5.3.0.2
11 April 2019

Bug Fixes
This is a bugfix release, for a full list of features see the Arnold 5.3.0.0 release notes.

- #8262 Improve robustness of NVIDIA driver version checks
- #8270 Log GPU stages as info
- #8252 Crash when disabling a light node in IPR
- #7696 Degenerate B-spline curves crash on GPU
- #8186 Corrupted ray differentials cause transparency artifacts on GPU
- #8220 Ray visibility masks not working on pre-Turing GPUs
- #8245 Fog does not affect the background color on GPU
- #8246 Incorrect self-only probe rays when shape has a transform on GPU
- #8248 Hang when re-rendering instances without a valid referenced node
- #8250 Displacement not working when driven by varying user data
- #8258 Gracefully handle CPU renders with insufficient GPU requirements for OptiX denoiser
- #8286 Hang when disabling / enabling shaders in IPR
- #8278 Mudbox UDIMs tile offset ignored on GPU
- #8246 Incorrect return value for `AiRenderGetHintBool()` for "progressive" and "progressive_show_all_outputs"

5.3.0.1
27 March 2019

This is a bugfix release, for a full list of features see the Arnold 5.3.0.0 release notes.

Bug fixes

- #8233: Support AA_sample_clamp on GPU
- #8122: Noise in normal AOV on GPU
- #8221: Fatal GPU message is not explicit enough
- #8216: NVIDIA driver check is misreporting version as X.00
- #8203: Rays/pixel render progress does not work
- #8207: Stack overflow regardless of continuation stack size
- #8211: Crash on render after setting null arrays to polymesh geometry data parameters
- #8215: Unable to enable nodes that were disabled in the first render pass
- #8227: Negative subsurface_color causes artifacts
- #8230: GPU cache population not reporting errors
- #8232: GPU: Crash with a mesh_light using a ginstance
- #8219: Mesh light is wrongly marked as unsupported on GPU

5.3.0.0
March 20th, 2019

Enhancements

- GPU rendering (BETA): You can now switch between CPU and GPU render devices interactively and expect visually similar results. NVIDIA® GPUs from Turing™ to Maxwell™ architectures are supported, and Arnold will take advantage of multiple GPUs, NVLink™ and NVIDIA® RTX™ hardware accelerated raytracing if available. Note that due to beta status of GPU rendering, a number of features are missing, performance is not final, and use in production is not advised. We plan to gradually improve this in subsequent releases and would appreciate your feedback. For a complete description of requirements, features and caveats, see https://docs.arnoldrenderer.com/x/iwnfB.
- Improved adaptive sampling: Adaptive sampling now uses a more effective criterion. With the new metric, pixels are being dropped gradually in a smoother and much more predictable manner, resulting much lower render times for identical noise levels. In addition, the
adaptive sampling criterion windowing that was previously available only in progressive rendering mode is now also available in non-progressive mode, resulting in higher quality sampling, with fewer sampling "holes". (#7972, #7950)

- **Improved skydome sampling**: Arnold now takes into account the normal of the shading point when importance sampling the skydome. Therefore bright areas in the skydome are less likely to "steal" samples when they are below the normal. Even for more uniformly colored skydomes, the improved sampler will waste fewer samples in directions that are below the hemisphere, allowing for lower skydome light sample rates to be used which should give a noticeable speedup. For the same number of shadow rays cast, the new sampler has just a small ~2% performance overhead. Note that skydome samples need to be lowered to 70% or less from their original value (3 instead of 4) to roughly keep the number of shadow rays and associated cost the same. (#6669)

- **Visible lights**: The quad_light, disk_light, cylinder_light and point_light now have a camera and a transmission attribute, allowing these lights to become visible to camera and transmission rays. These attributes are left to 0 by default, thus not changing the default behaviour of the non-visible lights. (#2269)

- **Microfacet multiple scattering**: The GGX microfacet BSDF used in the standard_surface shader has been improved to account for multiple scattering between the microfacets, which is more physically correct and reduces energy loss on reflection, especially at higher roughness settings. The reflection from rough metals in particular will be significantly brighter and more saturated as a result. Unfortunately there is a slight increase in noise on account of the more difficult sampling when including the multiple scattering component. Disabling the global option enable_microfacet_multiscatter will restore the previous look. (#7207)

- **Improved random-walk SSS**: A new randomwalk_v2 SSS mode has been added that scatters more accurately and deeply through highly-transparent/optionally-thin objects, which produces SSS with more saturated colors around fine surface detail and heavily backlit regions of an object. Note that renders will be more costly and noisier than with the original method, since random walks will be on average longer and more random. (#7550)

