4.2.0.0

Milestone 4.2

Enhancements

- **Stereo/multicam rendering**: The output statements have been extended to accept an optional camera name as the first parameter in an outputs string. Custom exporters or host application plugins can use this to setup multi-camera renders. The new additional syntax is:

  ```
  "camera AOV_name AOV_type filter driver" # for a regular driver
  "camera driver" # for a raw driver
  ```

  Multi-camera renders of the same scene can share the reading in and processing of files, acceleration structures, tessellation, displacement and texture cache by having the different cameras render sequentially. Interleaved bucket rendering for different cameras or sending the results from multiple cameras to the same multi-layer driver is not yet supported. (#2404)
- **Improved motion blur sampling quality**: Motion blur sampling has been improved, resulting in less noise given the same AA sample and time budget. The improvement is most prominent in scenes where motion blur is the primary source of noise in the image. There are now two possible values for the AA_motionblur_pattern option: the new default pattern, and the legacy jittered pattern which is provided as a backup until the new sampler is sufficiently battle-tested in production. (#2131, #4037, #4044, #4099)
- **Improved glossy sampling quality**: Some scenes will now render with slightly less noise given the same amount of samples and time. In particular, microfacet-based BRDFs viewed at grazing angles will show significant improvement. (#3571, #3888, #3900)
- **Improved CPU utilization**: The thread scheduler has been redesigned achieving much better CPU utilization in scenes that get stuck in a particularly slow bucket near the end of the frame. It is no longer necessary to use very small buckets to prevent these slowdowns. (#3873)
- **Improved memory performance and usage in Linux**: Linux-specific optimizations have been added which reduce memory usage and improve the threading scalability in certain situations, usually during scene preprocessing and shutdown. These require matched AiMalloc() / AiFree()allocations. If you allocate memory in your shader/procedural with AiMalloc() and then free it with free(), it is possible Arnold could crash. (#3779)
- **Faster ray accel build**: Acceleration structure construction is now up to 1.3x faster while using the same memory. (#3898)
- **Faster subdivision**: Catmull-Clark mesh subdivision is now up to 1.5x faster and uses about 5% less peak memory. (#3903, #3913, #3949)
- **Faster displacement**: Polymesh vertex displacement has been sped up, particularly for lower thread counts. On some highly subdivided models we've seen around 2x faster displacement even with 8 threads. In addition, vector displacement artifacts on UV seams have been eliminated. (#4008, #4017)
- **Faster polymesh processing**: Polymesh sanity checks are now about 3x faster, so we have removed the seldomly used options. mesh_sanity setting and now always perform these checks. Mesh processing/optimization time is now reported in the stats and this step is about 1.5x faster than before. (#3912, #3917, #3927, #3952, #3986)
- **Faster rendering of many-light scenes**: Scenes with many spaced out lights will now render faster thanks to improved culling of low contribution lights. We've seen up to 2x faster renders in some challenging production scenes. (#4100)
- **Faster SSS**: Raytraced diffusion-based SSS should now show higher performance on objects that have not been assigned to be a part of an SSS set via a "sss_setname" user data string, specially on scenes with spatially overlapping objects, like hair over skin. (#3808)
- **Faster image shader**: Texture map lookups coming from the image shader are now significantly faster. The noise shader has been made slightly faster too. (#3963, #4114)
- **Faster AiNodeGet* and motion vectors**: AiNodeGet*() is now substantially faster on multi-threaded machines. Before, excessively calling these functions would cause Arnold to not scale to multiple threads. Now Arnold is able to scale when these functions are called during rendering. There is still a substantial overhead for calling these functions and so AiNodeGet*() usage is still discouraged from being called during shader evaluate. Usage of AiWorldToScreenMatrix(), AiShaderGlobalsGetPixelMotionVector(), and AiDriverGetMatrices() are all substantially faster now and will scale to multiple threads. One benefit of this is that computing motion vectors is now substantially faster. (#3096, #3836, #3849)
- **Faster string processing**: We now scale better when handling strings. For instance, if many threads are frequently making calls that pass in strings, such as AiNodeDeclare(mynode, "some_string", "uniform RGB") we will now scale to all these threads and execute this call more quickly. This matters mostly in deferred loading of geometry from DSO/.ass procedural nodes. (#3921, #3957)
- **Faster AiNode()**: Performance scaling has been improved when many threads are rapidly calling AiNode(), as it can happen with deferred loading of geometry from DSO/.ass procedural nodes. Under normal situations, this bottleneck is not hit and so no speedup will be seen. (#3954)
- **Faster deep output using less memory**: Renders that output deep images are now up to 15% faster and use less than half the memory. (#4081, #4102)
- **Faster AOV composition with no memory overhead**: The memory overhead incurred when usingoptions.enable_aov_composition has been eliminated. This can save gigabytes in highly-threaded, high AA, multi-AOV renders. Complex renders with lots of transparency and AOVs could be up to 20% faster. (#4038, #4065)
- **Faster image output**: Taking advantage of the non-locking process_bucket API call introduced in 4.1, we have increased concurrency and improved the speed at which AOVs are written to disk, with a 15% to 30% speedup in built-in drivers like driver_exr, driver_tiff, etc. Custom drivers written by third-party developers should be updated accordingly for maximum performance. (#3857)
- **Faster backlighting**: The translucency, or "backlighting" computation in the standard shader, controlled with the Kb parameter, will use on average 50% fewer shadow rays. (#3890)
- **Skip skydome importance table if color is not linked**: Skydome lights will no longer build their importance tables when the color is not linked to anything. This can save a few seconds of preprocessing time when using high resolution tables. Also, IPR renders will now automatically respond to changing the skydome color. (#3020)
- **autobump in SSS**: Arnold can now optionally take into account the effect that autobump has on the ray traced BSSRDF's result via thesss_use_autobump render option. This helps capture the high frequency details of the surface more accurately, at the expense of slightly longer render times. For backwards compatibility and performance reasons, this option defaults to false. (#3872, #3894)
- **Smoother bump near shadow terminator**: We have fixed long-standing faceting issues near the shadow terminator boundary when using strong bump mapping. The improvement is most noticeable with detailed, high-frequency maps or when seen from far away. (#3891)

