Creating Projects on Windows XP Using Eclipse

• What is Eclipse ?

From the web : *"Eclipse is an extensible, open source IDE (integrated development environment). It is* completely *platform and language neutral. In addition to the eclectic mix of languages supported by the Eclipse Consortium (Java, C/C++, Cobol), there are also projects underway to add support for languages as diverse as Python, Eiffel, PHP, Ruby, and C# to Eclipse."*

Eclipse is a multi-platform workbench, you can use it to develop C/C++ applications and, of course, externals for MaxMSP. As the installation process of all the required tools is somehow tricky, this document describes, steps by steps, how to install and configure a stable development environment using the Eclipse workbench on Windows XP.

• Requirements

- The Eclipse platform (currently 3.1.1)
- The C Development Toolkit (CDT) plugin (currently 3.0.1)
- A C/C++ compiler : CYGWIN (or MinGW)
- The MaxMSP Software Development Kit

Warning : the steps order in the installation process is *critical*. If you don't follow the installation order, it will probably result in a corrupted architecture and *you will have to uninstall everything* you previously installed. As CYGWIN, for example, doesn't come with a handy uninstaller, you'll have to manually uninstall all of its components, including the registry keys and already running processes.

• Installing Eclipse

At this time of writing, Eclipse version is 3.1.1.

You can find it here : http://www.eclipse.org/downloads/index.php. You can choose to download it from the Eclipse foundation website or to use a torrent file (!). I suggest you install Eclipse in the root directory specifying which version you are currently using (i.e C:/Eclipse/3.1.1) and create a corresponding workspace folder *outside* the main directory (i.e C:/Eclipse/Workspace/3.1.1) for convenience. In fact, you'll probably have to update your Eclipse platform to a very different version soon and choosing good folder architecture is the best way to keep your work safe.

If you don't have a JRE, i wonder how you could use the new Java externals in MaxMSP ;) and I suggest you install it the usual way. Cycling '74 recommends the use of the SUN JRE/SDK.

• Setting up Eclipse for Java externals development (*optional*)

Now you are ready to test your brand new development platform. *The following lines are a top bonus in this document, describing how to setup a Java development environment for mxj externals and you can skip it if you don't care. But as Eclipse is, primary, a Java development tool, it is a good way to test your installation.*

- Choose File / New / Project / Java Project
- Name it "Test" or whatever
- Choose Libraries in the Java Settings pane, Add external jars
- Typically, add the max.jar from the ./common files/cycling '74/java/lib directory, the jitter.jar which should be in the same folder if you already own Jitter, and any other .jar files you will use in your Java external.

In the Package Explorer window on your left, you should see a new tree with the name of your project ("Test") on top with the JRE library and the max.jar (and any other libraries you've previously added) following. If not, right-click on your project, choose Properties / Java Build Paths / Order and Export, and check the boxes in front of the linked libraries. This should do the trick.

- Now go to the ./common files/cycling '74/java folder and edit the max.java.config text file : under the line <; add /Users/topher/myclasses to the dynamic classpath of MXJClassLoader ;>, replace by, i.e : <max.dynamic.class.dir "/Eclipse/WorkSpace/3.1.1/Test">, where "Test" is the name of your new Java Project located in your workspace directory. Save the edited text file.

Now you are ready to write your first Java external for MaxMSP. Right-click on your Project Folder in the Package Explorer, choose New / File, name it "Hello_World.java". In the editor window, write :

```
import com.cycling74.max.*;

public class Hello_World extends MaxObject
{

   public Hello_World()
   {
         post("Hello World!");
   }
}
```

Make sure you've checked the Build Automatically option under the Project menu, and launch MaxMSP. Type [mxj Hello_World] in an empty box. Et voila ! "Hello World!" should appear in the Max window.

Now that you are sure your Eclipse platform is properly installed, you will need a GNU C/C++ compiler / debugger and all the accompanying tools (make, binutils, GDB) available. You got the choice between Minimalist GNU for Windows (MinGW) and Cygwin toolkits. We have chosen the Cygwin toolkit, which is *"a UNIX-like environment for Windows that includes a GCC port along with all necessary development tools, including automake and the GNU Debugger (GDB). Cygwin is built around the* `cygwin1.dll` *library."*

I would like to warn you that it is critical that you install Cygwin just *after* the Eclipse platform and *before* the CDT plugin. If you don't, the CDT plugin won't be able to initialize itself around the Cygwin components and paths. It will result in a non-working setup and you will have to uninstall both the CDT and the Cygwin, this last one manually (see *How To Uninstall Cygwin* at the end of this document).

The installation process is not complex but no easy also. First, you'll have to find a working setup file for Cygwin. You can download it here : http://www.cygwin.com/setup.exe
This is not a package containing the Cygwin release but a remote setup engine that will let you choose the components you want to install. You can take a look at the very exhaustive tutorial located here http://www.ics.uci.edu/~stauro/courses/cse141/summer2005/CourseNotes/InstallingEclipse.htm and *follow exactly all the steps and choices described in it* or read the following, which makes a digest of this paper.

