6.2.1.0

22 Apr 2021

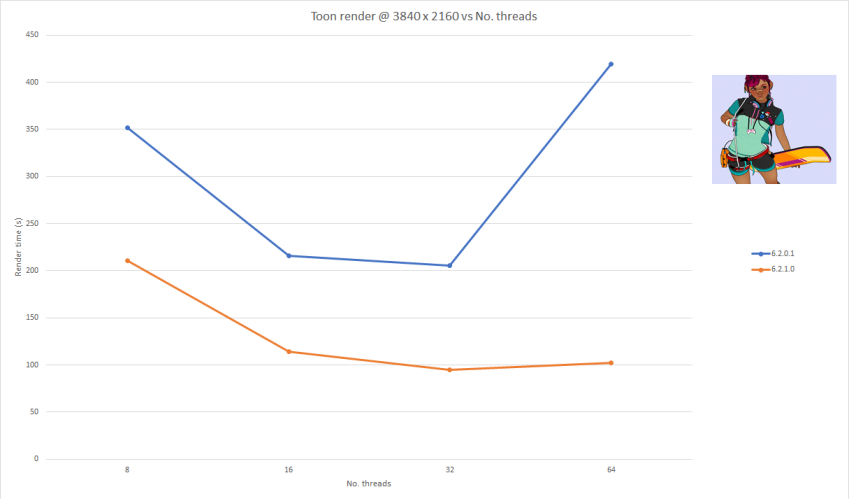
Arnold 6.2.1 is a performance release bringing important optimizations on GPU, toon shading, and imagers.

System Requirements

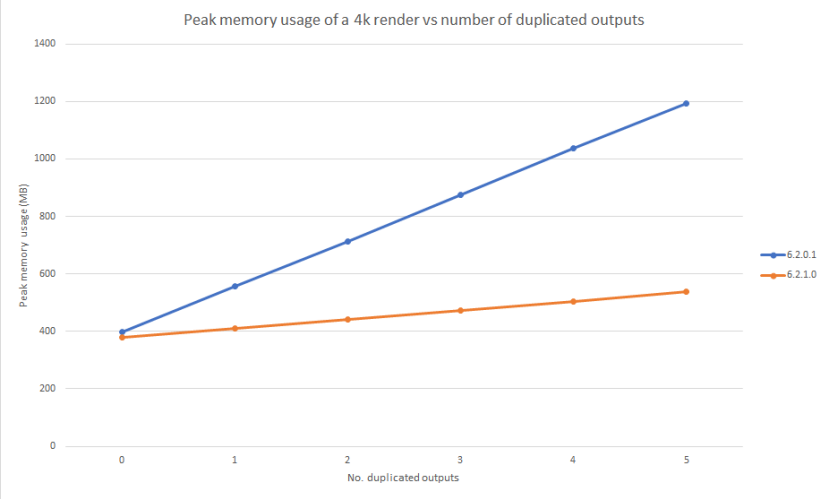
- Windows 10 or later, with the Visual Studio 2019 redistributable.
- Linux with at least glibc 2.17 and libstdc++ 4.8.5 (gcc 4.8.5). This is equivalent to RHEL/CentOS 7.
- macOS 10.13 or later
- CPUs need to support the SSE4.1 instruction set.
- GPU rendering works on Windows and Linux only and requires an NVIDIA GPU of the Ampere, Turing, Volta, Pascal, or Maxwell architecture. We recommend using the 460.39 or higher drivers on Linux and 461.40 (Quadro), 461.40 (GeForce), or higher on Windows. See Getting Started with Arnold GPU for more information.
- Optix™ denoiser requires an NVidia GPU with CUDA™ Compute Capability 5.0 and above.

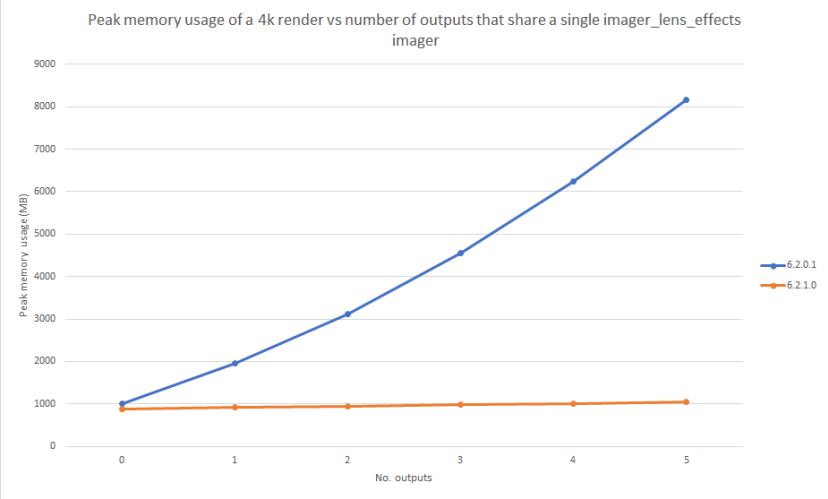
Enhancements

- **Toon shader (contour_filter) optimizations:** The toon shader has been optimized, especially for Windows machines with many cores and many small buckets, wherein in some situations we've seen over a 4x speedup (core#10414, #10242).