- **Reduced edge darkening in microfacet BSDFs**: Thanks to a new less aggressive shadowing/masking term, the built-in Cook-Torrance BRDF and microfacet BTDF now conserve a bit more energy and exhibit less edge darkening, particularly at high roughness settings. The impact is expected to be very subtle in most scenes. (#4043)
- **Anisotropic Cook-Torrance BRDF**: The Cook-Torrance BRDF functions in the shading API as well as the standard shader now actually make use of the anisotropic shading parameters. This BRDF has an anisotropic shading effect that is very similar to the anisotropy in the Ward-Duer BRDF in behavior, but with a greater amount of energy conservation and less edge darkening. (#4042)
- **Improved ambient and reflection occlusion**: Ambient occlusion artifacts around edges of low-poly objects have been cleaned up. In addition, the object visibility attributes (visibility, self_shadows) or the minimum occlusion distance (mint > 0 in the AiOcclusion call itself) can now be used to remove artifacts near silhouettes for reflection occlusion, or to disable occlusion from bump-mapped normals. (#3972)
- **Ambocc distance in utility shader**: Added a new ao_distance parameter in the utility shader to control the length of ambocc rays for scenes where the default of 100 units is too short. (#3861)
- **Refraction opacity AOV**: The standard shader now can output refraction opacity as an AOV. It is advised to write the complete RGBA set of channels, since the alpha component will constitute a mask that can be used to mix it in with the regular opacity, multiplying in refraction opacity to regular opacity based on the mask. (#2880)
- **Motion-blurred rolling shutter**: Cameras now have a rolling_shutter_duration parameter with which it is possible to control the duration of exposure of the scanlines in a rolling shutter camera. Valid values for this parameter are in the 0 to 1 range, where a value of 0 gives you an instantaneous exposure of each scanline (the default value and the rolling shutter's previous behavior), and a value of 1 exposes every scanline for the entirety of the camera's shutter interval (the same result that a camera without rolling shutter would give). (#4009)
- **Support for custom shutter shapes**: Added an additional curve value to the shutter_type enum in cameras. This, in conjunction with a new camera parameter shutter_curve, allows arbitrary shutter shapes, linearly interpolating shutter open/closed values between points. You can define as many points as needed. Coordinates are increasing in X from 0 (corresponding to shutter_start) to 1 (corresponding to shutter_end), and values in Y must be non-negative. (#3934)

```
# example shutter_curve usage
persp_camera
{
 ...
 # double-trapezoid shutter with outer edges being more gentle,
 # and the first trapezoid emphasized much more
 shutter_type  curve
 shutter_curve 8 1 POINT2
    0.0  0.0
    0.2  3.0
    0.3  3.0
    0.4  0.0
    0.6  0.0
    0.7  1.0
    0.8  1.0
    1.0  0.0
}
```