- **Anisotropy controls for coat in standard_surface**: We have added two new parameters, coat_anisotropy and coat_rotation, in the standard_surface shader for finer artistic control of the coat layer. (#7935)

- **Support for negative transmission_extra_roughness in standard_surface**: The parameter transmission_extra_roughness in the standard_surface shader now accepts negative values, allowing separate control of the specular and transmission roughnesses for artistic purposes. (#7936)

- **Linkable transmission depth in standard_surface**: The transmission_depth parameter is now linkable. (#8039)

- **Improved coat layer in diffuse reflections**: When caustics are disabled on the reflected object, the coat layer on the standard_surface shader is now correctly taken into account. (#8016)

- **Improved bump and normal mapping**: Built-in shaders such as bump2d, bump3d and normal_map now correct for non-physical shading normals while preserving detail. This should be more noticeable when rendering normal-mapped ocean surfaces or using extreme normal maps. (#7639)

- **Smart opaque**: Built-in shaders now set the object.opaque flag automatically based on whether or not the shader settings would require disabling the opaque flag on the object to render correctly. For instance, it's no longer necessary to manually disable the opaque flag to get transparent shadows for a glass shader. Notable exceptions are curves and points when min_pixel_width is in use and OSL shaders. Custom shaders can take advantage of this setup tagging shader attributes with metadata. As an example, the metadata for standard_surface follows:

```c
// This metadata setup from standard_surface:
AiMetaDataSetInt(node, "opacity", "opacity_term", 1);
AiMetaDataSetInt(node, "transparency", "transparency_term", 0);

// This additional integer brackets together terms with the same number.
AiMetaDataSetInt(node, "transparency_term", 0);
AiMetaDataSetInt(node, "transmission", "transmission_term", 1);
AiMetaDataSetInt(node, "opacity", "opacity_term", 1);
AiMetaDataSetInt(node, "metalness", "opacity_term", 1);

// is equivalent to setting:
// opaque = (opacity == AI_WHITE) && (transmission == 0) || (transmission_color == AI_BLACK) ||
// metalness == 1);
```

Note that setups with transparency but with the opaque flag set to true will now be considered transparent. To preserve the previous look you can make your shader opaque or use a ray switch to make it opaque only to shadows. You can also disable smart opaque with a user variable declared on the options node:

```c
options {
  declare disable_smart_opaque constant BOOL
  disable_smart_opaque ON
}
```

This is intended for old assets that can’t be modified. If possible, we recommend instead to specify opacity values in the shader itself. (#5966)

- **Operator connection on procedurals**: Operator graphs can now be connected to procedurals through the operator parameter. Only nodes in the procedural or nested procedurals are evaluated by the graph, where everything is performed relative to the procedural's name scope such as selection expressions. Furthermore, nodes created by the operators are put in the procedural's name scope. (#7747)

- **New include_graph operator**: The include_graph operator allows importing operator graph from an .ass file, where the target parameter specifies the target operator in the included sub-graph. The operator can also load shaders if they exist in the .ass file, where they are put in appropriate name scope depending on what the operator is connected to. (#7294)

- **Scoped child operators**: Child operators are now placed in the name scope of their parent operator, where a set of child operators can be connected to enforce a particular evaluation order. (#8075)

- **MaterialX operator improvements**: Updated the MaterialX library and operator (#7825). Swizzles can now be used to define the channel connections between shader nodes. Float array handling has been expanded to handle other array types such as color3array in order to
support the ramp shader (#7970). The search path for the Arnold node definitions is now set automatically (#8028). The operator matches assignments in accordance with the current name scope, i.e. procedural or global namespace (#7968).

