renderthreads Documentation

Release 0.1

timmwagener

March 25, 2016

Contents


RENDERTHREADS

CHAPTER 1

Installation

The installation is easy. Choose one of several options.

1.1 Manual installation

- Go to PyPI and download the package.
- Then extract the zip and go into the folder renderthreads-x.x.x (for example renderthreads-0.2.3). In there you'll find a folder called **renderthreads**.
- Take it and copy it into your Nuke **site-packages** (usually *NukeX.YvZ/lib/site-packages/*) or anywhere else on your Nuke Python path.

1.2 pip

If you have pip installed for your Python interpreter in Nuke you can just type the following:

```
pip install renderthreads
```

To verify the installation type:

```
pip list
```

and see if renderthreads is amongst the listed packages.

Note: In case you have pip but **NOT** for your Nuke Python interpreter, which might often be the case, i would still recommend doing the above procedure. **renderthreads** will then be installed into your default system Python and you have to copy it into your Nuke site-packages by hand **but** pip still provides an easy and clean way to quickly update **renderthreads**.

1.3 Run it

To start it open up your **Nuke Python script editor**, copy this and run:

```
from renderthreads import renderthreads
reload(renderthreads)
renderthreads.run()
```

In the case of a successful execution you should see the following window:

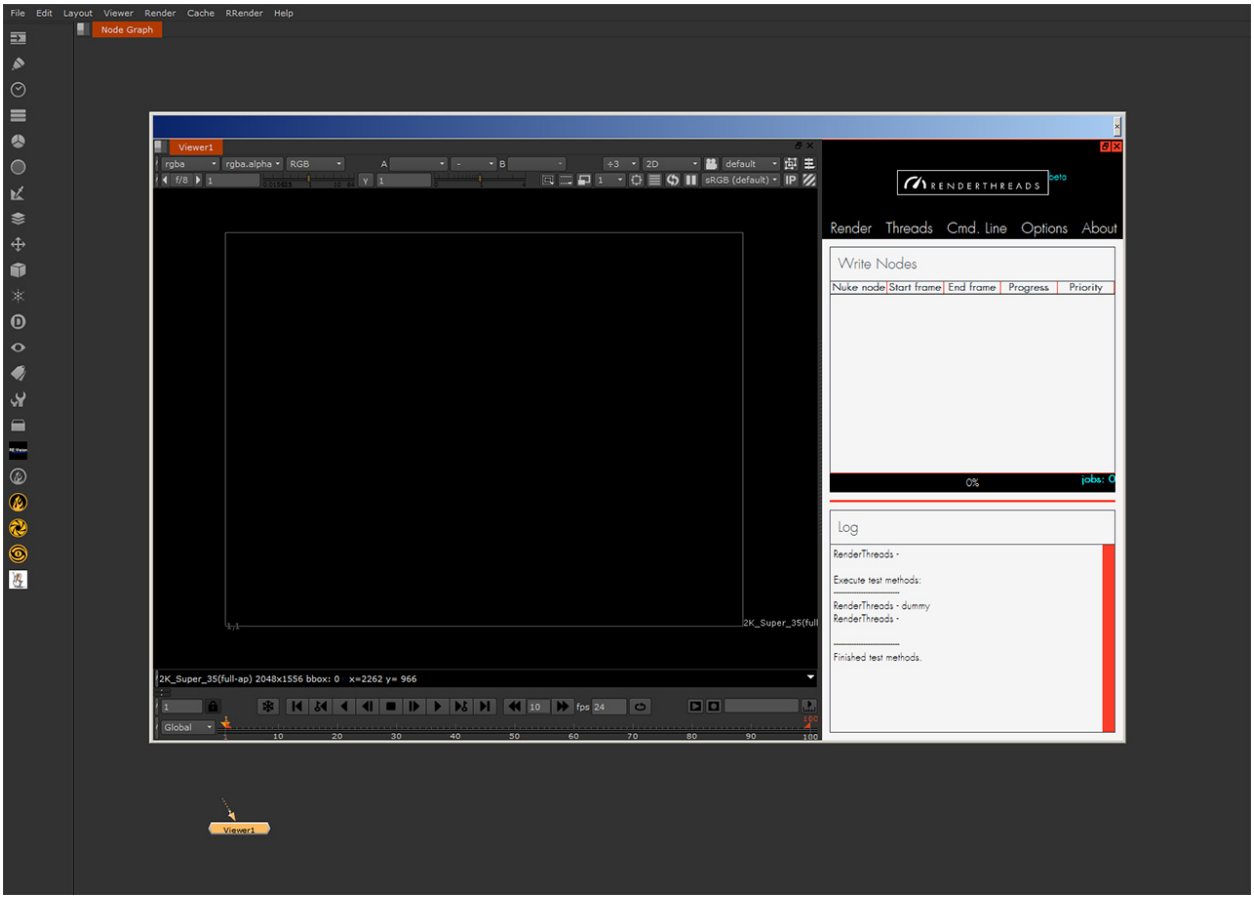


Fig. 1.1: You successfully installed renderthreads.

CHAPTER 2

Quickstart

Have your first renderings up and running in under a minute (...or 2). To start a command-line rendering with **renderthreads** do the following.