- **Volume plugins**: A new node volume has been added which allows volume plugin DSOs (specified via the dso parameter of a volume node). Volume plugins allow a plugin writer to wrap any volume format or generator for efficient rendering. Volume plugins provide volume creation, destruction, channel sampling, and gathering of ray extents, as well as an automatic bounding box and step size for ray marching (although these can be overridden by the user in the volume node). Ray extent computations inside the plugin will tightly bound volume data along the ray so that the number of volume shader evaluations are drastically reduced in sparse volume datasets. For a short tutorial on how to implement your own volume plugin DSO, see this article on the arnoldpedia. (#3466, #3978, #3996, #4045, #4046)
- **Density shader**: A new built-in shader density is available which uses AiVolumeSampleChannel() to sample from the new volume plugins. This allows volume sampling to work regardless of what source the volume data comes in (as long as there is a volume plugin for the source). Currently it has parameters for controlling RGB scattering, absorption, emission, forward vs backward scattering strength, interpolation quality, and offsetting the sampling position (for example with a noise shader to create the appearance of a higher resolution volume than what came from the source data). (#3466, #3994)
- **Static OIIO linking in Windows**: Just like we do in Linux and OSX, we now statically link the OpenImageIO library in Windows too; we don't distribute the file OpenImageIO.dll anymore. (#3969)
- **Scene update on "free" mode**: When using the "free" render mode, you can now force a scene update using AiRender (AI_RENDER_MODE_FREE), which will take into account parameter changes and nodes created or destroyed. (#3862)
- **Multithreading in procedurals**: The enable_threaded_procedurals option no longer affects the loading of the text-based .ass, .obj and .ply formats, which are now always loaded in parallel. This option will still apply to binary procedurals (.so/.dll/.dylib), forcing sequential processing when disabled. (#4096)
- **IES files in texture_searchpath**: If not found in the default path, photometric IES files are now searched for in texture_searchpath too. (#4036)

API additions

- **is_perspective metadata**: View-dependent geometry effects like min_pixel_width and adaptive subdivision are now supported in custom camera plugins that have a float fov parameter. This requires the developer of the custom camera to set the boolean is_perspective metadata to true. An example is given below. NOTE: we reserve the right to supersede this with a better API at some point in the future. (#2756)

```
node_parameters
{
   AiParameterFlt("fov", 60.0);
   AiMetaDataSetBool(mds, NULL, "is_perspective", true);
}
```