- **Subdivision creases support in Alembic procedural**: The Alembic procedural will now detect and translate crease data for subdivision surfaces. (#7665)
- **Improved ramp_rbg shader**: Several new interpolation modes were added, as well as an implicit_uvs parameter, which allows to use barycentric implicit UVs to drive the ramp, instead of regular UVs. This last option can be particularly useful with hair. (#7987)
- **New time mode in ramp shaders**: The ramp and ramp_rbg shaders now have an additional time mode that computes the input based on the current sg->time and the camera’s start and end shutter interval. (#7925)
- **New uv projection shader**: This shader allows to do planar, spherical, cylindrical, ball, cubic and shrink_wrap projections (#7920)
- **New matrix_interpolate shader**: This shader allows to produce motion blurred matrix parameters, by interpolating between matrix values. (#7948)
- **New coord_space control in camera_projection**: You can now choose to do camera projections in world or object space, or by using P user data, or linking any shader to the input position P parameter. (#7976)
- **UV coordinates for background**: During background shading, UV coordinates are now generated from the screen coordinates for camera rays. Thus it’s now possible to directly link a UV-based shader to options.background such as image, checkerboard or ramp. (#8007)
- **Denoising albedo**: A built-in AOV with an albedo optimized for denoising has been added as denoise_albedo. Both the Arnold (noise) and OptiX™ denoisers will use the new AOV if present. (#6955, #8168, #8069)
- **OptiX™ denoser improvements**: Updated to OptiX™ 6 which brings denoising improvements in quality, memory consumption and performance on Turing™ hardware. (#7965)
- **Faster sample generation**: Rendering with high camera AA samples combined with large numbers of lights or other secondary samples could sometimes result in very slow startup times. For instance, a scene with AA=18 and point light samples=18 was taking 11 minutes to start; it now takes a fraction of a second. (#7933)
- **Faster ass file writing**: Writing to .ass files, especially over some Windows networks, can be dramatically faster. One customer saw a 51x speedup. (#7801)
- **Faster OSL UDIM texture reads**: Reading UDIM textures in OSL is now much faster, with up to 10x reported speedups. (#7920)
- **Reduced memory usage for instances of curves**: The uvs and shidxs arrays in the curves node are no longer duplicated for every instance of a curves node, significantly reducing the memory footprint of instanced curves. (#7742)
- **Maketx monochrome map detection**: maketx now by default detects the frequent case of an RGB image that can be stored as single channel. This can result in smaller .tx files and render time memory savings. (#6671)
- **Motion vectors AOV uses the camera shader**: In order to facilitate using motion vectors, the global option options.ignore_motion now keeps motion keys and only sets the camera ray times to be equal to the options.reference_time - use this with motion vectors instead of setting shutter intervals to zero. The new global option options.ignore_motion will take over the pre-existing ignore_motion_blur functionality of ignoring all motion keys. (#7962)
- **New OCIO role**: arnold_srgb_equivalent has been added as a synonym for the older srgb_equivalent to make it clear this is an Arnold role. (#8092)
- **Automatic reinitialization of shapes**: Modifying a node parameter in a geometry (aka shape) node will now trigger automatic re-initialization of that node, thus no longer requiring re-exporting that node from the DCC or client code. (#7899)
- **Default AtrRGB/AtrRGBA assignment**: We now use the default compiler supplied assignment operator for the AtrRGB and AtrRGBA objects. This means these objects are now trivially copyable. This should not break preexisting code. (#7969)
- **Updated to OIO 2.1.0**: Updated to the latest OpenImageIO library in order to bring in several texture related bug fixes. (#7929)

**API additions**

- **Render hint for controlling interactive outputs**: Using the newer render API, there is an additional bool render hint progressive_show_all_outputs which will force all render outputs to be written to even during early blocky AA passes. By default it is off, and all but the main interactive output are skipped so that interactivity is improved. (#7986)
- **MaterialX API**: A new function AlMaterialWrite() bakes the shader and other look properties for one or more shapes to a .mtlx file along with the description of the shaders and shading graphs (#7883), and AlMaterialGetXGetLookNames() gets the list of Looks contained in a .mtlx document. (#8067)
- **Procedural scope from operator context**: Operator instances are now given a cook context (AtCookContext) which they can use to determine if they are being cooked using a procedural scope, i.e. the graph they are in is connected to a procedural, using AlOpCookContextGetCookScope(). Note that the same operator instance can be connected both to one or more procedurals as well as the scene options. An example use case is the MaterialX operator which uses this to make assignments relative to the scope. The operator also omits a node name prefix if a scope is found as the shaders it creates are automatically put in the procedural scope and thus do not require a unique name. (#7967)
- **Operator post cook callback**: A new operator function callback operator_post_cook has been added which is called once for each operator instance when all the operators have finished cooking. (#7973)
- **Match nodes against a selection expression**: AlOpMatchNodeSelection() is a function that can be used to check if a node’s name, type, and parameter match a selection expression. (#7909)
- **GPU cache API**: This new API manages the OptiX™ disk cache, which is automatically generated during JIT compilation prior to GPU rendering. It contains a store of previously compiled OptiX™ programs, so that subsequent renders start faster. The API provides a means to asynchronously pre-populate the cache and customize its location. This pre-population is triggered after installing a new Arnold version, updating to a new NVIDIA® driver, or changing the hardware configuration of GPUs on the system. (#7926)

