How to Read a Render Log

The Render Log prints the progress messages, warnings and errors to the console or to a file as kick renders an image. As well as showing the percentage completed, these detailed statistics can be used to optimize and debug scenes.

> The Arnold log provides detailed statistics that are useful for debugging, optimizing and benchmarking renders. It is the first thing to examine should you encounter errors and usually the first information to send to support.

The verbosity levels vary from kick to the plugins.

Kick	Plugins
0 - No Output 1 - Progress 2 3 4 5 - Detailed 6 - Debug	Errors Warnings + Info Debug

The log below contains the most common output. The first column of the log, `00:00:00`, shows the time elapsed in `hh:mm:ss`. The second column, `773MB`, shows the memory usage at that stage.

Initialization (Info)

This section shows the version of Arnold being used and its build details. It also lists the hardware specifications and operating system of the machine it was rendered on.

```
00:00:00   773MB         | log started Tue Jul 21 15:26:25 2015
00:00:00   773MB         | Arnold 4.2.7.4 windows icc-14.0.2 oiio-1.5.15 rlm-11.2.2
2015/06/15 09:39:31
00:00:00   773MB         | host application: MtoA 1.2.3.1 03a85380bec8 (Master)
MtoA-1.2.3.1 Maya 2016
00:00:00   773MB         | running on PC, pid=12188
00:00:00   773MB         |  1 x Intel(R) Xeon(R) CPU E5-1650 v2 @ 3.50GHz (6 cores,
12 logical) with 32712MB
00:00:00   773MB         |  Windows 7 Professional Service Pack 1 (version 6.1,
build 7601)
```

There is a single initialization process for the whole render session (only on the first render), with an updated process for every following render.

```
00:00:00   773MB         | [mtoa.session]     Initializing at frame 1.000000
00:00:00   773MB         | [mtoa] Exporting Arnold options
'defaultArnoldRenderOptions'
00:00:00   773MB         | [mtoa.extensions]  aiOptions Using translator
<built-in>, provided by <built-in>(<built-in>).
00:00:00   773MB         | [mtoa.session]     defaultArnoldRenderOptions     |
Exporting plug defaultArnoldRenderOptions.message for type aiOptions
```

Loading / Plugins

This section displays the results of importing any shaders or procedurals.

Just rendering through kick on it's own shows no plugin and the ass location.

```
00:00:00   773MB         | loading plugins from . ...
00:00:00   773MB         | no plugins loaded
00:00:00   773MB         | [ass] loading project/scenes/simplescene.ass ...
00:00:00   773MB         | [ass] read 11386 bytes, 11 nodes in 0:00.00
```

If a plugin is being loaded (and verbosity is high enough) the shaders will be listed:

```
00:00:00   773MB         | loading plugin:
C:/solidangle/mtoadeploy/2016/shaders/mtoa_shaders.dll ...
00:00:00   773MB         |  mtoa_shaders.dll: MayaMultiplyDivide uses Arnold
4.2.7.4
00:00:00   773MB         |  mtoa_shaders.dll: MayaClamp uses Arnold 4.2.7.4
00:00:00   773MB         |  mtoa_shaders.dll: MayaGammaCorrect uses Arnold 4.2.7.4
00:00:00   773MB         |  mtoa_shaders.dll: MayaCondition uses Arnold 4.2.7.4
....
00:00:00   774MB         |  mtoa_shaders.dll: volume_sample_float uses Arnold
4.2.7.4
00:00:00   774MB         |  mtoa_shaders.dll: volume_sample_rgb uses Arnold 4.2.7.4
00:00:00   774MB         |  mtoa_shaders.dll: driver_mplay uses Arnold 4.2.7.4
00:00:00   774MB         | loaded 90 plugins from 1 lib(s) in 0:00.00
```

License Check

If a license is available the log will display the following.

```
00:00:00   810MB         | [rlm] checkout of "arnold 20150615" in progress ...
00:00:00   810MB         | [rlm] checkout of "arnold 20150615" from server PC in
0:00.01
00:00:00   810MB         | [rlm] expiration date: 31-dec-2015 (164 days left)
```

However, if the RLM server is not running it will display a warning.

```
00:00:00   810MB WARNING | [rlm] could not connect to license server on
5053@localhost
```

Scene Contents

Here it will list the numbers of lights and objects (and their types) in the scene.

```
00:00:00   818MB         | there are 1 light and 2 objects:
00:00:00   818MB         |       1 persp_camera
00:00:00   818MB         |       1 distant_light
00:00:00   818MB         |       1 utility
00:00:00   818MB         |       1 lambert
00:00:00   818MB         |       1 driver_exr
00:00:00   818MB         |       1 gaussian_filter
00:00:00   818MB         |       1 polymesh
00:00:00   818MB         |       1 list_aggregate
00:00:00   818MB         |       1 MayaShadingEngine
00:00:00   818MB         |       1 renderview_display
```

