5.0.2.2

- #6257 s/twrap not working with <attr> expressions set in the image node filename
- #6411 Excessive memory use with compressed UVs
- #6417 Small PolyMesh memory leak at render shutdown
- #6422 OSL raytype mismatch
- #6440 Handle invalid characters when image tokens are resolved
- #6441 Backfacing normals not supported in normal_map
- #6442 Chained bumps using bump2d isn't working
- #6453 Custom procedurals cannot create nodes before proc_init
- #6454 Crash With Linear Subdivision, Vertex Normals, and Deformation Blur

5.0.2.1

- #6409 ASS metadata load is too slow
- #6412 Crash when setting an array user parameter with its current value

5.0.2.0

Enhancements

- **New sub-surface scattering algorithm**: A new, more accurate way of calculating SSS has been added. Unlike the current empirical BSSRDF method based on diffusion theory, this new method actually traces below the surface with a real random walk and makes no assumptions about the geometry being locally flat. This means it can take into account anisotropic scattering like brute-force volume rendering and produces much better results around concavities and small details. It can also be substantially faster for large scattering radius (i.e. large mean free path) compared to the old method. On the other hand, the new method can be slower in dense media (i.e. small mfp), does not support `sss_setname` for blending two surfaces together, may require redialing materials to achieve a similar look, and is more sensitive to non-closed meshes, "mouth bags", and internal geometry potentially casting shadows. This new algorithm is exposed in the `standard_surface` shader via the new parameters `subsurface_type` (with enum values `diffusion` and `randomwalk`) and `subsurface_anisotropy` (Henyey-Greenstein's eccentricity g from -1.0 to +1.0). The default is to use the old empirical diffusion method in order not to break the look of existing scenes.
- **Car paint shader**: We are now shipping a dedicated shader for car paint, which can be thought of as the combination of a simplified version of the `standard_surface` and `flakes` shaders. This shader can create a wide range of car paint looks without having to connect several nodes. For example, a pearlescent effect can be easily added to both the specular and flakes layers by simply tweaking a few parameters such as `specular_flip_flop` and `flake_flip_flop`. An arbitrary number of layers of flakes can be used (`flake_layers`). The flakes at a deep layer are covered by the ones closer to the surface and more tinted by pigments (specified by the `transmission_color parameter`).
- **Subdivision frustum culling**: Subdivision patches outside the view or dicing camera frustum will not be subdivided. This is useful for any extended surface that is only partially visible as only the directly visible part will be subdivided. Similarly, no subdivision work will happen if a mesh is not directly visible. This can be turned on globally by setting `options.subdiv_frustum_culling true` and can be turned off for specific meshes with `polymesh.subdiv_frustum_ignore true`. The global `options.subdiv_frustum_padding` adds a world space padding to the frustum that can be increased as needed to minimize artifacts from out-of-view objects in cast shadows, reflections, etc. Note that motion blur is not yet taken into account and moving objects might require some additional padding.
- **Improved accuracy of UV coords**: Mesh UV coordinates are now handled with higher numerical precision, fixing jagged artifacts that sometimes appeared when using high-resolution UDIM textures over very wide UV ranges.
- **Improved volume sampling of low-spread lights**: Low-spread quad and disk lights should now produce less noise and show increased performance when participating in atmospheric scattering effects.
- **Improved procedural namespace memory usage**: Reduced memory used by procedural namespacing by about 28KB per procedural primitive, so that procedurals now use very little additional memory. In a scene with 100K procedurals, this gave about a 2.7GB reduction in memory use.
- **Faster opacity masks**: Texture map-based opacity masks will now be faster to render. Testing indicates around 3-13% faster.
- **Faster IPR**: The message logging system has been optimized, resulting in about 10% faster performance in IPR, like while moving the camera. In addition, the IPR mode in `kick` has been made substantially more responsive.
- **Faster `AiNodeDestroy`**: we have optimized removal time for nodes contained in procedurals, greatly reducing the shutdown time at the end of a render in complex scenes.
- **Faster `triplanar` shader**: Texture filtering in the `triplanar` shader has been improved, giving better antialiasing and up to a 2x speedup, specially when using high-resolution texture maps.
- **Celullar option in `triplanar` shader**: The `triplanar` shader now supports projection through Voronoi cells using the new `cell` parameter. The rotation angle of the projected texture for each cell can be controlled with the `cell_rotate` parameter. Cells can be smoothly blended using the `cell_blend` parameter.