- **Faster renders using progressive rendering with adaptive sampling on CPU:** When using progressive rendering with adaptive sampling on the CPU, Arnold will now more quickly end the render when there is no more work left to do. For high AA_max renders this can result in noticeably faster renders (core#9022).
- **Faster imagers:** Imagers are faster, especially on many-core Windows machines where we have seen up to a 1.7x speedup with the imager_denoiser_noice. (core#10284, 10341, 10384).
- **Improved output memory usage:** The memory usage of outputs has been improved in a variety of cases. If a single output is sent to multiple drivers the memory is now shared, rather than duplicated (see the first graph below). The memory gains are even more substantial when imagers chains are shared between drivers. Again previously this would lead to much-duplicated memory that is now shared (see second graph below). (core#10214)

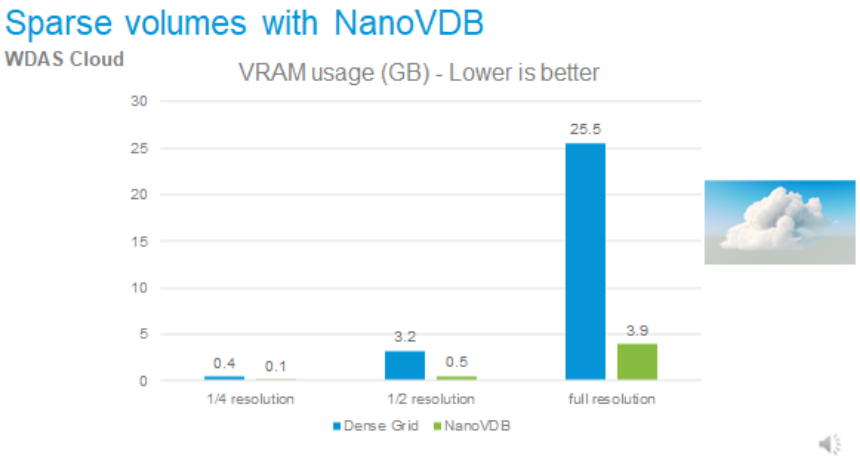


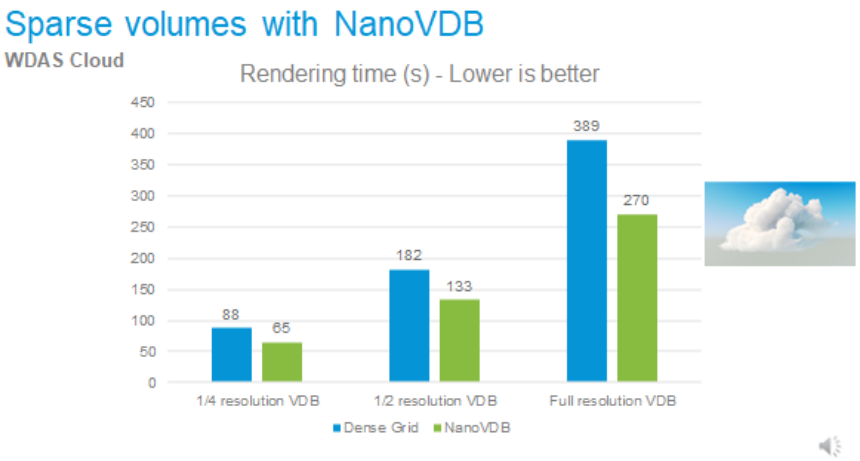


- **Better progress status updates for progressive renders:** Progressive mode renders, which are always enabled on GPU and on CPU are enabled with options.enable_progressive_render, now have more accurate progress reports when doing adaptive sampling. They will also now report to the log an estimated time before the render completes (core#10406).
- **OpenImageIO upgraded:** OIIO has been upgraded to version 2.3.2 (core#9705, #9481).

GPU Enhancements

- **Sparse volumes on GPU**: VDB volumes are now loaded on GPU in a sparse form using NVIDIA's NanoVDB library. This can lead to a very significant reduction in GPU memory usage in scenes with volumes, depending on the volume sparsity, compared to the previous dense grid implementation, as shown in the table below. This also produced a speedup due to the faster skipping of empty space possible with NanoVDB. (core#9626, core#10333).

