Tutorial 01: Setting up MonoDevelop to run Tao.(SDL, openGL) on Mono in Windows XP

Finally I have it working, but figuring it out has been chore. Here is what I have learned.

Mono

The first thing to do is download Mono from http://www.go-mono.com/mono-downloads/download.html. Click on the Windows icon and download *Mono for Windows, GTK#, and XSP*. I have created a directory c:\Programming, and have installed all the relevant programs there.

I can't remember whether you need to download the *GTK# for .NET* package or not. Run the Mono install and if it is a dependency you will need to download it. If Mono installs then it is not required. The install process is easy, just follow the instructions.

MonoDevelop

MonoDevelop has some dependencies that need to be fulfilled before the installation program will run. The dependencies and the IDE can all be downloaded from the Trunk Builds section at http://monodevelop.com/Download/Trunk_Builds.

The dependencies are:

.NET Framework 3.5
GTK# for .NET 2.12.9-2

Once they are installed MonoDevelop can be installed. Again the setup is simply a matter of following instructions.

TaoFramework

The TaoFramework is a set of wrappers around openGL and SDL among others. It can be downloaded from http://sourceforge.net/projects/taoframework/. If you download the zip file extract it somewhere (c:\Programming, eg). We will read the relevant dll into MonoDevelop. The framework comes complete with Sam Lantinga's c SDL dll files so you do not need to download these.

http://csgl.sourceforge.net/CSForDummy.html provides a good idea how the Tao framework wraps the raw dll files.

MonoDevelop Project

1) Open MonoDevelop and create a new Solution:

