SPE

Release 0.4.2

Stani's Python Editor

Python IDE with Blender and wxGlade support

Stani Michiels

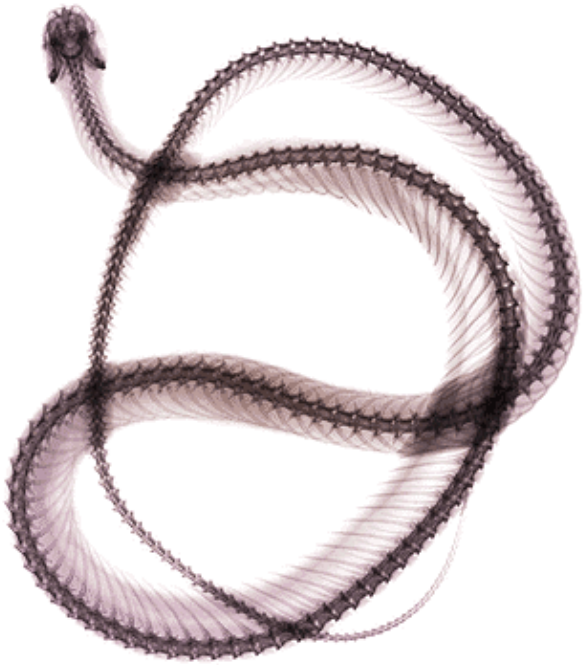


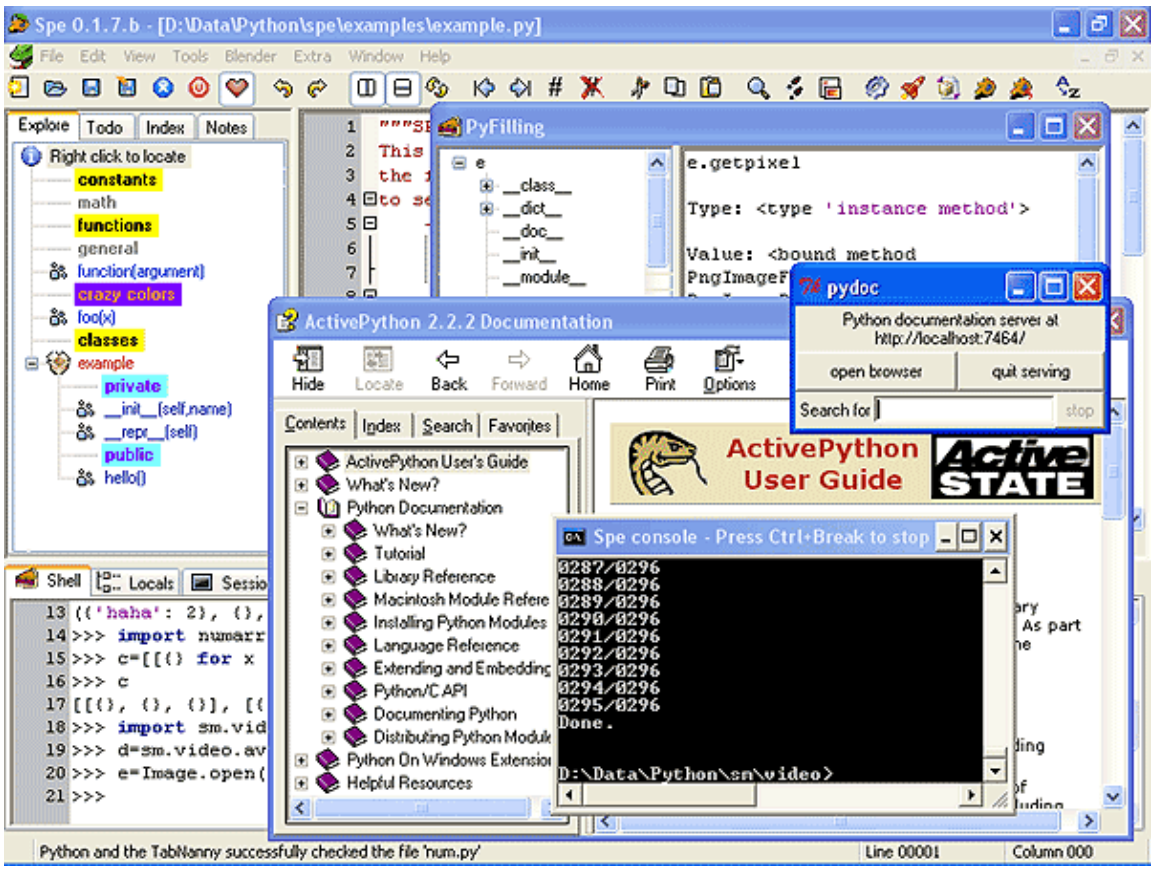
Table of Contents

1 Introduction

1.1 Spe

Stani's Python Editor with wxGlade and Blender support.

Spe is a python IDE with auto indentation, auto completion, call tips, syntax coloring, syntax highlighting, class explorer, source index, auto todo list, sticky notes, integrated pycrust shell, python file browser, recent file browser, drag&drop, context help, ... Special is its blender support with a blender 3d object browser and its ability to run interactively inside blender. Spe is extensible with boa or wxGlade.