**Incompatible changes**

- **Removed legacy point clouds**: An internal point cloud data structure (previously used for SSS) and its associated API have been removed. (#6558)
- **Removed SSS on curves**: SSS on curves was deprecated as it did not properly work and has now been removed. (#4254)
- **Removed support for NVIDIA® Kepler™ GPUs**: the OptiX™ denoser only supports GPUs with CUDA™ Compute Capability 5.0 and above. (#7964)
• Defaults changed for standard_surface: The specular_IOR parameter changed from 1.52 to 1.5 so that it matches the coat_IOR and thin_film_IOR. The specular_roughness changed from 0.1 to 0.2. The subsurface_type default changed from diffusion to randowalk. (#7627, #8133)

• MateralX shader reference type name changed: The attribute used to define the shader reference type in MaterialX documents (e.g. surf/aceshader/displacementshader) has been renamed context because of changed behaviour in the MaterialX library. (#7825)

• Operator merge order reversed: The last operator wins if multiple operator inputs are merged and override the same node parameter. (#7903)

• Specular AOV: The sheen component has been removed from the specular_* AOVs. (#7522)

• Custom operators: Custom operators need to be recompiled for this version of Arnold to include the new operator_post_cook() callback. (#7926)

To preserve the previous look you can make your shader opaque or use a ray switch to make it opaque only to shadows. You can also disable smart opaque with a user variable declared on the options node:

```python
options
{
  ...
  declare disable_smart_opaque constant STRING
  disable_smart_opaque "any-value-will-do"
}
```

This is intended for old assets that can’t be modified. If possible, we recommend instead to specify opacity values in the shader itself. (#5966)

• Improved skydome sampling: Skydome samples need to be lowered to 70% or less from their original value (3 instead of 4) to roughly keep the number of shadow rays and associated cost the same. (#6669)

Bug fixes

• #8198: Crash on operator modifying object nodes
• #7401: Triplanar: object coord space not working during displacement
• #7951: Arnold Crashes with certain multichannel TX files on Windows
• #8139: Triplanar: pref not working during displacement
• #8187: Alembic doesn't cleanup children cameras
• #6267: Dry screen differentials in displacement context are zero
• #6376: Parallel AiMakeTx can consume too much memory
• #6509: Setting a node parameter with the same current value should be ignored
• #6671: Add back --monochrome-detect to AiMakeTx when OIIO bug is fixed
• #7408: OpenEXR: standard chromaticities metadata was missing in output files
• #7694: Artefact when rendering inside a volume mesh
• #7741: Arnold crashes when you try to write out render stats
• #7750: Connected normal_map and bump2d breaks transmission
• #7760: Space transform screen space issue
• #7763: Subdivis: per face iterations should be ignored if type is not BYTE
• #7764: Artefact when rendering inside a volume mesh
• #7781: Crash when using polygon holes
• #7784: Operator nodes are not allowed within procedural
• #7790: No diagnostic showing license server used for successful checkout (RLM parity)
• #7794: Running multiple AiMakeTx hangs and crashes in 5.2.2.0
• #7803: Contour filter memory leak
• #7805: Change error to warning when OSL output keyword is missing
• #7806: makebx was not automatically doing --opaque-detect
• #7812: Transmission toon artifacts around triangle edges
• #7840: Forward references always take precedence over parameter overrides
• #7853: Transforming a procedural child node doesn't update bounds properly
• #7955: Color shifts in randomwalk sss when far from origin
• #7939: Add missing camera projection features
• #7947: Variance buffers are not cleared in progressive mode when updating sampling settings
• #7952: [alembic] Add triggers for filename, objectpath and fps changes
• #7955: Windowed adaptive disabled with 1-pixel filters
• #7982: Log reason why custom procedural fails to load
• #7987: Add missing attributes in ramp_rbg
• #8001: MaterialX: Add namespace prefix for materials when used in the global scope to avoid name clashing
• #8006: Curves: opaque override not working for non camera rays
• #8012: Ramp_rbg wrong indexing with shuffled positions
• #8016: Caustic avoidance in standard_surface messes up closure weights
• #8019: Deep driver: append does not work with half data channels
• #8026: Luminance conversions should take into account working color space
• #8032: Mismatch in python bindings for render API
• #8904: Address assignment expression issues#8097Add wrap_mode "none" for uv_transform
• #8142: Ramp in interpolation "constant" not including key positions #8170allow smaller SSS radius
• #8175: Uniform user attributes not selectable by operators
• #7788: skydome_light crashes when ignore_textures is enabled on GTC robot scene
• #8009: Reset shader assignment using set_parameter
• #8047: AiNodeReset should not reset the node's name
• #8089: AtArray interpolation of matrices with one key ignored index
• #8191: AiNodeGetParent missing from python bindings