5.0.0.0

Enhancements

Shaders

- **Standard Surface**: a new `standard_surface` shader was added, with intuitive and energy conserving parameters, a secondary specular coat with separate normal, metallic Fresnel, thin surface support and more. The `standard` shader is still available but considered deprecated. (#5372)
- **Standard Hair**: a new physically based `standard_hair` shader was added, with much more accurate specular and transmissive scattering, better diffuse scattering for dirty hair, melanin randomization, and simple and intuitive parameters. The older `hair` shader is still available but considered deprecated. (#5370, #5851)
- **Standard Volume**: a new `standard_volume` shader was added, usable for rendering a wide variety of volumes. The shader provides independent control of density, scattering color and transparency, in a way that is energy conserving by default. Fire can be rendered using blackbody emission driven by temperature. Displacement can be used to add more detail to volumes. This supersedes the `density` volume shader, which has been removed. (#5090)
- **Shader mixing**: surface, hair and volume shaders now output closures rather than colors. This makes it possible to mix shaders efficiently, using the new `mix_shader` node to blend or add two shaders including all their light AOVs. Adopted from the common shaders, there is also a vanilla color mixing shader `mix_rgba`. The ray switch shader has been split up into two `ray_switch_shader` and `ray_switch_rgba` nodes, for switching between shaders and one for switching between texture colors respectively. (#5494, #5495)
- **Bump shaders**: `bump2d` and `bump3d` now output a normal vector that can be linked to new `normal` parameters in `ambocc`, `lambert`, `standard_surface` and `utility` shaders. These bump shaders no longer function as passthrough shaders. The `@before` syntax for bump shaders has been removed as well. (#5599)
- **Noise shader**: The `noise.coord_space` parameter now has a `uv` enum value for texturing using the object's local UV coordinates. Note that this calls the faster 2D noise API, not the 3D noise like all other coordinate spaces. In addition, an arbitrary coordinate space can be specified manually by linking another shader into the new `P` parameter. The shader can now output colors, in `scalar` mode by blending between `color1` and `color2`, and in `vector` mode with a separate noise signal per color channel. A new `time` parameter can be used to smoothly vary noise over time. (#5195, #5235, #5247)
- **Utility shader**: the `utility` shader has been enhanced with a new `metal` shading mode, and a `roughness` control has been added that affects plastic and metal modes. (#5018)
- **Legacy standard shader**: the Beckmann specular distribution in the (deprecated) `standard` shader now uses more correct per-microfacet fresnel, as was already used for GGX. `sss_profile` has been removed and the empirical BSSRDF is now always used. (#3285)
- **Legacy hair shader**: `uparam` and `vparam` have been removed from the (deprecated) `hair` shader. Instead, curves now support UV sets. This shader does not fully support light path expressions. (#5646)
- **Env vars in `image` filename**: the `image` node filename field will now expand environment variables of the form `[var]`, in addition to it already expanding them when they are in the texture searchpath. (#5671)
- **New tags in `image` filename**: the `image` node filename field now accepts `<utile>` and `<vtile>` tags. They take an optional integer offset `<utile:2>` that adds to the tile number; e.g. an offset of 2 from a U coordinate of 1.4 will output tile number 3. (#5753)
- **Renamed `image` missing tile parameters**: on the `image` texturing node, `ignore_missing_tiles` has been renamed to `ignore_missing_textures`, and `missing_tile_color` has been renamed to `missing_texture_color`. The handling of missing textures has been extended to non-tagged textures. It is still recommended to not ignore missing textures and fix them (the render will emit an error to the log and abort if `options.abort_on_error` is left on). However in some cases missing UDIM or other tiled textures may be a deliberate choice. These options allow control, per image node, what happens in those cases. (#5753)
- **Built-in utility common shaders**: The common utility shaders that shipped at least in part with MtoA, C4DtoA, HtoA and KtoA have been integrated into the core. A few shaders from the set have not been added, such as `print`, `linearize`, `ln` and the matrix shaders, and a few have improved parameter defaults, and others have been optimized, but they are largely the same. (#5714, #5749)
- **Color Jittering shader**: a new `color_jitter` shader can be used to generate random colors within a specified gain, hue and saturation range. Colors may be randomized per face, object, procedural instance or user data, or a combination. (#5793)
- **Triplanar Shader**: a new `triplanar` shader can be used to quickly map image textures without needing UV coordinates. The texture is projected onto the object from the 6 sides, and smoothly blended together at the seams. (#5770)
- **Dedicated normal and vector map shaders**: The common shader `space_transform` had extra functionality for dealing with tangent-space normal or vector maps, and that has been removed. Instead, there are now two new shaders, `normal_map` and `vector_map` that take typical normal map and vector displacement map data, respectively, and prep them for their typical use later in a shading network, similar to how the bump shaders would be used. (#5714)
- **`osl` shader**: while it is recommended to compile .osl files to .oso and place them in the shader path, OSL shader code can be referred to directly via the `osl` node or even set as a string parameter. The shader node will then generate all the parameters of the OSL code with the `param_` prefix in front of each. (#5471)
- **Skydome light camera visibility**: New `camera` and `transmission` parameters set the amount of light contributed to camera and specular transmission rays, so that it is no longer required to use a separate background shader for such purposes. A new `shader` parameter may be used to link closure shaders to control color and transparency. Skydome lights are now preferred over background shaders, as they provide the same functionality with better sampling. (#5744)