1.2 BlenPy

BlenPy = Blender + python

Access python tools from within Blender & python browser. Blenpy provides an interface to script actions (edit, reference, run and import) and to a variety of object browsers for python in Blender. It integrates existing python development technologies like Idle, PythonWin, Pycrust, Boa, Leo, ... in an extensible framework. Besides it provides a (unfinished) module to ease the creating of gui's for python. As such this package is aimed at python developpers. Requires Blender and full python.

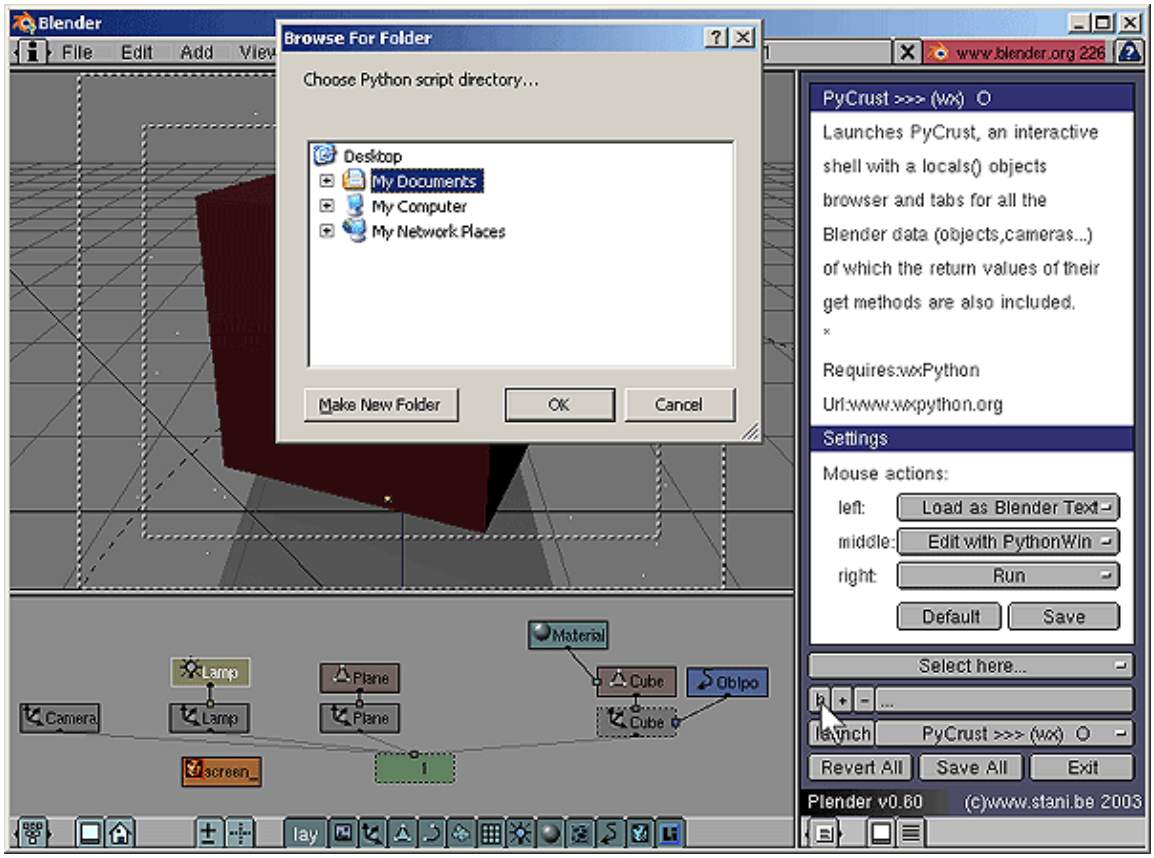
Strongly recommended:

◊ PythonWin extensions (already included in ActivePython)

◊ wxpython (www.wxpython.org)

Optional:

◊ Boa Constructor
◊ Leo
◊ SciTE
◊ ...



1.3 Internet links

- Homepage: http://spe.pycs.net
- Website: http://projects.blender.org/projects/spe/
- Screenshots: http://spe.pycs.net/pictures/index.html
- Mailing list: http://www.blender.org/mailman/listinfo/spe−user
- Forum: http://projects.blender.org/forum/?group_id=30
- RSS feed: http://spe.pycs.net/weblog/rss.xml
- Report bugs: http://projects.blender.org/tracker/?atid=193&group_id=30&func=browse
- Changelog: http://spe.pycs.net/stories/1.html

2 Requirements

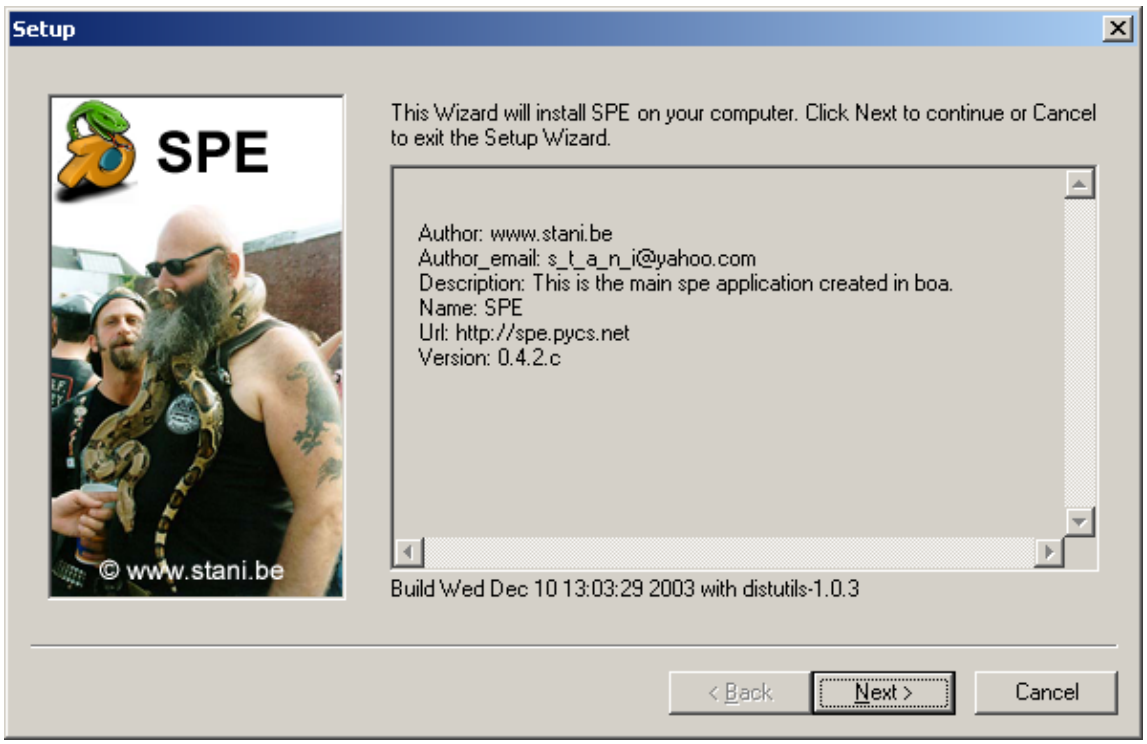
- *Python 2.2:*
  I recommend ActivePython distribution because of its excellent helpfiles: http://www.activestate.com/Products/ActivePython/index.html Spe also will work with Python 2.3, but not inside Blender, which requires Python 2.2
- Wxpython 2.4.1.2 (optional for blenpy)
- Blender 2.31 (optional for spe)

