4.2.2.0

Enhancements

- **Subdivision fractional soft creases and vertex creases**: OpenSubdiv-compatible fractional soft creases are now supported through a new polymesh.crease_sharpness attribute that holds the sharpness value for each crease listed in polymesh.crease_idxs. A crease sharpness value bigger than polymesh.subdiv_iterations will de facto encode an infinitely sharp crease. If crease_sharpness is not present, all creases will be assumed to be infinitely sharp. Additionally, it is possible to set vertex sharpness by duplicating the vertex in crease_idxs. For example:

  ```
  polymesh
  {
    crease_idxs 1 2 3 4 5 5
    crease_sharpness 1e6 2.5 4.0
    ...
  ```

  marks the edge between vertices 1 and 2 as infinitely sharp, the edge between vertices 3 and 4 as having sharpness 2.5, and sets the sharpness to 4.0 for vertex 5. (#3736)
- **Multi-threaded loading of pre-expanded procedurals**: It is now possible to load multiple procedural/standin nodes in parallel at node initialization time, before ray tracing begins, i.e. when either procedural.load_at_init or options.procedural_force_expand are enabled. For big scenes that rely heavily on procedurals, this can dramaticaly speed up scene loading. This is an experimental feature, and as such is disabled by default. Procedurals that are not thread-safe, or procedurals that depend on other nodes having been previously initialized, could result in crashes and undefined behaviour. This feature can be enabled with the new option parallel_procedural_init. We might remove or rename this option based on production feedback. (#2957, #4280)
- **Multi-threaded importance tables in textured quad_light**: The importance sampling tables needed by textured quad area lights can now be computed in parallel. This can result in important speedups in scenes with many lights that use high-resolution tables. (#4227, #4280)
- **Faster thread creation/destruction**: The overhead when creating and destroying rendering threads has been reduced. This is specially noticeable when using interactive rendering with 32 or more threads. (#4188)
- **Adjustable thread priority in OS X**: OS X now allows for options.thread_priority to adjust the priority of the render threads. It defaults to lowest, which is lower than the system default that was previously being used. Previously this option only existed in Windows. (#4292)
- **Reduced texture IO**: Texture IO is lower for texture lookups seen through glossy reflections and for displacement textures. This can result in noticeable speedups in texture-heavy scenes. (#4311)
- **Faster kick display**: The -interactive mode renders faster and navigation is smoother with less flickering. OpenGL graphics drivers are now required to open the kick render window, but batch rendering without a window does not require OpenGL to be installed. On Mac OS X, it is no longer required to install X11 to run kick. (#3063, #4210, #4304).
- **Faster SSS sets**: Sub-surface scattering is now faster for rendering overlapping hair and skin together if the skin objects were included in an SSS set. As an added bonus, this fixes long-standing darkening artifacts when using SSS sets near dense hair areas. (#2988, #3906, #4324)
- **More robust UDIM tile handling**: A couple of new attributes have been added to the image node. By default, missing UDIM tiles will still generate an error. However, certain workflows require missing tiles to be ignored and replaced with a default texture. This can now be done by setting image.ignore_missing_tiles true and linking or setting the image.missing_tile_color attribute to a constant color. Additionally, the code is now more robust to specific illegal UV mappings resulting in negative indices near polygon edges. (#4346, #4332)
- **Removed image.cache_texture_handles**: The image shader now always uses fast texture handles under the hood, and this can no longer be manually turned off. The only case where texture handles are not used is if the filename string is linked, but we don't recommend this as multi-threading performance would suffer. (#4351)
- **Improved matte/holdout support**: All objects now have a matte boolean parameter to turn them into a holdout. Shaders will not run on the object anymore and it will output all-black (including the alpha), except if the opaque parameter is off in which case shaders will be run just to compute the opacity. Note that even AOVs output by its shaders in that case will be black. (#4299)
- **Volume instancing**: The volume node didn't forbid instancing previously, but didn't work. We now formally support instancing of volume nodes via ginstance nodes as usual. (#4277)
- **Self-only mode for occlusion**: The ambient_occlusion shader makes use of the new AiSelfOcclusion() API function when requested to only gather occlusion against the same object. (#4285)
- **Spiral buckets by default**: The default options.bucket_scanning has changed from top to spiral, a more IPR-friendly mode. If rendering to scanline-based EXRs in batch mode on the render farm, the top mode might still be a better option. (#4274)
- **AiUniverseCacheFlush(AI_CACHE_TEXTURE) always works**: Textures can now be flushed during rendering. If a texture flush is requested during rendering, Arnold will record this request and when it finishes rendering, the texture flush request will be passed on to the texture system. Generally, a texture flush should be paired with an AiRenderInterrupt() so that the renderer can immediately stop and flush the textures. (#4341)
- **OIIO improvements**: OpenImageIO has been upgraded to 1.4.14. Texture lookup performance under many threads has been improved when performing many invalid texture lookups, for instance, when performing displacement mapping with an invalid displacement texture. Fixed a bug in OIIO which was causing random crashes in Windows during texture lookups. (#4202, #4282, #4296, #4312)
- **Updated to RLM 11.1BL2**: We have upgraded the license server and the external library controlling the licensing subsystem from version 10.1BL2 to 11.1BL2, a more stable release fixing various crashes, bugs, hangs and memory leaks. (#4088, #4258)