Sampling

- **Dithered sampling**: most samplers (e.g. soft shadows, indirect illumination, depth of field) will now take advantage of dithered sampling, which improves the visual distribution of noise specially at low sample rates. (#5047, #5412)
- **Quad lights**: sampling has been improved, reducing noise for surface lighting. For comparisons with the previous sampler, `options.enable_new_quad_light_sampler` can be disabled. (#4961, #5780)
- **Cylinder lights**: sampling has been improved, significantly reducing noise for cylinder lights in volumes or where cylinder lights are located near other surfaces. (#5347)

- **Disk lights**: a novel sampling algorithm has been implemented for disk lights which can greatly improve their rendering quality when shading points near the disk's surface, particularly in volumes. (#5316, #5856, #5862)
- **Russian roulette**: on average better performance for hair, transmission and volume scattering. (#4052, #5789, #5690)
- **Motion blur time samples**: rendering motion blur is now faster when using non-default `deform_time_samples` or `transform_time_samples`. (#5336)
- **Caustic noise reduction**: a new method was added to reduce noise from caustics. Noise from GGX microfacet speculars in particular is reduced. `options.indirect_specular_blur` controls the trade-off between more accurate noisy renders at 0.0, and more blurry biased renders with reduced noise at higher values. (#4498)
- **Faster opacity mapping**: opacity-mapped transparent surfaces, such as tree leaves, are now sampled more efficiently and can render up to 20% faster, specially in machines with many threads. (#4704)

OSL, Closures and AOVs

- **Open Shading Language**: shaders can now be written in Open Shading Language, an advanced shading language for production GI renderers. OSL shaders placed in the shader search path are automatically registered as Arnold shader nodes, with their parameters converted to Arnold parameters. Once loaded, they can be inspected, instantiated and linked in exactly the same way as C++ shaders. OSL shaders can be used to implement texture patterns and materials using closures. See the OSL documentation for more details. (#4357, #5400, #5487, #5526, #5471)
- **Closures**: a new closure parameter type has been added, which shaders can output instead of final colors. There are BSDF, BSSRDF, emission, matte, transparency and volume closures. See the API documentation and examples for more details on how to use these. Linking a color to a closure parameter will automatically create an emission closure with that color. A closure parameter however can't be linked or converted to a color, as the integrator only computes lighting after shader evaluation. (#4793)
- **Light group AOVs**: surface shaders now natively support light group AOVs, previously this feature was only available for volume shading. (#4305)
- **Light path expressions**: light path expressions are used to write lighting components into separate AOVs. No longer should individual shaders define AOVs for direct/indirect light and various layers, rather a regular expression syntax is used to define the subset of all scattering and emission events in the scene that should be written to each AOV. Built-in AOVs are available for the common cases. See the LPE documentation for details. (#5491, #5581, #5582)

Color Management

- **Color Managers**: custom `color_manager` nodes can now be implemented to handle input and output color transforms. A color manager is a package similar to SynColor or OpenColorIO. If no color manager is specified in `options.color_manager`, Arnold will use the built-in one which supports `linear` (no transform), `sRGB` and `Rec709` (non linear) (#5262, #5527)
- **OpenColorIO**: color spaces defined in OpenColorIO configurations can now be used in .ass files through the built-in `color_manager_ocio` node. (#5598)
- **Linear color spaces**: any linear color space can now be specified through the color manager for any output driver that supports halfs or floats. (#5552)
- **Color space metadata**: OpenEXR files have additional metadata fields with color manager type (`arnold/color_manager`), color space name (`arnold/color_space`) and chromaticities (`ColorSpace/Red Primary`, etc) when known. (#5263)
- **Rendering color space**: using specific implementations of a color manager it is now possible to specify Arnold's rendering color space. Arnold's default rendering color space is `Linear sRGB`. For other color spaces Arnold will try to auto-detect chromaticities and illuminant, and will expect all data (textures and .ass file parameters) in that same color space. An important note is that different rendering color spaces will yield different render results when compared with the default `sRGB linear`. These color and intensity shifts will be more visible with transmission, but will also affect illumination and subsurface. Because of this, shader adjustments and texturing should happen in the same color space as the rendering one. (#5262, #5527, #5819)
- **Color space-independent spectral effects**: effects that perform spectral integration, like refractive dispersion, `thin_film`, `blackbody` and `physical_sky` now take into account the rendering color space and provide accurate results for wider gamut spaces, or spaces with different illuminants. There might be some differences for these effects, like higher saturation, due to improved color computations. (#5819, #5835, #5836, #5837)
- **Listing available color spaces**: `kick -lcs` prints a list of available color spaces. (#5832)

Namespaces

- **Namespaces**: Arnold scenes are organized in hierarchies of procedurals, which previously could lead to naming conflicts when identically names nodes existed in different procedurals. Each procedural now has its own naming scope, with the following rules: (#2712)
  - In procedurals .ass files, node names are looked up first in the current scope. If not found, they are looked up in the parent scopes until the scene root is reached.
  - Alternatively, an absolute path may be specified to look up a node starting from the scene root, for example: `"^root_node^proc1^proc2^node_name"`.
  - `AiNodeLookUpByName()` now has a `parent` parameter to specify the scope where recursive lookup begins.
  - Procedural plugins creating nodes using `AiNode()` and `AiNodeClone()` must now set the `parent`parameter, to indicate that the node is a child of the current procedural node.
- **Procedural Node Overrides**: nodes inside a `procedural` can be replaced by other nodes with the `override_nodes` parameter. This may be used for example to replace shaders in an existing .ass procedural. When the parameter is enabled, nodes in the immediate parent scope of the procedural will replace identically named nodes inside the procedural. (#2712)