    File -> New -> Solution

2) In the following screen select C# and Empty Project. Give the Solution a Location directory and a Name. Click Forward and leave the options in the following screen unselected.
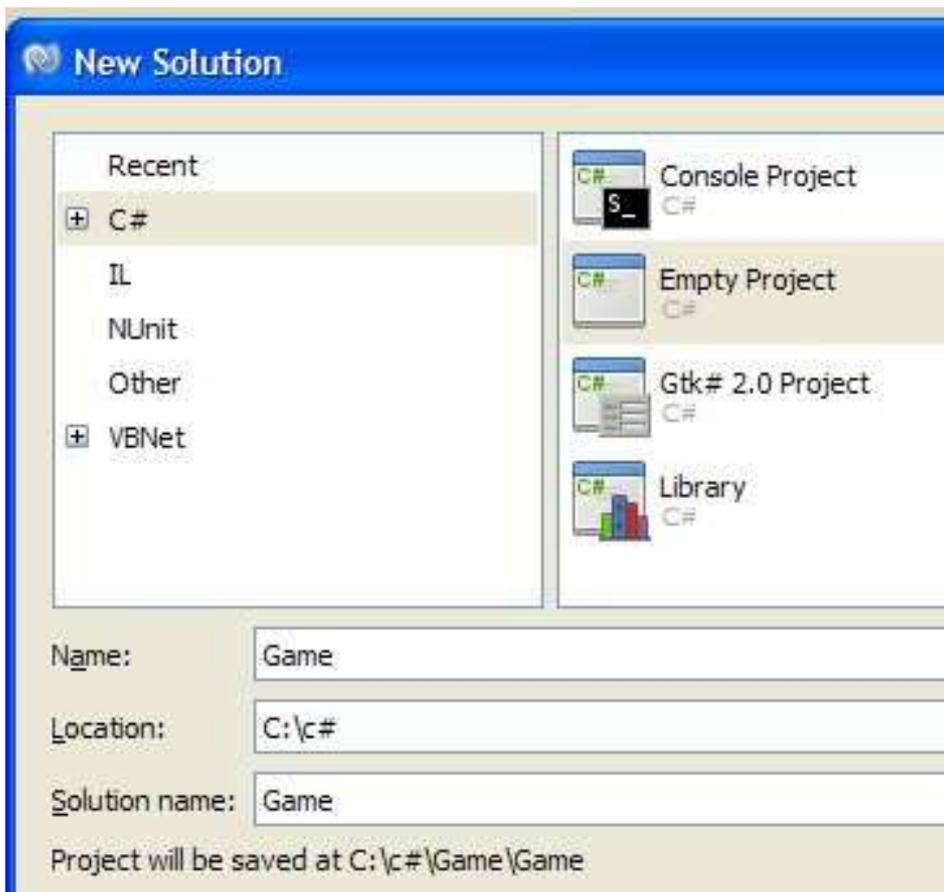
3) Create a class by right clicking on the Game icon and selecting Add -> New File. Select Empty Class from the next screen and give the class a name. I named mine App.
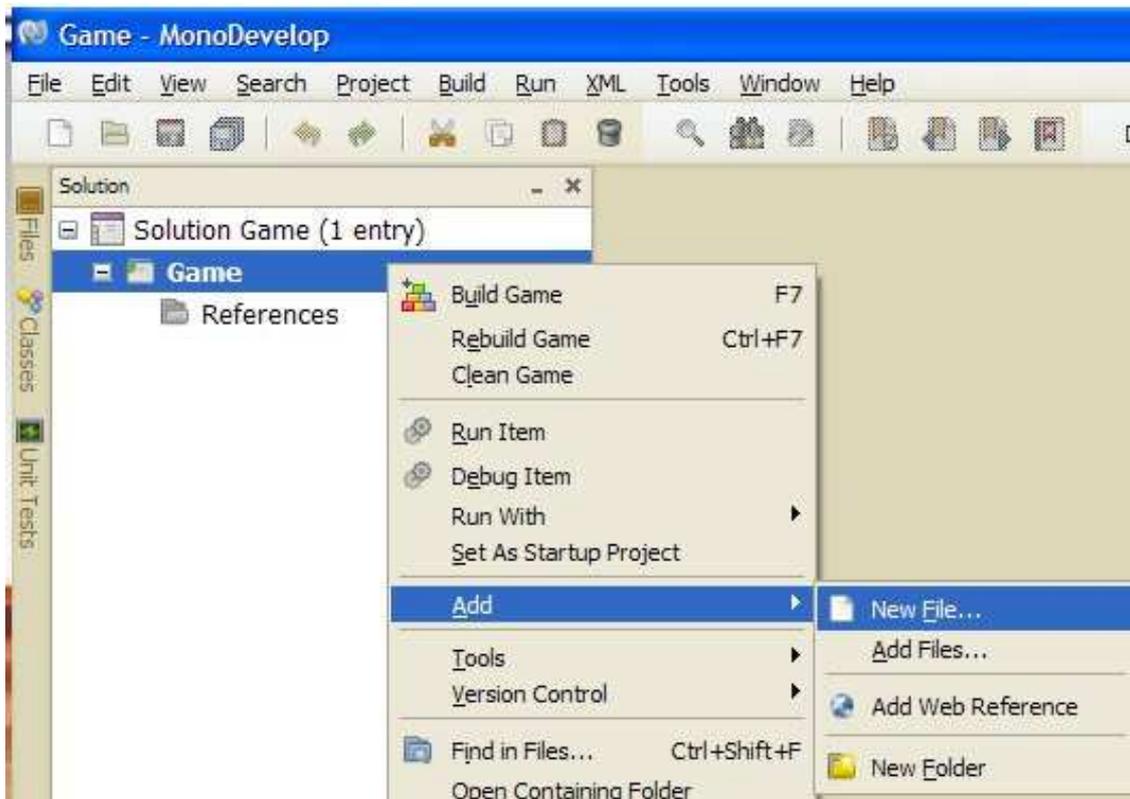
4) If you try to compile and run the program (CTL F5) you will get an error because there is no Main entry as yet. Just to make sure the environment is set up as it ought to be, copy the following code as a replacement for yours (you will need to change the namespace and class titles back to yours).

```
using System;

namespace Game{
     public class App{
           public App (){}

           [STAThread]
           public static void Main(){
                 Console.WriteLine("Every journey begins from
under one's feet");
           }
     }
}
```

Now the program should compile. The output will be printed on the Application Output screen that will appear at the bottom of the IDE.

