5.2.0.1

5.2.0.1

Released 10 Sep 2018

Bug fixes

- #7001 Add expression cache to the operator runtime
- #7293 MaterialX: Node graph output channels not supported
- #7443 MaterialX generated shaders are scoped under their node graph names
- #7446 alembic crash with make_instance enabled and changing frames
- #6221 Export full path instead of node name when writing to .ass while expanding procedurals
- #7388 metadata lexer should ignore more than three # in a row
- #7413 alembic files with single entry array attributes should be considered not arrays
- #7415 Noice: issues with certain crop windows combinations
- #7416 Noice: incorrect handling of variance AOVs in a separate file
- #7417 Noice: support additional channel suffixes and warn if the suffix is unsupported
- #7418 Noice: skip denoising AOVs with no associated variance
- #7419 make flatness check for quad_light more robust
- #7422 transform motion in procedural that only contains lights can give invalid lights
- #7431 Volumes: register new AOV volume_Z
- #7432 Volume AOVs: shadow rays interfere with new Z and ID AOVs
- #7436 Toon Render: random issues with keylight initialization
- #7445 Cell noise time not evaluated when linked
- #7456 Thread safety of alembic procedural tokenize function
- #7466 MaterialX: Supported parameter arrays are not processed by operator and node definition generation
- #7467 Render callback does not support an "empty" callback
- #7470 d'Eon BCSDF must be evaluated when `base_color` is small
- #7477 Gain function in range shader is inverted
- #7442 Copy id to child alembic nodes

5.2.0.0

Release 09 Aug 2018

Enhancements

- **Texture baking**: A new type of node called `uv_camera` has been added that will produce an image of a given polymesh's shaded UV space as output, which can be useful for texture baking. (#6091, #7206)
- **Improved sampling of spherical lights**: A new technique for sampling point lights has been added which can show significant reductions in noise, especially for large lights illuminating surfaces at grazing angles (rim lighting, for example). (#5534)
- **Faster adaptive subdivision**: Adaptive subdivision is now up to 2x to 3x faster even on a single thread. In addition, the adaptive codepath has been multi-threaded to fully take advantage of machines with many cores. The aggregated speedup in such machines can be 15x or more. (#2311, #7186, #7201, #7229)
- **Improved EXR read performance**: Threaded read performance and scaling of OpenEXR files has been greatly improved. (#6605)
- **`noice` denoiser improvements**: The stability and usability of the high-quality `noice` denoiser has been improved thanks to various bugfixes and improved error checking. In particular, the original metadata, display windows, bitdepth and compression are preserved in output files. (#7226)
- **OptiX denoiser improvements**: The GPU memory consumption of the fast OptiX denoiser has been greatly reduced proportionally to the number of denoised AOVs. Fringing artifacts around HDR pixels have been reduced. (#6885, #7100, #7190, #7333, #6880)
- **Sheen in `standard_surface`**: The `standard_surface` shader supports a new, energy-preserving sheen effect designed to render cloth-like microfiber materials such as velvet. The sheen effect is layered on top of the diffuse and subsurface components. (#7234)
- **New `cell_noise` shader**: A new `cell_noise` shader has been added which can create many different useful cell-like patterns. The color of each cell is mapped to a `palette` parameter, enabling the easy creation of patterns with colors chosen from a specific palette. (#5985, #6051)
- **New controls in `range` shader**: The range shader has been augmented with parameters to control contrast, bias and gain. (#7277)
- **RGB clamping in `clamp` shader**: The clamp shader can now be configured to either a scalar or color mode. (#7278)
- **Matrix shaders**: The `matrix_multiply_vector` and `matrix_transform` shaders have been reinstated. (#7243)
- **Built-in Cryptomatte**: Cryptomatte AOV shaders and filters are now being included as a part of the Arnold core package. (#7301)
- **New built-in volume AOVs**: The Z depth for the first volume contribution can now be output in a flat AOV with `volume_Z` (depth AOV for volumes was already available in deep files). Also, `ID` now works for volumes. (#7326, #7327)
- **New control in `toon` shader**: Edge detection can now be controlled using a `STRING` type user data called `toon_id`. This feature is enabled when `user_id` is checked. Otherwise, the detected edges will be driven by the object's own name as a toon-specific ID. (#7125)
- **Alembic procedural improvements**: The Alembic library has been updated to 1.7.5 in this release. User data parameters that clash with shape parameters will now get an underscore prefix instead of a warning. Added an `object_transform` parameter to allow additional transformations on the generated geometry. Added a `make_instance` parameter so that the Alembic procedural will automatically create instances of objects present in multiple Alembic procedurals (experimental; disabled by default) (#6916, #6947, #7076, #7109, #7163, #7242, #7261, #7286)

