Log Files

Table Of Contents

Introduction

During its execution, Redshift will print out a multitude of useful messages in your 3d app's script/console window. In order to avoid clutter, Redshift's default behavior is to only print out a subset of all the messages it generates. If you want to view *all* the messages in the script window, the user can set Redshift's verbosity level to "Debug". This option can be found in the System tab.
Apart from the 3d app's script/console window, Redshift stores all messages in log files.

Learning the structure and messages of the log file can be useful when diagnosing rendering errors and warnings such as an aborted render, unrecognized shading nodes, missing textures, etc. It can also help better understand the rendering process and know how (and when) to optimize your renders.

Log File Location

Redshift writes log files to a subdirectory of the log root path which is specified as follows:

Windows

```
%REDSHIFT_LOCALDATAPATH%\Log
```

Linux and macOS

```
$REDSHIFT_LOCALDATAPATH/log
```

If the environment variable `REDSHIFT_LOCALDATAPATH` is not defined, the default location is:

Windows

```
C:\ProgramData\Redshift\Log
```

Linux and macOS

```
~/redshift/log
```

Within the Redshift log folder you'll find a list of folders that might look like this

```
Log.Latest.0
Log.20150904_1236.0
Log.20150904_1231.0
Log.20150903_1950.0
… and so on
```

Each of these log folders represents a "render session". Assuming Redshift is installed and set to automatically initialize when your 3d app is launched, a new log folder will be generated each time you launch your 3d app. The most recent Redshift session is in the "Log.Latest.0" folder. The other folders will contain previous sessions.

The first part of the folder name is the date in Year-Month-Day form and the second part is the time. So, on the list shown above, the first folder with the name Log.20150904_1236.0 was created on 4 Sept 2015 at 12:36.

The number at the end of the folder name is used when multiple log files 'conflict' within the same hour:minute time. For example, if you open and close your 3d app very quickly then there will be multiple log files created in the same minute. When Redshift detects that, it will increment this last number of the folder name so you'll see folder names ending in .1 or .2 and so on. Such log folder conflicts can also happen when you're rendering with multiple instances of your 3d app – as example, when simultaneously rendering multiple frames via multiple instances of Maya.

