run the command `bin/updateArtisynthLibs`:

```
> cd /home/roger/artisynth_core
> bin/updateArtisynthLibs
```

The process may take a few minutes, depending on network speed.

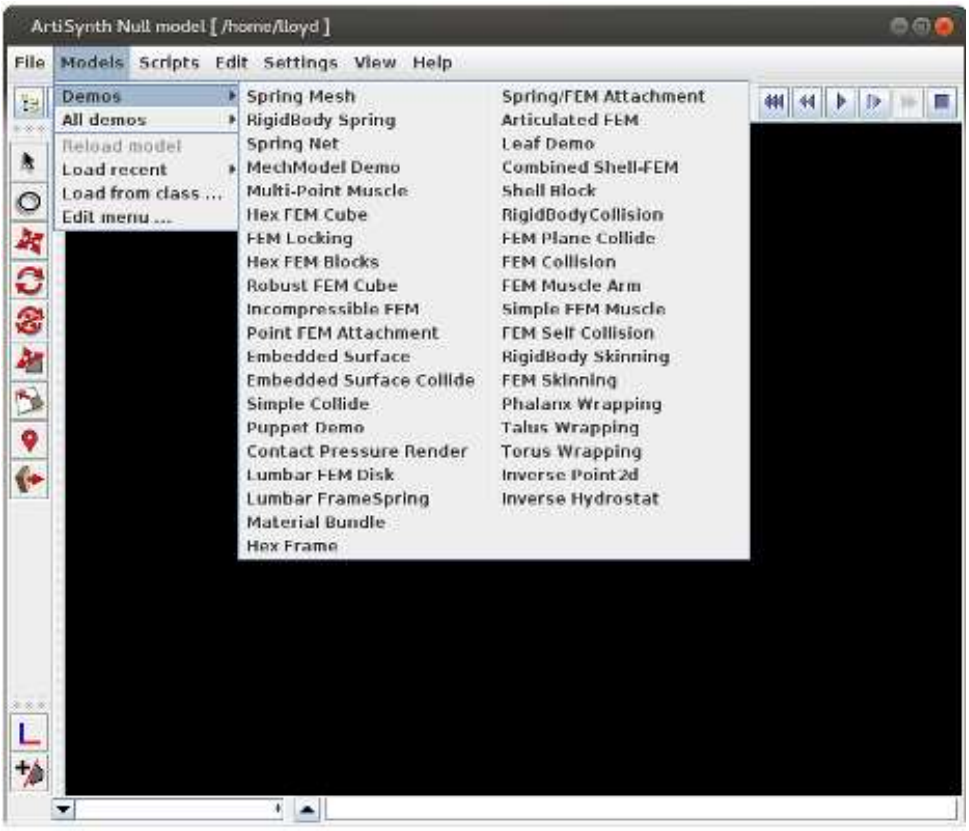
When the libraries are loaded, return to Eclipse, click on the `artisynth_core` project to select it, then "refresh", either by right clicking and selecting Refresh, or by hitting the F5 key. Eclipse should now find the libraries and compile ArtiSynth; a green progress bar will appear at the lower right while compilation is in progress.

5 Running ArtiSynth

You should now be able to run ArtiSynth. From the main menu, select `artisynth_core` by clicking on it in the Package Explorer, and then from the main menu select Run > Run, and the ArtiSynth application will start up (perhaps a little slowly the first time).

> On older versions of `MacOS`, errors similar to the following may appear on the Eclipse Console output:
>
> ```
> 2021-03-12 11:01:16.949 java[585:230b] invalid drawable
> ```
>
> This is due to a problem in the default version of the Java-OpenGL (JOGL) interface. To fix it, go back to the terminal window, and from the ArtiSynth installation folder run the command
>
> ```
> > bin/installJOGL2.3.2
> ```
>
> Then return to Eclipse, select `artisynth_core` in the Package Explorer, refresh it (right-click > Refresh or F5), and wait for it to recompile.