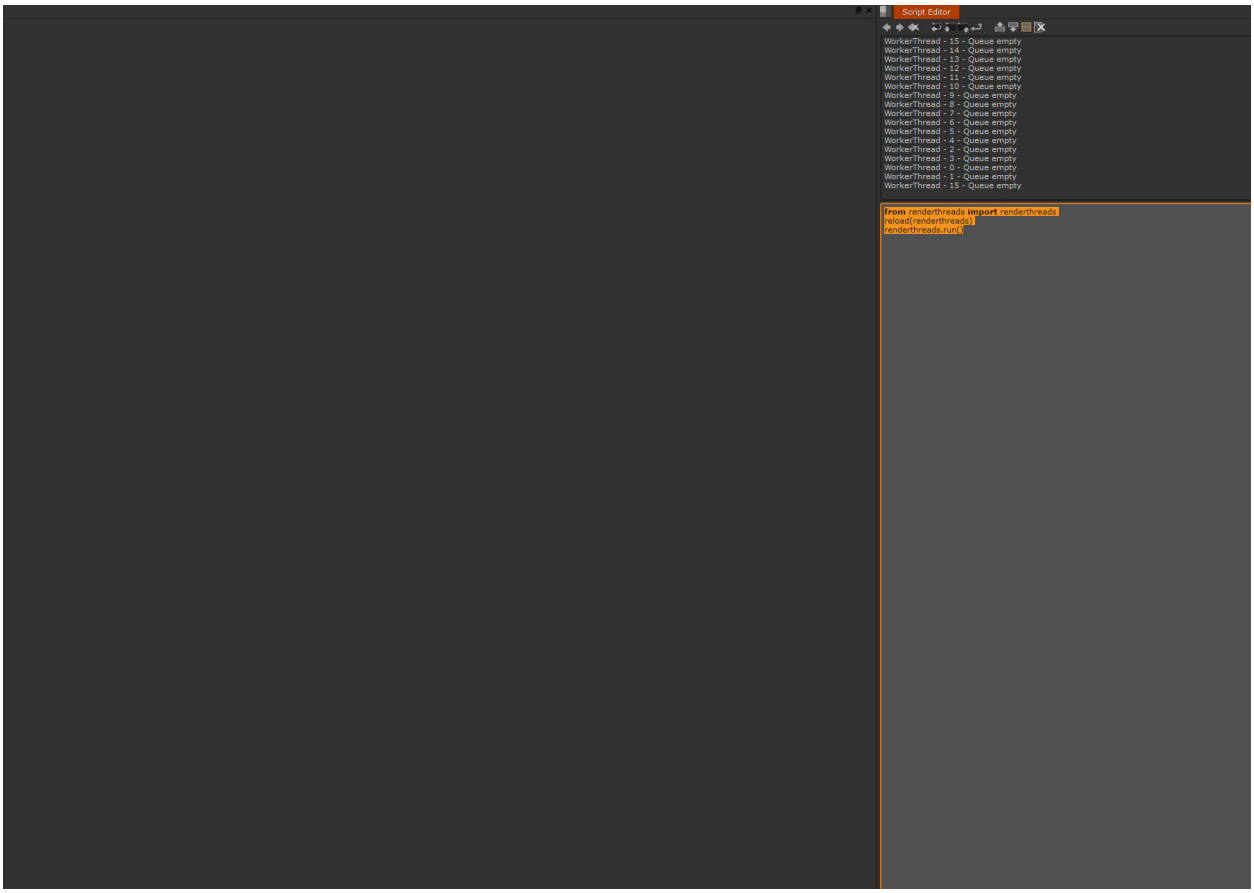


Fig. 2.1: Run the startup script like so. You may want to make a button out of it and integrate it in a menu. (Forgot how to install it? *Here's how*.)

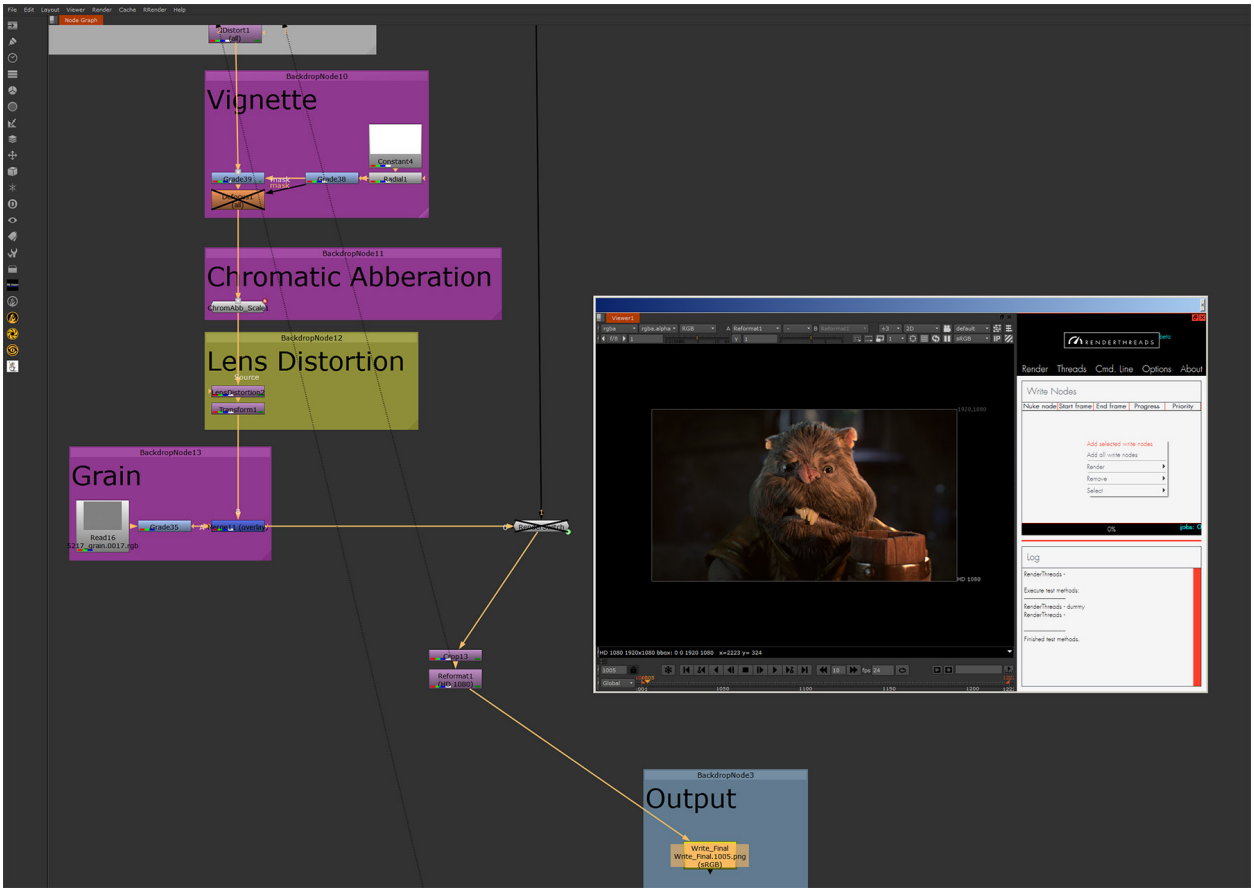


Fig. 2.2: Select a write node and add it like so.

