14.1 Advanced - Scripting & Extending Blender

Scripting & Extending Blender

- Introduction
  - General information
  - Getting Started - Manual links
  - Getting Started - External links
  - Extending Blender
- Add-ons
  - Searching
  - Enabling and Disabling
  - Installation of a 3rd party Add-on
  - File locations
  - Development guidelines
- Scripting & Security
  - Scripts in Blend Files
  - Controlling Script Execution

Introduction

Python is an interpreted, interactive, object-oriented programming language. It incorporates modules, exceptions, dynamic typing, very high level dynamic data types, and classes. Python combines remarkable power with very clear syntax.

Python scripts are a powerful and versatile way to extend Blender functionality. Most areas of Blender can be scripted, including Animation, Rendering, Import and Export, Object Creation and the scripting of repetitive tasks.

To interact with Blender, scripts can make use of the tightly integrated API (Application Programming Interface).

General information

Links that are useful while writing scripts.

- http://www.python.org/ - Python.org - General information about Python.
- http://www.blender.org/documentation/250PythonDoc/ - Blender Python API - Official API documentation. Use this for referencing while writing scripts.
- http://www.blender.org/documentation/blender_python_api_2_72_release/info_quickstart.html - API introduction - A short introduction to get you started with the API. Contains examples.
- http://wiki.blender.org/index.php/Dev:2.5/Py/Scripts/Cookbook - CookBook - A section of handy code snippets (yet to be written)

Links that deal with distributing your scripts.

- http://wiki.blender.org/index.php/Dev:Py/Sharing - Sharing scripts - Information on how to share your scripts and get them included in the official Blender distribution.
- http://wiki.blender.org/index.php/Dev:2.5/Py/Scripts/Guidelines/Addons - Creating Add-ons - Add-ons are used to encapsulate and distribute scripts.
- https://projects.blender.org/projects/bf-extensions/ - Extensions project - Project to maintain a central repository of extensions to Blender.

Getting Started - Manual links

The following links take you from the basics to the more advanced concepts of Python scripting for Blender.

- *Text Editor*
- *Python Console*

Getting Started - External links

Here are external links containing a lot of good information to start learning how to write scripts for Blender.

- http://sites.google.com/site/satishgoda/blender/learningblender25/introduction-to-blender-python-api - Introductory tutorial by Satish Goda - Takes you from the beginning and teaches how to do basic API manipulations.
- http://www.youtube.com/watch?v=vmhU_whC6zw - Ira Krakow’s video tutorials - First video in a series of video tutorials.

- http://en.wikibooks.org/wiki/Blender_3D:_Blending_Into_Python/2.5_quickstart - Quickstart guide - A quickstart guide for people who already have some familiarity with Python and Blender.
- http://blenderartists.org/forum/showthread.php?t=164765 - Examples thread - A forum thread containing many short working script examples.
- http://cgcookie.com/blender/2011/08/26/introduction-to-scripting-with-python-in-blender/ - Introduction to Python - A one hour video tutorial introducing Python and the Blender API.

Extending Blender

Add-ons

Add-ons are scripts you can enable to gain extra functionality within Blender, they can be enabled from the user preferences.

Outside of the Blender executable, there are literally hundreds of add-ons written by many people:

- Officially supported add-ons are bundled with Blender.
- Other **Testing** add-ons are included in development builds but not official releases, many of them work reliably and are very useful but are not ensured to be stable for release.

For an overview of all add-ons is available in, see Scripts Catalog and Extensions tracker.

http://wiki.blender.org/index.php/Extensions:2.6/Py/Scripts
https://projects.blender.org/projects/bf-extensions/

Scripts

Apart from add-ons there are also scripts you can use to extend Blenders functionality:

- Modules: Utility libraries for import into other scripts.
- Presets: Settings for Blender’s tools and key configurations.
- Startup: These files are imported when starting Blender. They define most of Blender’s UI, as well as some additional core operators.
- Custom scripts: In contrast to add-ons they are typically intended for one-time execution via the *text editor*

Saving your own scripts

File location

All scripts are loaded from the `scripts` folder of the *local*, *system and user paths*.

You can setup an additional search path for scripts in File Paths (*User Preferences –> File Paths*).

Installation

Add-ons are conveniently installed through Blender in the *User Preferences –> Add-ons* window. Click the *Install from File...* button and select the `.py` or `.zip` file.