Before continuing, cards on the table, I am come from the Java side and C# is new to me; albeit it very similar which is why I am here. If I do stupid C# then treat me as a monkey that can learn not as an inferior form of life. Frankly I don't care for the conventions if the code written does the work; if they don't do the work efficiently then please inform me.

5) We are ready to add the two dll from the Tao: OpenGL and SDL. If you try to type "using Tao." under the using System directive the auto-complete will not work because MonoDevelop has not been told about Tao:

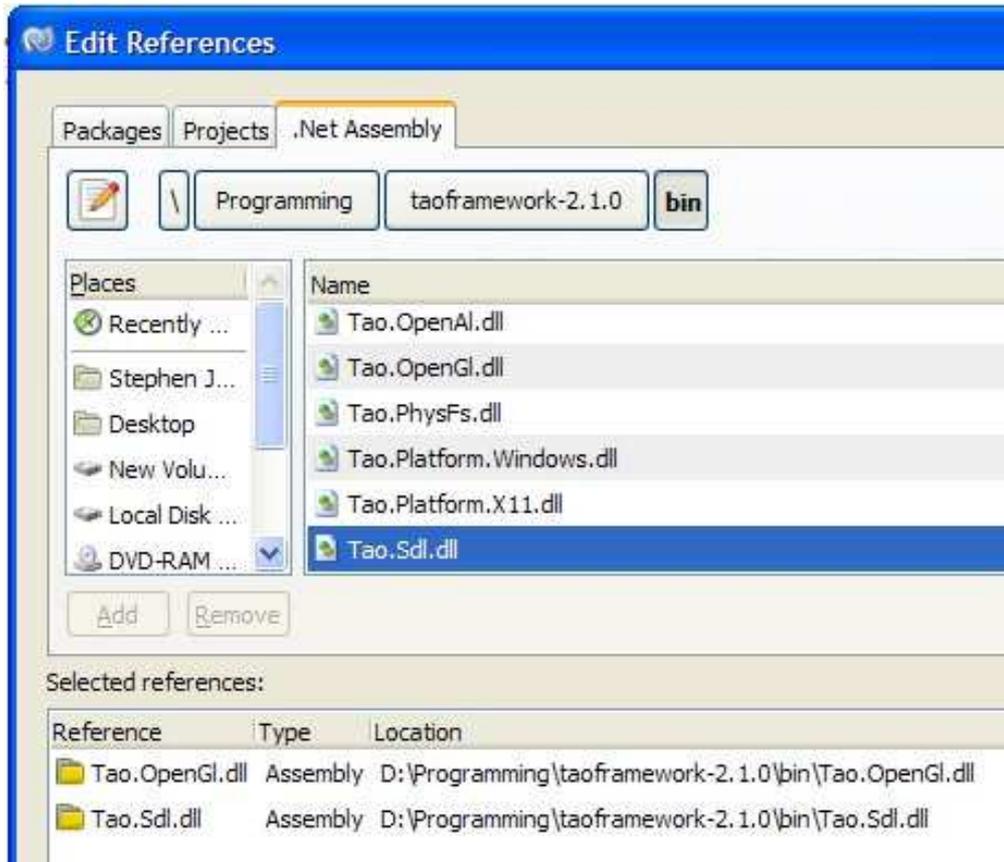"The name that can be named is not the eternal name" *Lao Zi*