Structure of a typical log file

Initialization

The first part of the log file prints out info about the Redshift version, the path locations and some basic CPU and GPU information. These messages are printed as part of Redshift's initialization stage.
A quick diagnostic is run on each GPU to measure its PCIe (PCI express) performance. PCIe (also known as 'the bus') is the computer component that connects your GPU with the remaining computer, so it can affect rendering performance. A bad motherboard driver or a hardware issue can adversely affect PCIe performance which, in turn, can reduce rendering performance. Ideally, the pinned memory bandwidth should be close to 5 or 6GB/s or higher.
The last line printed during initialization stage is also very important! It tells us how much GPU memory (aka "VRAM") Redshift will be able to use while rendering. If you have other 3d apps running at the same time, Redshift will be able to use less VRAM. As we can see here, even though our videocard is equipped with 4GB of memory, only 3.2GB of these could be used by Redshift. This is because this computer had other 3d apps as well as the Chrome web browser running – both of which can consume big quantities of VRAM. For more information about VRAM usage, please read this document.

```
14:17:01    481MB INFO:       Redshift for Maya 2016
14:17:01    481MB INFO:       Version 2.0.88, Mar 23 2017
14:17:01    481MB DETAILED:   Plugin filename: C:\ProgramData\Redshift\Plugins\Maya\2016\nt-x86-
64\redshift4maya.mll
14:17:01    481MB DETAILED:   Plugin path: C:\ProgramData\Redshift\Plugins\Maya\2016\nt-x86-64
14:17:01    481MB DETAILED:   Local data path: C:\ProgramData\Redshift
14:17:01    481MB DETAILED:   Core data path: C:\ProgramData\Redshift
14:17:01    481MB DETAILED:   Primary license path: C:\ProgramData\Redshift
14:17:01    481MB DETAILED:   Procedurals path: C:\ProgramData\Redshift\Procedurals;C:
\ProgramData\Redshift\Procedurals
14:17:01    481MB DETAILED:   Preferences file: C:\ProgramData\Redshift\preferences.xml
14:17:01    481MB DETAILED:   Extensions path: C:\ProgramData\Redshift\Plugins\Maya\2016\nt-x86-64\extensions;C:
\ProgramData\Redshift\Plugins\Maya\2016\nt-x86-64\extensions
14:17:01    493MB DETAILED:   Loading Redshift Maya extensions...
14:17:01    493MB DETAILED:           From path: C:\ProgramData\Redshift\Plugins\Maya\2016\nt-x86-64\extensions\
14:17:01    493MB DETAILED:                   C:\ProgramData\Redshift\Plugins\Maya\2016\nt-x86-
64\extensions\ornatrix_redshift.dll
14:17:01    493MB DETAILED:                   C:\ProgramData\Redshift\Plugins\Maya\2016\nt-x86-
64\extensions\shaveNode_redshift.dll
14:17:01    493MB DETAILED:                           Unable to load extension
14:17:01    493MB DETAILED:                   C:\ProgramData\Redshift\Plugins\Maya\2016\nt-x86-
64\extensions\xgen_redshift.dll
14:17:01    519MB DETAILED:           From path: C:\ProgramData\Redshift\Plugins\Maya\2016\nt-x86-64\extensions\
14:17:01    519MB DETAILED:                   C:\ProgramData\Redshift\Plugins\Maya\2016\nt-x86-
64\extensions\ornatrix_redshift.dll
14:17:01    519MB DETAILED:                   C:\ProgramData\Redshift\Plugins\Maya\2016\nt-x86-
64\extensions\shaveNode_redshift.dll
14:17:01    519MB DETAILED:                           Unable to load extension
14:17:01    519MB DETAILED:                   C:\ProgramData\Redshift\Plugins\Maya\2016\nt-x86-
64\extensions\xgen_redshift.dll
14:17:01    519MB DETAILED:           Done!