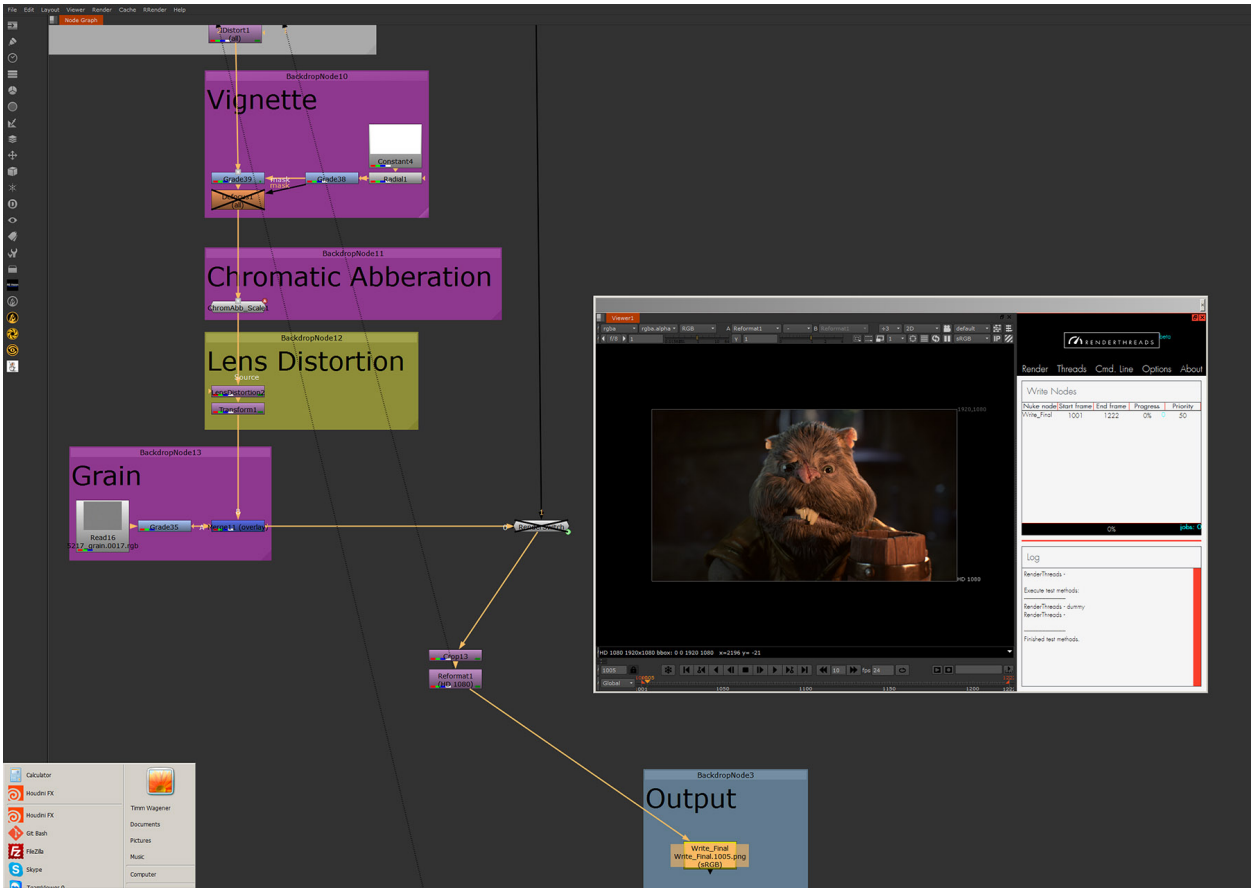


Fig. 2.3: Success. Your Write node is in **renderthreads** along with a **frame range** and a **priority**.

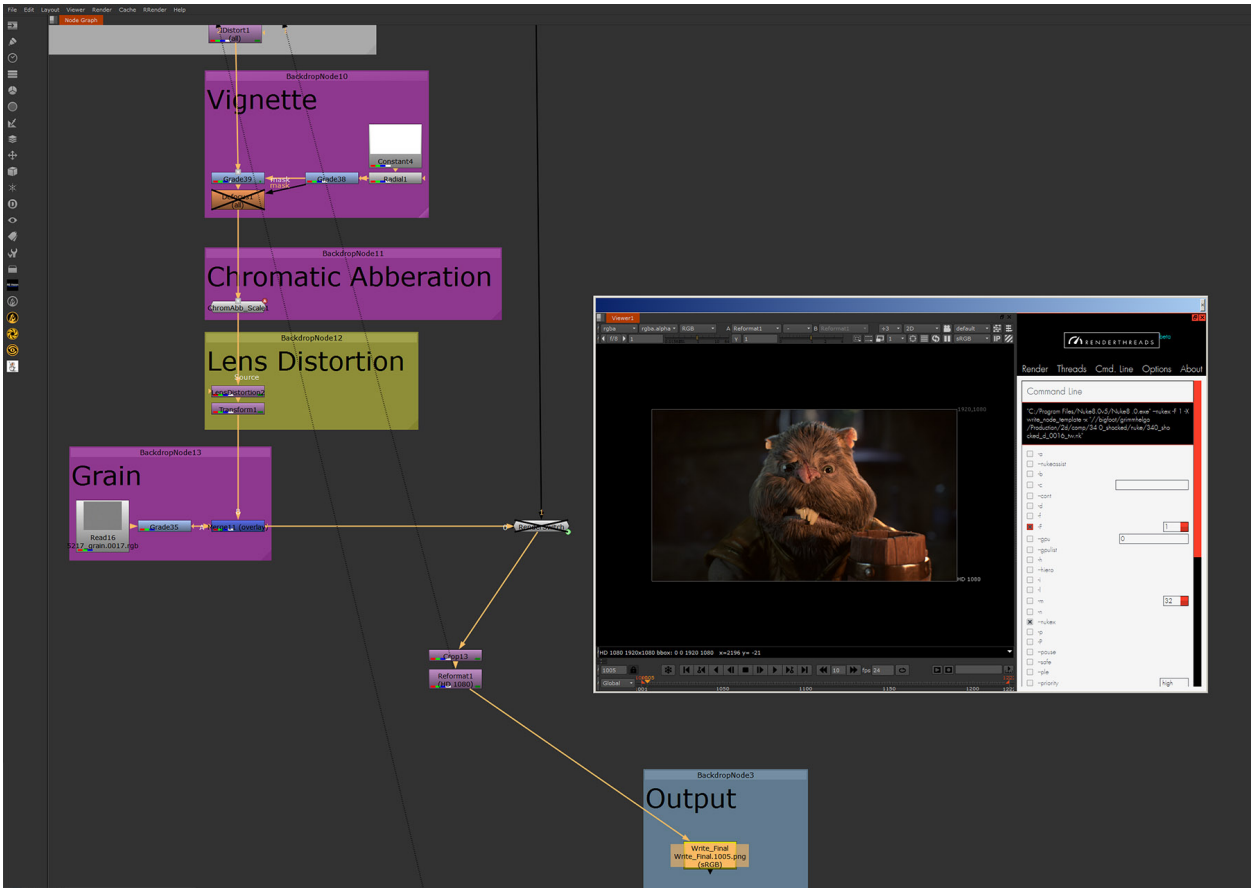


Fig. 2.4: **Optional:** Check the command line menu if the command line settings are fine. For example you could tell Nuke to use render licenses over interactive ones. Each command line flag has a tooltip to (more or less) explain it. The explanations are copied from *../Nuke.exe -help*