3 Installation

3.1 Windows

Installation:

Spe should be installed with the windows binary. It will probably install Spe in C:\PythonXX\lib\site−packages. Afterwards, you can register 'Edit with spe' in the context menu of Windows and create shortcuts, by running the script spe/winInstall.py



Removal:

If you registered spe in windows explorer context menu or created shortcuts, you have to remove them first with the script spe/winUninstall.py Afterwards, go to Control Panel > Add/Remove Programs and uninstall spe (probably listed with Python).

3.2 Linux, FreeBSD, Mac Os X, ...

Installation:

Run the 'setup.py' script. This will install spe and blenpy in the standard library directory of python:

```
/usr/local/lib/pythonX.X/site−packages
```

A wrapper script called 'spe' will be installed to PREFIX/bin. If necessary add PREFIX/bin to your PATH environment variable. PREFIX is determined by the install location of the modules, i.e. for the above PREFIX=/usr/local.

When spe is launched in Blender, what might be missing in the PYTHONPATH, is

/usr/local/lib/python2.2/site−packages. If you add this one in your .bashrc/.tcshrc/... to the PYTHONPATH variable everything should be fine (the subdirs spe,sm,etc. aren't needed). Though you must start blender from a bash − e.g. desktop menus usually don't read the .bashrc/.tcshrc/... and therefore blender does not know about your user defined environment variables. If you set the PYTHONPATH in /etc/profile instead of .bashrc/.tcshrc/... then starting spe/blenpy from blender will work also from menus.

***Removal**:*

Just run the spe/linuxUninstall.py script (use it at your own risk!) or by removing manually the directories spe, blenpy and sm of the standard library directory:

```
/usr/local/lib/pythonX.X/site−packages
```

One needs also to remove /usr/local/bin/spe manually.

4 SPE

4.1 Start SPE

- *autonomous*
  Windows:
  Open the spe folder and type 'python spe.py' at the command prompt or make a shortcut to your desktop.
  Unix (Linux, FreeBSD, ...):
  Type 'spe' on the commandline (assuming PREFIX/bin is on your PATH)
  Debugging mode:
  If you have problems starting up spe, type at the command prompt 'python spe.py −−debug' and send me the error message.
- *inside Blender:*
  Open spe.blend and press Alt+P in the corresponding text window.

4.2 Features

- *Sidebar:*
  - class/method browser (jump to source)
  - automatic todo list, highlighting the most important ones (jump to source)
  - automatic alphabetic source index of classes and methods (jump to source)
  - sticky notes
- *Tools:*
  - interactive shell
  - locals browser (left click to open, right click to run)
  - seperate session recording
  - quick access to python files in folder and its subfolders (click to open)
  - unlimited recent file list (left click to open, right click to run)
  - automatic todo list of all open files, highlighting the most important ones (jump to source)
  - automatic alphabetic index of all open files (jump to source)
- *Python:*
  - syntax−checking
  - syntax−coloring
  - auto−indentation
  - auto−completion
  - call−tips
- *Drag&Drop:*
  - drop any amount of files or folders...
  - ... on main frame to open them
  - ... on shell to run them
  - ... on recent files to add them
  - ... on browser to add folders
- *General:*
  - context help defined everywhere
  - add your own menus and toolbar buttons
  - exit & remember:all open files will next time automatically be loaded (handy for Blender sessions)

- wxpython gui, so should be cross−platform (Windows,Linux,Mac)
- scripts can be executed in different ways: run, run verbose and import
- after error jump to line in source code
- remember toggle: remembers open files between sessions (heart icon on toolbar)