14:17:02    519MB DETAILED:   Cache path: C:\Users\pzobo\AppData\Local\Redshift\Cache
14:17:02    519MB DETAILED:   Redshift Initialized
14:17:02    519MB DETAILED:           Windows Platform (Windows 10 Pro)
14:17:02    519MB DETAILED:           Release Build
14:17:02    519MB DETAILED:           Number of CPU HW threads: 12
14:17:02    519MB DETAILED:           CPU speed: 3.50 GHz
14:17:02    519MB DETAILED:           Total system memory: 63.91 GB
14:17:02    520MB INFO:       License acquired
14:17:02    520MB INFO:       License returned
14:17:02    520MB INFO:       License for redshift-core 2017.11 valid until Nov 11 2017
14:17:02    521MB DETAILED:   Creating CUDA contexts
```

```
14:17:02    521MB DETAILED:           CUDA init ok
14:17:02    521MB DETAILED:           Ordinals: { 0 1 }
14:17:02    632MB DETAILED:   Initializing GPUComputing module (CUDA). Ordinal 0
14:17:02    632MB DETAILED:           CUDA Ver: 8000
14:17:02    632MB DETAILED:           Device 1/2 : Quadro GP100
14:17:02    632MB DETAILED:           Compute capability: 6.0
14:17:02    632MB DETAILED:           Num multiprocessors: 56
14:17:02    632MB DETAILED:           PCI busID: 3, deviceID: 0, domainID: 0
14:17:02    632MB DETAILED:           Theoretical memory bandwidth: 0.000000 GB/Sec
14:17:02    640MB DETAILED:           Measured PCIe bandwidth (pinned CPU->GPU): 5.655509 GB/s
14:17:02    640MB DETAILED:           Measured PCIe bandwidth (pinned GPU->CPU): 6.028213 GB/s
14:17:02    648MB DETAILED:           Measured PCIe bandwidth (paged CPU->GPU): 3.602716 GB/s
14:17:02    648MB DETAILED:           Measured PCIe bandwidth (paged GPU->CPU): 4.543489 GB/s
14:17:02    648MB DETAILED:           Estimated GPU->CPU latency (0): 0.031825 ms
14:17:02    648MB DETAILED:           Estimated GPU->CPU latency (1): 0.025376 ms
14:17:02    648MB DETAILED:           Estimated GPU->CPU latency (2): 0.024788 ms
14:17:02    648MB DETAILED:           Estimated GPU->CPU latency (3): 0.025170 ms
14:17:02    632MB DETAILED:           New CUDA context created
14:17:02    632MB DETAILED:           Available memory: 13175.9094 MB out of 16384.0000 MB
14:17:02    637MB DETAILED:   Initializing GPUComputing module (CUDA). Ordinal 1
14:17:02    637MB DETAILED:           CUDA Ver: 8000
14:17:02    637MB DETAILED:           Device 2/2 : TITAN X (Pascal)
14:17:02    637MB DETAILED:           Compute capability: 6.1
14:17:02    637MB DETAILED:           Num multiprocessors: 28
14:17:02    637MB DETAILED:           PCI busID: 2, deviceID: 0, domainID: 0
14:17:02    637MB DETAILED:           Theoretical memory bandwidth: 480.480011 GB/Sec
14:17:02    645MB DETAILED:           Measured PCIe bandwidth (pinned CPU->GPU): 5.701497 GB/s
14:17:02    645MB DETAILED:           Measured PCIe bandwidth (pinned GPU->CPU): 6.073571 GB/s
14:17:02    653MB DETAILED:           Measured PCIe bandwidth (paged CPU->GPU): 4.174829 GB/s
14:17:02    653MB DETAILED:           Measured PCIe bandwidth (paged GPU->CPU): 4.557991 GB/s
14:17:02    653MB DETAILED:           Estimated GPU->CPU latency (0): 0.030136 ms
14:17:03    653MB DETAILED:           Estimated GPU->CPU latency (1): 0.028127 ms
14:17:03    653MB DETAILED:           Estimated GPU->CPU latency (2): 0.024278 ms
14:17:03    653MB DETAILED:           Estimated GPU->CPU latency (3): 0.024197 ms
14:17:03    637MB DETAILED:           New CUDA context created
14:17:03    637MB DETAILED:           Available memory: 10247.7844 MB out of 12288.0000 MB
```