Within ArtiSynth, you can load a demonstration model by selecting Models > Demos from the main menu and choosing a demo model:



Once a model is loaded, it will appear in the viewer, and simulation can be controlled using the "play" controls located at the upper right of the application window:

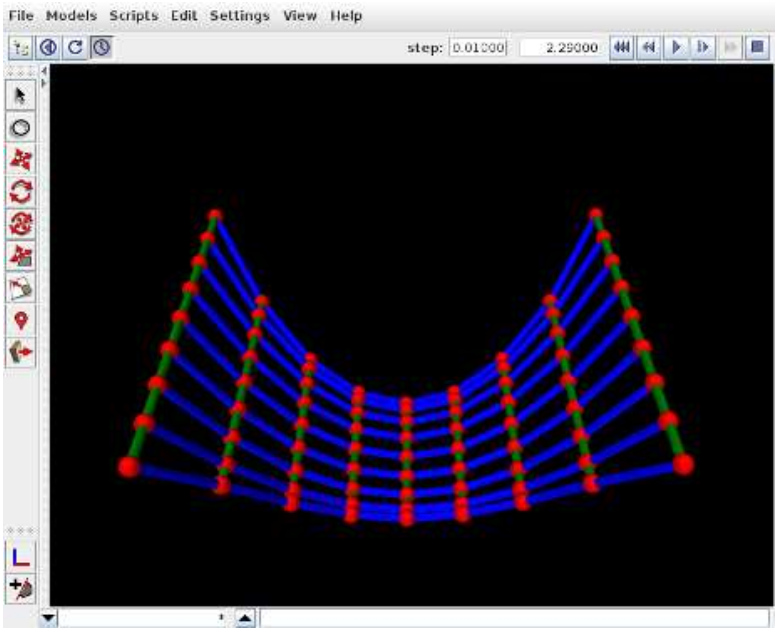
From left to right, these are: step size control, which controls the simulation step size (in seconds); current simulation time (in seconds); and the reset, skip-back, play/pause, single-step and skip-forward buttons. Starting and stopping a simulation is done by clicking play/pause, and resetting it to time 0 is done by clicking reset. The single-step button advances the simulation by one time step.

> On newer versions of `MacOS`, another problem that may occur when trying to run ArtiSynth models is that MacOS may complain about using a nonvalidated external library. This will take the form of a console error that looks like this:
>
> ```
> ...
> /Users/lloyd/git/artisynth_core/lib/MacOS64/libPardisoJNI.11.1.2.1.dylib)
> not valid for use in process using Library Validation: library load disallowed
> by system policy
> ...
> ```
>
> The problem here is that one or more of the native libraries are not "known" to Apple and are therefore not trusted. General information on working around this problem is described here:
>
> support.apple.com/en-ca/HT202491
>
> The short version is to immediately open your Security and Privacy settings after the error occurs, and then, near the bottom of the General tab, you should see a notification about the blocked application with a button to the right labeled Open Anyway. Clicking that button will grant the application a security exception.

A model can also be loaded by specifying either its defining class or a `.art` file containing its textual representation. For details, see the section "Loading, Simulating and Saving Models" in the ArtiSynth User Interface Guide. The

model menu can also be customized, as described in “Customizing the Model Menu”. The interface guide also contains information on using the ArtiSynth GUI for model visualization, navigation, editing, and simulation control. Instructions on how to build a model using Java are given in the ArtiSynth Modeling Guide.

The image below shows ArtiSynth with the Spring Net demo loaded:



6 Making the Java JDK visible to your system

If the `java` and `javac` commands of your Java JDK are not visible from a terminal command window, you can set the “default” JDK by setting the `JAVA_HOME` environment variable. This can be done inside the initialization file for whichever command line shell you are using.

Assume that the desired JDK has version number `1.8.0_281` and that your home folder is `<HOMEDIR>`. Then for the `bash` shell, one can use a plain text editor to edit `<HOMEDIR>/.bashrc` and insert a line of the form

```
export JAVA_HOME=‘/usr/libexec/java_home -v 1.8.0_281‘
```

while for the `csh` or `tcsh` shells, one can edit `<HOMEDIR>/.cshrc` and insert a line of the form

```
setenv JAVA_HOME ‘/usr/libexec/java_home -v 1.8.0_281‘
```

Setting `JAVA_HOME` can also be done directly within the shell; doing it within the initialization file simply avoids the need to do so each time a new terminal window is opened.