Resolution / Samples

This section displays the output resolution and details about the number of samples.

```
00:00:00   818MB         | rendering image at 640 x 480, 3 AA samples
00:00:00   818MB         |   AA sample clamp   <disabled>
00:00:00   818MB         |   diffuse           samples  3 / depth  1
00:00:00   818MB         |   glossy            samples  3 / depth  2
00:00:00   818MB         |   reflection        samples  1 / depth  2
00:00:00   818MB         |   refraction        samples  2 / depth  2
00:00:00   818MB         |   volume indirect   <disabled by depth>
00:00:00   818MB         |   total                          depth 10
00:00:00   818MB         |   bssrdf            <disabled>
00:00:00   818MB         |   single scatter    samples  5
00:00:00   818MB         |   transparency      depth 10 / threshold 0.99 / fast
opacity off
```

Nodes

Before rendering starts nodes are initialized. The stats here give insight into the objects and lights in the scene and can show warnings if anything is configured incorrectly. Note that some data is not immediately initialized, some computations are delayed until the first ray hits the object, so further information can be shown during the render.

```
00:00:00   818MB         |  initializing 11 nodes ...
00:00:00   818MB         |   creating root object list ...
00:00:00   818MB         |   scene bounds: (-0.995465517 -1.01209259 -0.995465755)
-> (1.00453472 0.98790741 1.00453484)
00:00:00   818MB         |  node initialization done in 0:00.00 (single-threaded)
00:00:00   818MB         |  updating 12 nodes ...
00:00:00   818MB         |   directionalLightShape1: distant_light using 1 sample
00:00:00   818MB         |  node update done in 0:00.00 (single-threaded)
```

Drivers / AOVs

Here are the drivers and AOVs used in the scene, and any warnings if they are set up incorrectly.

```
00:00:00   818MB         |  [aov] parsing output statement: "RGBA RGBA
defaultArnoldFilter@gaussian_filter defaultArnoldDisplayDriver@renderview_display"
00:00:00   818MB         |  [aov] parsing output statement: "RGBA RGBA
defaultArnoldFilter@gaussian_filter defaultArnoldDriver@driver_exr.RGBA"
00:00:00   818MB         |  [aov] registered driver:
"defaultArnoldDisplayDriver@renderview_display" (renderview_display)
00:00:00   818MB         |  [aov]  * "RGBA" of type RGBA filtered by
"defaultArnoldFilter@gaussian_filter" (gaussian_filter)
00:00:00   818MB         |  [aov] registered driver:
"defaultArnoldDriver@driver_exr.RGBA" (driver_exr)
00:00:00   818MB         |  [aov]  * "RGBA" of type RGBA filtered by
"defaultArnoldFilter@gaussian_filter" (gaussian_filter)
00:00:00   818MB         |  [aov] done preparing 1 AOV for 2 outputs to 2 drivers
(0 deep AOVs)
```

Progress

Now that the render is starting, the log will show how many buckets are being used and at what size. If any importance tables need to be generated, in this case for the skydome, they will be done so now, showing the average energy value. Percentage completed is shown in 5% increments along with the average rays per pixel. This value will show at a glance if your sampling settings are too high.

```
00:00:00   825MB         |  starting 12 bucket workers of size 64x64 ...
00:00:00   835MB         |     0% done - 9 rays/pixel
00:00:00   835MB         |   [accel] bvh4 done - 0:00.00 - 400 prims, 1 key
00:00:00   836MB         |     5% done - 9 rays/pixel
00:00:00   836MB         |    10% done - 9 rays/pixel
00:00:00   836MB         |    15% done - 9 rays/pixel
00:00:00   837MB         |    20% done - 9 rays/pixel
00:00:00   837MB         |    25% done - 9 rays/pixel
00:00:00   837MB         |    30% done - 9 rays/pixel
00:00:00   837MB         |    35% done - 10 rays/pixel
00:00:00   837MB         |    40% done - 10 rays/pixel
00:00:00   838MB         |    45% done - 12 rays/pixel
00:00:00   838MB         |    50% done - 12 rays/pixel
00:00:00   838MB         |    55% done - 13 rays/pixel
00:00:00   838MB         |    60% done - 13 rays/pixel
00:00:00   838MB         |    65% done - 13 rays/pixel
00:00:00   838MB         |    70% done - 13 rays/pixel
00:00:00   838MB         |    75% done - 13 rays/pixel
00:00:00   838MB         |    80% done - 13 rays/pixel
00:00:00   838MB         |    85% done - 13 rays/pixel
00:00:00   838MB         |    90% done - 13 rays/pixel
00:00:00   838MB         |    95% done - 13 rays/pixel
00:00:00   838MB         |   100% done - 13 rays/pixel
00:00:00   824MB         |  bucket workers done in 0:00.38
```