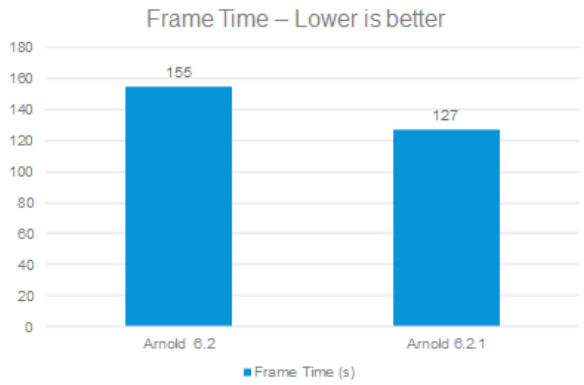




scene	GPU volume render time (min:sec)		GPU volume memory usage (MB)	
	dense grid	**NanoVDB**	dense grid	**NanoVDB**
wdas_cloud(quart)	1:28.65	1:05.91 (**1.35x** faster)	420	92 (**4.6x** smaller)
wdas_cloud(half)	3:02.00	2:12.33 (**1.38x** faster)	3222	576 (**5.6x** smaller)
wdas_cloud(full)	6:19.02	4:30.64 (**1.40x** faster)	25568	3974 (**6.4x** smaller)
explosion	1:18.20	1:10.71 (**1.11x** faster)	108	48 (**2.3x** smaller)
bunny level set	0:08.36	0:06.93 (**1.21x** faster)	788	102 (**7.7x** smaller)
buddha level set	0:14.29	0:12.97 (**1.10x** faster)	1514	202 (**7.5x** smaller)
crawler level set	1:38.56	1:11.08 (**1.39x** faster)	11276	1906 (**5.9x** smaller)

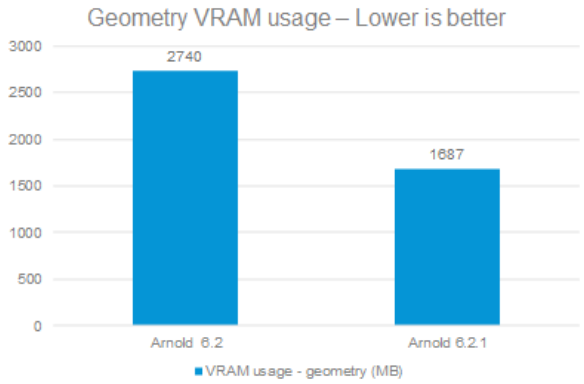
- **Faster SSS on GPU**: This overall GPU optimization is most beneficial to scenes with lots for subsurface scattering, where we have observed up to 1.2x speedups (core#9917).

Improving performance



- **Reduced VRAM use for polymeshes**: The amount of VRAM used by polymesh is significantly lower in this version. Scenes with heavy subdivisions will use around 33% less GPU memory for geometry (core#10462).

Reducing polymesh memory usage



Model: Jaeyeon Nam

- **Faster GPU render passes:** Each render iteration on the GPU has some CPU overhead which has been reduced. This will result in an overall GPU render time reduction up to 1.7x in fast to render scenes. In particular, when the resolution is low or adaptive sampling has reduced the amount of work left to do (core#10405).
- **PRef AOV:** The PRef AOV is now available when rendering on GPU (core#10444)).

USD Enhancements

- **Light and Shadow linking:** The render delegate now supports light and shadow linking. (usd#412)
- **Motion blur for the Point Instancer:** The render delegate now calculates motion blur when using the point instancer. (usd#653)
- **Half and Double precision:** Storing data using half or double precision is now supported in both the render delegate and procedural. (usd#669)
- **Pause and Resume:** Pausing and resuming renders are now supported in the render delegate. (usd#595)
- **NodeGraph schemas:** The procedural now supports using the NodeGraph schema for shader networks. (usd#678)
- **Crease Sets:** The procedural now supports crease sets on polymesh. (usd#694)
- **Purpose:** Usd Purpose is now supported in the procedural. (usd#698)
- **Transform2D:** The procedural now supports remapping UsdTransform2D to built-in Arnold nodes. (usd#517)
- **Write with default values**: The scene format now supports optionally writing parameters with default values. (usd#720)
- **Velocity blur**: The procedural now uses the velocity attribute to create motion keys for point-based shapes, when there are no position keys or the topology changes between frames. (usd#221)
- **Multi-Threading**: The procedural now uses USD's WorkDispatcher which improves the performance of multi-threaded expansion in many cases. Examples of the performance improvements. (usd#690)