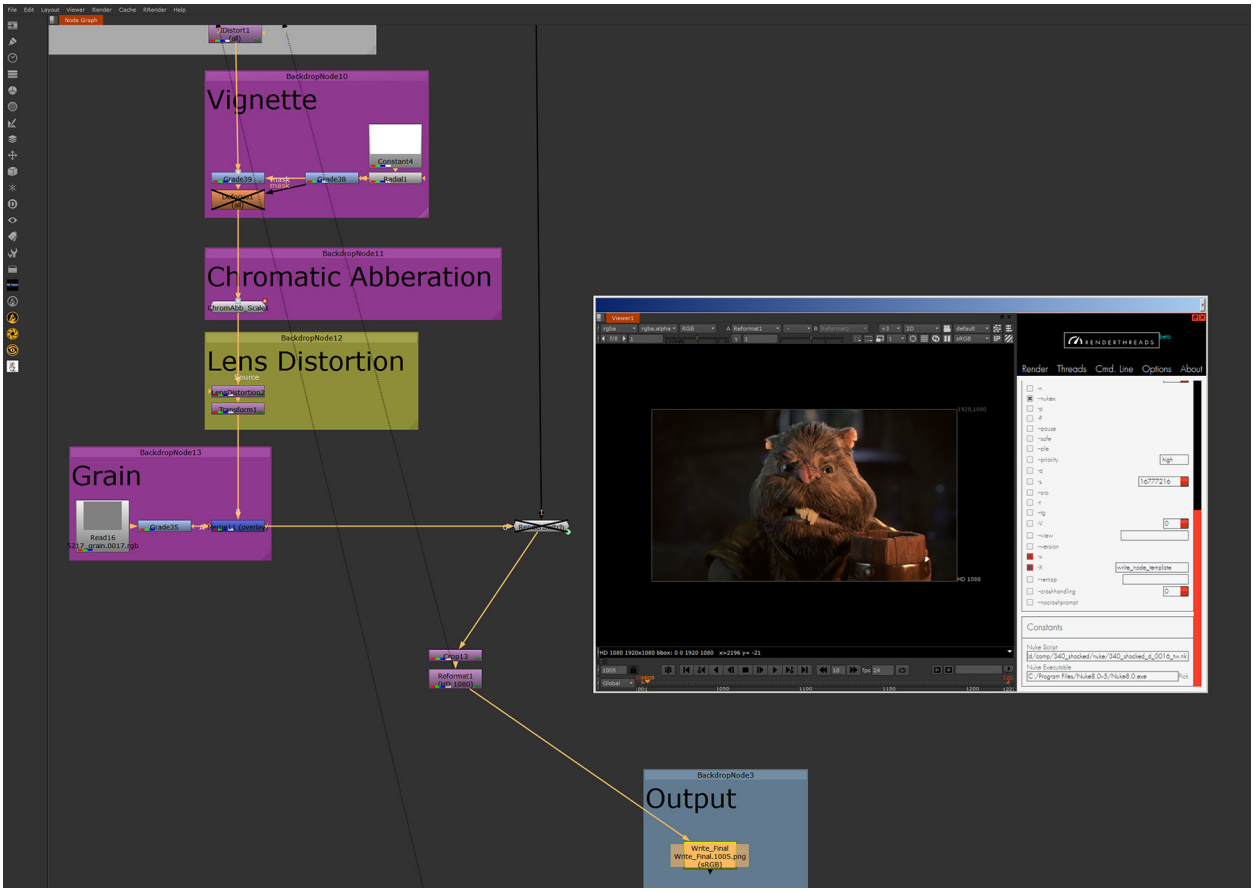


Fig. 2.5: **Optional:** Check the script name and the Nuke path under **Constants**. The nuke script is always the one you have open right now. If it says **Root** you might be prompted to save before rendering can start. The Nuke version can be choosen.

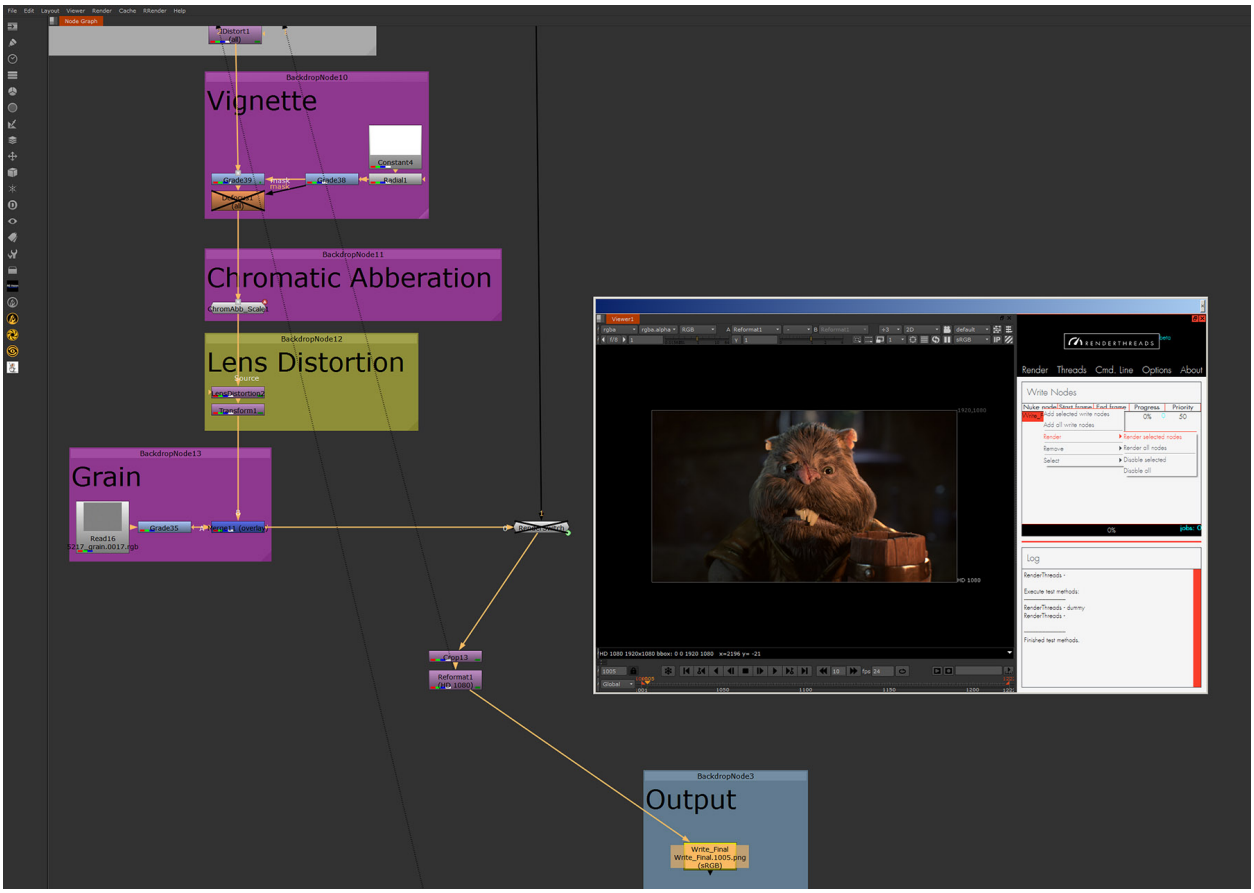


Fig. 2.6: Start the rendering like so. By default rendering will happen on half of the cores you have available. (You can adjust the number of cores in the *Threads* menu)