Scene Extraction

Scene extraction is the process during which Redshift is getting data from your 3d app into its own memory. That data includes your polygons, lights, materials, etc. If the scene has lots of objects or polygons the extraction stage can be long. If that's the case with your scene, we recommend using Redshift proxies.
Redshift also prints out the frame number which can be useful if the log file contains messages from several frames.
Redshift for Maya 2015 Version 1.2.90, Sep 3 2015 Rendering frame 1 Scene extraction time: 0.01s

Rendering – Preparation Stage

This is the first step of Redshift's rendering stage. Redshift does a first-pass processing of the scene's meshes for the ray tracing hierarchy. The ray tracing hierarchy is a data structure used during ray tracing in order to make rendering fast. Therefore, the building of the ray tracing hierarchy is a very important stage! If the scene contains many objects and/or polygons, the ray tracing hierarchy construction can take some time. Certain stages of tessellation /displacement can also happen here.
After the initial ray tracing hierarchy construction, Redshift does a pass over the scene textures and figures out if it needs to convert any of them. Redshift, by default, will convert all textures to its own internal format. If you want to avoid this kind of automatic processing, please use the texture processor and preprocess your textures offline.
For information on the GPU memory allocation, please read the next section.

```
18:16:51   1483MB INFO:
================================================================================================
18:16:51   1483MB INFO:         Rendering frame...
18:16:51   1483MB INFO:
================================================================================================
18:16:51   1486MB INFO:         License acquired
18:16:51   1486MB INFO:         License for redshift-core 2017.11 valid until Nov 11 2017
18:16:51   1486MB DEBUG:                0.0s
18:16:51   1486MB DEBUG:
18:16:51   1486MB DEBUG:        Preparing ray tracing hierarchy for meshes
18:16:51   1486MB DEBUG:                Time to process 0 meshes:   0ms
18:16:51   1486MB DEBUG:        Allocating GPU mem...(device 0)
18:16:51   1486MB DEBUG:                Done (CUDA reported free mem before: 12212 MB, after: 648 MB)
18:16:51   1486MB DEBUG:        Allocating GPU mem...(device 1)
18:16:51   1486MB DEBUG:                Done (CUDA reported free mem before: 9700 MB, after: 579 MB)
18:16:51   1486MB DEBUG:
18:16:51   1486MB DEBUG:        Allocating GPU mem for ray tracing hierarchy processing
18:16:51   1486MB DEBUG:                Allocating VRAM for device 0 (Quadro GP100)
18:16:51   1486MB DEBUG:                        Redshift can use up to 10240 MB
18:16:51   1487MB DEBUG:                        Geo/Tex: 9619 MB / 256 MB
18:16:51   1487MB DEBUG:                        NRPR: 262144 (364 MB)
18:16:51   1487MB DEBUG:                        Done! (0.0s). CUDA reported free mem: 655 MB
18:16:51   1487MB DEBUG:                Allocating VRAM for device 1 (TITAN X (Pascal))
18:16:51   1487MB DEBUG:                        Redshift can use up to 8714 MB
18:16:51   1488MB DEBUG:                        Geo/Tex: 8093 MB / 256 MB
18:16:51   1488MB DEBUG:                        NRPR: 262144 (364 MB)
18:16:51   1488MB DEBUG:                        Done! (0.0s). CUDA reported free mem: 579 MB
18:16:51   1488MB DEBUG:
18:16:51   1488MB DEBUG:        Ray Tracing Hierarchy Info:
18:16:51   1488MB DEBUG:                Max depth: 128. MaxNumLeafPrimitives: 8
18:16:51   1488MB DEBUG:                Extents: (-1.000000 -1.000000 -1.000000) - (1.000000 1.000000 1.000000)
18:16:51   1512MB DEBUG:                Time to create tree: 11 ms (0 11 0)
```

Rendering – Prepasses

Before Redshift can render the final frame, it has to execute all the prepasses. In this example, it computes the irradiance point cloud.
Other possible prepasses are:

- Photon mapping
- Irradiance Cache
- Subsurface scattering

At the start of each prepass, Redshift has to configure the GPU's memory (VRAM). Notice that, in this case, Redshift can use roughly up to 2.8GB of VRAM. You might ask "hold on, a few paragraphs above you said Redshift could use up to 3.2GB!". Well, the lower VRAM is because of two reasons:

- Redshift doesn't use 100% of the GPU's free VRAM. By default, it uses 90%. But you can grow or shrink that percentage in the Redshift System tab under the Memory section.
- Once Redshift initializes, it needs to use some VRAM (a few tens of MBs) for its operation. That memory is not counted here.

After VRAM has been allocated, Redshift prints out how much VRAM it didn't allocate and is now free. In this case it's 336MB. It's important to leave some VRAM free so that the 3d app and the operating system can function without issues.
At the end of each prepass, Redshift prints out how long the prepass took. In this case, the irradiance point cloud took 1.36 seconds.
Irradiance point cloud... Allocating VRAM for device 0 (Quadro K5000) Redshift can use up to 2860 MB Geo/Tex: 2028 MB / 256 MB NRPR: 262144 Done! (0.03s). CUDA reported free mem: 336 MB Total num points before: 55 (num new: 55) Total num points before: 85 (num new: 37) Total num points before: 144 (num new: 59) Total num points before: 209 (num new: 65) Total num points before: 318 (num new: 109) Total num points before: 442 (num new: 124) Total num points before: 560 (num new: 118) Total num points before: 678 (num new: 118) Total num points before: 799 (num new: 121) Total irradiance point cloud construction time 1.36s

