4.0.16.0

Milestone 4.0.16

Enhancements

- **Optimized geometry storage**: Certain types of meshes now require less memory thanks to improved mesh compression. In some cases, we've seen dramatic reductions in the storage for normals and UVs. In exchange, this will increase the preprocessing time for mesh loading by around 3 seconds per 50 million faces. In addition, the constant per-object memory overhead for geometry nodes has been slightly reduced which can help in complex scenes with millions of instances, polymeshes or procedurals. (#3428, #3460, #1833)
- **Faster area lights**: Area lights more consistently and correctly work with options.low_light_threshold. This can result in significantly faster renders when there are many lights. Compared to previous versions, there is a chance that the resulting image might be brighter or dimmer when using a very aggressive low_light_threshold. If it is brighter, then the image is actually closer to the true reference image (when low_light_threshold is 0), but if it is darker then low_light_threshold was too aggressive and will need to be lowered. For renders using less aggressive settings, such as the default .001 threshold, the image should be identical but render times should be improved. (#3471, #3477)
- **Initial support for IES lights**: We have added a new type of light, the photometric_light node. This light can initially load IESNA LM-63 photometric data through the parameter filename. (#2906, #3524)
- **Per-strand shader assignments on curves shape**: Similar to how polymesh shapes with multiple shaders are defined, the curves shape now has a shidxs parameter with which shaders from the shader array may be assigned to each separate strand. (#3456)
- **Improvements to the physical_sky shader**: Added a sun_size parameter to the physical_sky shader that allows specifying the angular diameter of the sun in degrees. Also added X,Y and Z parameters, similar to those of the sky shader, with which it is possible to define the orientation of the physical_sky shader, which was previously hardcoded to "Y is up". (#3452)
- **User-data access for volume shaders**: It is now possible to read user data fields from volumetric shapes, allowing things like per-particle user data on volumetric spherical point clouds to affect the result of volumetric shading. *Note: As is the case of surface shaders, accessing user data from a volume shader is still much slower than access to the shader's own parameters, so discretion is advised.* (#3327)
- **Per-face-vertex user data**: A new storage class, *indexed* can be used for declared user data, where it may be different across shared edges of polygons. This allows for such things as alternate UV sets, pre-baked tangent vectors, and so forth. It works the same way as the built-in vlist/vidxs, uvlist/uvidxs, nlist/nidxs arrays. You only need to declare the indexed data, e.g. *declare altuvlist indexed POINT2*, and then both the list and indexes array can then just be used: *altuvlist 4 1 POINT2 ... altuvidxs 16 1 UINT ...*. Note that it follows a strict naming convention; the list array can be named what you want, or end with *list*, and the *list* suffix will be removed and *idxs* appended for the actual index array. Examples of pairs would be *altuvlist* with *altuvidxs*, and *altuv* and *altuvidxs*. (#2866)
- **Multiple UV sets in the image node**: With the new indexed user-data storage class, alternate UVs can be created as POINT2 data on polymesh nodes. These alternate UVs can be selected in the image node by setting their name in the new uvset parameter. For example, if you have created a secondary UV set in a polymesh node and named it "UVset2", then you can use this UV set by setting image. uvset equal to "UVset2". By default, when the uvset parameter is empty, the primary UV set in the polymesh will be used, i.e. the one stored in the built-in polymesh parametersuvlist and uvidxs, so that backwards compatibility is preserved. (#3500)
- **Warn of millions of warnings**: Warning messages, even if hidden by explicitly setting a low maximum number of warnings, still take time to process. If there are millions of warning messages, typically from a custom shader's shader_evaluate method that is being too verbose, then this can introduce a sizable render time overhead. Even worse, this overhead does not scale well with increasing numbers of threads. When there are more than a million messages, we now print a performance warning with the amount of predicated time overhead the messages are causing. We have also improved the scaling, but not the overhead, when the majority of the messages are of the same type (the most common case we have seen in practice). Still, the most efficient solution is to never print warnings from a shader's shader_evaluate method. (#3499, #3501)
- **Upgraded OIIO to 0.10.19**: There have been a few bugfixes and performance improvements since the last OIIO version we shipped with Arnold (which was 0.10.16) including:
    - Minor performance improvements
    - Detailed OIIO stats sometimes causing crashes (#3473)
    - maketx: Fix an overflow problem when running on images larger than 32k x 32k (#3506)
- **kick command line in logs and display window**: Log files generated from kick now contain the entire command-line that was used which can help with debugging. In addition, the command-line is also printed in the title bar of the render display window that is created by the kick process. (#3457)

