4.0.11.0

Milestone 4.0.11

Enhancements

- **Faster processing of big light groups**: Light groups and shadow groups with many lights no longer have as high an overhead to process. In a production shot with thousands of objects and light groups of 300 lights on each object, node initialization time went from 30 secs to 4 secs and total rendering time went down by 4%. (#3148, #3166)
- **Texture lookup cache**: Texture lookups are now cached for each shading point, so redundant texture lookups to the same texture map will not be as expensive. We have seen 20% speedups in certain texture-heavy production shots with complex shader networks. However, it is extremely important that shaderglobals are not reused across shading points otherwise the texture cache will continue to grow with unrelated texture lookups and performance can plummet. If the shading context is set to AI_CONTEXT_VOLUME, then the texture lookup cache is not used which allows the sharing of a shaderglobals struct for marching rays across a volume. (#3003)
- **Improved texture derivatives**: Texture derivatives are now properly propagated through reflections and refractions of the polymesh, sphere and point-sphere objects. This improves texture antialiasing and can dramatically reduce disk I/O, provided that the texture maps were already tiled and MIP-mapped. The propagation of texture derivatives for both pointcloud-based and raytraced SSS has also been improved, with the addition of a global option texture_sss_blur that lets users trade texture blurring vs I/O. (#2980, #3163, #3180)
- **Faster texture lookups on many-core computers**: We have improved the texture engine's thread scalability, which previously was a bottleneck on highly-threaded renders. On a 32-core machine with 64 logical cores, we saw one render that was not able to scale past 8 threads and actually got slower as more threads were added. By improving the scalability of texture access, especially in the case of string-based lookups, we can now scale to all 64 threads which in this instance gave a 24x speedup for string-based texture lookups when using 64 threads and a 3x speedup for handle-based texture lookups. (#3186, #3190)
- **Faster volume scattering with textured gobos**: Texture lookups during volumetric scattering, e.g. those coming from spotlights with texture mapped gobos, are now much faster, with little to no perceivable difference in shading quality. We have seen 10-20% speedups in simple scenes with a few textured spotlights, depending on how many volume samples are used. (#3181)
- **Auto-detection of max open texture files limit on Linux**: Most Linux distributions have file descriptor limits that are higher than the other platforms by default, but texture_max_open_files was defaulted so low as to not take advantage of this. For scenes with many textures, setting this option higher can result in significant performance gains. The default for Linux is now set higher (512, was 100). If you manually set texture_max_open_files higher than is safe, a warning about potential crashes will be issued. Also, when the option is set to zero Arnold will automatically increase file descriptor limits to the OS-reported hard limit for the process, and then allocate a reasonable chunk of those to textures, usually much larger than 512. Note that you may increase the process descriptor limits yourself on Linux using the ulimit -Sn shell command in bash andlimit descriptors in csh/tcsh prior to running kick or another process hosting Arnold, and they will inherit these values for that session. (#3172)
- **Disable memory reporting on Linux**: On older Linux distros, such as CentOS 5, accurately computing the amount of memory used can be very inefficient and in certain situations can amount to a substantial overhead. We have measured this overhead to be up to 10% in complex scenes with long log files. In these situations, it can be beneficial to disable the memory usage calculations by setting the new option, enable_memory_reporting, to false. When disabled, memory used will return 0MB and the true peak memory will not be computed. When Arnold terminates, the reported "total peak" memory will not really be the peak, but instead only the memory used at the end of rendering. On CentOS 6 and other recent Linux distros, this option has no effect and memory will continue to be accurately reported even if set to disabled. On Windows and OS X this option does not exist since memory reporting is always fast. (#3159)
- **Bump/autobump time reporting**: To aid with debugging, the time spent in bump mapping and autobump calculations, including the time spent inside the 3 shader calls to the displacement shader, is now reported in the render time stats as long as the new global option shader_timing_statsis enabled. There is a bit of overhead (around 1-2%) in the act of measuring itself, which is potentially performed many times per pixel, so we are disabling this option by default. (#3189)
- **New plastic shade mode in utility shader**: In addition to the lambert mode, there is now a plastic shading mode in the utility shader, which has both diffuse (0.7) and specular (0.1) components. The specular component is hardcoded to a Cook-Torrance BRDF with MIS support. This mode can be helpful when debugging and optimizing glossy materials, as it can be quickly assigned to all objects in the scene with the simple kick commands: -is -sm plastic. (#3167)
- **Optimized node iterator for lights**: When creating a node iterator over all the lights in the scene with AiUniverseGetNodeIterator (AI_NODE_LIGHT), we now return much faster as we no longer do a linear search over all the (potentially millions of) nodes in the scene. This can help in interactive relighting applications. (#3200)
- **Improved kick -tree**: In addition to an ASCII-art printout of the given shader network, we now include an annotated summary with useful debugging information like maximum depth, total shaders, cycles (if any), shader count by type, and one-to-many connections. (#3187)
- **Upgraded Windows compiler to icc13**: We have upgraded the compiler used to build Arnold on Windows from icc11 to icc13. Initial testing shows speedups in the 5-14% range. (#2986, #3150)
- **Upgraded OIIO to 0.10.15**: This bugfix release supports some rare TIFF extensions from Adobe products (#3122), prints out detailed tile loading statistics per MIP-map level (#3124) and fixes a bug with file wrap mode support (#3064).
- **Upgraded RLM to 9.4BL4**: We have upgraded the license server and the external library controlling the licensing subsystem from version 9.3BL2 to 9.4BL4, a more stable release fixing various crashes, bugs, hangs and memory leaks. (#3111)

