3.3.5.0

Milestone 3.3.5

Enhancements

- **Support for linear subdivision**: Added a new linear enumerated value in the subdiv_type parameter of the polymesh node, which allows for polygonal subdivision without undergoing Catmull-Clark smoothing. This can be useful when displacing flat surfaces. (trac#2040)
- **New modes of UV subdivision**: Added linear and smooth enum values in the subdiv_uv_smoothing parameter of the polymesh node. The linear mode finally makes Arnold compatible with textures created with Z-Brush's linear UV subdivision format. (trac#2118)
- **Added shader_searchpath option**: This is similar to the existing {procedural|texture}_searchpath parameters of the options node. .ass files that reference custom shaders are now self-contained and portable, without the need to manually specify the search path with other methods such as kick -l <path>. (trac#2142)
- **Polymesh hit refinement**: Added a mode where Arnold will attempt to refine the intersection point between a ray and a polymesh. This might help rendering artifacts in scenes with very large scene extents. This is enabled by setting enable_hit_refinement on the options node. (trac#2165)
- **Opacity support in utility shader**: The default shader is now able to render transparent objects. This may be useful when benchmarking scenes with many partially opaque surfaces. This feature is controlled by a new float opacity parameter. This setting has no effect until the object is tagged as opaque=false. (trac#2132)
- **Removed excessive warnings from uninstalled nodes**: When loading .ass files exported with node types that have not been installed /cannot be found, a "node is not installed" warning will now be printed out only once, potentially eliminating a large amount of redundant warnings. (trac#2134)
- **Optional warnings for shaders returning NaNs**: The render option shader_nan_checks has been added to selectively check for NaN results from each shader evaluation which can be used to help debug shading trees. This option has a slight impact on performance and should be activated only when necessary. (trac#1855)
- **Added option -qres to kick**: This is a handy command to quickly reduce the rendering resolution (i.e. quarter the number of pixels) when for example debugging scenes. (trac#2174)
- **Improved licensing messages**: The logging of licensing messages (of info, warning, and error types) has been changed to improve brevity and clarity. In addition, temporary licenses now report the expiration date along with the remaining days for which the license is valid. (trac#2127, trac#2154)
- **License heartbeat/timeout**: Arnold now emits periodic "heartbeats" to the license server, which allows for licenses that were checked out by clients which did not properly release them to be freed by the server after a given timeout. *Users who would like to use this new* TIMEOUT *fe ature are encouraged to visit the License server timeout wiki for configuration guidelines.* (trac#2140, trac#2143)

API additions

- **Move metadata specification to a separate file**: We have implemented support for external metadata files, which are text files with a simple format. For more information about the file format and how it works, see the Arnold Metadata files wiki. (trac#1937)
- **Inlined C++ function versions of AtColor, AtRGBA and AtVector macros**: Many utility macros for colors and vectors have been changed to inlined C++ functions. This has several advantages over C macros (like referencing, type safety and return values) allowing client code, especially shaders, to become shorter and more readable. For example, we can turn this code:

```
AtVector T;
AiV3Normalize(T, sg->dPdv);
AtVector c;
AiV3Cross(c, a, b);
AtColor color;
AiColorLerp(color, sg->v, root_color, tip_color);
AtColor t;
AiColorClamp(t, transmittance, 0, 1);
```

Into this:

```
AtVector T = AiV3Normalize(sg->dPdv);
AtVector c = AiV3Cross(a, b);
AtColor color = AiColorLerp(sg->v, root_color, tip_color);
AtColor t = AiColorClamp(transmittance, 0, 1);
```

Note that we haven't *removed* anything from the API, we have only added new symbols. Existing code using the older C-style macros should compile without changes. However, the older macros are now deprecated and will eventually be removed in a major release. The arithmetic macrosAiV3Add(), AiColorScale(), etc have also been deprecated in favor of the overloaded operators +, *, etc which are much easier to use and result in much more readable code. (trac#2160)
- **Array operators for AtPoint and AtRGB**: Both colors and points/vectors now provide indexed access to their components using the [] operator. This can help simplify some algorithms where component access is determined on the fly. (trac#2187)

```
AtColor c;
c[0] = 1.f;      // equivalent to: c.r = 1.f;
float b = c[2];  // equivalent to: b = c.b;

AtPoint point;
p[0] = 1.f;      // equivalent to: p.x = 1.f;
float y = p[1];  // equivalent to: y = p.y;
```

Bug fixes

Ticket	Summary	Owner	Priority	Version	Created	Modified
#2136	Ignore 'abort_on_license_fail' when 'skip_license_check' is enabled	angel	major	3.3	8 weeks	8 weeks
#2141	NaNs/Infs at low roughness settings in the standard shader	alan	major	3.3	7 weeks	4 weeks
#2146	Python binding for AiArray{Set|Get}Mtx() crashes	oscar	major	3.3	6 weeks	3 days
#2149	framebuffer memory not accounted for in the statistics	marcos	major	3.3	5 weeks	5 weeks
#2161	NaNs in transmittance effects of the standard shader	alan	major	3.3	3 weeks	3 weeks
#2166	specular_anisotropy affecting roughness of isotropic BRDFs in the standard shader	alan	major	3.3	3 weeks	12 days
#2170	Crash when the color parameter of a quad light is mapped with MIS on and a volume_scattering atmosphere shader	alan	major	3.3	2 weeks	6 days
#2181	fully transparent surfaces at max auto transparency depth causing artifacts	alan	major	3.3	6 days	4 days
#2182	AiParseParameterString() not thread safe	alan	major	3.3	4 days	4 days