4.0.12.0

Milestone 4.0.12

Enhancements

- **Improved numerical robustness**: We have made the renderer more numerically stable in several important cases. First, we have removed shadow artifacts in convex regions that occurred when the object being shaded was far away from the origin. Now it should be much more robust; however, extreme distances from the origin will still breakdown and cause problems due to the limited precision of floating point numbers. This fix also removes the need for setting ray_bias for shadow rays and in fact ray_bias is now unused by shadow rays. Second, we have improved the robustness of matrix transformations for static objects far away from the origin. Third, the bump3d shader now uses improved differencing that allow it to work correctly when the shading point is even many thousands of units away from the origin without adjusting bump3d.epsilon, whereas previously the bump effect would start to disappear at even a couple hundreds units away. And fourth, we have improved the encoding of vertex normals at no cost in memory nor speed so that the encoding error is reduced by up to 2x and the vectors round-trip correctly when exported and read again. (#3253, #3281, #3288, #3274)
- **SSS blurring across objects**: It is now possible to tag multiple objects as belonging to the same SSS "set" so that illumination will blur across object boundaries. A common use case might be blurring between teeth and gum geometry. This feature is only available when using raytraced SSS (sss_bssrdf_samples>0). It is enabled by adding the constant STRING userdata sss_setname to the same value on the objects in the set. (#2413)
- **Improved raytraced SSS in standard shader**: The raytraced SSS technique used by the standard shader has been improved to more robustly handle situations where there is a great difference between the R, G and B color components of the sss_radius and /or Ksss_color parameters. This can sometimes result in large reductions of noise. Future versions will include a new API so that shader writers can take advantage of this optimization in their own shaders. In addition, a number of small optimizations have been added to the raytraced SSS code which can result in a 5-10% speedup. (#3265)
- **Improved sampling of raytraced gaussian SSS**: Raytraced gaussian SSS diffusion now uses a better importance sampling technique that can significantly reduce noise when compared to the old technique for a given number of samples. (#3266)
- **Per-light scaling of diffuse, specular, SSS**: Three new float parameters have been added to all lights to better control contributions in specific light transport situations: diffuse, specular and sss which can scale the light contribution to each of those independently. These options supersede the now-deprecated boolean light parameters affect_diffuse and affect_specular, since the new options can be set to zero to disable their effect. Finally, parameters bounces and bounce_factor have been renamed to max_bounces and indirect to better reflect their purpose, although the old names still work as synonyms. (#3231)
- **Fresnel from IOR**: A new boolean parameter Fresnel_use_IOR has been added to the standard shader, which will calculate Fresnel reflectance based on the IOR parameter, ignoring the values set in Krn and Ksn. (#3164)
- **Secondary specular in hair shader**: The hair shader has gained a secondary glossy specular lobe, together with the new controls spec2, spec2_color and gloss2. In addition, both specular lobes can now be shifted relative to each other with the new spec_shift and spec2_shiftcontrols; usually the first specular lobe is shifted by -5 to -10 degrees, and the secondary lobe is typically spec2_shift = spec_shift * -1.5 even though they are decoupled for greater artistic control. (#3243)
- **Cheaper Oren-Nayar BRDF**: The math in the Oren-Nayar BRDF has been greatly simplified without changing the look, which may result in a slight speedup. (#3242)
- **Reduced texture I/O for glossy reflections**: Glossy reflections using the Cook-Torrance BRDF will now 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 data being brought in from disk and lessens the need for using the texture_glossy_blur option. In fact, we now recommend setting texture_glossy_blur to zero and will likely remove or replace this option in the future. (#3195)
- **texture_max_sharpen**: To address some users' concerns that texture lookpups are noticeably blurry, we have added the global option texture_max_sharpen. It is set to the default of 1.0, which produces the same behavior as before. As it is raised, the textures will appear sharper, but at the expense of increased texture file I/O. The *theoretical* optimum setting for sharpest results is to set this to AA_samples, but under most practical situations where texture I/O must be controlled, setting this to around 1.5 already gives sharp results at a moderate cost. See https://trac.solidangle.com/arnoldpedia/wiki/texture_max_sharpen for more details. (#3184)
- **texturetime AOV**: A new built-in AOV called "texturetime" has been added which can be used for profiling texture-heavy scenes. This is similar to the "cputime" AOV but instead will produce a float value with the total time (in microseconds) spent on texture access in each pixel. Note that for this AOV to work, the global option shader_timing_stats must be enabled first. (#3249)
- **Additional shader timing**: More timings have been added when options.shader_timing_stats is enabled which will be printed in the stats section at the end of the log file. It now includes time spent in texture lookups, volume shading, and irradiance caching for pointcloud-based SSS. (#3223)
- **Smaller, faster .ass files**: The storage of .ass files on disk has been optimized by extending base-85 binary encoding to BYTE/INT/UINT arrays, in addition to the already encoded float-based arrays. This can significantly reduce the size of exported .ass files, as well as improve their read and write times. Depending on scene content, you can expect up to 40% smaller files, up to 2x faster reads, and up to 4x faster writes. (#2863)