Other Enhancements

- **OpenVDB**: rendering of .vdb files is now supported in the Arnold core, using OpenVDB 4.0. The `volume` node now has a `filename` parameter that accepts .vdb files. The `volume_implicit` node may be used to render OpenVDB volumes as an implicit surface. The parameters for both nodes are the same as in the previous OpenVDB plugin, except for the velocity shutter start and end which has been replaced by `motion_start` and `motion_end` parameters shared with other shapes. Velocity grids named `v`, `vel` or `velocity` are automatically detected

and used if motion blur is enabled in the scene. `options.texture_searchpath` is used to resolve relative filepaths. (#4844, #5801, #5814)

- **Improved OpenVDB startup time**: On Linux when rendering with many threads, large OpenVDB datasets could cause significant slowdowns at the start of rendering. Startup times are noticeably improved, resulting in buckets rendering faster especially for low AA samples such as during IPR. (#5816)
- **Faster implicit surfaces**: Implicit surfaces, through the `implicit` geometric primitive, render faster, particularly when using the uniform solver. The uniform solver requires fewer solver samples to accomplish much-improved quality, and the levelset solver got slightly faster as well. (#3221, #5472, #2097)
- **Faster curves**: The `curves` geometric primitive now renders about 5 to 15% faster. In addition, when rendering dense hair clumps, `min_pixel_width` is now up to 2x faster and results in more accurate shadowing, at the expense of a slight increase in sampling noise. (#4417, #5725, #5806, #5684, #5806)
- **Faster object initialization**: The initialization time for motion blurred objects is now about 700ns faster. For a scene with 1M motion blurred objects, that's a savings of about 0.7s. (#5379)
- **VR camera**: a new `vr_camera` node is now available that generates stereo views suitable for virtual reality in a variety of formats and projections. (#4906)
- **Quad and spot light roundness**: Quad and spot lights now have a `roundness` parameter, going from a square shape at 0, to rounded corners, to a disk shape at 1. Quad lights also have a new `soft_edge` parameter to soften the edges, similar to the `penumbra_angle` for spot lights. (#5144, #5222)
- **Lazy evaluation of shader array linking**: When calling `AiShaderEvalParamArray()`, no shaders are evaluated anymore. Reading one of the array values later on will trigger the corresponding shader evaluation, and the result is cached to avoid multiple shader evaluations. This means performance no longer depends on the number of links, which can be scaled as necessary. (#2924)
- **RGB camera filter maps**: The data type of camera filter maps has changed from float to `AtRGB` so that colored effects are possible. (#5299)
- **Object `transform_type`**: Shape objects (`ginstance`, `polymesh`, etc.) now have a `transform_type` parameter that specifies what type of motion the object has. Options are `linear`, `rotate_about_center`, and `rotate_about_origin`. `linear` corresponds to the linear interpolation between matrices that Arnold 4.2 used to do when `options.curved_motionblur=false`. `rotate_about_origin` corresponds to `curved_motionblur=true`. Unlike `rotate_about_origin`, which sets the rotation pivot at the origin, `rotate_about_center` will rotate about the object's center. This is the default mode and is useful for wheels, propellers, and other objects which spin. (#3962, #5376)
- **Hair UVs**: `curves.uvs` is a new parameter to set `sg->u` and `sg->v` default texture coordinates, similar to polymeshes. UVs may be specified per curve or per point, with an array that has the same length as uniform and varying user data respectively. `sg->bu` and `sg->bv` still contain the UV coordinates along the width and length of the curve respectively. The `hair` parameters `uparam` and `vparam` have been removed, as they can now be set through `curves.uvs`. (#5646)
- **`ARNOLD_PLUGIN_PATH`**: This environment variable is now used automatically in `AiBegin()` to load plugins, and to find procedural and volume DSOs. Previously this was a convention used by `kick` and the DCC plugins. (#5282)
- **`kick` camera controls**: New command-line flags to tweak the current render camera have been added. You can now set the position (`-position`), up vector (`-up`) and look at point (`-lookat`) for the current camera. This means that `kick -turn` now works for cameras defined with a matrix if you use the `-lookat` flag. (#5311)
- **`kick` exposure keys**: The `[` and `]` keys can be used to decrease or increase exposure in `kick` when in `-ipr` mode. (#5380)
- **`kick` abort on error**: `kick` now aborts on errors during loading of .ass files by default, to avoid rendering when plugins are missing or files are corrupted. Previously it would only abort on errors during rendering. Aborting can still be disabled with `-set options.abort_on_error off`. (#5736)
- **Polygon holes**: a new attribute `polymesh.polygon_holes` is available to add holes to given faces of a mesh. For instance,

```
polymesh
{
 name quad_with_hole
 nsides 2 1 UINT 4 4
 polygon_holes 2 1 UINT 1 0
 ...
```

  indicates that the face with index 1 will be a polygon hole on face 0. Multiple holes per face are supported. The winding of a face used as a hole does not need to be reversed. Also, note that polygonal holes are ignored when a mesh is subdivided. (#2972)
- **Tagging deprecated nodes**: Nodes can be tagged as deprecated by adding boolean metadata named "deprecated" (with the parameter name empty) set to true. These nodes will be listed in `kick -nodes` with `[deprecated]`, and when they are instantiated in an Arnold scene they will issue a warning to remind the user. (#5746)
- **Updated to RLM 12.2BL2**: We have upgraded the license server and the external library controlling the licensing subsystem from version 12.0BL2 to 12.2BL2, a more stable release fixing various crashes, bugs, hangs and memory leaks. (#5754)
- **Updated to OIIO 1.7.12**: We have upgraded from OIIO 1.7.7 to OIIO 1.7.12. (#5672, #5826)