Rendering – Final image

This is the final rendering stage where Redshift renders the image block-by-block. Blocks are also known as "buckets".
Notice that Redshift has to reallocate VRAM for this stage too, similar to what it did for the prepasses.
While blocks are rendered, Redshift will print out a line for each block so you can track the general progress. It also prints the time it took to render the block. Depending on the scene complexity, some blocks can take a longer time to render than others.
Like with the prepasses, Redshift finishes by printing out the total time taken to render the final image.

```
18:16:51   1510MB DEBUG:        Rendering blocks... (resolution: 960x540, block size: 128)
18:16:51   1510MB DEBUG:                Allocating VRAM for device 0 (Quadro GP100)
18:16:51   1510MB DEBUG:                        Redshift can use up to 10240 MB
18:16:51   1512MB DEBUG:                        Geo/Tex: 9125 MB / 256 MB
18:16:51   1512MB DEBUG:                        NRPR: 262144 (364 MB)
18:16:51   1512MB DEBUG:                        Done! (0.0s). CUDA reported free mem: 662 MB
18:16:51   1512MB DEBUG:                Allocating VRAM for device 1 (TITAN X (Pascal))
18:16:51   1512MB DEBUG:                        Redshift can use up to 8714 MB
18:16:51   1513MB DEBUG:                        Geo/Tex: 7676 MB / 256 MB
18:16:51   1513MB DEBUG:                        NRPR: 262144 (364 MB)
18:16:51   1513MB DEBUG:                        Done! (0.0s). CUDA reported free mem: 579 MB
18:16:51   1518MB DEBUG:                Block 2/40 (0,1) rendered by GPU 0 in 4ms
18:16:51   1519MB DEBUG:                Block 1/40 (0,0) rendered by GPU 1 in 5ms
18:16:51   1519MB DEBUG:                Block 3/40 (1,1) rendered by GPU 0 in 3ms
18:16:51   1519MB DEBUG:                Block 4/40 (1,0) rendered by GPU 1 in 4ms
...
...
...
18:16:52   1568MB DEBUG:                Block 39/40 (6,0) rendered by GPU 0 in 19ms
18:16:52   1568MB DEBUG:                Block 40/40 (6,1) rendered by GPU 1 in 47ms
18:16:52   1563MB DEBUG:                Time to render 40 blocks: 0.5s
18:16:52   1562MB INFO:         Rendering time: 1.3s (2 GPU(s) used)
```

Statistics

This is a very important part of the log file! It provides various scene statistics such as the number of proxies, meshes, lights, unique polygons, polygons with instancing, etc.Knowing such statistics can be useful when profiling a scene. As an example, scenes with many lights can take a longer time to render than scenes with few lights. Using the statistics, you can quickly find out if the scene contains many lights.

You can also quickly find out how many unique polygons or strand segments (for hair) there are in the scene. Please note that "unique" means "non-instanced". Say you're modelling a forest and you have the same 1-million-polygon tree instanced 1000 times, you will see something like "1 million unique triangles" and "1 billion total triangles".

Perhaps the most important data entries are the ones referring to GPU memory.

At the very bottom of the log file we can see the "Ray acceleration and geometry memory breakdown". These figures refer to the polygons and splines (primitives) in our scene. The sum of these figures is how much GPU memory would be required to fit all the primitives without requiring any out-of-core-access.

A few lines higher, we can see the "GPU Memory" section. The "uploads" figures in it tell us how much PCIe traffic was required to send the data to the GPU. Those figures might be smaller than the "Ray acc. And geometry memory breakdown" total. This is because Redshift only sends data it absolutely needs to the GPU. In this example, less PCIe traffic was required because there were was a percentage of polygons that wasn't visible by the camera. There are other cases, however, where data might have to be re-sent to the GPU due to out-of-core access (or for other reasons). In such cases, the 'uploads' can potentially be (much) larger than the sum of 'Ray acc. And geometry memory breakdown'.