• *Blender:*
- redraw the Blender screen on idle (no blackout)
- Blender object tree browser (cameras,objects,lamps,...)
- add your favorite scripts to the menu
- 100% Blender compatible: can run within Blender, so all these features are available within Blender

• *Windows:*
- spe registers itself in the windows explorer context menu
- optional creation of desktop and quick launch shortcuts

4.3 Information

It is recommended to check out all the context help, to get familiar with the features of spe. Some information which didn't fit there, comes here:

• *Blender:*

To run spe from within Blender, just type:

```
import spe
spe.main()
```

...and press Alt−P When spe is active, the Blender screen will always be redrawn automatically. So the results of any command you type in the interactive shell or of any program you run within spe, will be visible in the Blender window. Unfortunately it is not possible to interact with Blender directly when Spe is active. So it is impossible to rotate for example the view with the mouse. (Maybe one time the python development team could solve this issue and let run blender and spe in parallel. However this might be an illusion.)

• *Psyco:*

If you don't know the python psyco module, you can ignore this item, as it won't have any effect for you. Psyco programs can't run in spe, as they disable the 'locals()' function. Of course you can edit programs using psyco in spe, but if you want to run them, comment the psyco activation code out.

• *Refreshing:*

Spe has a lot of features like explore tree,index,todo list, and so on... This gets updated every time the file is saved or every time the refresh command is given. This can be done by pressing F5 on the keyboard, the refresh toolbar button or clicking the View>Refresh menu.

• *Remember option:*

This can be activated by checking File>Remember or by pressing the heart toolbar button. It will open automatically the scripts which were open in the last session. Useful for Blender if you have to switch continously between Blender and spe.

• *Running files:*

- *Run (F9):*

  Use this by default, unless you have specific reasons to use the other ones. It will run in the namespace of the interactive shell. So all the

objects and functions of your program become available in the shell and in the locals browser (the tab next to the shell).

- *Run with profile (Ctrl−P):*
  Same as above but with a profile added. A profile is a report of the program execution which shows which processes or functions are time consuming. So if you want to speed up your code, you can define the priorities based on this report.
- *Run in seperate namespace (Ctrl−R):*
  Like run, but all the objects and functions defined by the program will not become available in the namespace of the interactive shell. Instead they will be defined in the dictionary 'namespace' of the interactive shell. So if the file 'script.py' is run in this way, type namespace['script.py'] in the shell, to access this dictionary, or namespace['script.py'].keys() to get a list of all defined names, or namespace['script.py'].items() to get tuples of all the names and their values. More easy is to just browse 'namespace' in the locals browser.
- *Run verbose (Alt−R):*
  This is for very simple programs, which do not indent more than once. It will send all source lines, as if they were typed in the interactive shell. It is probably a good learning tool for beginners.
- *Import (F10):*
  Imports the source file as a module. For running files, they don't have to be saved. For importing files, it is recommended to save them first.

- *Syntax−checking:*
  Every time you save spe does syntax checking. If there is any error, spe will jump to the line in the source code and try to highlight the error.

4.4 Customize menus and toolbar

You can define your own menus and toolbar buttons, which can execute any python code and also external files. Look at 'framework/menus/Extra.py' for an example. Instructions:

1. Suppose you want to add a new menu with the name 'XXX' to the menubar. Create a new file with the name XXX.py in the 'framework/menus/' directory
2. Import or define some actions, with the following structure:

```
def action(script,app,event):
    ...
```

The arguments of the function are:

```
script      −> current script window
app         −> application window
event       −> event
```

Some usefull stuff:

```
script.fileName
script.source
  script.source.GetText()
  script.source.SetText(text)
  script.source.GetSelectedText()
```

```
  script.source.ReplaceSelection(text)
app.run(fileName)             -> runs an external file
app.new()                     -> creates a new file
app.open(fileName,lineno,col) -> opens a file at given position
app.message(text)             -> shows a dialog window with the
                                 text
app.messageEntry(text)        -> shows a dialog prompting an
                                 entry
app.messageError(text)        -> shows an error dialog window
                                 with the text
app.SetStatus(text)           -> sets the status text
```