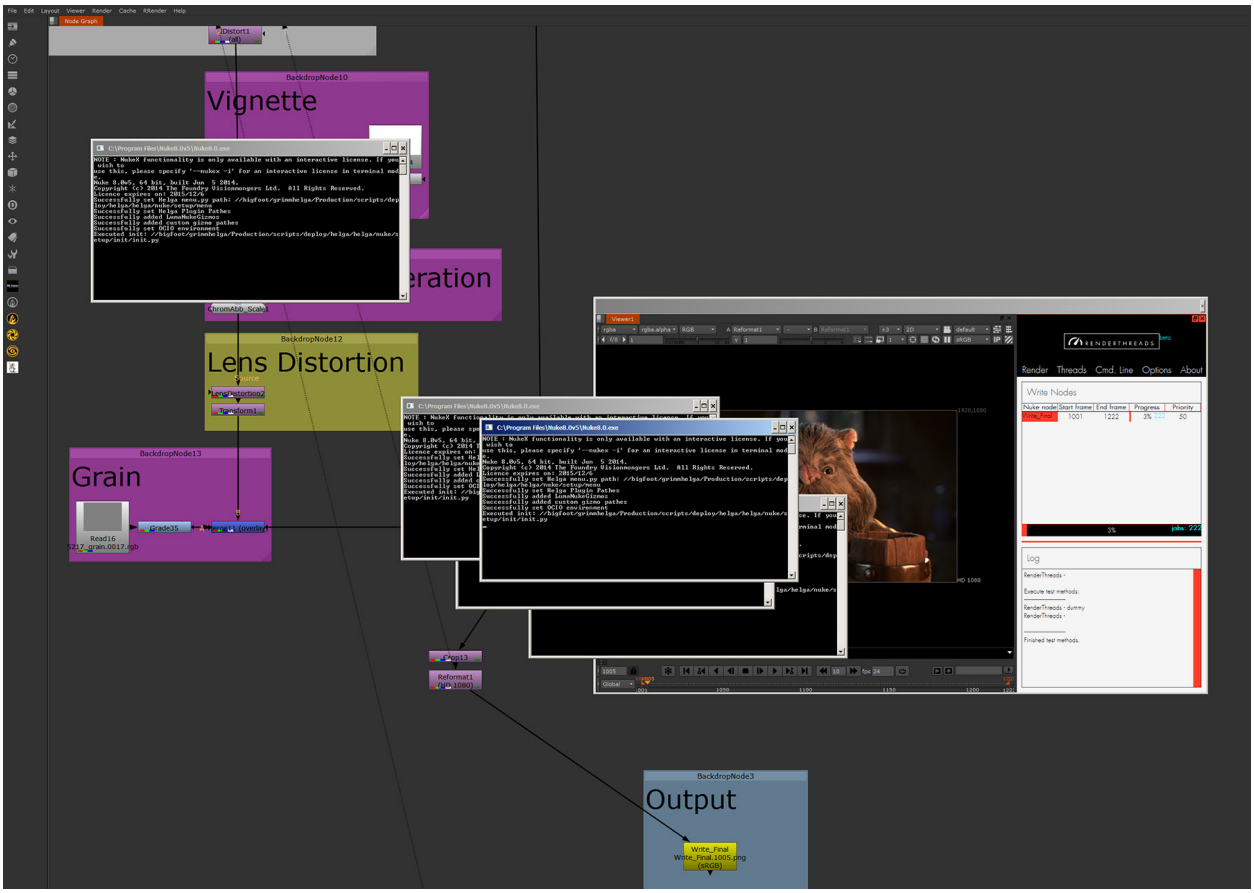


Fig. 2.7: Success, you are rendering. A lot of command shells pop up by default (No worries, you can disable this in the *Threads* menu). The progress is displayed in the interface.

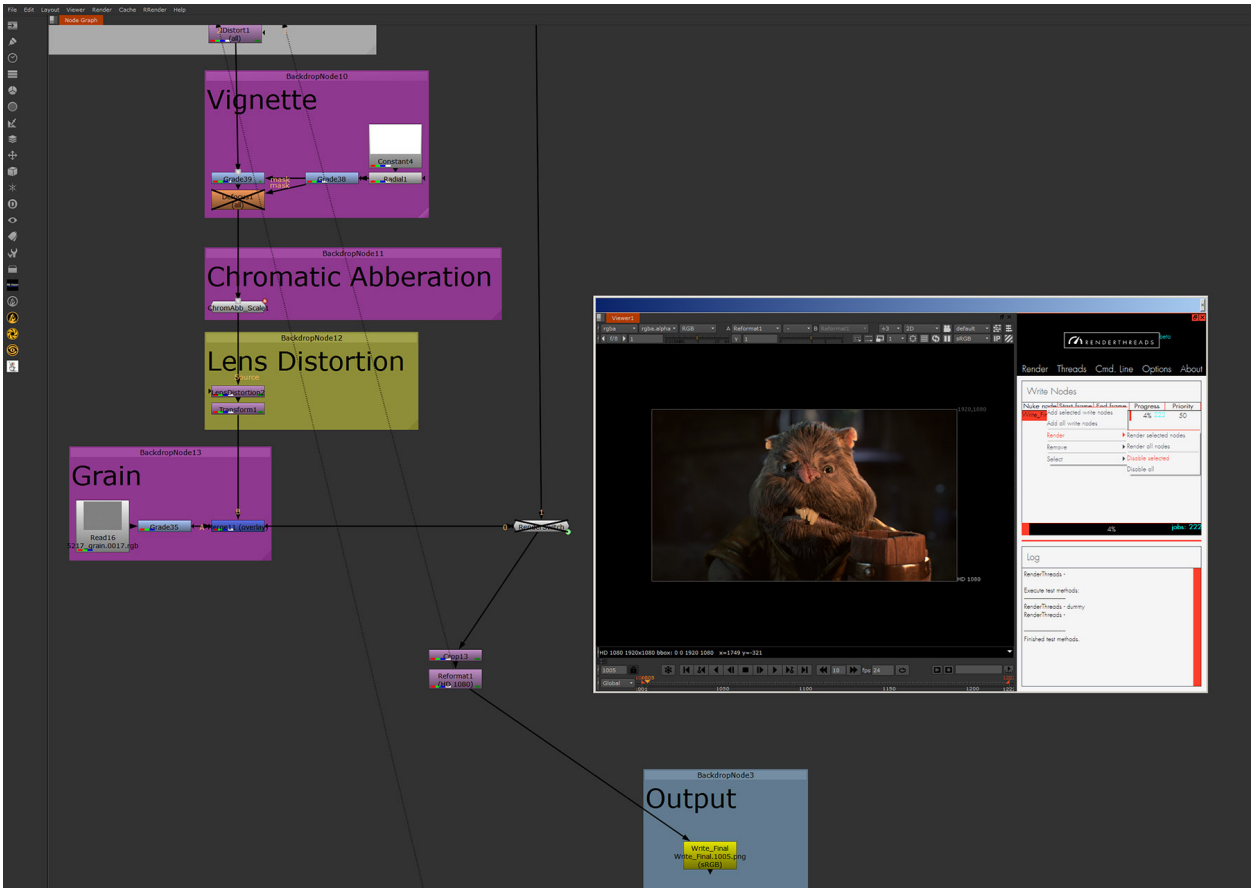


Fig. 2.8: You can disable the rendering anytime, like so. Disabling will kill the jobs currently running and not start those that are still queued up anymore.

