Installation Instructions

This document describes how to install the Inria version of AlphaZ which was forked for development at CSU.

Download Eclipse

This version of AlphaZ requires Eclipse 2022-06 (4.24). Follow the link below to the Eclipse download page, go to the "Eclipse SDK" section, and download the version which applies to your system. Note: the Windows, Power PC, and Linux AArch64 platforms are untested and may not work.

https://archive.eclipse.org/eclipse/downloads/drops4/R-4.24-202206070700/

The downloaded file is a `.tar.gz` file, so extract the contents. From a Linux command line, you can use the below command.

```
tar -xzf eclipse-SDK-4.22-linux-gtk-x86_64.tar.gz
```

This will extract all of the files to an `eclipse` folder, which contains all the files needed to run Eclipse. Move this folder to whatever location makes the most sense to you. If you plan on having multiple Eclipse installations (e.g., one for the Inria AlphaZ and one for the CSU AlphaZ), you can rename the folder to better identify this version of Eclipse.

Clone the Alpha Language Repository

Using Git, clone the main branch of the Alpha Language repository. This can be saved anywhere you like. The command below will clone the repository.

```
git clone git@github.com:CSU-CS-Melange/alpha-language.git
```

Add Plugin Sources

There are a handful of Eclipse plugins needed to run the AlphaZ compiler. Before they can be installed, Eclipse needs to know where to find them. There are two ways to do this. Importing them from a file is the most convenient, but if this doesn't work, they can be added manually.

Start by opening the unzipped folder in your file explorer and double-clicking the executable named `eclipse`. Alternatively, use a terminal to run the executable. If Eclipse asks you to select a directory to use as a workspace, select one. It is recommended that each copy of Eclipse has its own workspace. If you prefer, you may select the "Use this as the default..." option.

From the menu bar at the top, select `Help > Install New Software...`. Next, either follow the Quick Import or the Manual Import steps.

Quick Import

First, create a file anywhere on your computer with a ".xml" extension. Then, copy the following into the file, save, and close. TODO: make this a downloadable file.

```
<?xml version="1.0" encoding="UTF-8"?>
<bookmarks>
   <site url="https://csu-cs-melange.github.io/alpha-language/"
selected="true" name="Alpha Language"/>
   <site url="https://www.cs.colostate.edu/AlphaZ/gecos-mirror/gecos-tools-
emf/artifacts/" selected="true" name="GeCoS EMF"/>
   <site url="https://www.cs.colostate.edu/AlphaZ/gecos-mirror/gecos-
framework/artifacts/" selected="true" name="GeCoS Framework"/>
   <site url="https://www.cs.colostate.edu/AlphaZ/gecos-mirror/gecos-tools-
graph/artifacts/" selected="true" name="GeCoS Graph"/>
   <site url="https://www.cs.colostate.edu/AlphaZ/gecos-mirror/gecos-tools-
isl/artifacts/" selected="true" name="GeCoS ISL"/>
   <site url="https://www.cs.colostate.edu/AlphaZ/gecos-mirror/gecos-tools-
jnimapper/artifacts/" selected="true" name="GeCoS JNI Mapper"/>
   <site url="https://www.cs.colostate.edu/AlphaZ/gecos-mirror/gecos-tools-
tommapping/artifacts/" selected="true" name="GeCoS Tom Mapping"/>
   <site url="https://www.cs.colostate.edu/AlphaZ/gecos-mirror/gecos-tools-
tomsdk/artifacts/" selected="true" name="GeCoS Tom SDK"/>
   <site url="https://groovy.jfrog.io/artifactory/plugins-release/e4.24"
selected="true" name="Groovy Eclipse"/>
</bookmarks>
```

Back in Eclipse, in the "Available Software" window, click `Manage`. In the "Available Software Sites" window, click `Import`. Open the file you saved above. Click `Apply and Close`.