- **Launch the setup.exe** you have previously saved in a convenient place for future reuse. (Notice the absolute graphic design of the Cygwin icon and keep in your head you'll have to find a new icon set quickly on deviantart.com in order to keep the perfect look of your customized windows visual style).

- **Select *"Install from internet"***

- **Choose a root directory** to install your Cygwin. (I suggest you keep the C:/Cygwin for convenience). As I assume you're on windows, *be sure to check the DOS Default File Type option* and **click next**.

- **Choose a folder** to store the Cygwin installation package for future reuse

- **Select your internet connection type**

- **Choose one of the download mirrors**. It is sometimes a pain to find one that has a proper connection speed or even the good files on its server. If you got an error, don't panic, just re-launch the setup from the beginning and choose another mirror.

- Now comes the tricky part : you are in front of a hundred possible programs to download but you only need the C++ compiler / debugger components.

  Expand the Devel list by clicking on the + on the right hand side.

Here are the options you need to choose (to choose, click on the “Skip” to the left hand side of the column) :

NOTE : some options have already been pre-selected. Do not deselect any of these options unless you know what you are doing. I have included a list of all options that are set on my machine. Some may or may not have been pre-selected.

autoconf: Wrapper scripts for autoconf2.1 and autoconf2.5
autoconf2.1: Stable version of the automatic configure script builder
autoconf2.5: Development version of the automatic configure script builder
automake1.9: (1.9) a tool for generating GNU-compliant Makefiles
bashdb: Bash debugger
binutils: The GNU assembler, linker, and binary utilities
bison: A parser generator that is compatible with YACC
byacc: The Berkeley LALR parser generator
ctags: A C programming language indexing and/or cross-reference tool
expat: XML parser library written in C
gcc: C compiler upgrade
gcc-core: C compiler
gcc-g++: C++ compiler
gcc-mingw-core: Mingw32 support headers & libraries for GCC
gcc-mingw-g++: Mingw32 support headers & libraries for GCC C++
gdb: The GNU debugger
gettext: GNU Internationalization library and core utilities(PLUS LINK LIBS)
libiconv: GNU character set conversion library and utilities
make: The GNU version of the ‘make’ utility
mingw-runtime: MinGW Runtime
mktemp: Allows safe temp file/dir creation from shell script
pcre: Perl-Compatible Regular Expressions programs

I have personally added a few more features related to sound and graphics.

Click next and take a coffee, it could take some time.

<*Trumpets of victory.wav*> Cygwin is now installed on your computer. To test if your installation was successful, open the command prompt (go to Start / Run, type “cmd”) and type « make ». You should see the following message :

“`make:*** No targets specified and no makefile found.  Stop.`”

If not, you have to **modify your PATH environment variable**. Right-click on “My computer”. Go to Properties, Advanced Tab. Then in the left-bottom, select Environment Variables and find the “path” variable. Click edit. Append : “;c:\cygwin\bin”.

Now, re-open the command prompt and type « make » again…

Time to install the CDT plugin.

• Installing CDT plugin

Eclipse has its own tool for updating / installing new products.

In the *Help menu*, got to *Software Updates* and choose *Find and Install*.

The *Search for new features to install* option should be already selected, hit Next.

If you have previous features installed (like the Python Development plugin, or the JavaScript plugin), uncheck them.

Select New Remote Site. You will then have to manually enter the name of the feature (no matter) and the URL.

Type “Eclipse CDT” and add “http://download.eclipse.org/tools/cdt/releases/eclipse3.1”

Note : Be sure you got the latest version of Eclipse (currently 3.1.1) and that the CDT plugin version you have selected is compatible. If you have a doubt, go to http://www.eclipse.org/cdt/ and check.

Hit Ok. Then, select the “Eclipse CDT” remote site. It should look for the latest version of the CDT remotely and open a tree with two features : the tools and the SDK. Select both, click Next.

Accept the License, click Next, then Finish.

There will be a security Feature Verification. Click Install All.

It will download and install the CDT features. Time for another coffee, *it could be dramatically long*.

When it is done, you will have to restart Eclipse. Good boy.

Note : repeat this operation each time you want to install an extra feature (like the Python dev plugin).

Now you can test your brand new C/C++ development tool. Eclipse has its “personal” style and I suggest you check this page http://met.dnsalias.net:1111/teaching/cdt/ar01s04.jsp#installingcdt for building your first “Hello World!” with Eclipse. As we won’t discuss general C/C++ programming within the Eclipse environment in this document, you can also check http://eclipsewiki.editme.com/UsingCDTWithCygwin for an overview of the CDT features.

It’s time to configure the Eclipse CDT for MaxMSP externals development.

This is the final part which is a little bit tricky. It took me hours to find how to force the plugin to accept all the required commands. I hope this document will spare you some time and help you keep your temper.

- In Eclipse, go to the **File menu** and select **New / Manage C Make Project**. (It is a little bit confusing at the start because Eclipse will never mix C and C++ projects and the examples projects of the maxmspsdk are .c files, not .cpp… They won't build if you're in a C++ Project).