API Additions

- **Output information exposed via AtOutputIterator API:** Using the following new API functions, it is now possible to query the filter, floating-point bit depth, layer name, and camera of the currently active output of an AtOutputIterator. (core#9860)

```
AI_API AtNode*  AiOutputIteratorGetFilter(struct AtOutputIterator* iter);
AI_API bool     AiOutputIteratorIsHalf(struct AtOutputIterator* iter);
AI_API AtString AiOutputIteratorGetLayerName(struct AtOutputIterator* iter);
AI_API AtNode*  AiOutputIteratorGetCamera(struct AtOutputIterator* iter);
```

- **License availability check API**: The newly added function *AiLicenseIsAvailable()* checks if there are valid available licenses.

Incompatible Changes

- By default, Arnold licenses are now checked out using the **host computer name**. In previous Arnold versions, licenses were checked out with the name **arnold.user**

  This means that Arnold 6.2.1.0 and an older Arnold, like 6.2.0.1, won't share a license if you render with both versions on the same time on the same machine.

  To make Arnold 6.2.1 share a license with an older Arnold, set the environment variable **ARNOLD_LICENSE_CLM_USERNAME** to **arnold.user**

  ARNOLD_LICENSE_CLM_USERNAME supports the tokens **<username>** and **<hostname>**, so with Arnold 6.2.1 and later you can customize the name used to check out the license.

  It could be just **<username>**, which would expand to to something like "blairs"

  Or it could be something like **arnold.<username>.<hostname>**, which would expand to something like "arnold.blairs.NOVMJ07M9WM"

Bug Fixes

- core#10385 [Alembic] The transform_type parameter on the alembic procedural is not applied to child nodes
- core#10459 Crash in AiProceduralViewport with curves inside procedurals in bounding box mode
- core#10410 Crash when rendering EXR files in append mode at high resolution using many AOVs
- core#10386 Crash with single sided implicit objects
- core#7562 [GPU] Crash when destroying some procedurals
- core#10390 [GPU] Crash when loading empty points through a procedural
- core#10422 [GPU] Crash when switching shader presets in Bifrost Graph scene
- core#10478 [GPU] Crash with non-Nvidia GPUs with an old Nvidia driver installed
- core#9682 [GPU] Error during render when reaching the memory limit on one GPU with NVLink
- core#10431 [GPU] Incorrect result and crashes with motion blur on curves when not using the default shutter
- core#10352 [GPU] Incorrect Z AOV on background
- core#10328 [GPU] Overscan region is empty
- core#10419 [GPU] Texture behind glass is blurred
- core#10389 [GPU] Transforming procedurals in IPR does not update geometry position
- core#9751 [GPU] Wrong texture mipmap selected after ray goes through a transparent object
- core#10445 [Imagers] Artifacts with small filmic_shoulder_length value in imager_tonemap
- core#10461 [Imagers] imager_denoiser_noice does not write alpha when using output_suffix
- core#10253 [Imagers] Random crashes when interactively editing the light_mixer channels
- core#10382 [Imagers] Random crash in imager_denoiser_noice when feature buffers are not of type RGB
- core#10409 [MaterialX] Material assignments not working when using <surface_material> in the document
- core#3642 Minor memory leak (464 bytes) when starting a new render session
- core#10432 [OCIO] Unable to determine chromaticities using an aces_1.2 config without an sRGB curve
- core#10279 [OpenVDB] Block artifacts when VDB volumes overlap standard surface interiors
- core#10106 [OSL] Incorrect results when connecting G and B components of an OSL node to another OSL node
- core#10452 Rare hang with C4DToA on Linux
- core#10350 Render progress is not reset after AiRenderBegin
- usd#668 The render delegate does not convert HdInterpolationVarying primvars
- usd#651 Error rendering Usd file with samples in productName
- usd#615 Usd Writer crashes when node name contains a hyphen character
- usd#683 Don't apply skinning if the Usd stage comes from the cache
- usd#508 Nested procedurals ignore matrix in the viewport API
- usd#687 Crash with empty primvar arrays
- usd#679 Attribute subdiv_type should have priority over Usd subdivisionScheme
- usd#282 Primvars are not inherited from ancestor primitives
- usd#579 Subdivision settings not being passed to Usd procedural
- usd#660 Crease Sets and Subdivision scheme is not imported correctly
- usd#215 Issue with instanced primitives' visibility (procedural)
- usd#244 Curves with vertex interpolation on width (procedural)

- usd#718 Inactive render vars are still rendered when using the scene format
- usd#727 Arnold does not use wrapS and wrapT values on UsdUVTexture shader node when rendering UsdPreviewSurface
- usd#724 ID not passed to the shapes generated in the procedural