In the Solution pad right click on the References icon and click Edit References
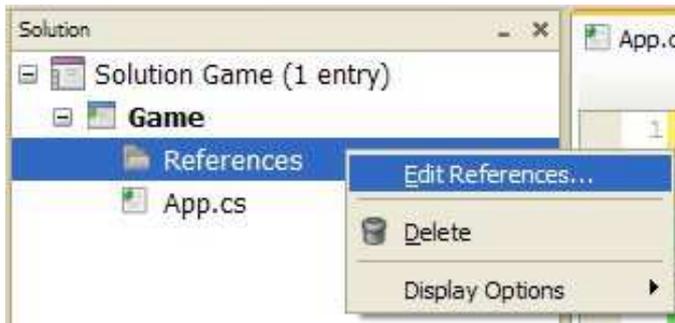
In the Edit References screen browse to your installation of the TaoFramework
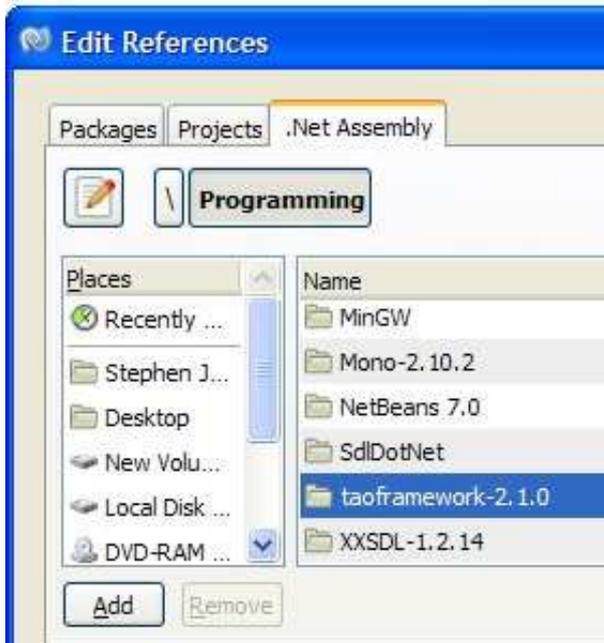
Open the Bin directory in the TaoFramework and Add the OpenGL and SDL dll files as illustrated.
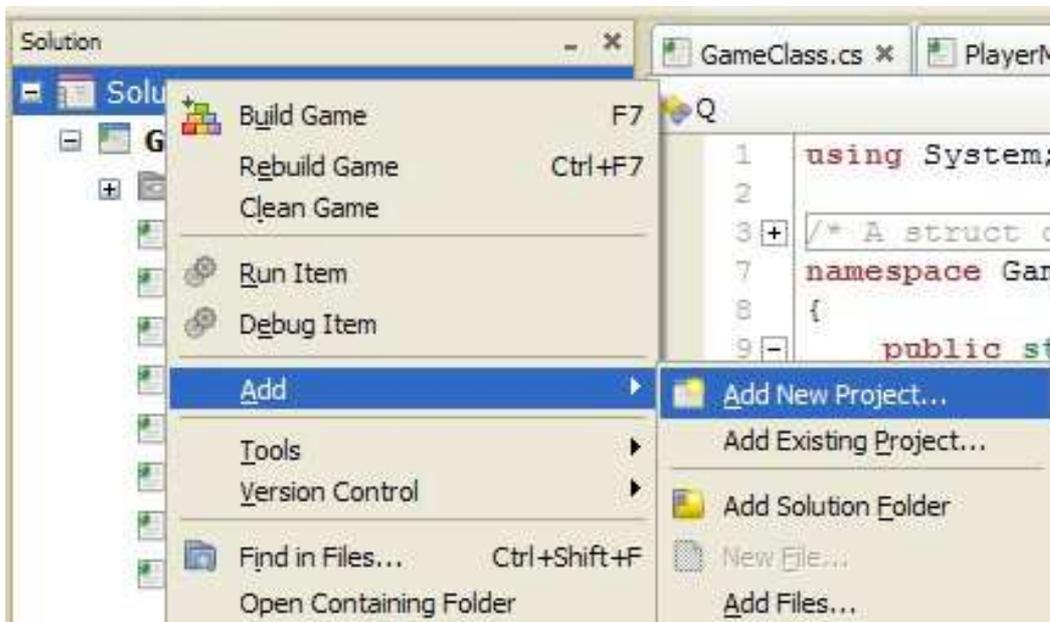