API additions

- **Added user-data derivatives with respect to screen x and y**: Derivatives of user data with respect to screen coordinates can now be queried. This should allow more accurate filtering based on user data, such as when using alternate UV sets for texture mapping. This is analogous to the texture derivatives of the primary set of UV coordinates that are stored automatically in sg->dudx,dudy,dvdx,dvdy. These derivatives will be zero except for varying (per-vertex) and indexed (per-face-vertex) user data. They also will be zero during displacement, which will effectively reduce any use of them to point sampling, which may negatively affect performance; we plan on addressing this in a future release. The following functions have been added to the shading API to obtain these derivatives, which will return *true* if the derivatives were available: (#3244)

```
bool AiUDataGetDxyDerivativesFlt(const char* name, float* dx_val, float* dy_val);
bool AiUDataGetDxyDerivativesRGB(const char* name, AtRGB* dx_val, AtRGB* dy_val);
bool AiUDataGetDxyDerivativesRGBA(const char* name, AtRGBA* dx_val, AtRGBA* dy_val);
bool AiUDataGetDxyDerivativesVec(const char* name, AtVector* dx_val, AtVector* dy_val);
bool AiUDataGetDxyDerivativesPnt(const char* name, AtPoint* dx_val, AtPoint* dy_val);
bool AiUDataGetDxyDerivativesPnt2(const char* name, AtPoint2* dx_val, AtPoint2* dy_val);
bool AiUDataGetDxyDerivativesArray(const char* name, AtArray** dx_val, AtArray** dy_val);
bool AiUDataGetDxyDerivativesMatrix(const char* name, AtMatrix* dx_val, AtMatrix* dy_val);
```

Incompatible changes

- **std::vector inclusion removed from ai_license.h**: This inclusion is no longer needed by ai_license.h itself and has been removed. This does not affect binary compatibility, but may cause client code that explicitly included ai_license.h to require explicitly including std::vector if recompiled. (#3488)
- **ai_version.h inclusion removed from ai_node_entry.h and ai_nodes.h**: That inclusion is not really needed in those files. This does not affect binary compatibility, but may cause client code that were not explicitly including ai_version.h and using its API to require explicitly including it now. (#2082)
- **Renamed physical_sky parameters**: The parameter solar_direction has been renamed to sun_direction, and visible_solar_disc has been renamed to enable_sun. (#3452)

Bug fixes

Ticket	Summary	Component	Owner	Priority	Version	Created
#3429	mesh intersection can create bad barycentric coordinates	arnold	ramon	major	4.0	2 months
#3433	Cloning a node with a linked parameter causes a crash when destroying nodes	arnold	angel	major	4.0	2 months
#3448	Cannot load Arnold Python module with unset LD_LIBRARY_PATH	arnold	oscar	major	4.0	2 months
#3458	shadow ring artifact with non-black shadow_color	arnold	thiago	major	4.0	8 weeks
#3472	lights should take into account _normalize when computing falloff radius	arnold	thiago	major	4.0	6 weeks
#3473	Printing detailed OIIO stats sometimes crashes	oiio	ramon	major	4.0	6 weeks
#3475	Add pixel aspect ratio to EXR images	arnold	ramon	major	4.0	6 weeks
#3482	MIS samples identical for transparent surfaces along a ray	arnold	ramon	major	4.0	6 weeks
#3483	distortion in skydome light's sample mapping causing severe flickering	arnold	alan	major	4.0	6 weeks
#3486	infinite lights should not be affected by low_light_threshold	arnold	thiago	major	4.0	5 weeks
#3491	Polymesh Differentials generate NaN / Inf	arnold	ramon	major	4.0	5 weeks
#3504	atmosphere lit by disk light not rendering against background until geometry is hit	arnold	thiago	major	4.0	4 weeks
#3506	maketx crashes with big images	oiio	ramon	major	4.0	3 weeks
#3509	time-based displacement does not work with duplicate deformation keys	arnold	ramon	major	4.0	2 weeks
#3513	Division by zero in the density-based volume sampler	arnold	alan	major	4.0	13 days
#3515	User data is not preserved by the procedural cache	arnold	angel	major	4.0	12 days
#3518	volume scattering dark or missing when scaled to a large size	arnold	alan	major	4.0	11 days
#3519	Memory leaks when updating lights	arnold	oscar	major	4.0	8 days
#3522	undefine MIN/MAX macros in public API to avoid symbol clashes	arnold	thiago	major	4.0	7 days
#3526	mesh light crashes while in free render mode	arnold	thiago	major	4.0	24 hours
#3527	Fix shader message accumulation in free mode	arnold	ramon	major	4.0	6 hours
#3508	remove harmless warning when using base85 encoding of packed INT arrays	arnold	angel	minor	4.0	3 weeks