Note : As you see there are two kinds of C/C++ projects on Eclipse. Manage and Standard. If you never looked at the links I've added in the previous section, I suggest you do so now. Standard Projects let you write your own makefiles while Manage Projects write them for you.

- In the **New Project window**, choose a **Project Name**. We will use "My_Proj". Hit **Next.**

- In the **Project Type** scrolling menu, select **Shared Library (GNU on Windows)**.

- Leave the Debug / Release configurations checked (it allows you to create two different folders related to different building configurations. You will probably never change your configuration between these two, but how knows). Hit **Next**.

- As this is your first project, there will be no other projects listed in the **Referenced C/C++ Projects window**, so you may not care at this time. Just go to the **C/C++ Indexer Tab** and make sure the **Full C/C++ Indexe**r option is selected. Hit **Finish**.

As you left the **Build Automatically** option selected in the **Project menu**, Eclipse will now try to build the empty project and you should see the following message in the console window :

```
**** Full rebuild of configuration Debug for project My_Proj ****

Nothing to build for project My_Proj
```

This is good. You will also notice Eclipse created an **Includes folde**r under your My_Proj project folder in the **C/C++ Projects window** (usually on the left). This is even better. *If you don't have these two things by now, something bad happened and I'm afraid you will have to uninstall the CDT plugin and the Cygwin tools, then redo the installation process again.*

- Now, you *must* unselect the **Build Automaticall**y option in the **Project menu**.

- **Locate** your maxmspsdk folder, find the [**01. plussz]** project in the ./maxmspsdk/max-projects folder, **drag** *only* the following files : plussz.c and plussz.def, into your Eclipse My_Proj project folder. The files are copied in your ./Eclipse/Workspace/3.1.1/My_Proj folder on disk.

- **Double-click** on the plussz.c file in your My_Proj folder. It will load and open the .c file in the main **Editor window**. You will notice a message in the **Problems window** (middle-bottom) :

```
C/C++ Indexer Problem: Preprocessor Inclusion not found: ext.h in
file:
.\eclipse\WorkSpace\3.1.1\My_Proj\plussz.c on line: 15. plussz.c
My_Proj line 15
```

This problem will be reported in every fields of your project. A yellow triangle points the same on your plussz.c file in your My_Proj folder and a yellow ? rectangle points the #include ext.h line 15 of the .c file in the main editor window.

That's absolutely normal : we haven't linked the maxMSP library yet to the project. And that's what we will do now.

- **Right-click** on your My_Proj folder in the **C/C++ Projects window** (on the left).

- Go to **Properties**. You will be presented with a **Properties For My_Proj window**.

- Select the **C/C++ Build pane**.

- Now in the **Tool Settings pane** (on the right), go to the **GCC C Linker** options, select **Libraries**.

- In the **Libraries (-l) field**, hit the **Add button** and enter "MaxAPI".

- In the **Library search path**, hit the **add button** and enter the path were the max library is located on your disk. For me : "C:\maxmspsdk\c74support\max-includes"

- Go back to the **GCC C Compiler pane**, select **Symbols** and **add** *both* of the following to the **Defined Symbols (-D) field** :

  ```
  WIN_VERSION
  WIN_EXTVERSION
  ```

- Select **Directories** and re-enter the max library path in the **Include Paths (-l) field**, like you have done before in the GCC C Linker / Libraries.

- Select **Optimization** and choose the Optimize (-O1) level.

- Go back to the main **GCC C Compiler pane** and in the **Command field** on the right, enter :

  ```
  gcc -mno-cygwin -fpack-struct=2
  ```

  "`-mno-cygwin`" means use the Microsoft standard C libraries, instead of Cygwin standard C libraries. This step is important if you wish to distribute your extern to people that might not have Cygwin installed.

  "`-fpack-struct=2`" means set the structure member alignment to 2 bytes.

- Go to the main **GCC C Linker pane** and in the **Command field** on the right, enter :

  ```
  gcc -mno-cygwin ../plussz.def
  ```

  "`../plussz.def`" means use a .def file to export only the main function of the DLL

- Now, in the general **C/C++ Build pane**, select the **Build Settings.**

- In the **Build output** section enter the name of your external in the **Artifact name** field : plussz

- Change the **Extension** from DLL to .mxe

Everything is done for the Debug configuration. Hit **Ok**. You can do all the same for the Release configuration or choose not to take care of.

Now, **right-click** on your My_Proj folder and select **Build Project**.

You got a brand new plussz.mxe external for windows, ready to be added to the ./Common Files/Cycling '74/externals folder to be tested in maxMSP.

```
"On a dark desert highwaaaaaaaay,
  Cool wind in my haaaaaaaair,
    Oh yeah yeahhhhhhh…"
```

françois-eudes chanfrault – paris - december 05 2005

• Resources

online :

http://www.ics.uci.edu/~stauro/courses/cse141/summer2005/CourseNotes/InstallingEclipse.htm

http://www.eclipse.org/cdt/

http://met.dnsalias.net:1111/teaching/cdt/ar01s04.jsp#installingcdt

http://eclipsewiki.editme.com/UsingCDTWithCygwin

maxmspsdk files :

writingExternals.pdf
Cygwin_gcc_Howto.rtf