4.2.9.0

Milestone 4.2.9

Enhancements

- **Refractive dispersion**: The standard shader now supports chromatic dispersion through refraction. The new parameter dispersion_abbe specifies the Abbe number of the material, which describes how much the index of refraction varies across wavelengths. For glass and diamonds this is typically in the range 10 to 70, with lower numbers giving more dispersion. The default value is 0, which turns off dispersion. The chromatic noise can be reduced by either increasing the global AA samples, or the refraction samples. (#4835)
- **Volumetric scattering AOVs**: Direct and indirect scattering in volumes can now be rendered separately using the new volume_direct and volume_indirect AOVs. (#4806)
- **Perspective OpenVDB volumes**: The accompanying OpenVDB volume DSO provided with our DCC plugins has now been optimized to render perspective grids, also known as frustum buffers. The speedup depends on the specific perspective transformation, but we have seen 4x-6x faster rendering in tests. (#4827)
- **Faster volumes**: Volume lookups with multiple channels render 5% faster in various tests. (#4516)
- **Faster curves**: Curves in thick mode now render 2-20% faster. Curves with non-zero min_pixel_width now render up to 2x faster. (#4823, #4824)
- **Faster deep EXR**: The driver_deepexr node has been optimized, resulting in up to 25% faster rendering, most notably when using volumes. (#4811)
- **Faster subdivision**: Several optimizations were added to the Catmull-Clark subdivision engine. Adaptive subdivision mode is now 4% faster, and smooth derivatives (as needed by anisotropic surface shading) are 27% faster while using 8% less memory. (#4832, #4837)
- **Faster opacity**: The arnold core shaders (lambert, standard, etc...) now make use of AiShaderGlobalsApplyOpacity() which can result in significant speedups when shading with partial opacity, or when using opacity-mapped tree leaves. This optimization is controlled byoptions. enable_fast_opacity. We have also improved AiShaderGlobalsApplyOpacity() so that it is faster and higher quality than before. User shaders are advised to make use of this function in order to benefit from the faster opacity using code such as the example below. (#4434, #4775, #4800, #4814)

```
   AtColor opacity = AiShaderEvalParamRGB(p_opacity);
   if (AiShaderGlobalsApplyOpacity(sg, opacity))
      return;
   opacity = sg->out_opacity;  // the new opacity is contained in sg->out_opacity
```

- **Driver time stats**: Output driver time is now reported in the render time section of the log file stats. This can be used to detect bottlenecks related to deep image output, excessive number of AOVs or networking issues. (#4840)

API additions

- **AtString volume API**: AiVolumeSample functions in shaders and the volume plugin Sample method now accept AtString channel name parameters, for better performance. (#4516)
- **Separate direct and indirect SSS**: Variants of the SSS/BSSRDF shading and lighting API that return both the direct and indirect results separately now exist. These new API functions, denoted by the Separate suffix, accept two variables by reference in which the direct and indirect components are stored separately, instead of returning their sum as the function's result. The following functions were added:

```
AI_API void AiBSSRDFCubicSeparate(const AtShaderGlobals* sg, AtRGB& direct, AtRGB& indirect, const float
* radius, const AtColor* weight, unsigned int num = 1);
AI_API void AiBSSRDFGaussianSeparate(const AtShaderGlobals* sg, AtRGB& direct, AtRGB& indirect, const f
loat* variance, const AtColor* weight, unsigned int num = 1);
```

  The built-in standard shader and the external skin shader have also been extended to make use of this new API by means of separate direct and indirect SSS AOVs. (#3217)
- **AiTextureLoad() EXPERIMENTAL API**: We have introduced a new API that allows for easily reading an entire texture file into a pre-allocated image buffer. The main use for this is for Maya/MtoA to support .tx texture maps. This can be called outside of AiBegin/AiEnd() blocks so that it can be used at any time. The corresponding AiTextureGetResolution(), AiTextureGetNumChannels(), AiTextureGetChannelName(), AiTextureGetFormat(), AiTextureInvalidate(), and AiTextureGetMatrices() can now also be called outside of AiBegin/AiEnd() blocks. Note that this is currently an EXPERIMENTAL API addition, so we might modify this in the near future. As such, code that uses this might stop working and need to be updated with newer versions of Arnold. (#4831)

```
AI_API bool AiTextureLoad(const AtString filename, const bool use_float, void* image);
```

Incompatible changes

- **Depth of field**: Perspective cameras have a new persp_camera.flat_field_focus attribute which is set to true by default to match the standard thin lens camera model. This prevents overblurring away from the optical axis. Some renders might change, specially with wide FOV angles or very shallow DOF. Set persp_camera.flat_field_focus false to get the previous behavior. (#4810)
- **Opacity in utility shader**: utility shaders with opacity will now render less transparent than before, previously it was incorrectly rendering too transparent. The new opacity behaviour better matches other shaders like standard and lambert. (#4836)
- **Bump and object scale**: The bump2d, bump3d shaders and the AiShaderGlobalsEvaluateBump() API function now take into account object scale by default. Previously the bump displacement height would not scale along with the object, which was inconsistent with displacement and autobump. The new global options.bump_space is set to object by default, but can be set to world for backwards compatibility. This new global option will likely be removed in the next API-breaking major release. (#4784)

Bug fixes

Ticket	Summary
#2807	subdiv_smooth_derivs doesn't work in polymeshes with adaptive subdivision
#4523	Free mode crash using autobump and multiple light loops
#4546	Random procedural transform resets with options.parallel_node_init
#4627	load_at_init procedural inside a delayed load procedural hangs Arnold
#4784	Bump2d is not affected by scale
#4797	Destroy the contents when destroying a procedural node
#4799	Wrong procedural contents reported in the log file when writing to .ass
#4801	Potential crash when initializing multiple instances of the same procedural
#4805	Autobump crash with NaNs in displacement texture
#4807	Shadow matte broken when there are disabled lights
#4808	AiAOVEnabled crash in free mode
#4809	Light acceleration was missing in free mode
#4811	Inefficient string comparison in deep driver
#4815	Poor sampling quality when tracing camera rays in free mode
#4818	Matte property not working in built-in AOVs (P, N, Z, ...)
#4819	crash due to nodes being initialized multiple times on different threads
#4820	Adaptive subdivision crash if vertex valence exceeds maximum
#4821	Mesh processing corrupts smooth derivatives
#4826	Subdivision crash with user data and free-floating vertices
#4836	Incorrect opacity in utility shader
#4837	Subdivision: exact derivatives not needed for user data and user normals