- **Improved `flakes` shader**: The `size` parameter is replaced by the `density` parameter, which makes it easy to control the size and number of flakes. Alpha channel can be used as a mask. The new `flakes` shader supports non-disc shapes and 3D flakes, which are useful to render gemstone inclusions like goldstone, for example.
- **Improved `shadow_matte` shader**: We have revamped and simplified the shader to make it easier to use, and fixed a number of long-standing issues: Indirect illumination now fills the global `diffuse_indirect` and `specular_indirect` AOVs, so we have removed the shader's (confusingly named) `indirect_diffuse` and `indirect_specular` AOVs. Self-reflections are no longer rendered. A new `specular_IOR` parameter was added that controls Fresnel reflection. Parameters `offscreen_color` and `background_type` were removed. The new enum parameter `background` can be set to either `scene_background` (default) or `background_color`, which allows to connect a specific texture in the `background_color` parameter slot. Parameter `alpha_mask` was added to control whether the alpha must be opaque or if it has to contain the shadow mask.
- **Support for more OSL attributes**: OSL shaders now support `getattribute()` lookups of standard camera attributes (e.g. `camera:fov`, `camera:resolution`, etc) as well as the geometry attributes `geom:type`, `geom:name`, `geom:bounds`, and `geom:objbounds` on objects.
- **Transmit AOVs and Alpha**: The `standard_surface` shader with transmission can now pass through AOVs, by enabling the `transmit_aovs` parameter. If the background is transparent, then the transmissive surface will become transparent so that it can be composited over another background. Light path expression AOVs will be passed through, so that for example a diffuse surface seen through a transmissive surface will end up in the `diffuse` AOV. Other AOVs can also be passed straight through (without any opacity blending), which can be used for creating masks for example.
- **Improved multi-threaded render time stats**: Render times when using more than one thread did not really work. We have improved this so that render times are now much more reliable and useful and can now be confidently used to determine what parts of Arnold are the most expensive. In particular, the subdivision and displacement times will now show how much total time was used as well as what fraction of that time was spent with threads unable to do useful work. This "threads blocked" time can often be lowered by using larger buckets or the `randor bucket_scanning` ordering.
- **Custom procedural namespaces**: Procedurals can now declare a custom namespace using the new `namespace` parameter. This custom namespace can be used instead of the procedural name, to reference contents through absolute or relative paths. Multiple procedurals can share the same namespace by using the same custom name. Also, they can declare an empty name and they will use the global namespace. (#6085)
- **Added `-turn_smooth` option to `kick`**: When using `kick -turn`, the `-turn_smooth` flag can be added to smoothly start and stop the movement as the original position is reached with a cubic ramp.
- **Added `-laovs` option to `kick`**: Using `kick -laovs file.ass` will display a list of all the built-in AOVs and all the AOVs registered by this .ass file.
- **Added cputime heatmap view to `kick`**: When using `kick` you can now toggle between viewing kicks default output and a cputime heatmap with the `T` key. The mapping of the heat map can be scaled with the `[` and `]` keys.
- **Support for wasd keys in `kick -ipr m`**: Running `kick` in Maya-style IPR mode with `-ipr m` now supports "wasd" style keyboard movement. This was previously only available in Quake-style mode, `-ipr q`.
- **`maketx` version info**: The custom `maketx` binary that ships with Arnold now reports, both in the command-line and in the embedded EXR headers, that it was built specifically for "OpenImageIO-Arnold", to distinguish it from the official "OpenImageIO" one.
- **`maketx` color spaces**: The custom `maketx` that ships with Arnold now supports OCIO and SynColor (when available) through the `colorengine`, `colorconfig` and `colorconvert` flags. See `maketx --help`.
- **`AiMakeTx` reports info messages**: `AiMakeTx` now also prints out OIIO informational messages (generated in verbose mode, for instance) in addition to errors.
- **`AiMakeTx` releases input file lock sooner**: `AiMakeTx` will now close the input texture as soon as possible instead of waiting for all the maketx jobs to finish.
- **`AiMakeTx` and `maketx` optimized flags**: Both `AiMakeTx` and `maketx` now always run with the flags `--monochrome-detect --opaque-detect --constant-color-detect --fixnan box3 --oiio`, which can result in smaller .tx files that are faster to load and take less memory.
- **Report when textures are changed during render**: The log files now report when texture modifications during a render cause a texture read error, which can happen in certain pipelines.
- **OCIO color space family support**: The OCIO color manager now implements color space enumeration by family. This is useful for UI drop down organization.
- **OCIO view/display enumeration**: The OCIO color manager can now enumerate view/display combinations using the "View (Display)" family. This lets client programs filter color spaces when only a display transform is appropriate.
- **.ass metadata from compressed files**: We now support loading metadata from `.ass.gz` files through the `AiMetadataStoreLoadFromASS()` API function.
- **Upgraded to OIIO 1.7.17**: OpenImageIO has been upgraded to OIIO 1.7.17.