CHAPTER 3

Options

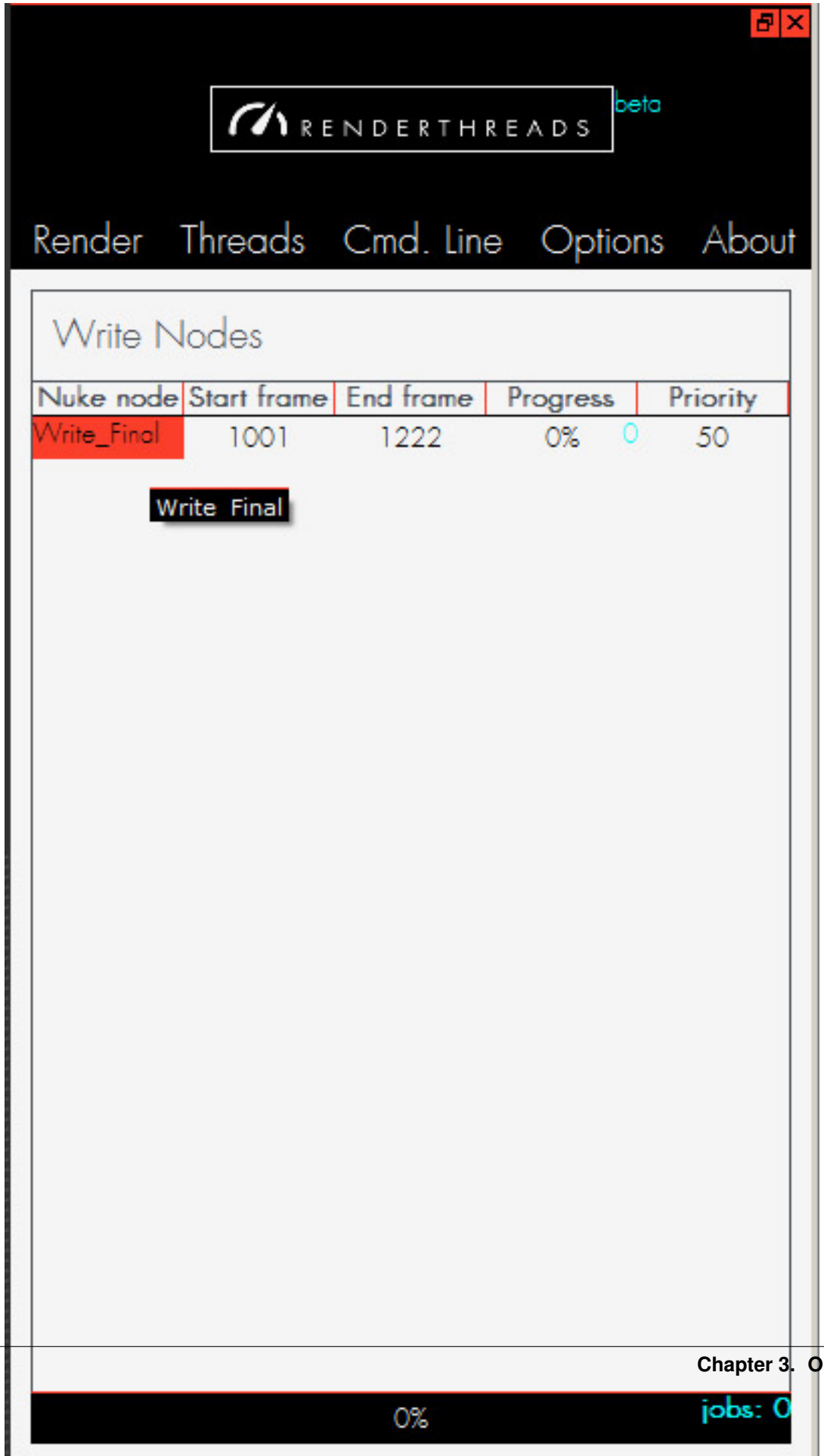
renderthreads gives you a few menus to adjust things and influence the rendering. All the widgets provide reasonable **tooltips** that tell you what they are doing. Anyways, here's another source of information on what those settings do.