- **Improved operator assignments**: Assignment expressions in operators have improved functionality with regards to reference and string types. (#7284, #7287)
- **Upgrade to OSL 1.9.9**: This upgraded version of OSL addresses several reported limitations involving locales, the availability of certain noise types, and compatibility issues with utility functions like `transformc` being promoted to built-in function definitions. (#6225)
- **Updated to RLM 12.4BL2**: The RLM license server and library have been upgraded from version 12.2BL2 to 12.4BL2, which fixes sporadic access violations and hangs. (#7350, #7120)

API additions

- **Sheen closure**: The `AiSheenBSDF()` function has been which provides the sheen closure used by `standard_surface` so that the same effect can be easily obtained in custom shaders. This function also returns a weight that can be used to layer the sheen closure onto other closures in an energy conserving fashion. (#7234)
- **Multiple Universes**: A new concept has been added to Arnold's API which gives it the capability of creating nodes in any number of user-defined workspaces called `AtUniverse` that can be created and destroyed via the `AiUniverse()` and `AiUniverseDestroy()` functions, respectively. Upon creation, nodes are assigned a universe which will take on their exclusive ownership, and this ownership can be inspected via the `AiNodeGetUniverse()` function. Please note that the "null" universe is the "default" universe, which for the time being is the only one which permits rendering and procedural expansion. (#4129)

Incompatible changes

- **Code compatibility**: Code made for Arnold 5.1 should still function in Arnold 5.2 after a recompilation.
- **Binary compatibility**: Even though 5.2 is technically an API-breaking release, Arnold 5.0 and 5.1 plugins (shaders, filters, etc) in the majority of cases will still be compatible with 5.2 and can continue to be used without being recompiled, while procedural plugins on the other hand will in the majority of cases require recompilation. (#6822)
- **Multiple Universes**: In order to support multiple universes, several existing API functions have a new `AtUniverse` pointer as first parameter indicating which universe these functions must operate on. This change to function signatures implies a break in binary compatibility with any client code compiled for previous versions of Arnold that were using these functions. However a set of function overloads that preserve the previous signatures has been added to the C++ headers such that the mismatched signatures can be resolved through a code recompilation. The API functions affected by this change are:
    - `AiASSWrite()`
    - `AiASSWriteWithMetadata()`
    - `AiASSLoad()`
    - `AiNode()`
    - `AiNodeLookUpByName()`
    - `AiUniverseCacheFlush()`
    - `AiUniverseGetOptions()`
    - `AiUniverseGetCamera()`
    - `AiUniverseGetSceneBounds()`
    - `AiUniverseGetNodeIterator()`
- **Removed texture blur options** : The `texture_specular_blur`, `texture_diffuse_blur` , and `texture_sss_blur` options have been removed after having defaulted at 0 for some time. They are no longer needed for improving texture performance since Arnold is able to automatically blur the textures as needed without the over-blurring that these options would produce. (#4706)
- **GPU denoising options renamed**: The GPU-related render options used by the OptiX denoiser have been renamed to use the `gpu_*` prefix . (#7190)
- **ENUM params now byte-sized**: Enum parameters are now internally stored as bytes instead of ints, and will be reported as such by the `AiParamGetTypeSize()` API. (#7114)

Bug fixes

- #7325 procedural loading a .ass containing many nodes that reference other nodes is very slow
- #6472 OSL issues with Locale
- #6916 curves in Alembic proc cut short
- #6947 Skip invalid curves
- #7012 Noice: black pixels around variance spikes
- #7027 Non-ascii characters prevent Open EXR images from being loaded
- #7076 Rename user data parameters that clash with shape nodes in alembic expansion
- #7154 Adaptive Subdivision: crash with linear patches and different position and UV topologies
- #7179 Adaptive Subdiv: irregular patch crash with different position and uv topologies
- #7226 Noice: support inputs with different data windows
- #7286 Alembic object transform
- #7303 Python binding fixes for functions that return AtNode types
- #7312 Raw drivers crash (including Cryptomatte's manifest driver)
- #7341 Crash when computing the render stats for procedural
- #7355 Alembic curve user data expansion