API additions

- **Color space family enumeration**: Existing color space families for the current config can be enumerated using the new API methods `AiColorManagerGetNumFamilies` and `AiColorManagerGetFamilyNameByIndex`. The addition of these new API methods requires any existing custom color managers (which we know are very rare) to be recompiled.
- **`AiAOVSampleIteratorGetPixel()`**: Custom filters can now determine what pixel is being filtered with the new API method `AiAOVSampleIteratorGetPixel()`.
- **`transparent` LPE label**: When setting the `transparent` LPE label on a BSDF, the surface will act as if it is transparent and pass through AOVs. This would typically be used for transmission BSDFs, as it is in the `standard_surface` shader.
- **Deprecated API warnings**: Defining `AI_ENABLE_DEPRECATION_WARNINGS` will cause the compiler to emit warnings if deprecated Arnold API functions are used.
- **Random walk SSS closure**: The new random walk SSS algorithm is exposed in the C++ API as `AiClosureRandomWalkBSSRDF()`. The corresponding OSL closure is `randomwalk_bssrdf`.

Incompatible changes

- **`autotile` disabled by default**: We found that the `autotile` code in OIIO does not scale with high-resolution textures. In order to avoid very slow loading of untiled textures (such as JPEG) when autotile was enabled, we have now changed the `autotile` default setting to 0, which effectively disables it. In very rare cases, when rendering with a large number of high-resolution untiled textures, this change might degrade performance as the texture cache will blow up. The real solution is to never use untiled files, and instead convert all untiled textures to .tx tiled files.
- **`maketx` dependencies**: The custom `maketx` that ships with Arnold now depends dynamically on `libai.so` and optionally on `syncolor_shader.so`, therefore to work correctly it needs to be run from its installation folder, like `kick`, or alternatively the new dependencies should be copied to where `maketx` is running from.
- **Light groups and volume shading**: Just like with surface shapes, Arnold now obeys light group assignment on volume shapes and surfaces with volumetric interiors. While in many cases desirable, this can produce an unexpected change in the final image in scenes with light group assignments.
- **`flakes` shader**: It was not easy to control the number of flakes with the `size` and `scale` parameters because they were mutually dependent. Now this can be easily done using the new `density` parameter. The shape of each flake has been changed from disc to Voronoi cell, which is more suitable to render inclusions of gem stones. The shader output type has been changed from RGB to RGBA to support a mask.
- **`motionvector` AOV**: The motion vector scaling factor in the built-in `motionvector` AOV has changed. This was required to fix a bug that caused zero motion vectors for certain shutter positions. The output from the `motion_vector` shader is unchanged: it can be used as a workaround in your old scenes if you require the previous scaling.
- **`shadow_matte` changes**: The shader AOVs `indirect_diffuse` and `indirect_specular` were removed, since the shader now fills the corresponding built-in AOVs. Parameters `offscreen_color` and `background_type` were removed. Specular reflection is now affected by Fresnel. To roll back to the previous specular behaviour, set `specular_IOR` to a high value like 100, which effectively disables the Fresnel effect. See notes in the enhancements section above.