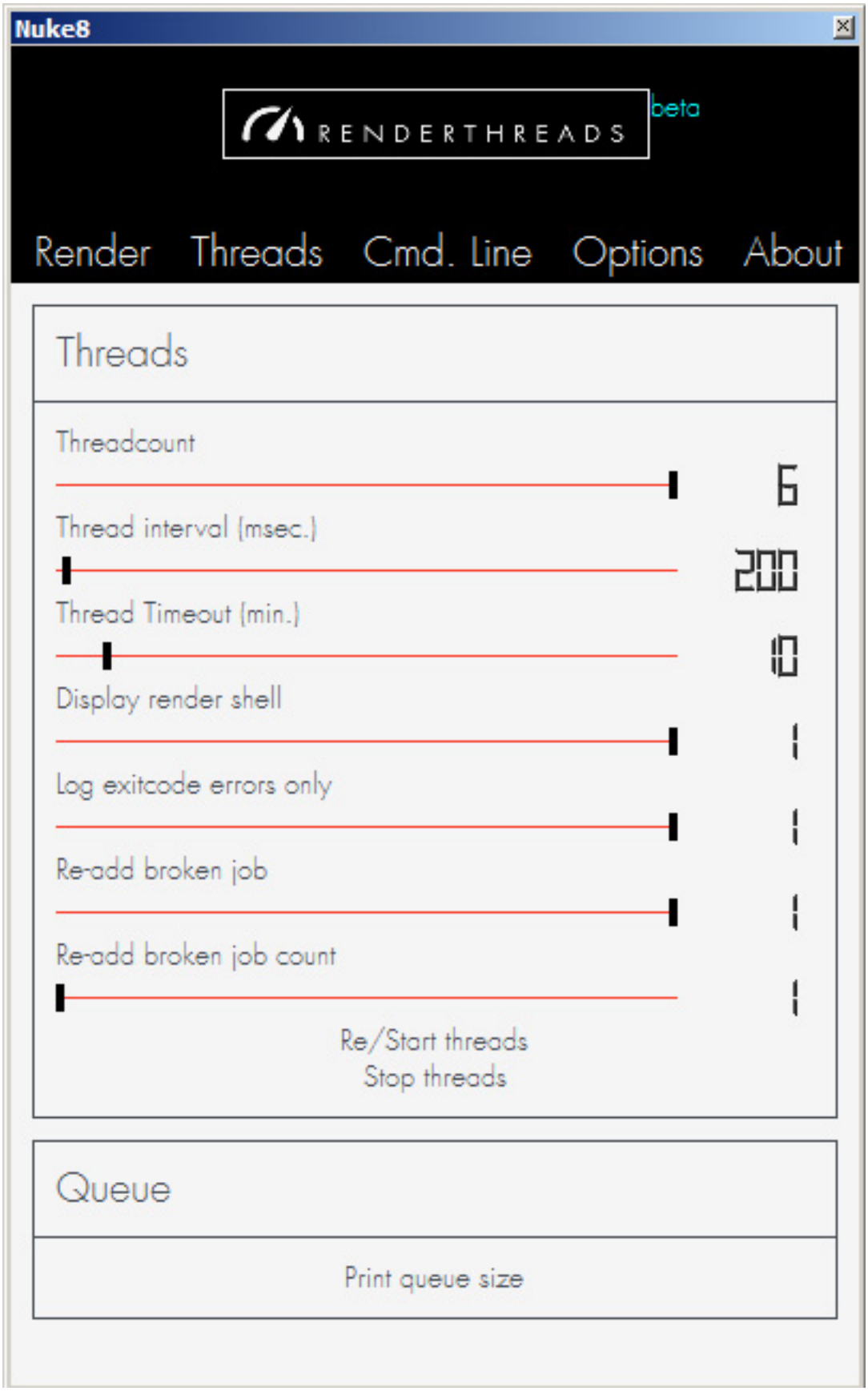
3.1 Render

- **Nuke node** is the name of the write node you are about to render. You can edit it from within renderthreads. Write nodes within nested groups are also valid.
- **Start frame** of your render sequence. The initial value is taken from the root node.
- **End frame** of your render sequence. The initial value is taken from the root node.
- **Progress** of the rendering of your specific write node. The blue number is the total amount of jobs/frames. The progressbar at the bottom displays the **sum of all write nodes**.
- **Priority** of the rendering of the specific write node. This cannot be adjusted once the jobs are added. In case of equal priorities the jobs are prioritized by the write node alphabetic order.
- **Log** displays success/fail messages for the command-line jobs. By default only error messages are logged, but you can change that in the **Threads** menu.

3.2 Threads

- **Threadcount** lets you adjust how many threads/cores you want to use for rendering. Each core renders a single frame. By default the number of cores is **half the number of the available cores in your system**, which is a conservative setting ment to prevent **memory overflow**.


RENDERTHREADS
beta
Render
Threads
Cmd. Line
Options
About
Write Nodes
Nuke node
Start frame
End frame
Progress
Priority
Write_Final
1001
1222
0%
0
50
Write Final
0%
jobs: 0


Nuke8
RENDERTHREADS
beta
Render
Threads
Cmd. Line
Options
About
Threads
Threadcount
6
Thread interval (msec.)
200
Thread Timeout (min.)
10
Display render shell
1
Log exitcode errors only
1
Re-add broken job
1
Re-add broken job count
1
Re/Start threads
Stop threads
Queue
Print queue size

> **Warning:** Be aware that each command-line render opens your Nuke script to process it. Therefore it needs a certain amount of memory. When too many jobs are started that eat a lot of memory you might want to pick a lower threadcount to prevent **memory overflow**.

- **Thread interval** is the rate at which a new job is started by a thread after the thread finished the current job. The should rarely be a reason to increase this value.
- **Thread timeout** is a more **important value**. It sets the time in minutes that a rendering is allowed to take before it is forcibly closed. The default is 10 minutes, but demanding nuke comps can easily take longer. **If you wonder why all your renderings are terminated, a too low timeout setting might be the reason.**
- **Display render shell** lets you adjust wether or not you want shell windows popping up.
- **Log exitcode errors only** Wether you just want errors and their exit codes logged (process terminated, timed out, was disabled...) or also sucess messages.
- **Re-add broken job** Wether or not to add a job to the queue again that terminated unsuccessfully. (With an exitcode different from 0).
- **Re-add broken job count** How often such an error job gets added again. The default is 2.
- **Start/Stop threads** Pause rendering. Will finish all currently running jobs and then re/start or stop rendering. All the jobs remain, so this is like a **pause** function.
- **Print queue size** Print complete count of all jobs.

3.3 Command-line

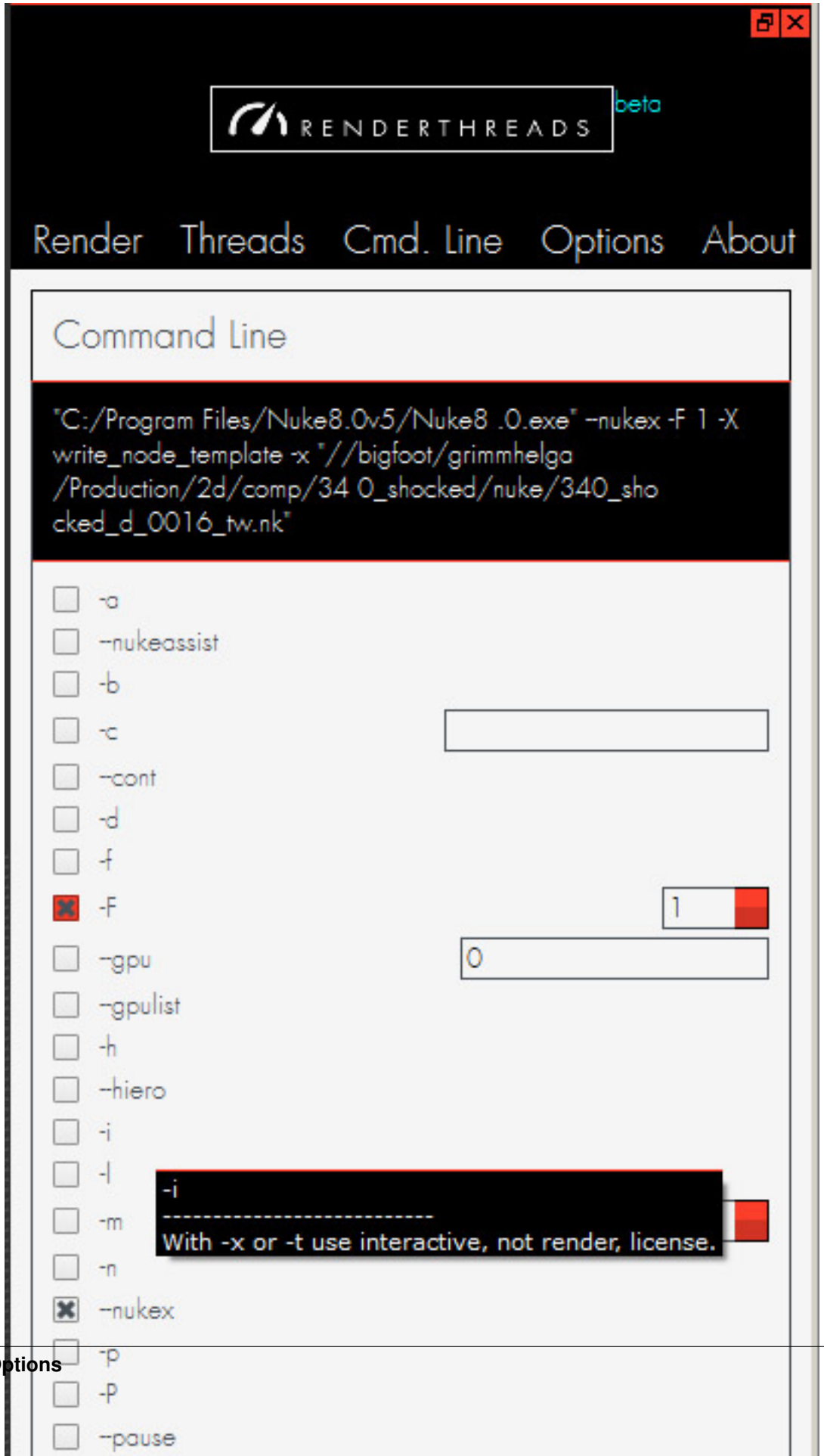
- **Command-line flags** Assemble the Nuke command line used for rendering. Here you have the default command line options (Taken from Nuke 8.05). At the top you see the command line that you assembled. Some values like **-X, -x or -F are locked and cannot be changed. They are handled internally**. Each flag has a tooltip that describes what it does.