API additions

- **Refraction ray differentials**: In order to benefit from the refraction ray differentials, a new function, AiRefractRay(), must be called. This will compute the new refracted (or total internal reflection) ray direction as well as compute the proper ray direction differentials. Similarly, AiReflectRay() has been added which computes the reflection direction and ray reflection differentials; however AiMakeRay() is already setting these reflection differentials, so it is not necessary to call this unless the normal has changed, as might happen when reusing an AtRay to cast multiple reflection rays from the same shading point but with different shading normals. Previous code that computed refraction rays would change from:

```
AtRay ray;
AtVector T;
if (AiRefract(&sg->Rd, &sg->Nf, &T, n1, n2))
{
   AiMakeRay(&ray, AI_RAY_REFRACTED, &sg->P, NULL, AI_BIG, sg);
}
else
{ // Total internal reflection
   AtVector R;
   AiReflectSafe(&sg->Rd, &sg->Nf, &sg->Ng, &R);
   AiMakeRay(&ray, AI_RAY_REFLECTED, &sg->P, &R, AI_BIG, sg);
   // But classify it as a refracted ray
   ray.type = AI_RAY_REFRACTED;
}
```

to:

```
AtRay ray;
AiMakeRay(&ray, AI_RAY_REFRACTED, &sg->P, NULL, AI_BIG, sg);
AiRefractRay(&ray, &sg->Nf, n1, n2, sg);
```

- **Seeded AtSampler**: A new function, AiSamplerSeeded(), has been added that provides the same functionality as AiSampler() but has an additional integer seed parameter. For shaders and shader networks that use multiple AtSampler instances, each of those would give back the same sample pattern, leading to correlations where the sample patterns were used for very similar effects. Using samplers with unique seeds can help reduce correlations in those cases. (#3131)

Incompatible changes

- **Removed options.unload_plugins and procedural.unload_dso**: The practical utility of these two legacy parameters was dubious and therefore they have now been removed. The default behaviour is preserved: dynamically-loaded libraries (whether for shaders or procedural DSOs) will always be unloaded in AiEnd(). (#3210)

Bug fixes

Ticket	Summary	Component	Owner	Priority	Version	Created
#3120	recent changes in abort/interrupt mechanics broke re-rendering	arnold	oscar	critical	4.0	8 weeks
#3199	Empty procedurals cause their library to be reloaded	arnold	oscar	critical	4.0	6 days
#2970	Wrong visibility when instancing procedurals	arnold	angel	major	3.3	5 months
#3018	seams and other artifacts when "blur" is used with textures	oiio	ramon	major	4.0	4 months
#3059	bump nodes don't bridge constant values when 'shader' is not linked	arnold	oscar	major	4.0	3 months
#3117	Support for light nodes in procedurals broken by procedural cache	arnold	angel	major	4.0	2 months
#3122	maketx crashes with TIFF tagged as 16-bit float	oiio	ramon	major	4.0	8 weeks
#3125	diffuse edge darkening in standard shader with Fresnel reflection	arnold	marcos	major	4.0	8 weeks
#3135	min_pixel_width opacity being set too low for points primitives	arnold	thiago	major	4.0	6 weeks
#3136	anisotropic sometimes pulls in overly high-res mip levels	oiio	thiago	major	4.0	6 weeks
#3142	upgrade "degenerate polygons" AI_LOG_DEBUG warnings to AI_LOG_WARNINGS	arnold	thiago	major	4.0	6 weeks
#3143	crash caused by degenerate triangles in displacement	arnold	thiago	major	4.0	6 weeks
#3144	disp_zero_value affecting faces with NULL displacement shaders	arnold	alan	major	4.0	6 weeks
#3146	receive_shadows and self_shadows cannot be overriden in procedurals	arnold	angel	major	4.0	5 weeks
#3148	Lightgroup/shadowgroup membership checks too expensive in scenes with many lights	arnold	thiago	major	4.0	5 weeks
#3154	AI_AOV_BLEND_NONE mode not working for deep AOVs	arnold	ramon	major	4.0	5 weeks
#3156	Wrong disp_padding warnings for motion-blurred meshes	arnold	ramon	major	4.0	4 weeks
#3163	fix texture derivatives for sss_irradiance_shader in pointcloud mode	arnold	mike	major	4.0	4 weeks
#3166	Speed up array simplification during node initialization	arnold	mike	major	4.0	4 weeks
#3174	common radius optimization ignored for points with motion	arnold	thiago	major	4.0	3 weeks
#3176	IPR crash in utility shader when changing shade_mode from ambocc	arnold	angel	major	4.0	3 weeks
#3197	crash with multi-threaded displacement	arnold	alan	major	4.0	8 days
#3207	make ndoteye mode shade black for secondary rays	arnold	marcos	major	4.0	3 days
#3208	perform more thorough error checking when writing ass files	arnold	angel	major	4.0	2 days
#3064	file texture Wrap Mode not working	oiio	ramon	minor	4.0	3 months
#3130	remove annoying kick warnings about null driver when not writing to disk	kick	marcos	minor	4.0	7 weeks
#3188	Arnold hangs when it can no longer write to a tiled EXR file	arnold	ramon	minor	4.0	2 weeks