Incompatible changes

Options

- Added `options.subdiv_dicing_camera`, replacing the per polymesh adaptive subdivision dicing camera with a global one. (#5029)
- Replaced `options.aspect_ratio` with `options.pixel_aspect_ratio`. The new pixel aspect ratio follows the usual definition `pixel_width / pixel_height`. Note that the new attribute is the inverse of the old `options.aspect_ratio`. (#5392)
- Renamed `options.shader_searchpath` to `options.plugin_searchpath`, all types of node plugins are loaded from this path. (#5779)
- Removed `enable_fast_opacity`, so that the fast opacity mode is always used. (#5158)
- Removed `auto_transparency_threshold`: The `options.auto_transparency_threshold` has been removed. As long as shaders make sure to use `AiShaderGlobalsStochasticOpacity()`, there is no need for manually specifying a transparency threshold. (#4842)
- Removed `ignore_direct_lighting`: The AOV system provides various indirect lighting AOVs that can achieve the same result as this old debugging option. (#5029)
- Removed `options.curved_motionblur`, in favor of a per object `transform_type` parameter. (#5437)

- Removed `options.bump_space` and `options.bump_multiplier`. Bump mapping is now always in object space. (#5029)
- Removed `options.GI_fallof_start_dist` and `options.GI_fallof_stop_dist` have been removed. (#4793)
- Removed `options.light_gamma` and `options.shader_gamma`. All shader and light attributes are now assumed linear. The `always_linear` metadata is therefore no longer needed and is no longer supported. (#5245)
- Removed rarely used `bottom`, `right` and `woven` modes from the `options.bucket_scanning` enum. (#5642)
- Removed `options.enable_threaded_procedurals`, this was previously already enabled by default. We now require clients to make their geometry procedurals thread safe. (#5029)
- Removed `options.CCW_points`. If polygons have a clockwise winding order, they are now expected to be converted to counter-clockwise when translating the geometry. (#5029)
- Removed `options.shadows_obey_light_linking`, this is now off as was already the default. To have light groups affect shadows, set the shadow group to the same value as the light group. (#5029)
- `options.preserve_scene_data` is no longer propagated when writing to an .ass file. (#5651)
- Removed `options.procedural_force_expand`, since procedurals are always expanded during initialization now. (#5612)

Lights

- Lights with `normalize off` and radius zero now no longer emit any light, for consistency with lights with very small radii. Previously these would work as if `normalize` was on. (#3851)
- Light sampling now uses `options.GI_diffuse_samples` and `options.GI_specular_samples` for multiple importance sampling. If existing scenes render with more noise, increasing these samples maybe be necessary. (#5423)
- Removed constant light decay, and the `decay_type` parameter, so that all lights now have quadratic decay. (#5147)
- Removed support for negative lights (negative `color`, `intensity`, or `shadow_color`). Negative values will now be clamped to zero. (#5162)
- Removed `affect_diffuse`, `affect_specular` and `affect_volumetrics` parameters. Instead `diffuse`, `specular`, `sss` and `volume` multipliers can be set to zero. (#5423)
- Renamed `gobo` parameters `scale_s`, `scale_t`, `wrap_s`, `wrap_t` to `sscale`, `tscale`, `swrap` and `twrap`. (#5183)

Ray Types

- The ray types used for ray visibility, the ray switch shader, GI depth and samples have changed. There are now 5 scattering ray types: diffuse reflection, diffuse transmission (includes SSS), specular reflection, specular transmission and volume scattering. In the options node there is now a GI depth and number of samples for diffuse, specular and transmission rays. `kick` now has arguments to set the diffuse, specular and transmission depth and samples. (#3462, #5572)

Motion Blur

- Arnold and its DCC plugins used to encourage the use of "canonical" 0..1 time for all data in .ass files and in Arnold. For example, if motion blur samples went from frame-relative time of -0.25 to 0.25, in Arnold-land the -0.25 time sample would become 0.0 and the 0.25 time sample would become 1.0. This simplified the data in .ass files, etc but made it difficult or impossible to share .ass standins among vendors or even shots that differed in their time ranges, and to know why the motion blur was mismatched. Now, the `transform_time_samples` and `deform_time_samples` arrays, along with `time_samples` arrays on lights and cameras have all been removed in favor of the simpler `motion_start` and `motion_end` float parameters on each object. These should always be frame-relative values, e.g. -0.25 and 0.25 respectively in the example above. *NOTE* that this means non-uniform motion time samples are no longer possible; if an object has N motion keys, the time interval will always be divided up into N-1 equally-space time intervals. Also, this means that transform motion blur and deformation motion blur will always use the same time range. (#4916)
- Volume motion blur is also controlled by `motion_start` and `motion_end` parameters. The `velocity_fps` should be set to the framerate, so that the velocity can be scaled to the length of the frame. With the frame relative convention, motion start and end should typically default to -0.5 and 0.5 respectively, indicating the velocity was sampled at the center of the frame. (#4844)