API additions

- **Self-only occlusion**: AiSelfOcclusion() was added to allow intersecting only against the same object which is being shaded. It takes the same parameters as AiOcclusion() and can be used as a drop-in replacement for those situations where this type of occlusion query is needed. (#4285)
- **Matte status**: AiShaderGlobalsIsObjectMatte() is now provided to quickly query the new matte parameter that objects have. This can be used in conjunction with rays of type AI_RAY_CAMERA to only compute opacity, and skip all other shader logic. (#4299)
- **Subsurface ray type**: AI_RAY_SUBSURFACE is a new internal ray type used for BSSRDF and single scattering probe rays. The ray counts section of the log includes these ray types as bssrdf and single_scatter. This internal ray type does not currently affect the object visibilityparameter. (#4324)

- **User parameter in shaders**: Shaders may now get a AtUserParamEntry* for a given user parameter via AiUDataGetParameter(const char*). This allows the shader to inspect various aspects of the user data, including the type, so that the appropriate AiUDataGet*() call may be issued. Retrieving the parameter will return NULL if the parameter doesn't exist as user data. (#4342)

Incompatible changes

- **SSS with zero diffusion radius**: previously such shaders would render black, now the result is the same as a diffuse BRDF. This affects the standard shader, as well as the AiBSSRDFCubic() and AiBSSRDFGaussian() functions. (#3272)
- **Sharper ray derivatives**: The directional derivatives (dDdx and dDdy) are now based off the distance between samples (clamped by texture_max_sharpen) and not the distance between pixels. The first implication of this is that texture_max_sharpen does not affect texture lookups for displacement and other places where the ray derivatives are not used in computing the texture derivatives. The second implication is that ray derivatives can no longer be directly used to compute pixel footprints. To do that, the derivatives would need to be widened by:

  ```
  if (options->AA_samples > 1)
     my_derivative *= MIN(options->texture_max_sharpen, static_cast<float>(options->AA_samples));
  ```

  However, beware that this could result in over widening for reflected rays. (#4311)

Bug fixes

Ticket	Summary
#4260	Overscan output invalid for non-tiled EXRs
#3272	correctly handle tiny/zero diffusion radius in SSS
#4202	soft invalidations do not work for relative paths
#4233	holes in the alpha channel when shading layers at auto_transparency_depth
#4243	Overriding per-face shaders in ginstance gets wrong indices
#4263	black dots/nans near center of spotlight with cosine_power > 0
#4265	8 and 16 bit gamma corrected textures generate some values above 1
#4267	rare random crash at start of rendering
#4269	Perlin noise returns NaNs for very large inputs
#4271	INT and BOOL AOV filtering not working in some cases
#4272	crash after deleting a shader node connected to "background" or "atmosphere"
#4275	Subdivision crash due to unsupported vertex valence (bigger than 255)
#4277	Volume crash when used with instancing
#4281	Curves UV discontinuity in linear mode
#4287	AiSamplerSeeded() not fully using seed so still has some correlation
#4288	AiBegin should check if it's already in a session
#4294	Remove utility.set_opacity
#4297	Potential race condition when creating internal shaders
#4308	Quad light artifacts with small or far away lights
#4309	Camera depth of field and other parameters not updating correctly in IPR
#4311	displacement texture derivatives should not be affected by AA samples
#4312	oiio crashes in windows
#4313	Volume shader override not working
#4314	Normals around hard subdiv creases are smoothed (non adaptive mode)
#4321	simple primitives don't update/rebuild properly
#4326	AiNodeResetParameter inheritance issues and missing python bindings
#4332	UDIMs and negative UVs
#4343	min_pixel_width fails for points and curves created in order before a camera
#4344	min_pixel_width silently fails for points nodes with a single point
#4349	varying string user data crashes arnold
#4325	remove extra newline in .ass file array declaration of strings
#4353	image multiply/offset should have 'linkable' metadata enabled