To manually install scripts or add-ons place them in the `add-ons`, `modules`, `presets` or `startup` directory according to their type. See the description above.

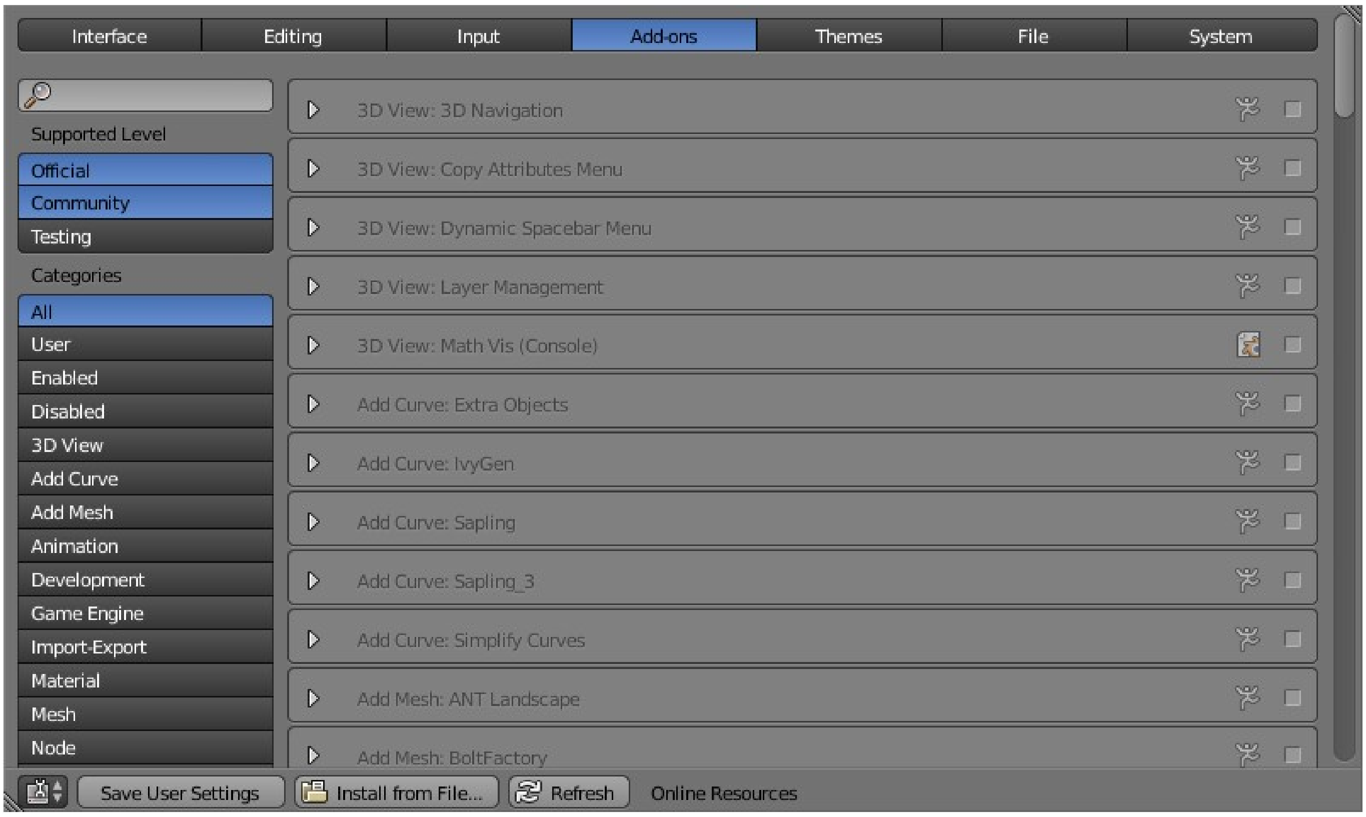
You can also run scripts by loading them in the *text editor* window.

Add-ons

Add-on is the general term for a script that extends Blender’s functionality. They are found in the *Add-ons* tab of the *User Preferences* window. This tab allows to search, install, enable and disable Add-ons.

Searching

Blender comes with some useful Add-ons already, ready to be enabled, but you can also add your own, or any interesting ones you find on the web.

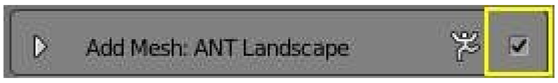


Add-ons tab in the User Preferences

The Scripts Catalog provides an index of Add-ons that are included with Blender as well as listing a number of external Add-ons.