Other Changes

- **Visual Studio 2015 redistributable**: In Windows, we now require the VS 2015 redistributable to be installed in order to run Arnold. We expect most of our users to have it already, but if not the installer can be downloaded from the following link: https://www.microsoft.com/en-us/download/details.aspx?id=48145 (#4445)
- **Geometry ID**: The type of geometry `id` attribute and of the associated built-in AOV has been changed from `INT` to `UINT`. (#5315)
- **AOV filtering**: Built-in linear filters (`gaussian_filter`, etc) will no longer work on AOVs of type `INT`, `UINT` or `BOOL`. `closest_filter` should be used instead. (#5175)
- **Custom cameras**: Custom cameras can now implement `camera_reverse_ray` to convert from camera space to raster space. This makes it possible for any custom camera to work as a dicing camera, support `min_pixel_width`, support accurate raster-space wireframe width, etc. (#4477)
- **`procedural` node**: The `procedural` node now only supports loading .ass, .obj, .ply files, and creating procedurals using methods provided through `funcptr`. The `dso` parameter has been renamed to `filename`. `options.procedural_searchpath` is now only used for this node. Procedural plugins instead register new node types now. (#5154)
- **Deferred procedurals**: Removed support for deferred procedurals. All procedurals are now loaded during scene initialization (right before rendering). Also, `load_at_init`, `min` and `max` parameters have been removed. (#5612)
- **Gamma options removed**:
    - Output drivers now take a `color_space` string attribute to specify which color space to use for rendered images. Built-in values are `linear`, `sRGB` and `Rec709`. The default value `auto` will use `sRGB` for 8 bit formats and `linear` otherwise. All `driver_*.gamma` have been removed. (#5244)
    - `image` nodes now have an `color_space` attribute to specify which color space the texture is assumed to be in. Built-in values are `linear`, `sRGB` and `Rec709`. The default value `auto` will use `sRGB` for integer (8 or 16 bit) formats and `linear` otherwise. The attribute `options.texture_gamma` has been removed. (#5254)

    - Legacy gamma related options in `kick` (-g/tg/sg/lg) have been removed. You can now set the output color space for a render with `kick -ocs <string>`. (#5267)
- **Removed `-log` legacy option in `kick`**: Use the more explicit `-logfile <path>` instead. (#5783)
- **32-bit Integer TIFF**: Removed the broken `int32` format from `driver_tiff`. Note that 32-bit integer TIFFs cannot be read by Nuke nor Gimp, and OSX Preview displays a reduced bit depth version, making this format pretty useless anyway. (#2976)
- **Removed `driver_display`**: The `driver_display` node, which was rarely used, has been removed. (#5270)
- **Removed legacy pixel filters**: The following pixel filters, which were rarely used, have been removed: `catrom2d_filter`, `cook_filter`, `cone_filter`, `cubic_filter`, `disk_filter`, `video_filter`. (#5673)
- **Always use auto-bounds for volumes and implicits**: Volume and implicit plugins already notify Arnold of the bounds around the data they provide. Users could previously override this with `min` and `max` parameters, but those have been removed for volume nodes and are unused for implicit plugins. All volume plugins will automatically have a `bounds_slack` parameter available to expand the natural bounding box around the data for purposes of adding positional noise in shaders, etc. Volume plugins are responsible for increasing their reported bounds by the `bounds_slack` amount. Also, the `load_at_init` parameter has been removed from both volume and implicit type nodes, as it had no effect in practice. (#5831)

API changes

General

- Many API functions now accept arguments by reference instead of by pointer. (#2159)
- Many API functions that used to have `char*` arguments or returns will now instead use `AtString`. Some of these changes should be transparent on account of the automatic conversion of `AtString` to `char*`, but in some situations, such as when the returned `AtString` is used by `AiMsg*()` or `printf` type variadic functions, preexisting code will not compile. In this case, the resulting `AtString` should have `.c_str()` called on it. (#5177)
- Added `AiMsgAddCallback()` to install an additional message handling callback alongside the default message processing. (#5736)
- Removed functions of the form `fooAtString(AtString str)` in favor of the overloaded `foo(AtString str)` forms. For instance, `AiNodeAtString(const AtString name)` is now `AiNode(const AtString name)` (#5177)
- Removed `AiLicense{Set|Get}Attempts()` and `AiLicense{Set|Get}AttemptDelay()`. (#2893)
- `AiMakeRay()` and other shader utilities now return their result. (#5300)
- Changes to support hierarchical namespaces:
    - `AiNode`: It now receives, in addition to the node entry name, the node name, and the parent (or NULL for a root node).
    - `AiNodeLookUpByName`: It receives a pointer to the parent node, so it can start relative search in the appropriate context (not necessary for absolute paths).
    - `AiNodeClone`: Since it creates a new node, it will now receive, like the AiNode function, the node name and (optionally) the parent node.
- Removed `AtProcInitBounds`/`procedural_init_bounds` method from the procedural API, since procedurals no longer have user provided bounds (they are always open during initialization). (#5612)

Arrays

- `AtArray` is now opaque: Any operation involving `AtArray` objects has to be done through its new API, and not accessing its struct members which are now hidden. (#4082)
- `AiArrayAllocate()` does not initialize memory: When an `AtArray` is created with `AiArrayAllocate()`, its contents are no longer initialized to zero and are instead left uninitialized. This results in slightly faster code. (#4507)
- `AiArrayResize()` was added, to resize an `AtArray` in place, preserving its existing data. (#2073)
- Removed `AiArrayModify()`. (#5279)