When profiling performance, you should mostly worry about the 'upload' figures, especially if they go into the "many gigabytes" range. This applies for both geometry and texture uploads.

```
18:16:52   1562MB DEBUG:        Scene statistics
18:16:52   1562MB DEBUG:
18:16:52   1562MB DEBUG:                General counts
18:16:52   1562MB DEBUG:                        Proxies:                                    0
18:16:52   1562MB DEBUG:                        Proxy instances:                            0
18:16:52   1562MB DEBUG:                        Meshes:                                     1 (1 TriMeshes,
0 HairMeshes)
18:16:52   1562MB DEBUG:                        Instances:                                  0
18:16:52   1562MB DEBUG:                        Point cloud points:                         0
18:16:52   1562MB DEBUG:                        Lights:                                     1
18:16:52   1562MB DEBUG:                        Volume grids:                               0 (0 unique)
18:16:52   1562MB DEBUG:                        Sprite textures:                            0
18:16:52   1562MB DEBUG:
18:16:52   1562MB DEBUG:                Geometry
18:16:52   1562MB DEBUG:                        Unique triangles pre tessellation:        760
18:16:52   1562MB DEBUG:                        Unique triangles post tessellation:       760
18:16:52   1562MB DEBUG:                        Unique points:                              0
18:16:52   1562MB DEBUG:                        Unique hair strands:                        0
18:16:52   1562MB DEBUG:                        Unique hair strand segments:                0
18:16:52   1562MB DEBUG:                        Total triangles:                          760
18:16:52   1562MB DEBUG:                        Total points:                               0
18:16:52   1562MB DEBUG:                        Total hair strands:                         0
18:16:52   1562MB DEBUG:                        Total hair strand segments:                 0
18:16:52   1562MB DEBUG:
18:16:52   1562MB DEBUG:                Largest triangle meshes:
18:16:52   1562MB DEBUG:                                760 triangles : pSphereShape1
18:16:52   1562MB DEBUG:
18:16:52   1562MB DEBUG:                GPU Memory
18:16:52   1562MB DEBUG:                        Device  0 geometry PCIe uploads:          3.02 KB
(cachesize:         8.91 GB)
18:16:52   1562MB DEBUG:                        Device  0 texture PCIe uploads:             0 B
(cachesize:       256.58 MB)
18:16:52   1562MB DEBUG:                        Device  1 geometry PCIe uploads:          3.02 KB
(cachesize:         7.50 GB)
18:16:52   1562MB DEBUG:                        Device  1 texture PCIe uploads:             0 B
(cachesize:       256.58 MB)
18:16:52   1562MB DEBUG:                        Matrices (for instances/points):            0 B
18:16:52   1562MB DEBUG:                        Rays:                                  364.72 MB
18:16:52   1562MB DEBUG:                        Sprite textures:                            0 B
18:16:52   1562MB DEBUG:                        Volume grids:                               0 B
18:16:52   1562MB DEBUG:
18:16:52   1562MB DEBUG:                Textures
18:16:52   1562MB DEBUG:                        Device  0 stream and upload time:           0ms
18:16:52   1562MB DEBUG:                                File loading time:                      0ms
18:16:52   1562MB DEBUG:                                File decompression time:                0ms
18:16:52   1562MB DEBUG:                                Average GPU cache hits:                 0%
18:16:52   1562MB DEBUG:                        Device  1 stream and upload time:           0ms
18:16:52   1562MB DEBUG:                                File loading time:                      0ms
18:16:52   1562MB DEBUG:                                File decompression time:                0ms
18:16:52   1562MB DEBUG:                                Average GPU cache hits:                 0%
18:16:52   1562MB DEBUG:
18:16:52   1562MB DEBUG:                GPU Ray Accel. And Geometry Memory Stats (rough)
18:16:52   1562MB DEBUG:                        Acceleration Structures:                 9.94 KB
18:16:52   1562MB DEBUG:                        Main primitive data:                     8.94 KB
18:16:52   1562MB DEBUG:                        Extra primitive data:                    3.02 KB
18:16:52   1562MB DEBUG:                        Primitive loading time:                     0ms
```

End Of Frame

When Redshift has finished rendering the frame, it saves the image and prints out how much time the frame took in total. The total time includes extraction, preparation, prepasses and final rendering times.

```
18:16:52   1561MB INFO:         Saved file 'C:\Users\pzobo\Documents\maya\projects\default\images\tmp\untitled.
iff' in 0.03s
18:16:52   1552MB INFO:         Rendering done - total time for frame 1: 1.40s
```