The auto-complete function will now recognize the Tao dll files you have added.
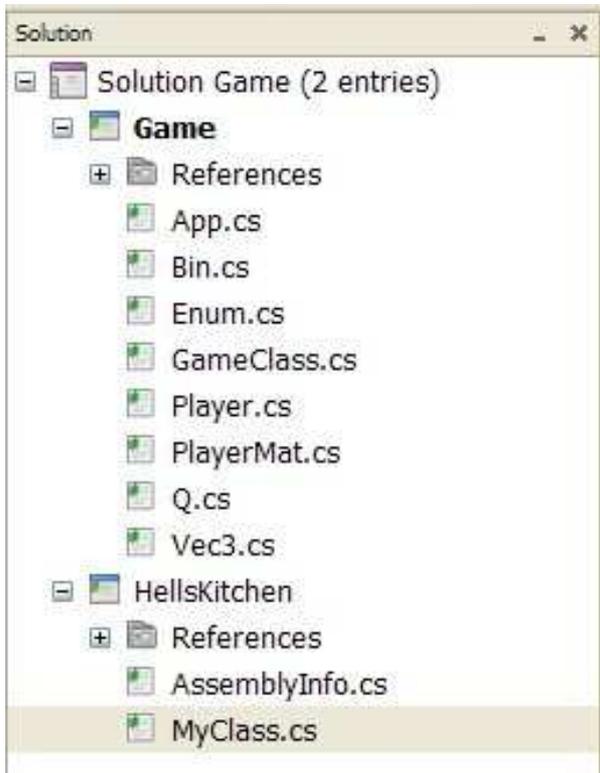
Adding a Driver File

I am thinking it makes sense to compile different game levels into separate dll files that can be loaded when needed. Essentially, a dll is an executable that contains no main function. If nothing else it provides for repetition of names made unique due to unique namespaces. To that end here is how to create and incorporate a dll in your project.

1) Add a new project. Select Library from the following screen and give it a name. I am naming the dll after the name of the location/level that they will code.

2) Your solution pad should now look something like this

```
Solution                           _  ×
□ □ Solution Game (2 entries)
   □ □ Game
      ⊞ □ References
         □ App.cs
         □ Bin.cs
         □ Enum.cs
         □ GameClass.cs
         □ Player.cs
         □ PlayerMat.cs
         □ Q.cs
         □ Vec3.cs
   □ □ HellsKitchen
      ⊞ □ References
         □ AssemblyInfo.cs
         □ MyClass.cs
```

HellsKitchen is the name I gave to the new project. I have altered MyClass to:

```
using System;

namespace HellsKitchen
{
     public class MyClass{
          public MyClass (){
               Console.WriteLine("Developing a level titled
HellsKitchen");
          }
     }
}
```

3) Now we need to link the dll back into Game, for Game contains the main function in the App class. Actually, there is an option allowing a program to run more than one executable main (right click Solution and choose Options, select Startup Projects from the left panel). By default it is set to Single Startup Project, which is what we want.

Linking HellsKitchen into the project is similar to linking the SDL and openGL dlls.

