3.3.0.0

Milestone 3.3

Enhancements

- **Hair diffuse cache**: Implemented a lazy caching scheme for diffuse illumination of hair, which can speed up hair-to-hair global illumination rendering by up to 3x with minimal impact on visual quality. The illumination is cached at spline control points. This lazy cache can be enabled or disabled using the new boolean parameter diffuse_cache in the built-in hair shader (conservatively disabled by default). For custom shaders, see the AiHairDirectDiffuseCache() API below. (trac#1857)
- **Better FOV control in the cyl_camera node**: There are now separate controls for vertical and horizontal FOV in the cyl_camera node. The new parameters are vertical_fov and horizontal_fov. We have provided a backwards-compatibility synonym for the old fov parameter name, which maps to the new horizontal_fov parameter name. There is also a new projective parameter that controls the angular mapping in the vertical axis; when set to true, the vertical mapping is projective, resulting in a standard cylindrical camera projection, and when set to false the vertical mapping is linear, resulting in a lat-long camera projection. Note that the projective parameter is enabled by default, which is a change in behaviour, as previously we always used a hardcoded linear mapping in the vertical axis. (trac#1863)
- **Kick command line shortcuts**: Added handy kick shortcuts to set the default color mode (-cm) and default shade mode (-sm). The following two lines are equivalent: (trac#1859)

  ```
  kick file.ass -set ai_default_reflection_shader.color_mode polywire
  kick file.ass -cm polywire
  ```

- **New framework for interactive node updates**: We have redesigned the API for nodes to better support interactive changes. This has solved a critical memory leak when rendering multiple frames. See a more detailed description of the API changes below. (trac#1861)

API additions

- **API for hair diffuse cache**: Custom hair shaders can call the following API function that returns a fast, cached value of the indirect diffuse component: (trac#1857)

  ```
  AI_API AtColor AiHairDirectDiffuseCache(const struct AtShaderGlobals *sg);
  ```

- **Refactoring of node initialization**: We have modified the semantics of the node_initialize and node_destroy methods to avoid inconsistencies and ambiguous behavior. Also, we have added a new node_update method for interactive modification of nodes (trac#1861). The new semantics are:
  - node_initialize is only called once per render session. It will be called right before the first call to AiRender().
  - node_destroy is only called once at the end of the render session, during the call to AiEnd().
  - node_update is called once per rendered frame. It is called at the beginning of AiRender().

> *Example:*
>
> ```
> node_initialize
> {
>    AtCustomData *data = (AtCustomData*)AiMalloc(sizeof(AtCustomData));
>    node->local_data = (AtVoid*)data;
> }
>
> node_update
> {
>    AtCustomData *data = (AtCustomData*)node->local_data;
>
>    // Evaluate data based on current scene state.
>    // This is called at the begining of each rendered frame
>    // (such as each "pass" of an interactive session with
>    // progressive refinement).
> }
>
> node_destroy
> {
>    AiFree(node->local_data);
> }
> ```

- **Support for separately linking shader components**: When creating links between nodes in a shader network, you can now specify a specific component (e.g. R, G, B) on the target parameter, so that the source shader will be linked only to that component. In addition, each component of a parameter can be linked separately, even to different shaders. (trac#1811)

> *Example:*

```
noise
{
   name noisesource
}

flat
{
   name shader1
   color.r noisesource
}

flat
{
   name shader2
   color.b noisesource
   color.r noisesource
}
```

> *At the same time, the AiNodeLink() and AiNodeGetLink() API functions have support for component specification in the parameter name:*
>
> ```
> AiNodeLink(shader1, "color.r", noisesource);
> ```

> *For now, only target parameters support separate linking of shader components. In the future we will also add support for specifying components in the source data (e.g. linking the red component of a node's output to the green component of a node's input parameter).*

- **AiStretchedPhongBRDF()**: The rarely-used retro parameter is now optional and will have FALSE as its default value, if the parameter is not set. (trac#1793)
- **AiCellular()**: The delta and ID parameters are now optional and will have NULL as their default value, if the parameters are not set. (trac#186
)

Incompatible changes

- **AiCameraInitialize()**: This function has been split in two. So, there is a new AiCameraUpdate() function, to be called from node_update. (trac#1861) The signature for both is:

  ```
  AtVoid AiCameraInitialize(AtNode *node, AtVoid *data);
  AtVoid AiCameraUpdate(AtNode *node, AtBoolean plane_distance);
  ```

- **AiFilterInitialize()**: This function has been split in two. So, there is a new AiFilterUpdate() function, to be called from node_update. The signature for both is:

  ```
  AtVoid AiFilterInitialize(AtNode* node, AtBoolean requires_depth, const char** required_aovs, AtVoid*
  data);
  AtVoid AiFilterUpdate(AtNode* node, AtFloat width);
  ```

- **node_finish**: The signature for this method has been modified, removing the (often misused) destroy_node parameter:

  ```
  AtVoid node_finish(AtNode *node);
  ```

- **node->connectors**: The type of this member variable in the AtNode struct has changed from AtNode* to AtConnectorData, to accomodate per-component linking. See ai_nodes.h for details.
- **AiNodeLink()**: Passing NULL in the source node as a way to "unlink" an existing node link is no longer supported, resulting in a warning message and the function doing nothing. You should instead use AiNodeUnlink(). (trac#1867)

Bug fixes

#1854	-INF values in alpha channel are not properly filtered on the beauty
#1856	possible deadlock with SSS and instancing
#1864	AiNodeReset() should remove all shader links
#1858	light sample cache can allocate the wrong number of samples in SSS