4.2.7.0

Milestone 4.2.7.0

Enhancements

- **Faster cutout opacity mapping**: options.enable_fast_opacity can be enabled to get faster textured mapped opacity mask renders, such as for tree leaves. Tests with production scenes have shown up to 25x speedups. This flag also toggles a fix for more accurate renders, where previously the object would be rendered more transparent when further away from the camera. When using this option, noise/flicker can occur on far away geometry, similar to when rendering far away high res geometry. If this happens, it is best to use a high amount of AA samples (and correspondingly lower amount of diffuse/glossy/light samples) to reduce flickering in animation. The fast opacity flag is enabled for anything that is connected to any RGB parameter called "opacity", so custom shaders that have an RGB opacity parameter will benefit from this. We expect that users will want to enable this mode all the time for new scenes and if no one complains, in a future release we will likely remove this flag and leave the optimizations and fixes in the "fast_opacity" mode always enabled. (#4696)
- **Multiple scattering for volumes**: Indirect light in volumes now supports an arbitrary number of bounces instead of being fixed to one bounce. It is now possible to render volumes such as clouds for which multiple scattering has a large influence on their appearance. The new options.GI_volume_depth parameter sets the number of bounces, defaulting to 0. The default value of options.volume_indirect_samples has been changed to 2. (#4594, #4682)
- **Support for deep volume output**: Volumes are now visible in deep renders, but note that older "atmosphere" shaders and volumetric mattes are not supported yet. Previously, volumetric samples were composited with the next surface sample. Now, in case you want to implement your own deep driver, volumetric samples are available as independent samples to raw drivers or filters. In order to query the end of a volumetric sample you can request the new built-in float Zback AOV channel. (#4654, #4655)
- **Per light volume contribution**: A volume contribution scaling parameter was added to lights, similar to the existing diffuse and specular parameters. (#4657)
- **New EXR compression modes**: Four additional compression modes, b44, b44a, dwaa and dwab have been added to driver_exr.compression. b44 is lossy for half data and stores 32-bit data uncompressed. b44a is an extension to b44 where areas of flat color are further compressed. dwaa and dwab correspond to JPEG like compression from DreamWorks Animation. Note that dwaa and dwab require the reading program to be compatible with OpenEXR 2.2 which is not yet widespread, Nuke 9.x will not read them. (#4634, #4638)
- **Faster UDIMs**: UDIMs accessed through the image shader node now internally use texture handles, which helps improve multi-threading performance. (#3058)
- **OIIO improvements**: Upgraded to OIIO to 1.5.15. 16-bit textures are now stored as 16-bit in the texture cache instead of as 32-bit floats. This halves the amount of memory 16-bit textures use. (#4619, #4671)
- **Spaces allowed in EXR metadata names**: driver_exr.custom_attributes now supports spaces in names, as in the example below. (#4631)

```
driver_exr
{
 name mydriver
 filename foo.exr
 custom_attributes "POINT2 'Chromaticities/Red Primary' 1.0 2.0"
}
```

API additions

- **Per texture cache invalidation**: Sometimes reloading a specific texture is needed, rather than flushing the whole texture cache. AI_API void AiTextureInvalidate(const char *filename) is a new API call that lets developers invalidate single textures. (#4698)

Incompatible changes

- **Raised minimum OS X platform to 10.8**: OS X 10.7 is no longer supported. You now need at least 10.8 (Mountain Lion) to run Arnold. (#4718)
- **Changed texture_sss_blur default to 0**: SSS bump textures using sss_use_autobump and any textures used in an sss_irradiance_shader will now render with more detail. This may lead to increased texture I/O, though we have found little to no impact in various production scenes. (#4664)
- **polymesh.nlist**: Values in this array must be normalized vectors now. (#4651)
- **Removed per-light volume_density**: This was an old hack that only applied to the volume_scattering atmosphere shader, and was not supported in the more recent object-based volume API. Note that you can still link volume_scattering.density to achieve the same effect, only globally instead of per-light. (#4676)
- **volume_indirect_samples**: In existing scenes with one bounce of volume indirect lighting, options.GI_volume_depth must be set to 1 to re-enable it.

Bug fixes

Ticket	Summary
#4506	Opacities for volume mattes not as expected
#4596	Crash interrupting render with mesh light
#4606	Crash when calling AiRendering() during AiEnd()
#4624	UDIMs should work when triangles span multiple tiles
#4636	interactive update of light and shadow linking fails randomly
#4644	"too many messages" warning is output for masked out messages
#4645	planar light_blocker does not properly use height_edge
#4652	Procedurals not working correct with _self traceset, self_shadows and receive_shadows
#4656	Overlapping curves objects in procedural sometimes missing in render
#4658	Faceting artifacts with SSS and bump mapping
#4661	Restore shaders (shift + i) in interactive kick not working
#4662	Crash in points primitive with small radii
#4664	excessive texture blur when using sss_use_autobump
#4665	Statistics are not reset for each render session
#4673	Remove non-working light blocker upper case X, Y, Z ramp axes
#4674	INT channels do not work with deep EXR driver
#4682	Volume indirect converges to wrong result due to correlated samples
#4695	clamp away invalid opacity in core shaders
#4697	Improve warnings for varying and indexed data on objects that don't support them
#4699	UDIM textures do not work with UV sets or linked uvs
#4630	silence compiler warning in Visual Studio for AiNodeGetStrAtString
#4659	Fix pedantic warnings in public API