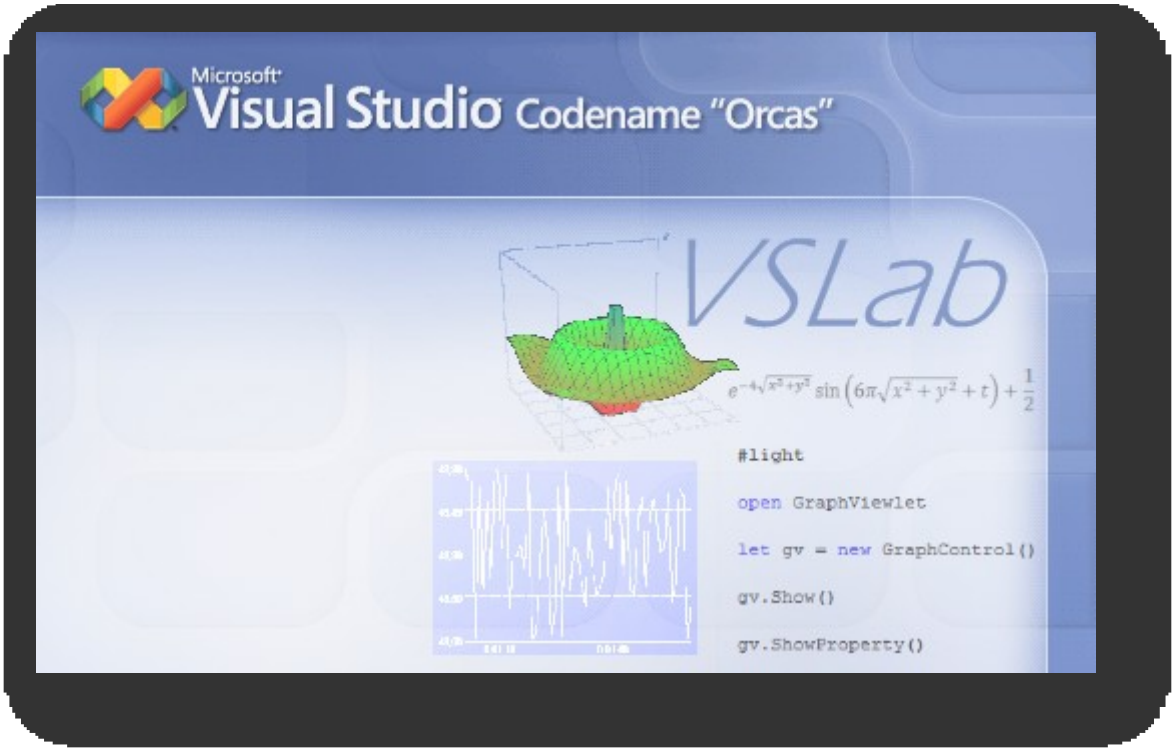
VSLab Visual Tutorial

Whitepaper



In this paper show how to use VSLab using screenshot. It is the quickest way to make VSLab do something interesting.

Author: Antonio Cisternino (cisterni@di.unipi.it), University of Pisa
Version: 1.1
Last update: 7/16/2008 12:25:00 PM