Manual Import

In the "Available Software" window, click `Add`. Copy and paste the name and location for the first row of the table below, then click `Add`. Repeat these steps for all rows of the table below.

Name	Location
Alpha Language	https://csu-cs-melange.github.io/alpha-language/
GeCoS EMF	https://www.cs.colostate.edu/AlphaZ/gecos-mirror/gecos-tools-emf/artifacts/

Name	Location
GeCoS Framework	https://www.cs.colostate.edu/AlphaZ/gecos-mirror/gecos-framework/artifacts/
GeCoS Graph	https://www.cs.colostate.edu/AlphaZ/gecos-mirror/gecos-tools-graph/artifacts/
GeCoS ISL	https://www.cs.colostate.edu/AlphaZ/gecos-mirror/gecos-tools-isl/artifacts/
GeCoS JNI Mapper	https://www.cs.colostate.edu/AlphaZ/gecos-mirror/gecos-tools-jnimapper/artifacts/
GeCoS Tom Mapping	https://www.cs.colostate.edu/AlphaZ/gecos-mirror/gecos-tools-tommapping/artifacts/
GeCoS Tom SDK	https://www.cs.colostate.edu/AlphaZ/gecos-mirror/gecos-tools-tomsdk/artifacts/
Groovy Eclipse	https://groovy.jfrog.io/artifactory/plugins-release/e4.24

Install Plugins

Now that the sources have all been added, the plugins can be installed. In the "Available Software" window, in the `Work with` dropdown menu, select "--All Available Sites--". Check the checkboxes for all of the plugins in the list below, then click `Next`.

Plugins to Install:

- Alpha
  - alpha Language
- EMF Tools
  - GeCoS EMF Tools
  - GeCoS EMF Tools Developer Resources
- Framework
  - GeCoS Framework
  - GeCoS Framework Developer Resources
- Graph Tools
  - GeCoS Graph Tools
  - GeCoS Graph Tools Developer Resources
- ISL
  - GeCoS JNI ISL bindings
  - GeCoS JNI ISL bindings Developer Resources
- JNI Mapper
  - GeCoS JNI Mapper Tools
  - GeCoS JNI Mapper Tools Developer Resources
- Main Package (required)
  - Eclipse Groovy Development Tools
- Modeling
  - EMF - Eclipse Modeling Framework Xcore SDK
  - Xtext Complete SDK
- TOM Mapping
  - GeCoS TOM Mapping Tools

    - GeCoS TOM Mapping Tools Developer Resources
- TOM SDK
    - GeCoS TOM Tools
    - GeCoS TOM Tools Developer Resources
- Uncategorized
    - GeCoS JNI Barvinok bindnigs
    - GeCoS JNI Barvinok bindnigs Developer Resources

At the "Installation Details" screen, click `Next`. At the "Review Licenses" screen, select the radio button for "I accept the terms of the license agreements" and click `Finish`. At the bottom-right of the Eclipse window, it should say "Installing Software". Wait for the installation to proceed.

During the installation, a "Trust" window should appear. Click the `Select All` button, then `Trust Selected` to allow the plugins to be installed. A new window should appear asking to restart Eclipse. Click `Restart Now` and wait for Eclipse to load again.

Import the Alpha Language Packages

To properly use the AlphaZ compiler, the latest copy of the Alpha Language repository must be imported into Eclipse. From the menu bar, select `File > Import`. In the "Select" window, select `General > Existing Projects into Workspace`. Next to the "Select root directory" dropdown menu, click `Browse`. Navigate to the Alpha Language folder cloned previously, then click "Open". In the "Import Projects" window, in the "Options" section, select "Search for nested projects". Click `Select All`, then `Finish`.

Fixing the Import

At the time of writing, several unnecessary projects have compile errors. These projects can all be closed. In the "Project Explorer" window, find each of the projects in the list below, right-click it, and select `Close Project`.