Output

Once the render is complete it will be written to file (or screen) using the selected output driver. The log will show the relative path.

```
00:00:00   824MB         |  [driver_exr] writing file
`C:/Users/Documents/example.exr'
00:00:00   821MB         | render done
```

Scene Creation Stats

These timings show the time taken to load plugins and .ass files.

```
00:00:00   821MB         | scene creation time:
00:00:00   821MB         |  plugin loading               0:00.07
00:00:00   821MB         |  system/unaccounted           0:00.12
00:00:00   821MB         |  total                        0:00.12  ( 1.05% machine
utilization)
```

Render Time Stats

These timings show the time spent performing various tasks. Typically pixel rendering will take up most of the time, if that's not the case then looking at the node initialization and geometry stats might reveal inefficiencies. Low machine utilization may indicate that other processes on the same machine are slowing down the render. But it can also indicate that a lot of time is spent performing single threaded tasks, for example in node initialization, procedurals or mesh processing. Another possibility is that there is a lot of (slow) file access, typically for textures or volumes. Looking at the texture stats might reveal problems.

```
00:00:00   821MB         | render time:
00:00:00   821MB         |  node init                    0:00.00
00:00:00   821MB         |   sanity checks               0:00.00
00:00:00   821MB         |  bucket rendering             0:00.38
00:00:00   821MB         |   mesh processing             0:00.00
00:00:00   821MB         |   accel. building             0:00.00
00:00:00   821MB         |   pixel rendering             0:00.38  (multi-threaded
render, this value may not be reliable)
00:00:00   821MB         |  system/unaccounted           0:00.17
00:00:00   821MB         |  total                        0:00.55  (69.23% machine
utilization)
```

Memory Stats

Here memory used at startup and peak memory for various types of data are listed. When using a plugin the startup memory shows how much memory the host application is using, when rendering from kick the startup memory is small which typically means larger scenes can be rendered. If memory usage is too high these stats indicate which data would be most helpful to reduce.

```
00:00:00   821MB         | memory consumed in MB:
00:00:00   821MB         |  at startup                    807.25
00:00:00   821MB         |  plugins                         2.52
00:00:00   821MB         |  AOV samples                    11.67
00:00:00   821MB         |  output buffers                  5.32
00:00:00   821MB         |  node overhead                   0.00
00:00:00   821MB         |  message passing                 0.02
00:00:00   821MB         |  memory pools                   13.54
00:00:00   821MB         |  geometry                        0.01
00:00:00   821MB         |    polymesh                      0.01
00:00:00   821MB         |  accel. structs                  0.02
00:00:00   821MB         |  strings                         0.30
00:00:00   821MB         |  texture cache                   0.00
00:00:00   821MB         |  unaccounted                     1.64
00:00:00   821MB         |  total peak                    842.29
```

Ray Stats

These are the number of rays of each type, per pixel and per AA sample. If the render time is high, these numbers give insight into which types of samples are most expensive to render and would be most helpful trying to reduce.

```
00:00:00   821MB         | ray counts:                           (/pixel , /sample)
(% total) (avg. hits) (max hits)
00:00:00   821MB         |  camera                    2318256 (   8.71,    1.00)
( 64.57%) (     0.11) (       1)
00:00:00   821MB         |  shadow                     228076 (   0.86,    0.10)
(  6.35%) (     0.00) (       0)
00:00:00   821MB         |  diffuse                   1043914 (   3.92,    0.45)
( 29.08%) (     0.00) (       0)
00:00:00   821MB         |  total                     3590246 (  13.48,    1.55)
(100.00%) (     0.07) (       1)
00:00:00   821MB         |  max depth                       1
```

Shader Stats

The number of shader calls in various contexts are shown here, which can be useful to figure out which shader calls are most helpful to speed up.

```
00:00:00   821MB         | shader calls:                         (/pixel , /sample)
(% total)
00:00:00   821MB         |  primary                    523006 (   1.96,    0.23)
(100.00%)
00:00:00   821MB         |  total                      523006 (   1.96,    0.23)
(100.00%)
```