Introduction

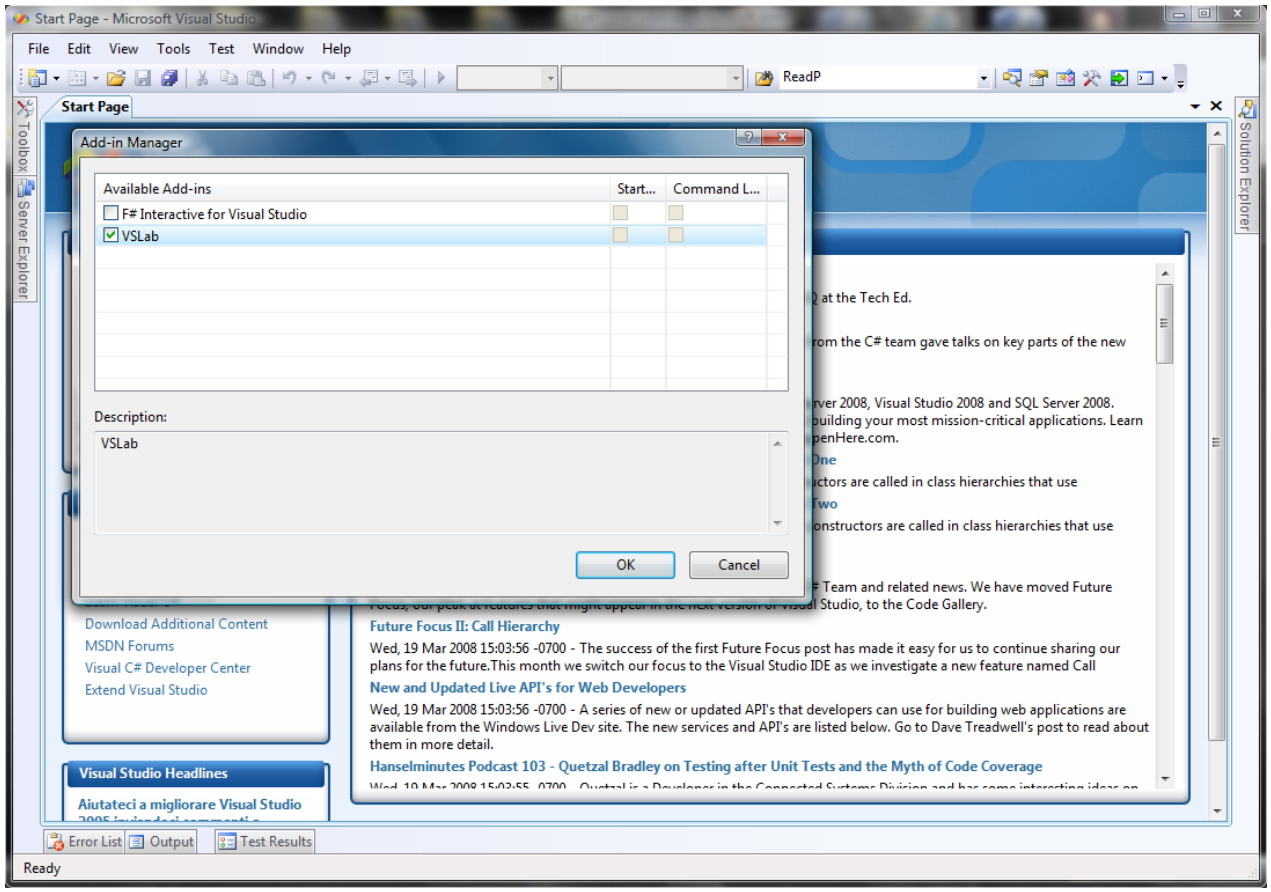
The easiest way to look at VSLab is a powerful version of F# interactive that allows opening Visual studio toolwindows and interactively drawing inside them. VSLab provides several facilities to create and manage *Viewlets*, Visual studio toolwindow that are interactively updated by F# functions. I think that the implementation is worth to study because *fsi.exe* run in a separate process and it isn't trivial at all to convince VS to accept efficient drawing from an external process. In fact VSLab has been developed to be an example of DTE use, the Visual Studio extensibility, and F# was the natural candidate for exploiting this power because of its interactive abilities. More technical documents will follow, for the moment let me introduce VSLab in the quickest way (assuming that you know a little bit of F#).