Enabling and Disabling

Enable and disable and add-on by checking or unchecking the box on the right of the add-on you chose, as shown on the figure.



Enabling an Add-on

The add-on functionality should be immediately available. If the Add-on does not activate when enabled, check the *Console window* for any errors that may have occurred.

You can click the arrow at the left of the add-on box to see more information, such as where it is located, a description and a link to the documentation. Here you can also find a button to report a bug specific of this add-on.

Tip

Saving Add-on Preferences

If you want an Add-on to be enabled every time you start Blender, you will need to *Save User Settings*.

Installation of a 3rd party Add-on

For add-ons that you found on the web or your own to show on the list, you have to install them first by clicking *Install from File...* and providing a .zip or .py file.

Alternatively you can manually install an Add-on, which is useful when developing your own add-ons. Move or link the files to `../scripts/addons` folder (where .. is the path to your Blender configuration folder).

File locations

For information on the location of blender directories see: *Configuration & Data Paths*

You can also create a personal folder containing new add-ons and configure your files' path in the *File* panel of the *User Preferences*. To create a personal script folder:

- Create an empty folder (i.e. 'script_addon_2-7x')
- Add one folder named 'addons'. It has to named like this for Blender to recognize it.
- Put your new add-ons in this 'addons' folder.
- open the *File* panel of the *User Preferences*.
- Fill the *Scripts* entry with the path to your script folder (i.e. 'script_addon_2-7x').

Development guidelines

If you are a script developer, you may be interested in the Add-ons development guidelines

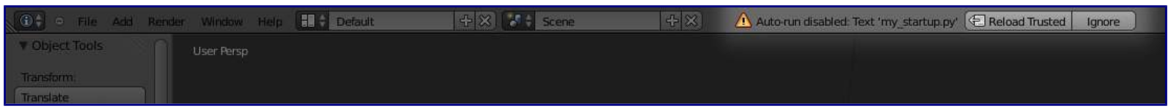
Scripting & Security

The ability to include Python scripts within blend files is valuable for advanced tasks such as rigging, automation and using the game-engine, however it poses a security risk since Python doesn't restrict what a script can do.

Therefore, you should only run scripts from sources you know and trust.

Automatic execution is disabled by default, however some blend files need this to function properly.

When a blend file tries to execute a script and is not allowed, a message will appear in the header with the option to **Reload Trusted** or **Ignore** the message.



Auto-run disabled: Text 'my_startup.py'

Scripts in Blend Files

Auto Execution

Here are the different ways blend files may automatically run scripts.

Registered Text-Blocks
A text block can have its *Register* option enabled which means it will load on start.

Animation Drivers
Python expressions can be used to *Drive* values and are often used in more advanced rigs and animations.

Game Engine Auto-Start
Scripts are often used for game logic, blend files can have *Auto Start* enabled with runs the game on load.

Manual Execution

There are other ways scripts in a `blend` file may execute that require user interaction (therefor will run even when auto-execution is off), but you should be aware that this is the case since it’s not necessarily obvious.

- Running a script in the text editor *(ok, this is obvious!)*.
- Rendering with FreeStyle - *FreeStyle uses scripts to control line styles*
- Running the Game-Engine.

Controlling Script Execution

Blender provides a number of ways to control whether scripts from a blend file are allowed to automatically execute.

First of all, the file-selector has the option **Trusted Source** which you can use on a case-by-case basis to control auto-execution.