3. Define the 'main' function:

```
def main(app):
    menu(app,
        item(label=<str:label that will appear in menu>,
             action=<function:that will be called>),
        item(...),
        SEPARATOR,
        item(...),
        item(...),
        ...
        SEPARATOR,
        item(...),
        ...
    )
```

   If you want this menu item also to have a toolbar button, than make a 16x16pixels png image (transparency is allowed). The image has to be located in the menu folder. Pass the the fileName with the toolbar keyword:

```
item(label=<str:label that will appear in menu>,
    action=<function:that will be called>,
    toolbar=<str:fileName of the toolbar image (optional)>)
```

4.5 Separators

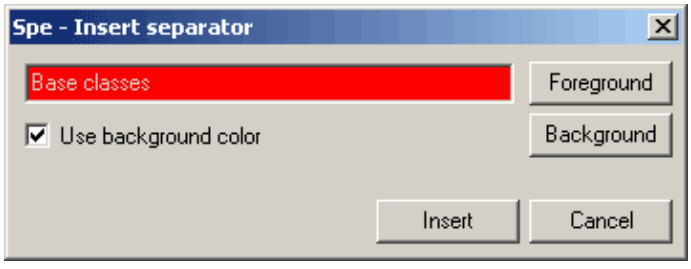
A separator is a label which appears in the explore tree of the sidebar to help structuring the script. An easy way to add separators is to use the 'Edit'>'Insert colored separator' wizard from the menu.

Syntax:

- ◊ normal: `#---label`
- ◊ coloured: `#---label---#foregroundcolour#backgroundcolour`
- ◊ highlighted: `# # # #label (4 times #, without spaces)`

Foreground and background colour are in html notation, eg:

red on blue label: #---red on blue---#FF0000#0000FF



4.6 Shortcuts

If you want to change the default keyboard shortcuts, open the file spe/framework/shortcuts.py and adapt it to your own taste. Backup this file so that when you install a new version of spe, you can copy it back.

4.7 Known Issues

- Undo might sometimes takes big steps back.
- editors.txt describes possibly installed editors. It may not work 'out of the box' for many people. If so adapt the file to your system.
- If spe fails to start up, after it worked well before. Delete all the files in the spe settings path.

5 BlenPy (Blender only)

5.1 Start blenpy

Open spe.blend and press Alt+P in the corresponding text window. Above settings you'll find a description of the current tool and in settings you can change the mouse configuration. Important: before using the blenpy gui, save your work. Sometimes blenpy might block/crash Blender. If you want to use the tools in another drawing, just type the following in a Blender Text window:

```
from blenpy.pyGui import PythonGui
PythonGui()
```

5.2 Modules:

- *blenpy.pyGui: blender python gui*
  This user interface attempts to integrate as tight as possible existing python editing and introspection tools (Idle, ObjectBrowser, Pycrust, PythonWin, Leo, SciTE...) in Blender. It targets people with yet some python experience. It aim is make developping python programs for Blender more easy. The screenshots might gave a good impression. To fully profit from the benefits of this package, the installation of Wxpython is recommended.
- *blenpy.gui: blender gui*
  A module to ease the making of gui's (graphical user interface) in Blender, wrapping the most important functions from the Blender BGL and Draw Modules. Features included:relative layout positioning, drawing filled rectangles with optional borders and automated registration of keyboard and button events. Supported are Buttons, Menus, Strings and Texts, rest will follow later. blPyGui was built upon blGui.

5.3 Manual

- *Tools:*
  Next to the launch button you can select a tool from the menu. If you select a tool, the description will appear above. Before launching the tool, make sure you have installed the required modules. Tools which have (tk) depend on Tkinter and tools which have (wx) on Wxpython. You don't have to install Tkinter as it is part of the standard python distribution. If extra modules are required, it is displayed in the description. If you have Wxpython installed, check out 'PyCrust >>>'. Press the launch button to launch the Tool. When working with the tool, the Blender application is not accessible (probably turns black). You can only return to Blender by closing ALL the tool windows.