Installation

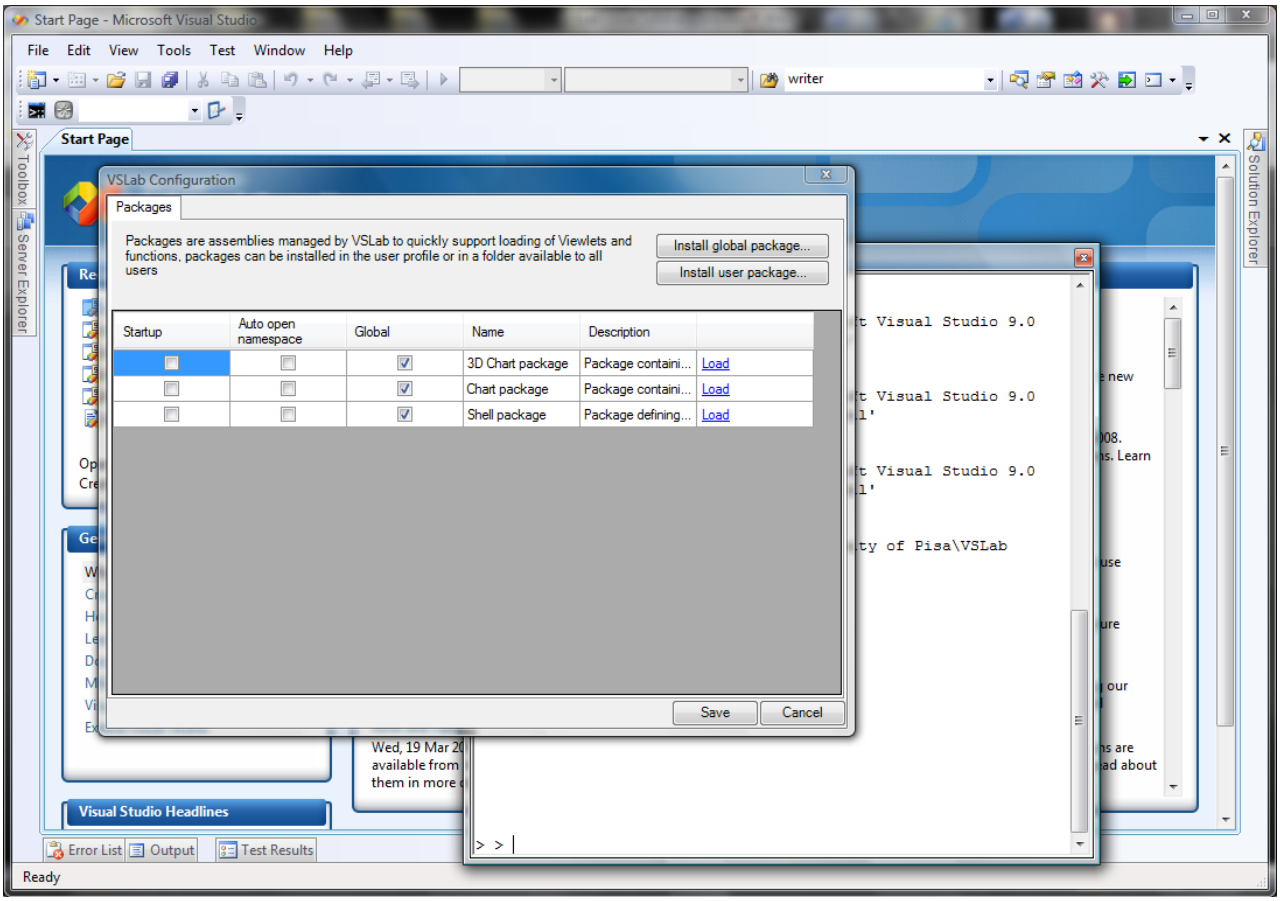
The current release of VSLab is codenamed R. Daneel and it is considered version 1.2.1. Installing VSLab is easy and can be done free of charge. Installation requires:

- Visual Studio Shell integrated mode (or Visual Studio 2008)
- Managed Direct-X (only if you want to run the 3D function viewer viewlet)
- F# (and VS integration)

The installer checks for prerequisites and then you simply next enough times to get VSLab installed (if you are interested in the options about *vslabfsc* and installed packages read the next section). When you start Visual Studio you create a VSLab project (or an F# project) and you simply load the VSLab add-in by opening Tools -> Add-in Manager and select VSLab. During the first load the add-in registers a number of Visual Studio commands that let you start VSLab by simply hitting Alt+Enter to evaluate a selected code fragment.



When VSLab addin is selected the VSLab version of F# interactive starts and the VSLab toolbar appears on the Visual Studio window. Clicking on the configuration button (the latest one) you get the VSLab configuration dialog that at the moment contains only the package configuration.



Using this dialog it is possible to load packages interactively (it is just a shortcut to #r directive). We suggest to mark the basic packages as loaded at startup and then click save. You can now either restart Visual Studio or re-open the configuration dialog and manually load the packages and you are ready to play with VSLab!

Installation options: a quick overview

During installation you can choose if override *fsc.exe* or not with a wrapper for the F# compiler; this wrapper is meant to ensure that .vslab files aren't compiled in a VSLab project. It is a hack because VSLab project files are in fact based on the F# project library which is not under our control.