Projects to Close:

- `alpha-language/bundles/alpha.model.wizard`
- `alpha-language/bundles/alpha.targetmapping.xtext`
- `alpha-language/tests/alpha.model.tests`
- `alpha-language/tests/alpha.targetmapping.tests`

Additionally, at the time of writing, one of the projects is missing some folders. In the "Project Explorer" window, find the `alpha-language/bundles/alpha.loader` project. Right-click the project and select `New > Folder`. Name the folder `src-gen` and click `Finish`. Right-click the project again and select `New > Folder`. Name this second folder `xtend-gen` and click `Finish`.

If Eclipse does not automatically compile, from the menu bar, select `Project`. If the "Build Automatically" option is checked, click `Clean` and clean all projects, then wait for Eclipse to finish cleaning and building all projects. If the "Build Automatically" option is not checked, click `Build All`.

Test the Installation

To test if the plugins installed correctly, we will make a new project and attempt to use the AlphaZ compiler.

From the menu bar, select `File > New > Other...`. In the window that appears, in the "Select a wizard" menu, select `Plug-in Development > Plug-in Project` and click `Next`. In the "Plug-in Project" menu, give the project a name, leave all other options as their default, and select `Next`. In the "Content" menu, leave all options as their default and select `Finish`. If an "Open Associated Perspective?" window appears, select `No`.

If Eclipse does not automatically open the `MANIFEST.MF` file, open it manually from the "Package Explorer" window by double-clicking `META-INF/MANIFEST.MF`. In the window with the manifest file information (should be the center window), go to the `Dependencies` tab (near the bottom of that window). In the "Required Plug-ins" portion of the window, click `Add`. Select the `alpha.commands` plugin, then click `Add`. Save the file either with the `Ctrl+s` keyboard shortcut or by clicking the "Save" icon at the top of the Eclipse window.

In the "Project Explorer" window, right-click the project you created and select `New > Folder`. Using the `Folder name` textbox, name the folder `resources` and click `Finish`. Right-click that folder and select `New > File`. On the "File" screen which appears, give the file a name such as "Install Test.alpha" and click `Finish`.

In the "Project Explorer" window, navigate to that newly created Alpha file, and double-click it. If a window appears asking if you want to convert the project to an "Xtext project", click `No`. Write any Alpha program in this file, such as the one below, then save the file.

```
affine InstallTest [N] -> {: N>0}
    inputs  X: [N]
    outputs Y: [N]
    let Y[i] = X[N-i-1];
.
```

In the "Project Explorer" window, right-click the project you created and select `New > Package`. In the "Java Package" screen that appears, give the package a name. If you added a package declaration in the Alpha program (the example did not), set the name of the package to be the same as the package declaration in the program. Leave all other options as the defaults, and click `Finish`. Right-click the newly created package and select `New > Other`. In the "Select a wizard" screen that appears, select `Xtend > Xtend Class` and click `Next`. Name the class ("InstallTestCompile" for example), select the checkbox to create the `main` method stub, leave all other settings as default, and click `Finish`.

If the file did not open automatically, open the newly created Xtext class by double-clicking it from the "Package Explorer" screen. Write some code which, at a minimum, reads the previously created Alpha program. An example program which reads and shows the example Alpha program is below.

```
package installTest

import alpha.model.AlphaModelLoader
import alpha.model.util.Show

class InstallTestCompile {
    def static void main(String[] args) {
        var root = AlphaModelLoader.loadModel("resources/Install
Test.alpha")
        println(Show.print(root))
    }
}
```

In the "Project Explorer", right-click the Xtend class file and select `Run As > Java Application`. When the program runs, Eclipse should show the console with the program output. In the example above, it will output the "InstallTest` Alpha program, with a few changes. The output of the example is below.

```
affine InstallTest [N] -> {  : N > 0 }
    inputs
        X : {[i0]: 0 <= i0 < N }
    outputs
        Y : {[i0]: 0 <= i0 < N }
    when {  : N > 0 } let
        Y = (i->N-i-1)@X;
.
```