Colors

- Removed `AiRGBCreate()` and `AiColor()`, replacing them with actual C++ constructors. For instance, `AtRGBA c = AiRGBACreate(1,0,1,0);` can now be written as `AtRGBA c(1,0,1,0);`. (#2158, #5271)
- Removed `AtColor`: `AtColor` has been deprecated in favor of `AtRGB`. It is now in ai_deprecated.h. (#5352)
- Removed `AiColorLerp()` and `AiRGBALerp()` since the preexisting `AiLerp()` already did the same thing. (#5141)
- Removed `AiColorReset()` and `AiRGBAReset()` since it's simpler to just assign AI_RGB_BLACK and AI_RGBA_ZERO. (#5279)
- Removed `AiColorIsZero`, replace with the equivalent `AiColorIsSmall`. To see if a color is exactly zero use `== AI_RGB_BLACK`. (#5305)
- Removed `AiColorEqual()` in favor of the cleaner `operator==` and `!=` methods. (#5279)
- Removed `AiColorGamma()` and `AiRGBAGamma()`. (#5267)
- Renamed `AI_RGBA_BLACK` to `AI_RGBA_ZERO`, to clarify that its alpha component is zero. (#3928)

Math

- Removed `AI_TYPE_POINT`: we no longer differentiate between points and vectors and so have removed `AI_TYPE_POINT` and all functions that operated on points when a similar vector function was available. For instance, instead of using `AiNodeSetPnt()`, you can directly use `AiNodeSetVec()`. Search and replace of "Pnt" with "Vec" and "POINT" with "VECTOR" will go most of the way towards updating the code. `AtPoint` and `AtPoint2` have been deprecated and moved into ai_deprecated.h. (#5059)
- Removed `AiM4Berp()`, `AiColorBiLerp()`, `AiRGBAbiLerp()`, `AiColorHerp()`, `AiRGBAHerp()`, `AiColorBiHerp()`, `AiRGBAbiHerp()` and `BIHERP()`. (#4646)
- Removed `AiVector()` and replaced them with actual C++ constructors. (#2158, #5271)
- Renamed `MIN()`, `MIN3()`, `MIN4()`, and corresponding `MAX` to `AiMin()` and `AiMax()`, which are overloaded and support 2, 3, or 4 arguments. (#5232)

- Renamed SQR to AiSqr , CLAMP to AiClamp, LERP to AiLerp and ACOS to AiSafeAcos. (#5595)
- AtBBox: The AtBBox related functions are now class member functions, so instead of writing something like: AiBBoxIsEmpty(bbox), we do: bbox.isEmpty() (#5446)
- AtMatrix: in the Python API, matrix elements are now accessed using the more natural matrix[i][j] syntax. (#5242)
- NaN/inf checking: We used to have three different names for NaN/inf checking of floats, colors and 3D vectors: IsFinite, Corrupted, and Exists. We have now consolidated all of those functions with a consistent name and meaning, following the float version. Therefore, AiV3Exists() has been renamed to AiV3IsFinite(), and AiRGB/ACorrupted() have been renamed to AiRGB/AIsFinite(), which reversed their meaning. (#2145)
- AiBuildLocalFrameShirley() has been replaced by AiV3BuildLocalFrame(), which is faster and has fewer discontinuities. (#3213)
- AtSampler: AiSamplerSeeded() has been renamed to AiSampler(), so that we now always require a random seed. The sampler now also supports 1D, 2D and 3D sample patterns, instead of only 2D. (#5451)
- AtVector and AtRGB/A comparison operators: AtVector, AtVector2, AtRGB, and AtRGBA now have support for the <, <=, >, and >= operators. After doing a comparison, calling AiAny() on the result will return a bool if any of the element comparisons were true and AiAll() will return true if all the element comparisons were true. For instance, AiAll(AtVector(1,2,3) < AtVector(0,10,20)) will return false since the first element comparison 1 < 0 is false. AiAny(AtVector(1,2,3) < AtVector(1,10,20)) on the other hand returns true. (#2157)
- AiFastPow() added: this fast power function is suitable for cases where speed is more important than accuracy. (#5573)
- Removed AiV3RayPoint(). You now need to do the math yourself, so AiV3RayPoint(out, origin, dir, t) becomes out = origin + dir * t. (#5089)

BSDFs

- The BSDF API has been completely revamped to support more advanced and optimized BSDFs, and to be usable for more advanced rendering algorithms. More detailed information is available in the BSDF documentation. (#4783)
- AiBSDFIntegrate() can integrate both direct and indirect light. If only one or the other is needed, passing NULL for the unneeded component will skip it. If both are needed it is fastest to compute them in a single function call. (#5423)
- AiOrenNayarBSDF(), AiMicrofacetBSDF(), AiMicrofacetRefractionBSDF() and AiMetalBSDF() are the built-in BSDFs. AiOrenNayarIntegrate(), AiMicrofacetBTDFIntegrate(), AiDirectDiffuse() and AiIndirectDiffuse() have been removed. (#4783, #5631)
- AiDielectricFresnel() and AiConductorFresnel() have been added. (#5631)
- Renamed AiFresnelWeight() to AiSchlickFresnel(). (#5631)
- Removed Cook-Torrance, stretched Phong, Ashikhmin-Shirley and Ward BRDFS: use Microfacet BRDF instead. (#5161)
- Removed AiHairDirectDiffuseCache() and hair shader diffuse_cache parameter, use diffuse lighting without cache. (#5161)