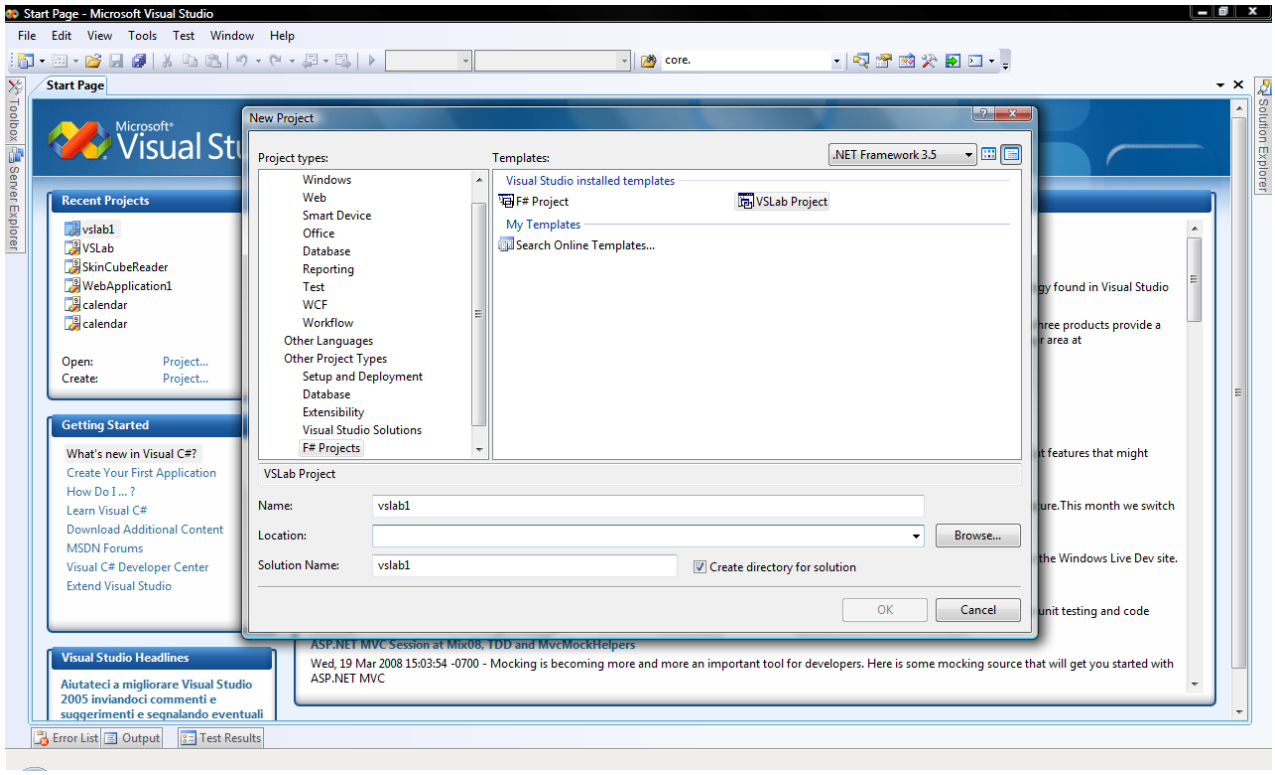
Another option is to decide which packages to install among those available. In the current version there are three packages:

- *VSLabChartPackage*: containing the graph viewlet, a plotter of values annotated with time
- *VSLabChart3DPackage*: containing the *Function3DViewlet* the plotter of 3D functions
- *VSLabShellPackage*: containing the *PerfMonViewlet* to monitor VS CPU and memory loads.

These packages are almost empty but they represent the beginning of the standard package distribution of VSLab; it is recommended to install all of them.

Getting started: a new VSLab project

We have defined a new F# project type called VSLab, it is an F# project and it is simply used to introduce .vslab files which are files that are added to the project but not compiled with the *fsc* compiler is invoked.