a) right click the References icon under the Game entry and select Edit Reference.
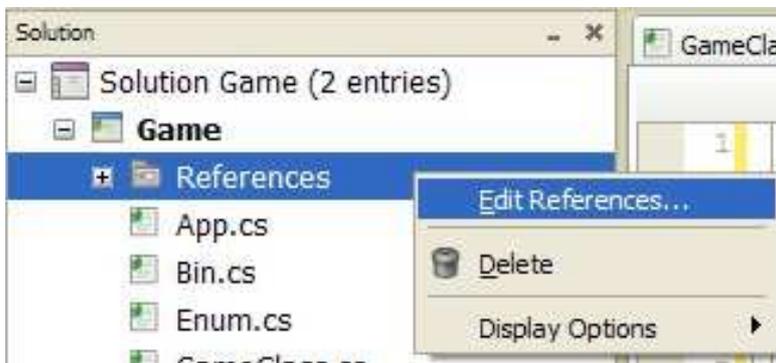
b) in the Projects tab select the library you want to incorporate.
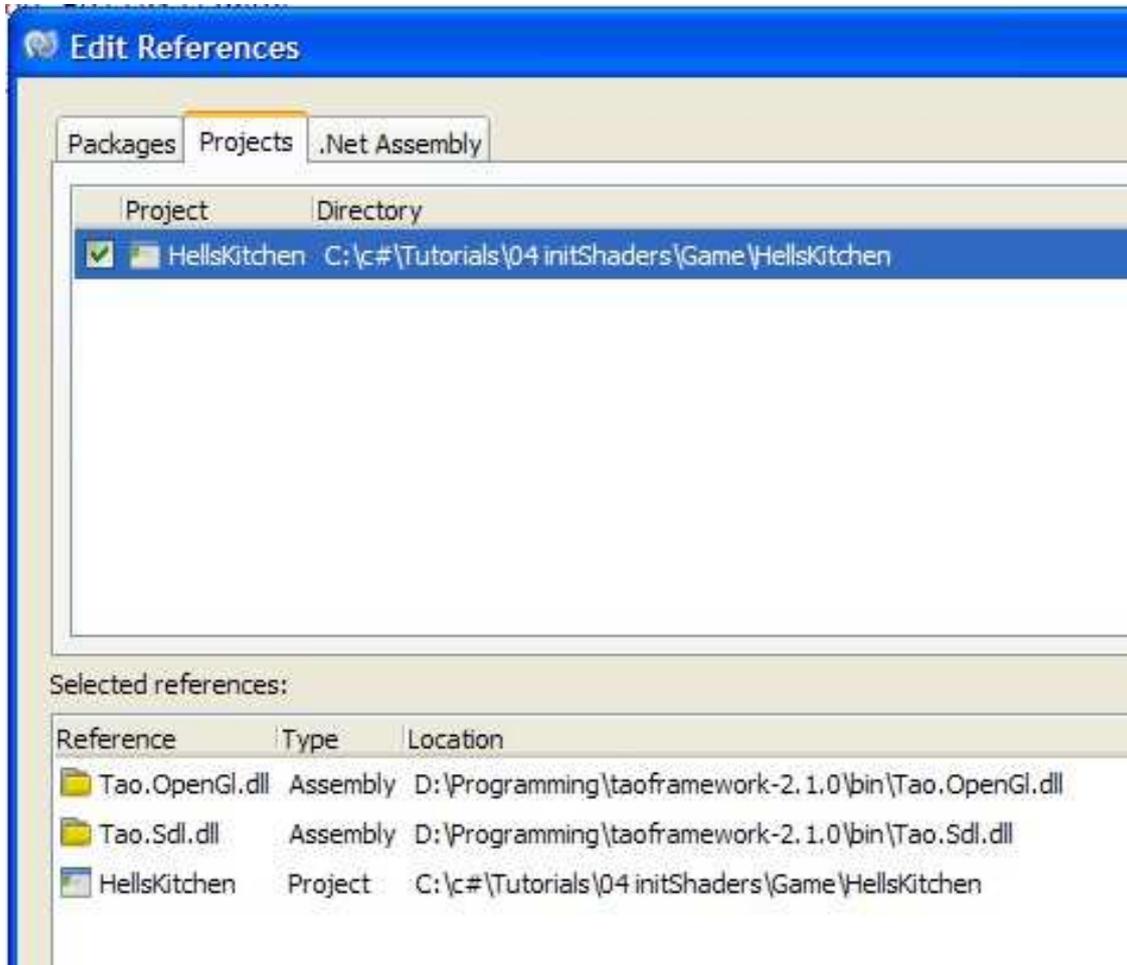
c) the driver can now be accessed by the Game project by using the namespace directive. In App.cs, for example:

```
using System;
using HellsKitchen;

namespace Game{
      public class App{
            private MyClass c;

            public App (){
                  c=new MyClass();
            }

            [STAThread]
            public static void Main(){
                  Console.WriteLine("Every journey begins from
under one's feet");
            }
      }
}
```

will write to the console at the bottom of MonoDevelop:

Every journey begins beneath one's feet

Developing a level titled HellsKitchen

That completes the environment setup. The link for the commented SDL/OpenGL code can be found at http://www.sjonesart.com/gl.php under the C# heading.