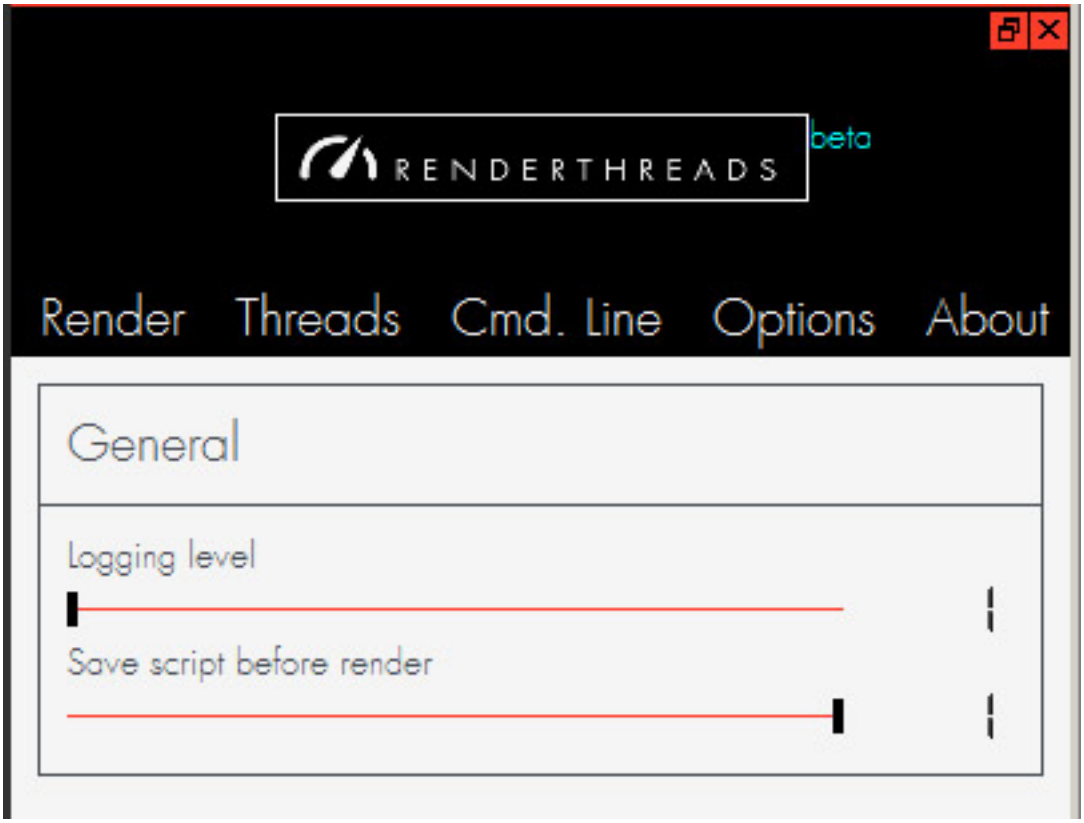
  > **Warning:** **renderthreads** does not restrict you in the combination of flags. Therefore you can set flags that are incompatible or make no sense (like enabling proxy mode while forcing full size rendering).

- **Nuke script** is the currently open script. If the value is Root (script not saved yet), you might be prompted to save before rendering starts. The value updates automatically when you save the script. You cannot change this value.
- **Nuke executable** The nuke exe that is used for rendering. You are free to pick the Nuke version you want. **Nuke is always started with the environment of the Nuke that is starting the jobs.**

3.4 Options

- **Logging level** How verbose **renderthreads** tells you about its state.


RENDERTHREADS
beta
Render
Threads
Cmd. Line
Options
About
Command Line
"C:/Program Files/Nuke8.0v5/Nuke8 .0.exe" --nukex -F 1 -X write_node_template -x "//bigfoot/grimmhelga/Production/2d/comp/34 0_shocked/nuke/340_shocked_d_0016_tw.nk"
-a
--nukeassist
-b
-c
--cont
-d
-f
-F
1
--gpu
0
--gpulist
-h
--hiero
-i
-l
-m
-n
--nukex
-p
-P
--pause
-i
With -x or -t use interactive, not render, license.



- **Save script before render** lets you adjust if you want to automatically save before before starting a render. The default is on. **If this is off you might not render the latest adjustments in your comp unless you saved manually**

CHAPTER 4

Compatibility

Here i will note environments (OS, Nuke versions) that **renderthreads** was tested in and is known to work with.....or not. If you encounter any issues running the tool with your version of Nuke or your operating system, feel free to drop me a line. Or file a bug report on the github issue tracker.

4.1 Operating systems

4.1.1 Compatible

- Windows7

4.1.2 Incompatible

4.1.3 Untested

- Linux
- MacOS

4.2 Nuke versions

4.2.1 Compatible

- 9.x
- 8.x

4.2.2 Incompatible

- 7.x/below: **renderthreads** needs **Python 2.7**