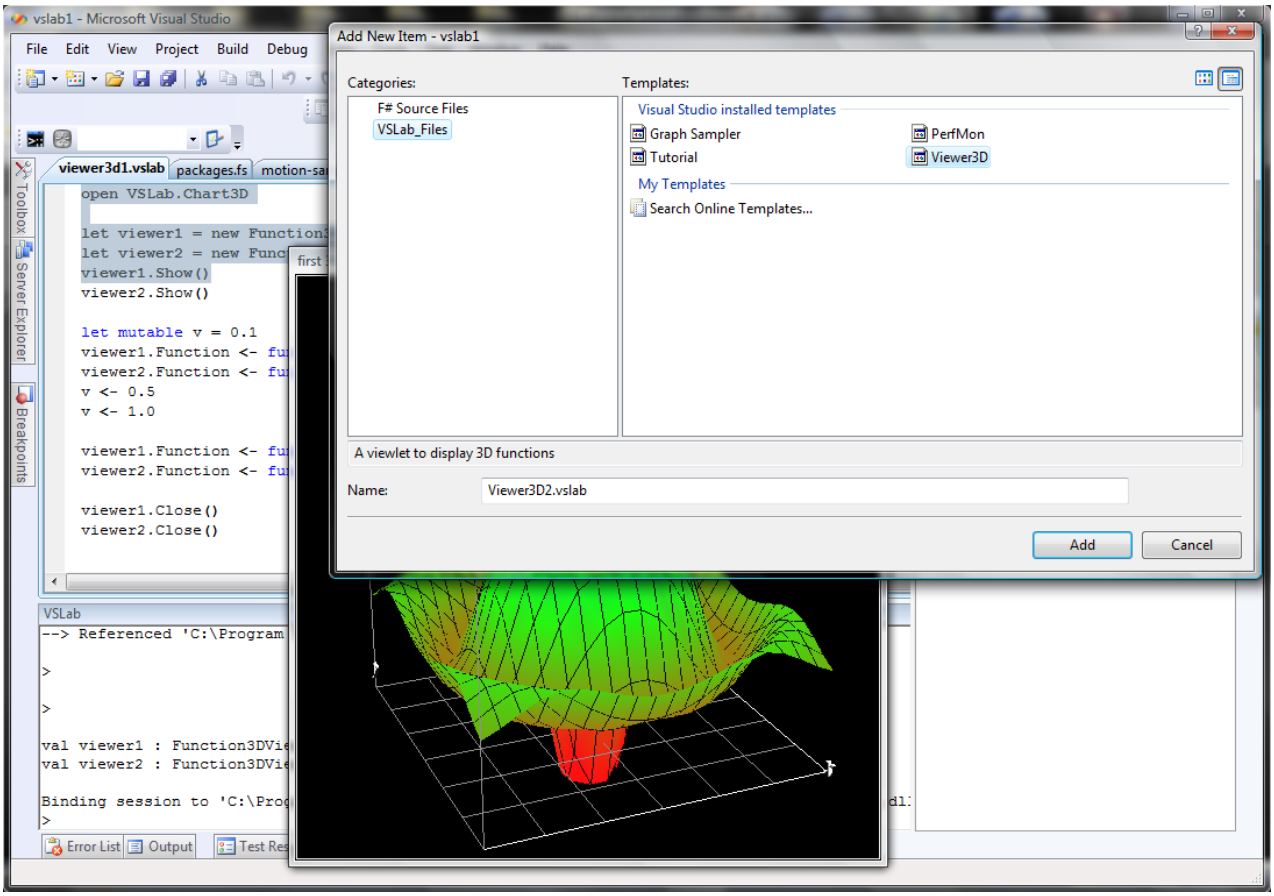


The rationale for this is that *.vslab* files act as a sort of whiteboard where you can experiment with code snippets without committing to use them in the final program. Recall that the overall goal of the project that is to have MatLab-like interaction in Visual Studio, therefore VSLab has been designed with experimentation in mind. However, we also support code consolidation that this a simple form of extrusion to move snippets into *.fs* files to be compiled.

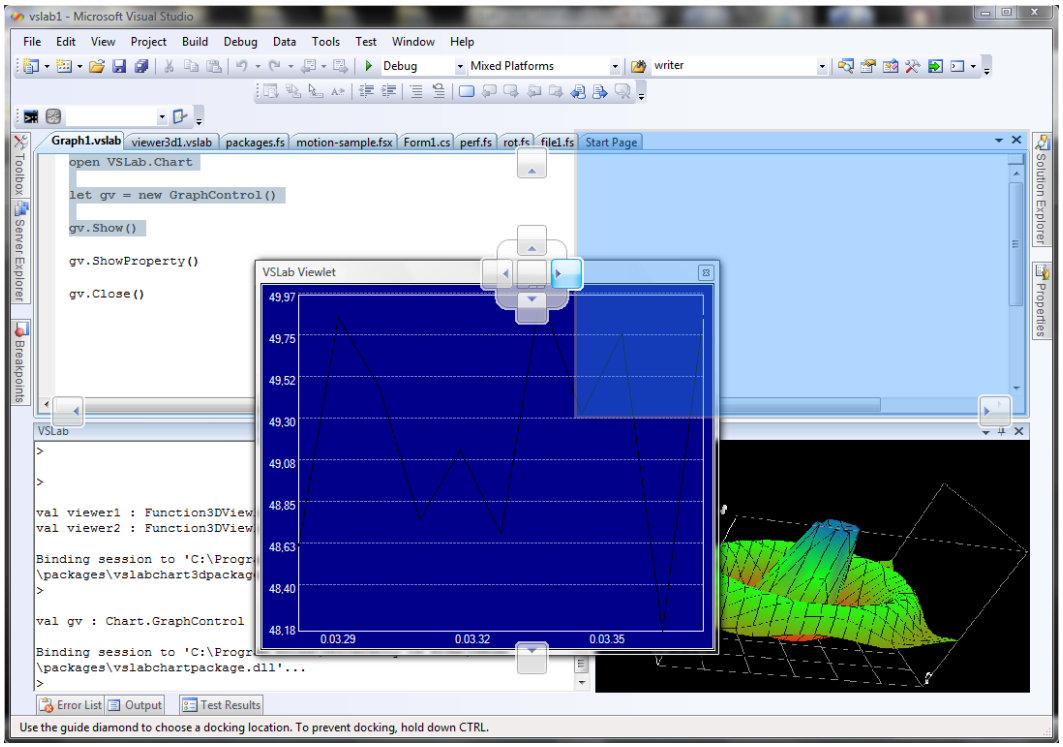
Viewlets

When you add a new file to your VSLab project you will notice a new category that allows you to choose among VSLab templates, a sort of code snippets to quickly insert viewlets in your project. Currently there are four templates:

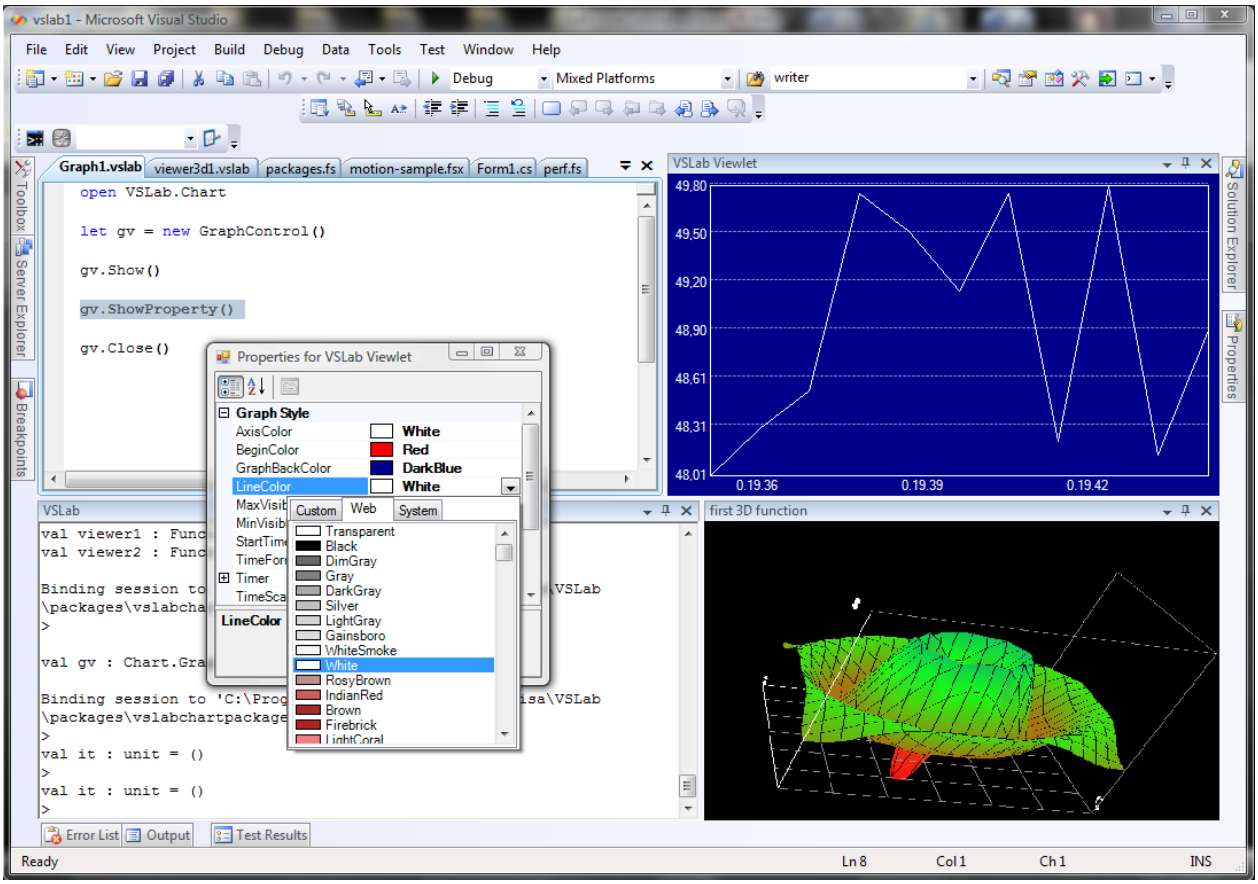
- Viewer 3D: a modified version of the wonderful DX example developed by Don Syme
- Graph sampler: a highly configurable graph control that plots data annotated with timestamp
- Performance monitor: a sample written for a Whitepaper that shows CPU and memory load of VS and F# interactive
- Tutorial: a set of useful VSLab code fragments showing features of the system