- **AiPoint() and AiVector()**: A new API function AiPoint() has been added which is aimed at replacing the old and deprecated API AiV3Create(). An equivalent function AiVector() has been added, simply for naming consistency. Analogously, 2D functions AiPoint2() and AiVector2() have also been added. These new APIs allow writing shorter, more concise code in certain cases. (#3936)
- **Volume plugin API**: Volume plugins provide callbacks very similar to procedural plugins. The callbacks are for initialization and takedown of the plugin, initialization and destruction of volumes, sampling of volumes, and providing tight ray extents along a ray where there is volumetric data to sample (via AiVolumeAddIntersection()). Volume plugins may query declared user parameters from the volume node to customize their loading of volume data as needed. Volume plugins are expected to provide a bounding box around the volume data, as well as a recommended step size for ray marching the data. Shaders may query the volume plugin for data on specific named channels via AiVolumeSampleChannel(). Please see the API documentation for details and this short tutorial article. (#3466)
- **External memory tracking**: A new API function AI_API void AiAddMemUsage(AtInt64 size, const char* category) is available for plugin writers and API users to inform Arnold when they allocate and deallocate memory. The string category used will show up in log files under the memory stats at the end of each AiRender() call. It is strongly advised that all shaders, plugins and procedurals that need to allocate memory make use of this new API so that log files can tell the true story of where memory is being used, otherwise it all goes into the "unaccounted" line of the memory stats. (#3948)
- **Procedural with variable number of nodes**: Procedural nodes are now allowed to return 0 from the NumNodes() callback, to avoid having to return a fixed number of nodes when that might not be known in advance. In this case, the GetNode() callback will be called in sequence until it returns NULL. (#3971)
- **Reset parameter to default value**: A new function AiNodeResetParameter() is provided to reset a node parameter to its default value, also removing any links to that parameter. (#4071)

Incompatible changes

- **Empty sss_setname strings ignored**: Objects with SSS that have been assigned to be part of an SSS set via an empty "sss_setname" user data string will no longer be considered as part of the same set, which may result in differences in the rendered result. (#3923)
- **Removed cubic projection map**: This old projection map is rarely used, has seam artifacts along cube edges, and is the least efficient of the various possible cubic formats, so we have decided to remove its support in the sky and skydome_light nodes. Note that we have left the C API intact,AiMappingCubicMap() and AiMappingCubicMapDerivs(), in order not to break binary compatibility in the rare event that anybody was using these API calls. These calls are now marked as deprecated and will be removed in a future release. (#3937)
- **edgelength changes**: The edge_length color mode in the utility shader now works on any mesh, regardless of whether it's subdivided or not. This visualization is handy for detecting over-tessellated objects. The old behaviour has been moved to a new mode called pixelerror, which is similar but based on how well the polygon matches subdiv_pixel_error. (#3956)
- **Early abort when no outputs are present**: When no output drivers are present the render will now be canceled. This has the side effect that progressive kick renders will be aborted when no display driver is present. If for debugging reasons you wish a render to continue even without valid outputs you can use options.abort_on_error false. (#3668)
- **Deep EXR driver tiled mode disabled**: While we upgrade to the next OIIO version we need to disable tiled support for the deep driver since it is crashing on OIIO 1.2.2. Note that tiled, deep EXR files are not supported by Nuke anyway. (#4013)
- **Parameters of pointer type not written to .ass**: Since any pointer to memory cannot be preserved by writing and later reading from an .ass file, we have disabled .ass writing for such pointer-type parameters. (#3860)
- **Removed enable_threaded_displacement option**: There have been no reported issues requiring this fallback option to be turned off, so we are retiring it and making threaded displacement always on. (#4054)
- **Anisotropic Cook-Torrance BRDF**: Now that the Cook-Torrance BRDF functions actually make use of the anisotropic shading parameters, they are more sensitive to "junk" data which could cause erroneous results or even crashes if the shader code is not careful. It is recommended that shaders making use of the Cook-Torrance BRDFs be revised so that the normal, tangent and bitangent parameters form an orthonormal basis with each other (when provided), and that both of the roughness parameters take on reasonable values. (#4042)
- **Removed specular_brdf from standard shader**: Since the Cook-Torrance is similar in performance to the Ward-Duer BRDF yet superior in quality due to its reduced edge darkening and higher energy conservation, the specular_brdf parameter has been removed from standard, leaving Cook-Torrance as the default (and only!) BRDF of the standard shader. Therefore, the look of existing standard shader-based materials that use the anisotropic Ward-Duer BRDF might change a little bit for the better. (#4042)
- **list bucket scanning mode**: Due to the new bucket threading code, when using the list bucket scanning mode, in-place modification of the bucket list during rendering is no longer possible. This was an unexpected usage of the API. In a future release, we will provide a method to dynamically select buckets during rendering. In the meantime, the list bucket scanning mode is assumed to be given a static list. (#3873)