Geometry Stats

These stats give insight into amount of geometry in the scene. If memory usage is high or scene setup takes a long time these can be used to find the objects that contribute to it most, and would benefit from being simplified or having their subdivision iterations reduced.

```
00:00:00   821MB         | geometry:                                (% hit )
(instances) (  init mem,  final mem)
00:00:00   821MB         |  lists                                 1 (100.0%) (
0) (      0.00,       0.00)
00:00:00   821MB         |  polymeshes                            1 (100.0%) (
0) (      0.02,       0.01)
00:00:00   821MB         |
-------------------------------------------------------------------------------
------
00:00:00   821MB         | geometric elements:                      (      min) (
avg.) (      max)
00:00:00   821MB         |  objects (top level)                   1 (        1) (
1.0) (        1)
00:00:00   821MB         |  polygons                            400 (      400) (
400.0) (      400)
00:00:00   821MB         |
-------------------------------------------------------------------------------
------
00:00:00   821MB         | triangle tessellation:                   (      min) (
avg.) (      max) (/ element) (% total)
00:00:00   821MB         |  polymeshes                          760 (      760) (
760.0) (      760) (     1.90) (100.00%)
00:00:00   821MB         |  unique triangles:                    760
00:00:00   821MB         |  memory use (in MB)                  0.01
00:00:00   821MB         |    vertices                          0.00
00:00:00   821MB         |    vertex indices                    0.00
00:00:00   821MB         |    packed normals                    0.00
00:00:00   821MB         |    normal indices                    0.00
00:00:00   821MB         |    uv coords                         0.00
00:00:00   821MB         |    uv coords idxs                    0.00
00:00:00   821MB         |    uniform indices                   0.00
00:00:00   821MB         |    userdata                          0.00
00:00:00   821MB         |  largest polymeshes by triangle count:
00:00:00   821MB         |          760 tris -- pSphereShape1
```

Texture Stats

Detailed statistics for image textures. These give insight into which textures use most memory, and which textures are untiled and would benefit from being converted to .tx files. The percentage of main cache misses should be very small for good performance, if it is high render time can be significantly increased. If the main cache misses are too high it helps to ensure only .tx files are used, reduce the number of textures, or tweak the texture settings.

```
00:00:00   841MB         | OpenImageIO Texture statistics
00:00:00   841MB         |   Queries/batches :
00:00:00   841MB         |     texture     :  261503 queries in 261503 batches
00:00:00   841MB         |     texture 3d  :  0 queries in 0 batches
00:00:00   841MB         |     shadow      :  0 queries in 0 batches
00:00:00   841MB         |     environment :  0 queries in 0 batches
00:00:00   841MB         |   Interpolations :
00:00:00   841MB         |     closest  : 0
00:00:00   841MB         |     bilinear : 398256
00:00:00   841MB         |     bicubic  : 701789
00:00:00   841MB         |   Average anisotropic probes : 2.8
00:00:00   841MB         |   Max anisotropy in the wild : 671
00:00:00   841MB         |
00:00:00   841MB         | OpenImageIO ImageCache statistics (000000003D574040) ver
1.5.15
00:00:00   841MB         |   Images : 1 unique
00:00:00   841MB         |     ImageInputs : 1 created, 1 current, 1 peak
00:00:00   841MB         |     Total size of all images referenced : 1.7 MB
00:00:00   841MB         |     Read from disk : 1.7 MB
00:00:00   841MB         |     File I/O time : 0.2s (0.0s average per thread)
00:00:00   841MB         |     File open time only : 0.0s
00:00:00   841MB         |   Tiles: 25 created, 24 current, 24 peak
00:00:00   841MB         |     total tile requests : 1454328
00:00:00   841MB         |     micro-cache misses : 35383 (2.43294%)
00:00:00   841MB         |     main cache misses : 25 (0.00171901%)
00:00:00   841MB         |     Peak cache memory : 2.3 MB
00:00:00   841MB         |   Image file statistics:
00:00:00   841MB         |          opens   tiles  MB read  I/O time  res
File
00:00:00   841MB         |       1    1       12      1.7      0.2s   768x 768x3.u8
C:/Users/Documents/example.tif UNTILED UNMIPPED MIP-COUNT [12,7,3,2,1,0,0,0,0,0]
00:00:00   841MB         |
00:00:00   841MB         |   Tot:     1       12      1.7      0.2s
00:00:00   841MB         |   1 not tiled, 1 not MIP-mapped
```

Shutdown

When Arnold is finished, it will release any resources, memory/threads and shutdown.

```
00:00:00   841MB         | releasing resources
00:00:00   831MB         |  unloading 27 plugins
00:00:00   829MB         |  unloading plugins done
00:00:00   828MB         | Arnold shutdown
```