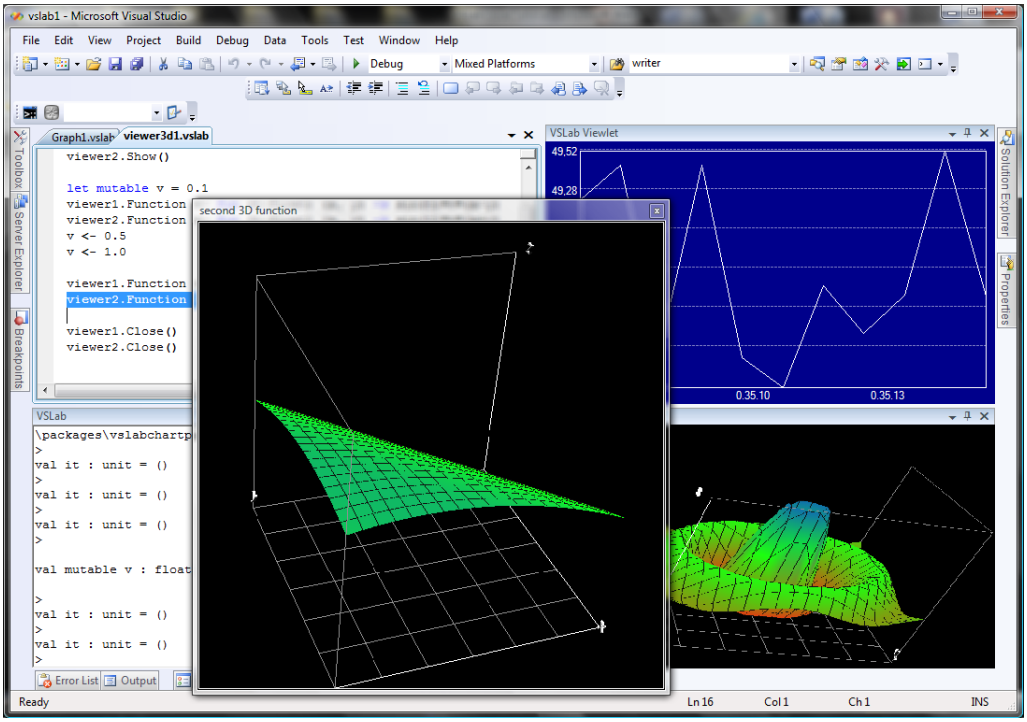
I would suggest you to start from the 3D viewer, with VSLab interactive loaded select the code from the beginning of the file until the line **viewer1.Show()** and evaluate it using *Alt+Enter*. Now add a *Graph sampler* item to your project and evaluate until the invocation of the **Show** method.



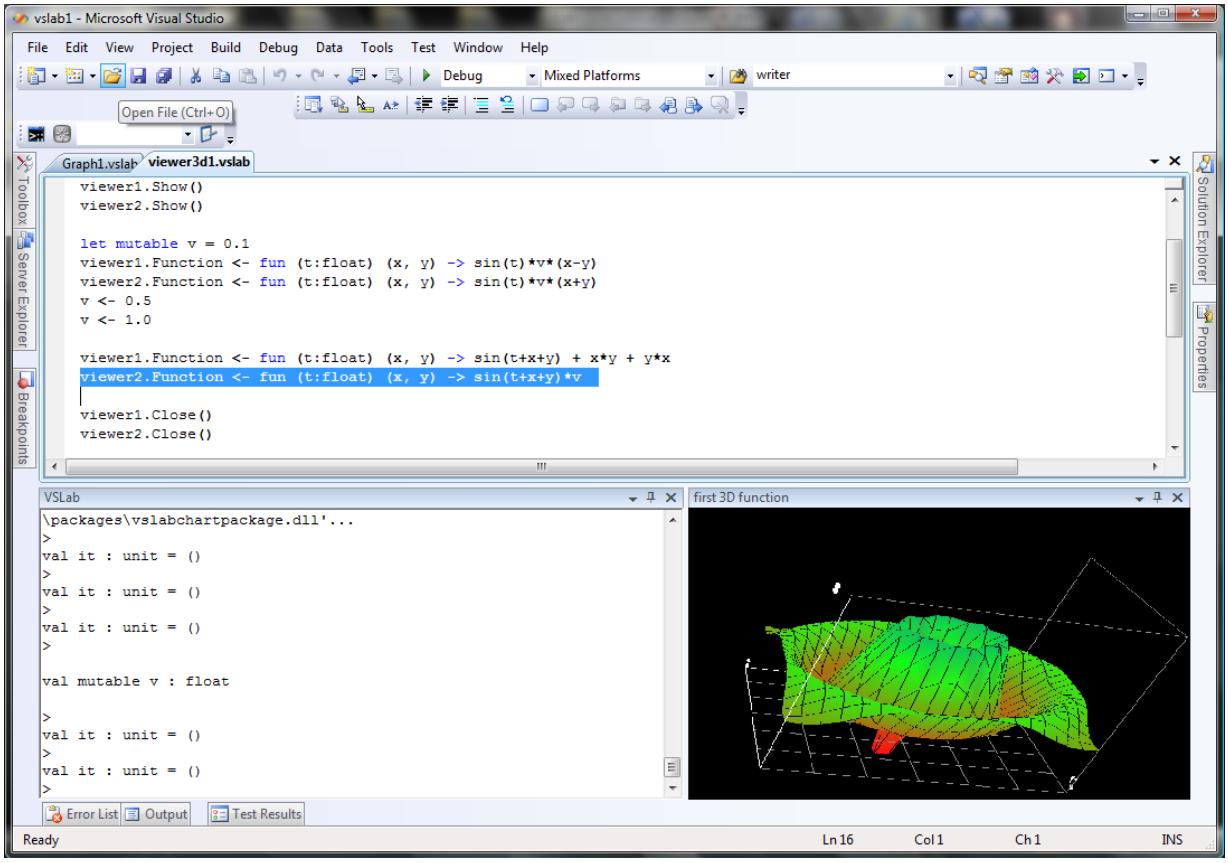
If you evaluate the **ShowProperty()** line you get the property grid associated with the viewlet and you can dynamically configure your viewlet while it is running!