API additions

- **Volume shading API**: A new class of volume shaders is available. These shaders differ from earlier atmosphere shaders in that they are limited to some region of space defined by a container shape, and that heterogenous scattering/absorption effects are now directly integrable by the core instead of volumes having to be ray marched and lit by the shader. These new shaders are point sampled by the core, provided a small subset of the shader globals (P,Po,Ro,Rd,Rl,dPdx,dPdy,M,Minv), and are expected to return their volumetric coefficients via the newly addedAiShaderGlobalsSetVolumeAbsorption(), AiShaderGlobalsSetVolumeEmission() and AiShaderGlobalsSetVolumeScattering() API functions. To use this new class of shaders, one must apply them to the shader parameter of a points, sphere or box shape and change the shape's step_size parameter to a non-zero value. More information is available on the arnoldpedia. *Note*: For optimal performance, it is recommended that the step size be no smaller than the smallest feature of the volume, which in the case of voxelized data is the size of a voxel in world space. (#3010,#3011,#3012,#3129,#3155,#3168,#3113)
- **Refraction API**: The previous release, 4.0.11 introduce AiRefractRay(). However, that required all refracted rays to use AiRefractRay(), otherwise the ray derivatives would remain uninitialized and incorrect results, including NaNs and crashes could occur. We have fixed this so that ifAiRefractRay() is not called, the same inefficient but at least safe behavior will occur. If AiRefractRay() is called after AiMakeRay(), then

the correct ray derivatives will be computed and performance and texture image quality will be improved. Examples of code usage and results can be seen in https://trac.solidangle.com/arnoldpedia/wiki/Refraction. (#3236)

- **Generic BRDF integrator**: Through the newly added AiBRDFIntegrate() function, it is now possible to easily apply the importance sampling Monte Carlo integration technique to compute the indirect lighting for BRDFs defined through the existing MIS API given the eval_sample, eval_brdf and eval_pdf callbacks. (#3284)
- **AtRGBA operators**: Added arithmetic operators +, -, *, / for AtRGBA similar to those already implemented for AtRGB. (#3233)

Incompatible changes

None.

Bug fixes

Ticket	Summary	Component	Owner	Priority	Version	Created
#2834	AiMakeRay() does not properly set up shadow rays	arnold	mike	major	3.3	10 months
#3109	Error when append and flush_buckets_on_halt are both true	arnold	ramon	major	4.0	4 months
#3133	Wrong normals when 'ignore_bump' used with autobump	arnold	oscar	major	4.0	4 months
#3147	Filtering long and narrow degenerate ellipses	oiio	ramon	major	4.0	3 months
#3184	textures using a perspective camera are too blurry	arnold	thiago	major	4.0	3 months
#3227	Append is not compatible with crop rendering	arnold	ramon	major	4.0	2 months
#3230	De-clutter logs: min_pixel_width optimization, hair diffuse cache	arnold	marcos	major	4.0	7 weeks
#3234	wrong mipmap level is used for sphere-points with deformation on radius	arnold	thiago	major	4.0	6 weeks
#3236	NaNs in derivatives with refracted rays	arnold	thiago	major	4.0	6 weeks
#3239	NaN's when light's 'indirect' = 0 and 'radius' > 0	arnold	oscar	major	4.0	5 weeks
#3250	Improve precision of time measurements in logged statistics	arnold	angel	major	4.0	4 weeks
#3253	Fix self-shadowing artifacts when far away from the origin	arnold	thiago	major	4.0	4 weeks
#3260	An array with a single element array is not properly written to .ass	arnold	angel	major	4.0	3 weeks
#3261	Arnold crashes when rendering after a texture cache flush	oiio	ramon	major	4.0	3 weeks
#3262	MIS not working with mesh_light if mesh not visible for glossy rays	arnold	ramon	major	4.0	3 weeks
#3263	Incorrect shadow_group override behavior for ginstances	arnold	mike	major	4.0	2 weeks
#3267	Accel structure can create enormous amounts of unaccounted memory	arnold	thiago	major	4.0	2 weeks
#3270	Cannot override "opaque" on procedurals	arnold	angel	major	4.0	2 weeks
#3274	packed normals are slightly wrong	arnold	thiago	major	4.0	9 days
#3281	Improve numerical precision of inv_matrix for static objects	arnold	thiago	major	4.0	7 days
#3288	Make bump3d more robust to scale	arnold	thiago	major	4.0	3 days
#3290	Mismatching matrix motion keys between source and ginstance should be ignored when inherit_xform is off	arnold	mike	major	4.0	3 days
#3295	degenerate motion key causes major slowdown	arnold	thiago	major	4.0	20 hours
#3297	optimize 3 unnecessary divisions in transmittance of standard shader	arnold	marcos	major	4.0	2 hours
#3211	Do not process abort flag when loading procedurals and ass files	arnold	oscar	minor	4.0	2 months
#3240	Mirror uv wrap mode not working for gobo	arnold	ramon	minor	4.0	5 weeks
#3212	Windows 8 is not correctly reported in the OS details of the log file	arnold	oscar	trivial	4.0	2 months