  You can add your own tools by editing pluginstools.py If this tool could be interesting for other users, please send me than a copy. So I can include it in the next release.
- *Browser:*
  In the mouse configuration: you can define actions for the mouse buttons. There are now 3 types of actions:editing(see more down), run and import. 'Run' executes the file and returns the exitcode (see Blender console). 'Import' imports the file as if it was a module. Reference is an interesting option if you want to edit the script outside Blender. It doesn't import the source code into Blender, but creates a python shorcut

script which will execute the saved file on the disk when you press Alt+P.

The 'Select here...' button display a path menu. By the default the current drawing directory is added as the 'working directory'. To add a directory, press the 'b' button, which will open a directory select dialog. Select the directory and press 'ok'. The directory is now added to the 'Select here' menu. (If you don't have tkFileDialog installed on your system, type the name in the field which contains '...'. Then press the '+' button. The directory is than added.)

Before proceeding, remember the configuration of the mouse actions. If you now select a path with the 'Select here...' menu, a table with all python files of that path and all subpaths will be displayed. By clicking above a file, the mouse action will be triggered.

With the '−' button you can remove the active directory from the path menu.

- *Editing strategies:*
  1. *'Edit with ...':*
     This provides an editor, from which programs can be immediately run in Blender. It also provides a shell which can be used to try out python in Blender interactively. The Shell has access to the Blender Namespace, which makes it possible to type 'import Blender' in the shell and than to use any Blender python command. Spe is probably the best solution with its auto−indentation, calltips and auto−completion. If you want to switch continously between Blender and the editing sessions, turn on the remember toggle on the toolbar (heart icon). This saves you from opening always all the files again. PythonWin (windows only) can only be used once.
  2. *'Reference' & 'Send to ...':*
     These two commands are very usefull together. 'Reference' doesn't load the source code in a Blender Text window, but places a short python code to run the script from the file. 'Send to ...' opens the file in an external editor. So you can edit your file and save it. To try the result immediately, just press Alt+P in Blender on the python shortcut.
  3. *'Load as Blender Text':*
     The old way of using python in Blender. It loads the source as a Blender Text, adding a comment on the first line: `#@filename`

     This comment is recognised by the 'Revert All' and 'Save All' buttons and will automatically load/save the source from/to these files. The 'Save All' button won't save the first comment line in the file on disk, so there is no need to remove it from blender. To exclude a file from to the 'Revert All' and 'Save All', just change '#@' in '#'. The 'Revert All' and 'Save All' can be combined with external editing through 'Send to ...' but it can be confusing to have the same file twice opened. It has the risk of overwriting a new file with an old file.

     The Blender Text window is a but 'unpythonic', as it doesn't support auto−indentation, calltips or auto−completion. Its clipboard operations (Cut Alt+X, Copy Alt+C and Paste Alt+V) are only working within Blender and do not interact with other applications. Therefore above strategies are preferable.

- *Adding editors:*
  You can add your own editors by editing pluginseditors.py for internal and pluginseditors.txt for external editors. If this change could be interesting for other users, please send me than a copy. So I can include it in the next release.
- *Notations:*
  '>>>':
  This means that the tool includes an object Browser with all objects in the drawing, of which all 'get' methods are recursively evaluated (see the '>>>' entries). In this way it is possible to see eg of an [Object 'Camera'] automatically its data by '>>> getData()', which could be [Camera 'Camera']. Although this can be very usefull, it can be too clumbersome for bigger drawings. So it is better to limit its use to light drawings.
  'Light':
  This means that the tool doesn't preload the Blender module and its object browser.
- *Customizing:*
  Power users can customize files like mouse.py, tools.py and editors.txt in the plugins folder. If you do, please if you changed these files please send me a copy, so it can be updated for the next release and all users can profit from it.

5.4 Appendix

- Files:

```
gui         General module for creating gui's
pyGui               User interface for python_ devellopers
doc\                Documentation
...
plugins\            You can edit these to add other editors or tools:
editors.txt External editors (Send to)
editors.py  Internal editors (Edit)
mouse.py        Mouse button actions (such as run, import,...)
tools.py            Tools
...
sm\         Blender_ independent modules
tk*         Tkinter specific
wxp*                Wxpython_ specific
```

- *Todo:*
  · Better documentation
  · Finish the gui module
  · ...
- *Known issues:*
  · Boa Constructor: can only run if you disable all Zope features.
  · pydoc: This will crash Blender. Don't use it, unless you know what you are doing.
  · PythonWin: Edit only works once within one Blender session.
- *Bugs in the Blender API:*
  One of these errors might appear in the Blender console, but you can safely ignore them:

  Internal error, Invalid prop entry Trying to access the attribute 'starColNoise' of the World class gives this error statement.

ipo member access deprecated,use self.getIpo() instead! Trying to access the attribute ipo gives this error statement.