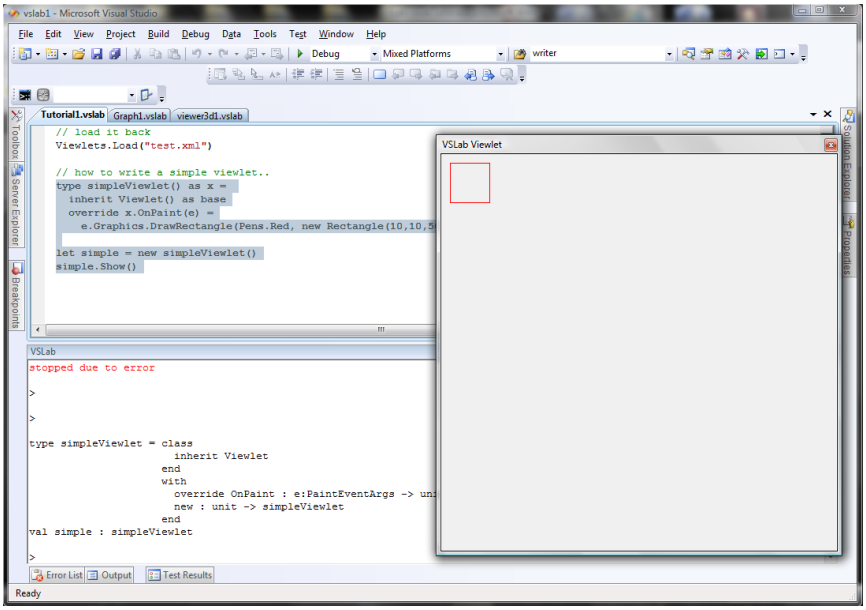
Viewlets can be instantiated multiple times, let's add another 3D viewer:



When you click **x** on a toolwindow it gets hidden, and you can restore hidden viewlets using the VSLab toolbar:



Using the same toolbar you can also show and hide the F# interactive toolwindow.
You can even dynamically define viewlets without any need for compilation:



Editor

After experimentation you will feel the need for consolidation of code fragments. If you select text and use right click you get the opportunity of sending code fragments to *.fs* files in the project that can be used to compile the final application. VSLab files remain as a memory of the process that has generated the final code.

Things you must know

VSLab does several hacks to make F# interactive interoperate with Visual Studio (for robustness they ran in different processes). Many of the hacks are restricted to VSLab and does not affect your installation with few exceptions:

- *fsc.exe* has been wrapped to ignore .vslab files when compiling from Visual Studio. The wrapper invokes *fsc.exe* after pre-processing the command line. If you run *fsc.exe* you get the instructions to disable this behavior (though I didn't find any issue so far). You can now decide wether to wrap or not *fsc.exe* during installation.
- *Alt+Enter* key binding starts VSLab addin running F# interactive instead of the original addin that ships with F#. It uses exactly the same objects thus it is not a problem.

When *fsi.exe* crashes or you quit getting a fresh instance you can use a button on the toolbar to invoke the startup code that setup the VSLab environment.

Trust the installer; it has been carefully developed and removes all the additions made during uninstall.

Acknowledgments

I wish to thank all the people that have contributed to this project: my students Davide Morelli and Sara Berardelli in the first place that have made possible the core mechanisms and the 3D viewlet respectively. Cristina Nardini and Emanuele Arpini from Microsoft that supported the idea, and Don Syme that developed such a beautiful piece of code (F #and the 3D viewer), and supported me during VSLab development. I also thank all other people that are not mentioned here but that have contributed in many ways.