Rubus: A compiler for seamless and extensible parallelism

Prerequisites

Download and install the following software.

1. **OpenCL Drivers:** https://software.intel.com/en-us/articles/opencl-drivers
2. **Java 1.7 (1.8 is not fully supported):** http://www.oracle.com/technetwork/java/javase/downloads/java-archive-downloads-javase7-521261.html
3. **Apache Ant**: https://ant.apache.org/bindownload.cgi

Project Structure:

```
libs        # Dependent Jar files
samples     # Code samples (Java / Transformed / Transformed Decompiled)
src         # Rubus source code
LICENSE     # License
build.xml   # Ant build script
```

How to build:

1. Extract rubus.zip to some directory.
2. Open Terminal app (linux/mac) or Command Prompt in Windows
3. Navigate to the rubus folder (having build.xml file)
4. Type **ant** and press enter. Make sure that apache ant is already in your system path and accessible by terminal / command prompt
5. Build process will be started and Rubus.jar will be generated in **"dist"** folder and dependent lib folder will also be copied there.

How to use Rubus:

Rubus is compiled into a single Jar file in dest folder using build.xml. Rubus can be used from command line or with GUI interface.

Command Line interface:
The command line interface has a number of optional arguments.
Syntax:
java Rubus [options...] arguments...

Arguments contains input files and destination dir.

Example:
java Rubus --dist <Output Dir> -auto -clean -debug -export -manual -open -path <Input Files>

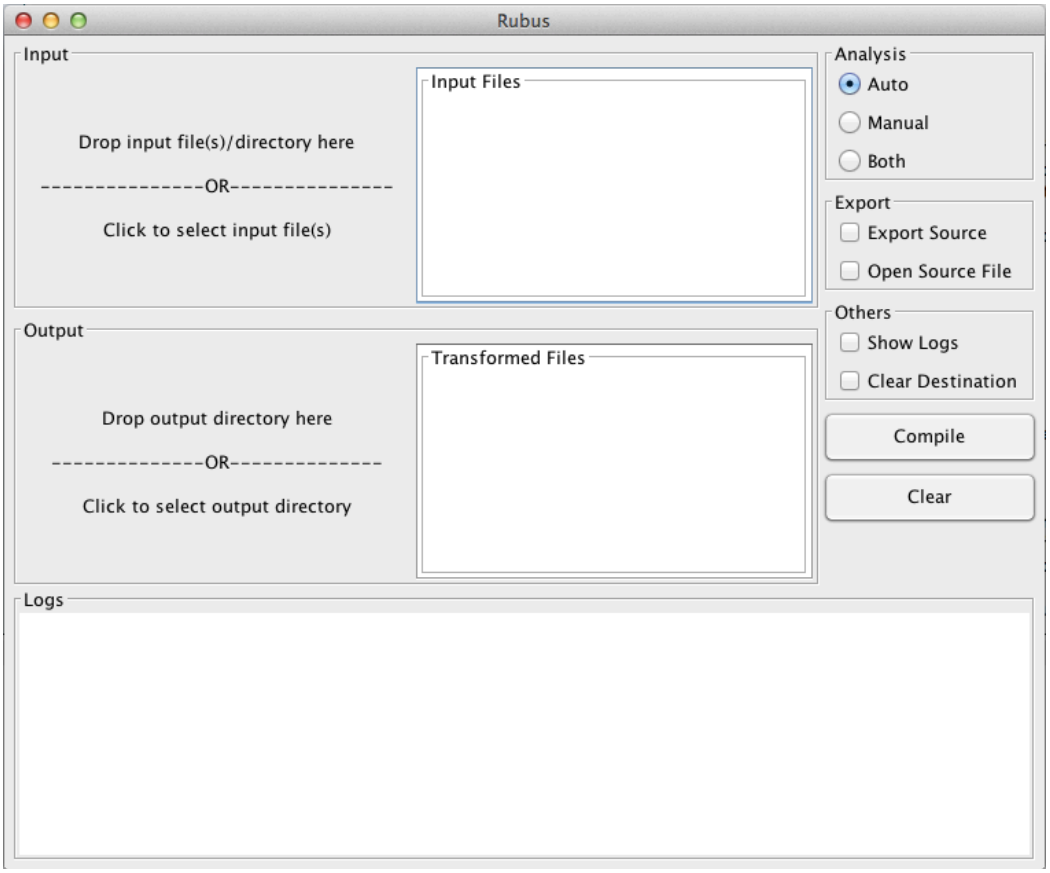
Possible Options for are as follow:

-auto #If -auto flag is given, Rubus will perform automated analysis.
-manual #If -manual flag is given, Rubus will perform annotation based analysis.
-clean #Destination folder will be deleted before transformation.
-debug #Rubus will show all log messages.
-export #Export generate kernel and executor Java code.
-open #Open generated source file after transformation.

--dist #Destination path to export transformed code.
--path #Class path to search class files from. All path should be separated by comma ','.

After options, all class and Jar files should be given, separated by the comma. To perform both auto and manual analysis, both -auto and -manual flags should be given.
All available arguments can be printed to the console using "Rubus.jar -help" command.

Graphical User Interface:
Graphical User Interface may be launched by double clicking on Rubus.jar. Alternatively, it can be run from command line using **java Rubus** command. All commandline options are also available in GUI environment.



You may drag input file(s) on "**Input"** area of the gui or click on input area to show a file picker.
Drag destination directory on **"Output"** area of the gui.

Select analysis type, Manual/Auto or Both.

If you want to export the source, check the "**Export Source"** checkbox.
If you want to open the exported source file, check the "**Open Source File"** checkbox.

To delete all files from destination directory before transformation, check "**Clear Destination**" checkbox.

To show logs in **"Logs"** area, check the **"Show Logs"** checkbox.
To apply transformation, Click "**Compile**".
To reset the GUI, click "**Clear**".

If any loop is expected for transformation, transformed class file will be generated in destination dir.

Note that the generated code use javacl to access OpenCL apis from java, thus a jar file **"javacl-core-1.0.0-RC3-shaded.jar"** must be in the class path to run the generated code. It is available in **libs** folder.

Detailed logs will be generated in a file named rubus.log that would be created automatically besides the Rubus.jar in dist folder.