Bug fixes

5.0.2.0

- #3319 Alpha not fully opaque in output images
- #4559 Non-linkable light colors should have the linkable metadata disabled
- #5974 Distant light not motion blurring direction
- #5981 Incorrect melanin absorption values for wide gamut rendering color spaces
- #6086 procedural memory overhead
- #6087 maketx fails when run on Windows with read only (mandatory) user profiles
- #6089 Deep EXR output of light path expressions missing volumes
- #6094 artifacts in latlong skydome_light
- #6095 Fresnel discontinuity in diffuse term when texture is connected to specular_color
- #6097 Subdiv: duplicate vertex index in face causes crash
- #6104 Quad lights do not work with projected textures
- #6116 Warn that images will be watermarked if license authorization fails
- #6124 mesh_light crashing when provided non-mesh node
- #6128 curvature shading differences when seen through glossy transmission/reflection
- #6135 crash during accel construction if there are over 65k overlapping primitives
- #6136 Indirect sample clamp not visible in AOVs
- #6138 light groups not supported by volume shapes and surface shader interiors
- #6141 "A" AOV is black when 8 light AOVs are used
- #6143 AiShaderGlobalsGetVertexUVs not working for uvsets in free render mode
- #6151 Black AOV output due to conflicting AOV type redefinition
- #6162 AiShaderGlobalsGetPositionAtTime, AiTraceBackground, and AiVolumeSample crash in background shading context
- #6166 AiM4Scaling does not work in python
- #6168 Overriding view direction for metal BSDF not supported

- #6170 Crash creating motion blurred min_pixel_width inside a procedural
- #6177 wireframe artifacts when not in raster-space
- #6182 MotionVector AOV empty for negative motion start / end
- #6184 MtoA crash when saving a scene using motion blur
- #6186 triplanar uses wrong mipmap level for some parts of the object
- #6187 triplanar uses overly high res mipmaps
- #6192 Crash when removing and recreating a procedural instance
- #6202 counter overflow crash in big scanline EXR images
- #6203 Camera corruption after substantial foward zooming with kick -ipr m
- #6219 photometric_light filename does not support environment variable expansion
- #6222 Metallic BSDF albedo is always white
- #6242 Crash when removing a procedural node
- #6245 shadow_matte AOVs
- #6248 Remove offscreen_color from shadow_matte
- #6249 Remove shadow_matte background_type
- #6250 Re-introduce background_color in shadow_matte
- #6251 shadow_matte self-reflections / self-shadowing
- #6254 trace_set: crash when destroying shader
- #6255 sss_irradiance_shader fails with closure-based shaders
- #6256 Add alpha_mask in shadow_matte
- #6286 Volume shader: crash when connected as atmosphere shader
- #6287 Crash with invalid options.atmosphere and options.background shaders
- #6288 Miscellaneous crashes (empty nurbs, empty implicit, atmosphere shadow_matte)
- #6292 Deep Driver: write errors hang the render
- #6293 Deep driver: append does not work with overscan renders
- #6294 Bucket call back: user bucket coords should be snapped to bucket grid
- #6295 Crash when saving empty ginstance with open_procs enabled
- #6297 Camera differential evaluation can cause crashes or hangs with OSL shaders linked to camera
- #6313 crash when 4-channel opacity texture has no alpha=0
- #6331 Crash on Windows when loading plugins without .dll extension
- #6353 barndoor light filter result not order-independent
- #6174 "host app" metadata item not readable by AiMetadataStoreLoadFromASS
- #6258 Range shader can sometimes ouput infinite or nan
- #6303 Support upper and mixed case versions of .ass / .ass.gz extensions