How to create your very own FUSEE plugin for Blender:

> 1. Install these four programs:

   a. Microsoft VisualStudio 2013 b. Python 3.4 (x64)
   c. SWIG for Windows 3.0.6 d. Blender 2.72 (x64)

> 2. Create and set system environment variables:

   a. Open: System Properties -> Advanced system settings -> Advanced -> Environment Variables

   b. Add the path to your SWIG and Python folders to the PATH variable -> e.g.: D:\swigwin-3.0.6\;D:\Python34\;

   c. Add a new variable BLENDER_ROOT and set it to the path of your Blender version folder -> e.g.: D:\Blender Foundation\Blender\2.72\

   d. Add a new variable PYTHON_ROOT and set its value to the path of your Python folder -> e.g.: D:\Python34\

   e. Add or change the value of the VS100COMNTOOLS variable to the path of your VisualStudio 2013 folder
   -> e.g.: C:\Program Files (x86)\Microsoft Visual Studio 12.0\Common7\Tools\

> 3. Open the solution SwigBlender.sln and change the build mode to Release x64. If you don't have read and/or write permissions for your Blender folder you have to run VisualStudio as administrator.

> 4. Build projects 01_CppApi, 02_Swig, and 03_CppWrapper in that order. Use "Project only" when building.

> 5. Add all files in \SwigIt\CsWrapper\Generated to the corresponding folder in the 04_CsWrapper project in VisualStudio.

> 6. Build projects 04_CsWrapper, 05_CsClient, 06_ManagedBridge, and 07_BlenderPlug in that order. Again, use "Project only".

> 7. Set 07_BlenderPlug as the starting project. Open the project properties and go to "Debugging". Set the command property value to the path of your blender-app.exe and the working directory property value to the path of your Blender folder.
   -> e.g.: D:\Blender Foundation\Blender\blender-app.exe
   and D:\Blender Foundation\Blender\

> 8. Click on the start button. Blender should open. Then, switch to the Scripting view (see upper menu, next to "Help").

> 9. You should see a black console. Type:

```
from fusee import uniplug
uniplug.init()
```

   This executes the code written in the CsClient project.

> 10. That's it! You can now start to add your own code to the CsClient.cs. All you have to do is to rebuild the CsClient and BlenderPlug projects and start Blender again (don't forget to close Blender before building).

   If you need more than one function (i.e. more than uniplug.init()) you can modify the Main.cpp of the BlenderPlug project and/or the two files ManagedBridge.h and ManagedBridge.cpp and rebuild the corresponding projects. Please do not change the uniplug_blender_api.h unless you know what you're doing.