4.2.1.0

Milestone 4.2.1

September 8, 2014

Enhancements

- **EXR overscan support**: Overscan is now supported by extending the render region beyond the regular image coordinates. E.g. a ten-pixel overscan for a 640x480 image in all directions can be achieved with options region_min_x,region_min_y,region_max_x,region_max_y of -10, -10,649,489. The old render region defaults of -1,-1,-1,-1 will continue to render the whole image, but new defaults (INT_MIN) are available going forward indicating the render region is not used. EXR output will fully respect the overscan region with differing data and display windows, but the other image formats will simply enlarge the final image resolution to incorporate overscan since they cannot accomodate data outside the display area of the image. (#3102)
- **Improved Cook-Torrance specular and refraction sampling**: Many scenes will now render with significantly reduced noise for glossy reflection and rough refraction with the microfacet-based Cook-Torrance BSDF. In particular, BSDFs viewed at grazing angles or involved in multiple glossy or refraction bounces are improved. (#4064, #4196, #4209)
- **Improved sampling of textured quad lights**: Quad lights now use a new sampling strategy that adaptively chooses the best sample distribution for each shading point. The noise reduction is most noticeable with textured lights, both on surfaces and in volumes. (#3567)
- **Automatic texture_max_open_files**: The options.texture_max_open_files is now set by default to 0, which means that the maximum number of texture files that can be simultaneously opened is automatically computed by Arnold using a new heuristic. This now works for Windows, Linux, and OS X. We expect that the majority of users will be able to leave this at 0 and get the best performance. If Arnold runs out of file handles, Arnold is now more likely to be able to recover without crashing. If the automatic texture_max_open_files that is set is deemed too low, then texture_max_open_files can still be set manually. However, a better solution (and often the only solution) on Linux and OS X is to increase, in the OS, the open file handle hard limit. (#4139, #4140, #4141)
- **Improved performance for shadow ray texture lookups**: For shades done to shadow rays (transparency), texture lookups now use less texture I/O. (#4167)
- **Improved performance for rough glossy texture lookups**: This improvement was supposed to have been in 4.0.12, but due to a bug, it was never actually enabled in the AiBRDFIntegrate() function used by the standard shader. Shaders that were manually calling the older APIAiCookTorranceIntegrate() were never affected by this bug and have been able to use this improvement since 4.0.12. Now both AiBRDFIntegrate()and AiCookTorranceIntegrate() make use of this feature. Now that the standard shader finally has this feature, users should find that glossy reflections will pull the correct texture mipmap level across all distances while maintaining an optimal amount of texture sharpness. This can result in dramatic reductions in texture I/O and lessens the need for using the texture_glossy_blur option, which we now recommend setting to zero. Unfortunately, third party shaders that do their own integration and manual ray tracing (such as alSurface and Kettle) will not be helped by these fixes, something that we hope to address in a future version. (#4164, #4176)
- **Faster IPR when rendering with many textures**: Time between IPR progressive frames has been decreased when there are many textures being used. This is especially evident when the textures reside on a high latency file server. (#4152)
- **Low-level optimizations**: Several optimizations were made to improve performance. Some of these are most noticeable on linux and/or when rendering empty or low-complexity regions. (#4193, #4212, #4228, #4149, #1231)
- **Upgraded OIIO to 1.4**: This new version comes with many little improvements and bug fixes. For instance, processing a 54k-resolution TIFF file withmaketx on Linux is now 4.5x faster compared to the previous version. However, we now requires a manual texture flush in order for changes made to texture files be reflected inside of arnold. (#3593, #3646, #3826, #4154)

API additions

- **Python 3 support**: The Arnold Python API and pykick are now fully compatible with Python 3, workarounds for module importing and unicode strings are no longer needed. (#4239)

Incompatible changes

- **Difference in microfacet BTDFs**: Rough refraction viewed at grazing angles or using Phong-smoothed normals renders slightly darker now, especially for high roughness values. This affects the standard shader and AiMicrofacetBTDFIntegrate(), and fixes a bug in the previous BTDF sampling. (#4196, #4231)
- **Handling of unassociated alpha for TGA and PSD**: OpenImageIO 1.4 has more conformant handling of unassociated ("unpremultiplied") alpha in TGA and PSD files. This results in improved texture filtering for typical files, but may also cause TGA files that are missing the appropriate metadata to render darker.

Bug fixes

Ticket	Summary
#3998	Writing an .ass with open procedurals will not preserve overrides
#4158	Indirect from volumes in SSS is extremely noisy
#3183	OIIO cache refuses to close texture files
#3355	Overridden opaque attribute on procedural not written out
#3605	extreme anisotropy in Cook-Torrance BRDF produces black
#3624	Calling AiUniverseCacheFlush(AI_CACHE_TEXTURE) outside AiBegin / AiEnd crashes
#3674	OIIO locking texture files between renders on windows
#3823	deepexr driver crashes in tiled mode
#3867	OIIO does not check validity of image SHA
#3893	light_gamma is not applied to skydome_light color
#4068	utility shader unconditionally overwrites opacity when opaque flag is off
#4119	faceting in Cook-Torrance and Ward-Duer speculars
#4123	NaNs in bump mapping when Ns and N differ very slightly
#4126	make backtrace handler more resilient to errors
#4130	zero-length or NaN tangents causing artifacts in Cook-Torrance and Ward-Duer MIS functions
#4131	Zero roughness crashes AiCookTorranceIntegrate
#4134	crash with non-existent procedural path and AI_RENDER_MODE_FREE
#4137	null tangent vectors potentially causing crashes in Cook-Torrance and Ward-Duer
#4142	Don't crash if OIIO runs out of file handles
#4144	Crash after cloning shader with linked array parameter
#4147	Support user data for mesh light texture
#4159	AiShaderGlobalsGet*() hang if called during displacement
#4160	Volume node should adjust rays to local space
#4161	Wrong AtPoint2 constructor in Python bindings
#4163	robustness fixes in OBJ procedural loader
#4164	excessive texture I/O in cook-torrance blurry reflections in standard shader
#4171	Sampling position offset should be object space in density shader
#4181	Interactive creation of an object in an empty scene does not work
#4183	Spherical quad sampling converges a different result than area sampling in quad_light
#4184	Enable tiled Deep EXR output
#4186	driver quantization API functions overflow with large float values
#4187	skydome_light crash in IPR and corruption with per-component linked color
#4190	Wrong shadow when using utility shader to modulate opacity of an object
#4194	crash when reporting error before the options node was initialized
#4198	Sky image texture returns NaN when looking at poles
#4205	Degenerate linear or curved motion transforms can cause crashes
#4211	Removing a node with unsolved references causes a crash
#4213	bump2d shifts with highly scaled UV coordinates
#4214	AiNodeGetArray/Matrix returns wrong matrices on instances
#4215	Deep EXR driver crash when reused in the same session
#4222	Shadow precision artifacts with lights with visible geometries
#4231	Microfacet refraction has sharp total internal reflection
#4240	SSS invalid or crashes on non-polymesh objects
#4244	Camera exposure affecting AOV alpha channels
#4251	Skydome interactive enable/disable not working
#4252	Don't write 'options' node default values to .ass file
#4157	Need to warn when passing empty filename as texture name
#4192	Cannot set metadata for "node" parameter on an .mtd file