6 Contact

Spe is still under development. If you use Spe, please post a message on the appropiate forum on http://projects.blender.org/forum/?group_id=30 describing the platform, the problems that occur and possible solutions if you know. If Spe runs without any problems, I'm also interested to get a notice. I developped spe under windows xp and have no access to Linux, Mac, FreeBsd or any other platform. So any help for these platforms is highly appreciated. If you would like to contribute to spe in any way, send me an email with your skills (programming, graphics,icons,3d,html,...) and I'm sure you can help me out.

If you want to get the last version, which is not yet released publicly, just send me an email.

7 Credits

Thanks to the following components Spe was made possible:

- *Blender*
  - 3D modeling, rendering, animation and game creation package
  - Copyright 2003 Blender Foundation − Ton Roosendaal
  - http://www.blender.org
- *Boa Constructor*
  - a python IDE and Wxpython GUI builder
  - Copyright 1999−2003 Riaan Booysen
  - http://boa−constructor.sourceforge.net
- *HtmlDoc*
  - Html document processing to Pdf or PostScript
  - Copyright 1993−2003 Easy Software Products
  - http://www.easysw.com/htmldoc/
- *Kiki*
  - free environment for regular expression testing (ferret)
  - Copyright 2003 Project 5 − Andrei
  - http://come.to/project5
- *PyChecker*
  - a python source code checking tool
  - Copyright (c) 2000−2001, MetaSlash Inc.
  - http://pychecker.sourceforge.net/
- *Pycrust*
  - the flakiest python shell (Patrick K. O'Brien)
  - Sponsored by Orbtech − Your source for python programming expertise.
  - http://www.wxPython.org
- *Pycs*
  - python community server
  - Copyright 2002 pycs
  - http://www.pycs.net
- *Pyds*
  - python desktop server
  - Copyright 2002 Georg Bauer
  - http://pyds.muensterland.org
- *Pyframe guide to Wxpython*
  - Documentation about wxStyledTextCtrl
  - Copyright 2003 Jeff Sasmor
  - http://www.pyframe.com/wxdocs/
- *Pythonwin*
  - python IDE and GUI Framework for Windows
  - Copyright 1994−2003 Mark Hammond
- *Scintilla*
  - Copyright 1998−2001 by Neil Hodgson
  - http://www.scintilla.org
- *Sky icons*
  - KDE icon theme made with gimp
  - Copyright 2002 David Vignoni
  - http://projects.dims.org/%7Edave/icon_sky5.html

- *wxGlade*
  - wxGlade is a GUI designer written in Python with the popular GUI toolkit wxPython, that helps you create wxWindows/wxPython user interfaces. At the moment it can generate Python, C++ and XRC (wxWindows' XML resources) code.
  - Copyright 2003 Alberto Griggio, Marco Barisione, Marcello Semboli, Richard Lawson, D.H.
  - http://wxglade.sourceforge.net/
- *wxPython*
  - python extension module for wxWindows GUI classes
  - Copyright 1997−2003 Robin Dunn and Total Control Software
  - http://www.wxPython.org

Special thanks to Tina Hirsch (Linux feedback).

Python is Copyright (c) 2000−2002 ActiveState Corp:

```
Copyright (c) 2001, 2002 Python_ Software Foundation.
All Rights Reserved.

Copyright (c) 2000 BeOpen.com.
All Rights Reserved.

Copyright (c) 1995−2001 Corporation for National Research Initiatives.
All Rights Reserved.

Copyright (c) 1991−1995 Stichting Mathematisch Centrum, Amsterdam.
All Rights Reserved.
```

This program uses IDLE extensions by Guido van Rossum, Tim Peters and others.