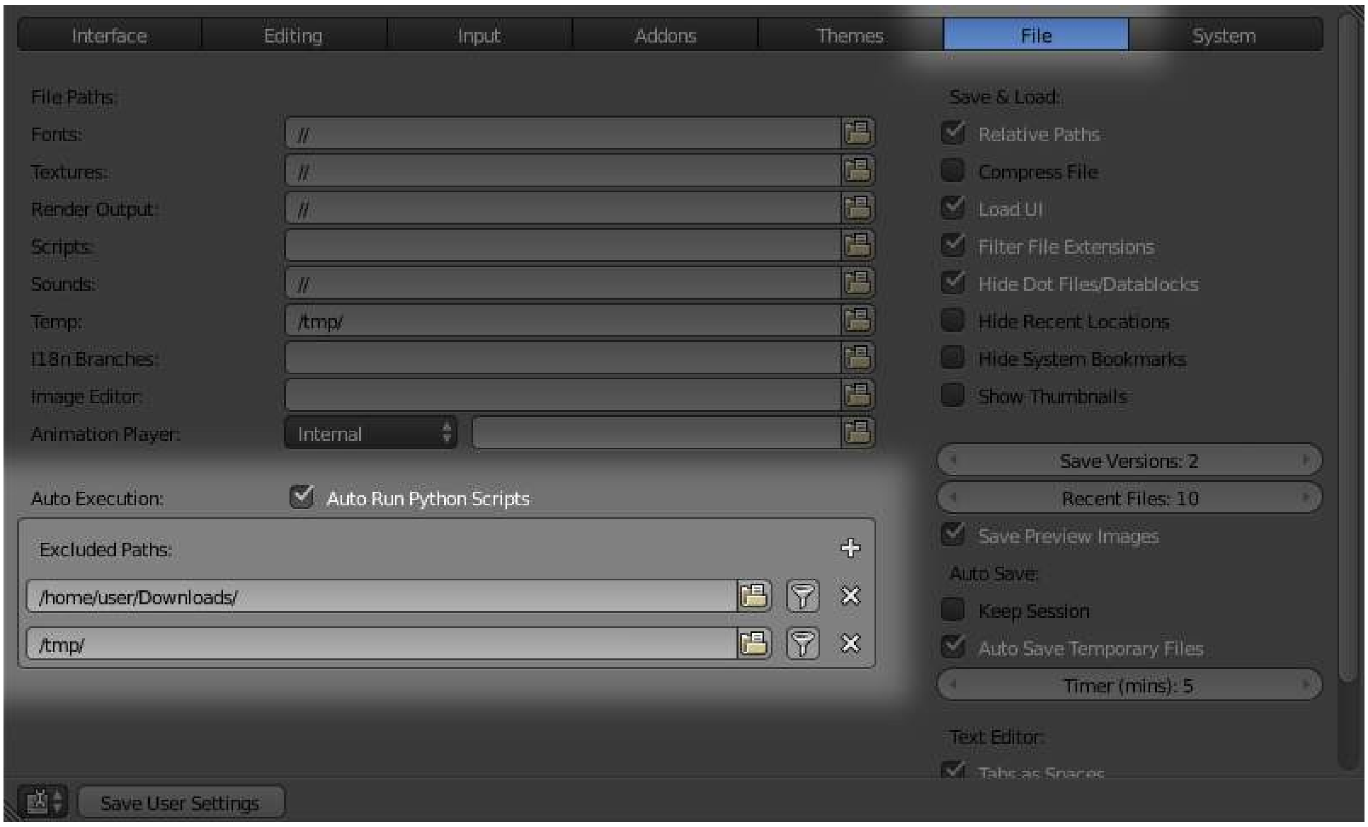
However you may forget to set this, or open a file without going through the file selector - so you can change the default (described next).

Setting Defaults

In the **File** section of the user-preferences there is the toggle **Auto-Run Python Scripts**.

This means the **Trusted Source** option in the file-selector will be enabled by default, and scripts can run when blend files are loaded without using the file selector.

Once enabled you have the option to exclude certain directories, a typical configuration would be to trust all paths except for the download directory.



File Paths:
Fonts: //
Textures: //
Render Output: //
Scripts:
Sounds: //
Temp: /tmp/
I18n Branches:
Image Editor:
Animation Player: Internal

Auto Execution: ✓ Auto Run Python Scripts
Excluded Paths:
/home/user/Downloads/
/tmp/

Save & Load:
✓ Relative Paths
Compress File
✓ Load UI
✓ Filter File Extensions
✓ Hide Dot Files/Datablocks
Hide Recent Locations
Hide System Bookmarks
Show Thumbnails
Save Versions: 2
Recent Files: 10
✓ Save Preview Images
Auto Save:
Keep Session
✓ Auto Save Temporary Files
Timer (mins): 5
Text Editor:
✓ Tabs as Spaces

Command Line

You may want to perform batch rendering or some other task from the command line - running Blender without an interface.

In this case the user-preferences are still used but you may want to override them.

- Enable with `-y` or `--enable-autoexec`
- Disable with `-Y` or `--disable-autoexec`

Example - rendering an animation in background mode, allowing drivers and other scripts to run:

```
blender --background --enable-autoexec my_movie.blend --render-anim
```

Note

These command line arguments can be used to start a regular blender instance and will still override the user-preferences.