SSS

- Removed AiBSSRDFCubic() and AiBSSRDFGaussian(). AiBSSRDFEmpirical() may be used instead. The cubic radius multiplied by 0.13 approximately matches the mean free path for the empirical BSSRDF. (#5161)
- Removed AiSSSTraceSingleScatter(). The empirical BSSRDF that includes an approximation for single scattering may be used instead, or for brute force volume scattering volume closures may be passed to the interior parameter of AiClosureTransparent() or AiMicrofacetRefractionBSDF(). (#5161)
- Removed AiSSSEvaluateIrradiance(), use raytraced SSS instead. (#4796)

Volumes

- Removed AiShaderGlobalsSetVolume*. Instead volume closures must be used instead, which have the advantage of making volume shaders mixable. (#5503).
- Removed sg->Vo and sg->Ci. Instead AiClosureVolumeAtmosphere() must be used instead. (#4793)
- Renamed volume_scattering node to atmosphere_volume to clarify its purpose. (#5495)
- Removed volume shape node with dso parameter. Plugins will instead register custom node types now, for example volume_openvdb. (#5154)
- Volume plugins all now have a bounds_slack parameter. The plugins are responsible to read that parameter and apply it to the bounds they report in the AtVolumeData struct. (#5831)

Lights

- AiBSDFIntegrate() can be used to integrate both direct and indirect light. If more low level access is needed, it is still possible to manually implement light loops and multiple importance sampling using the light sampling API. See the documentation for an example of custom direct light integration. However it is strongly suggested to use closures and leave the light sampling to Arnold instead, to benefit from sampling optimizations and support light path expressions. (#5423)
- AiLightsGetSample() now returns an AtLightSample with all light sample data, rather than writing it to shader globals. For light filters the AtLightSample in sg->light_filter can be used to read and modify the current light sample. (#5423)
- AiLightsTrace() and AiLightsTraceRayTypes() have been added for low level direct light integration. (#5423)
- Removed AiEvaluateLightSample(). (#5423)
- Removed AiLightGetAffectDiffuse() and AiLightGetAffectSpecular(). AiLightGetDiffuse() and AiLightGetSpecular() now automatically return zero for when light.affect_diffuse or light.affect_specular are off. A new AiLightGetInfluence() utility function can also be used instead of these functions, returning the right influence depending on the shading context and ray type. (#5423)
- AiLightsIntegrateShadowMatte() was added to compute more useful shadow mattes. The new matte has color information if needed, correctly weights light occlusion when multiple lights are present and takes into account the surface's BSDF. (#5496)
- Removed AiLightsGetShadowMatte(), use AiLightsIntegrateShadowMatte() instead. (#5496)

Plugins

- Procedural and volume plugins now work more similar to other plugins such as shaders, drivers and cameras. They are automatically loaded from paths passed to `AiLoadPlugins()`, `ARNOLD_PLUGIN_PATH` and `options.plugin_searchpath`, registering new node types and defining parameters when they are loaded. The available methods are the same, but the syntax has changed, for details see this procedural API example. (#5154)
- Derived types and type-names of the newly introduced node-like implicit, procedural and volume shape nodes are now exposed through corresponding `AtNodeEntry` functions in the API (#5686)
- Optional `node_plugin_initialize` and `node_plugin_cleanup` methods have been added. These methods can be used for initializing a plugin, if there is some initialization or data to be shared between nodes of the same type. These methods are only called if a node of this type is created. `AiNodeGetPluginData()` can be used to retrieve the plugin data created in `node_plugin_initialize`. (#5154)
- `node_initialize` and `node_update` no longer provide a `params` parameter, and `AiNodeGetParams()` has been removed too. `AiNodeGet*/Set*` and `AiShaderEvalParam*` must now be used to access parameters. For best performance these parameters should be read and stored with `AiNodeGetLocalData()` in `node_update`, and then retrieved though `AiNodeGetLocalData()` in performance critical methods such as shader evaluation and pixel filtering. (#3796, #4853, #5145)
- Removed deprecated parameter creation functions with uppercase types `AiParameterENUM()` etc; use the lowercase versions instead, e.g. `AiParameterEnum()`. (#5182)
- `AiMetadataSet???`: these API functions now operate on a node entry, for symmetry with the `AiMetaDataGet???` functions. The prototype for `node_parameters` has been modified to receive a `AtNodeEntry* nentry` parameter. (#5292)
- `AtParamValue` values are now accessed as a function instead of directly, so for instance, `param.RGB = my_color` is now `param.RGB() = my_color`. (#5194)
- Removed `AiCameraGetLocalData()`, `AiFilterGetLocalData()` and `AiDriverGetLocalData()`. Instead, `AiNodeSetLocalData()` and `AiNodeGetLocalData()` can be used now to attach custom data to camera, filter and driver nodes, the same as for shaders. (#5281)
- Removed `AiCameraDestroy`, `AiFilterDestroy` and `AiDriverDestroy`. They no longer need to be called for custom cameras, filters or drivers, as this de-initialization now happens automatically inside Arnold. (#5278)
- Removed `AtVolumePluginVtable.SampleDeprecated()`, use the `Sample()` method instead. (#4180)
- Removed `AiLoadPlugin()` since `AiLoadPlugins()` does the exact same thing. (#4646)