Bug fixes

Ticket	Summary
#3866	crash with multiple texture-mapped quad area lights
#3884	Crash with polymesh index arrays bigger than 16M elements
#3950	Crash when interrupting rendering during BVH build
#3987	AiHairDirectDiffuseCache() non-deterministic
#3205	User data inheritance on procedural networks is not working
#3237	Fully transparent samples not properly accounted for in raw/deep drivers
#3268	trace_sets not working for procedurals
#3294	bump2d does not work with Pref when UV coords are not defined
#3370	broken motion_vector AOV in objects with deform keys and ignore_motion_blur
#3766	Deep EXR driver is missing a ZBack channel for Nuke compatibility
#3787	Setting sss_bssrdf_samples to 0 should disable SSS computations
#3814	Crash with invalid camera matrix
#3825	Missing referenced disp shader causes mesh to disappear

#3829	Deep EXR driver uses too much virtual memory
#3830	ignore_motion_blur not removing extra keys
#3835	several second startup delay on certain linux systems even for trivial renders
#3840	curved motion blur is sometimes using a slightly off rotation
#3841	linking quad_light.color to a shader not working with multiple lights
#3843	inaccurate "rays/pixel" progress report during rendering
#3850	unnormalized lights with zero radius should be treated as normalized
#3854	Deep EXR crash when defining data tolerances but no data channels
#3856	maketx crash (windows)
#3863	Light filters incorrectly modify surface UVs
#3864	Volume RGB output fading as step size decreases
#3865	Crash in AiEnd() in free mode after destroying geometry nodes
#3875	Fixed render checkpointing for rare case with multiple AOVs
#3879	Correct memory accounting for sample stores
#3883	falloff radius for constant decay lights should be infinite
#3885	Deep EXR driver with high tolerance has bogus extra sample per pixel
#3888	remove correlation artifacts between pixels
#3896	ignore_motion_blur not enabled at reference_time 0
#3904	Kick display driver crash on Windows with render region
#3905	Faster render abort
#3909	AiThreadSelf() not working properly on Windows
#3915	subdivision numerical precision regression at high iterations
#3916	reported displacement time is larger than it should be
#3919	Deep EXR and raw drivers missing samples from user-defined AOVs
#3923	empty sss_setname not being ignored
#3924	Inconsistent mesh_light intensity when mesh is scaled
#3929	RGB/RGBA user data is gamma corrected when writing to .ass
#3930	Missing AtRGBA operators in python bindings
#3931	Crash writing to .ass with a string param of max allowed length
#3942	allow composition of RGB AOVs connected to RGBA drivers and viceversa
#3946	reported unaccounted memory is too low
#3958	Out-of-range pixel coords in sample store access
#3965	Lights in a procedural with no geometry are not transformed
#3966	Lights in a 2-level procedural network not correctly transformed
#3968	Kick should skip progressive renders when there is no display window
#3972	AiOcclusion artifacts with regular and reflection occlusion
#3975	NaN samples with zero value transmittance in the standard shader
#3989	AiMsg* was ignoring messages before AiBegin
#3997	Interrupting rendering during subdivision causes extraneous warnings and garbled mem stats
#4008	Vector displacement incorrect on UV seams
#4032	Interruption not terminating quickly for subdiv and displacement
#4057	out-of-range error in spotlight when using duplicate keys
#4061	Add pid to logs
#4070	Arnold doesn't accept floats with dot as the last character in IES and metadata files
#4084	Deep AOV sample memory reporting was wrong
#4089	Add missing operators in Python bindings
#4091	Wrong procedural user data with procedural cache enabled
#4093	Light occlusion should behave consistently
#4094	Add parameter type to API interpolation templates
#4101	Deep data tolerances incorrectly set when alpha is also a data channel
#4105	A ginstance referencing a non-shape node causes a crash
#3743	Report cycles on instances and avoid crash
#3940	Typo in python bindings
#3991	Emit warning when using an array value on a non-array parameter
#4014	Better message when disp_padding is too small
#4087	AiSSSTraceSingleScatter darkening for albedo 1
#4109	rays/pixel stat should be rounded to nearest integeer