Shader Globals

- Added `AiShaderGlobalsGetUniformID()`, for getting a unique per face, subdivision patch, curve or point ID. This uniform ID may for example be used for randomizing colors per hair curve. DCC app plugins can benefit from this by removing the "curve_id" user data export that is used by certain shaders, and instead using this function, which would reduce memory usage a bit. (#2436)
- Removed `AtShaderGlobals.out_opacity`. Use `AiClosureTransparent()` instead to create mixable shaders with transparency. (#4793)
- Removed `AtShaderGlobals.area`, use the `AiShaderGlobalsArea()` method instead. (#4180)
- Removed image space coordinates from `AtShaderGlobals`: the `sx` and `sy` fields in `AtShaderGlobals` have been replaced with subpixel camera sample coordinates `px` and `py`. If `sx` and `sy` are needed they can be computed using the following code. (#5278)

```
const float sx = -1 + (sg->x + sg->px) * (2.0f / AiNodeGetInt(AiUniverseGetOptions(), "xres"));
const float sy =  1 - (sg->y + sg->py) * (2.0f / AiNodeGetInt(AiUniverseGetOptions(), "yres"));
```

- `AtShaderGlobals.bu` for curve objects has changed to cover the entire width of the curve from 0 to 1. If UVs are specified through `curves.uvs`, they will affect the `u` and `v` texture coordinates. The parametric coordinates `bu` and `bv` always range from 0 to 1 along the width and length of the curve respectively. (#5212)
- `AiShaderGlobalsGetVertexUVs()`: `uvset` parameter added, an empty `AtString()` returns the default UVs. (#5653)
- `AiShaderGlobalsStochasticOpacity()` now applies `geo_opacity` for min pixel width, and should no longer be manually applied by shaders. Instead, calling `AiShaderGlobalsStochasticOpacity()` in the shader will handle the opacity correction required by `min_pixel_width`, and any manual application must be removed to avoid double opacity correction. (#5399)
- For any `opacity` or equivalent parameters in shaders, `AiShaderEvalParamOpacity()` should be used to evaluate it. This enables optimizations for rendering of transparent surfaces. Failing to call this function will result in very slow rendering of opacity-mapped cutouts such as tree leaves or other overlapping transparent surfaces. (#5357)

```
const AtRGB opacity = AiShaderGlobalsStochasticOpacity(sg, AiShaderEvalParamOpacity(p_opacity));
```

Bug fixes

5.0.0.0

- #3851 unnormalized lights with zero radius should NOT be treated as normalized
- #3962 Curved motion blur can result in large excursions
- #4108 number of pixels stat is too high
- #5166 fast_opacity results in noise for low auto_transparency_depth

- #5169 change default bump2d height to 1.0 so it matches bump3d
- #5181 kick -info should print default values for matrix parameters too
- #5189 Fix assorted memory leaks
- #5212 Curve barycentric and uv coordinates inconsistency
- #5289 AiNodeClone increases instance overhead memory stats
- #5297 Include detailed reason for failure in append (checkpointing) errors
- #5326 Kick should support -o for raw drivers
- #5367 Subdivs: bad limit surface near vertices connected to both boundaries and creases
- #5368 Subdivs: creased vertices are not correctly handled for UVs or indexed user data
- #5393 Assert triggered when overriding multiple shader assignment with single shader
- #5396 Drivers with append ON should not touch completed files
- #5402 Deep driver: use smaller minimum alpha value for samples
- #5420 Remove incorrect texel offset from persp_camera.uv_remap
- #5425 Subdiv: adaptive mode should work with orthographic cameras
- #5427 Subdiv: adaptive mode not working with very small subdiv_adaptive_error
- #5430 Corrupted output for high resolution EXR with many AOVs in scanline mode
- #5432 Displacing meshes with high valence vertices (> 256) can crash
- #5448 crash in AtNode::getUserDataMemoryUsage()
- #5454 AiIrradiance returns too much indirect light
- #5457 Mesh light not evaluating UDIMs properly
- #5458 Fix minor memory leaks
- #5459 Account correctly for mesh lights importance table shader calls
- #5468 Different binary outputs from same scene
- #5478 Slow render with infinite velocity values in OpenVDB grids
- #5538 Crashes when accessing the camera in free render mode
- #5551 Portals not working with unconnected skydome light color
- #5562 Skydome light outside portals should be blocked
- #5577 Race condition in procedural population
- #5593 crash in AiTextureAccess when texture_autotile is changed
- #5643 Array sanity check happens with partially filled array data
- #5644 Crash in procedural creating an object with more than 2 motion keys (5.0)
- #5664 Use GGX instead of Beckman for plastic/metal mode in the utility shader
- #5703 Missing nodes should report an error and abort the render
- #5710 Curve minimum pixel width not working for some camera positions
- #5721 face_visibility crash
- #5730 Crash using Arnold 5 in C4DtoA from thread other than main
- #5733 Watermarked deep file hangs render
- #5768 AiMakeTx should close file handles of input textures
- #5773 Volumes missing IPR update when changing min and max parameters

- #5792 AiMakeTx should output error messages
- #5794 Add d'Eon and Zinke hair BSDFs in API
- #5815 motion parameters should not be inherited
- #5818 Artifacts with refractive levelset implicits
- #5827 Automatic reloading of a procedural with lights doesn't work
- #5836 Blackbody: should match for all rendering color spaces
- #5837 Refractive dispersion: maximize match for all rendering color spaces
- #5851 Add residual terms to `standard_hair`
- #5863 Free mode: crash when logging is on
- #5867 UINT AOVs not correctly written out
- #5871 Only warn once for deprecated nodes
- #5881 Allow changing thread_priority in Windows
- #5896 AiTraceProbe not evaluating opacity
- #4955 OpenEXR driver: only warn about long metadata names if they are actually written out to output file
- #5858 IPR moving in kick doesn't work in flat scenes
- #3928 Rename AI_RGBA_BLACK to AI_RGBA_ZERO