4.1.1.0

Milestone 4.1.1

November 14, 2013

Enhancements

- **Reduced polymesh memory footprint**: We now have slightly better compression of polygon mesh data. Up to 1.1x smaller mesh sizes are commonly seen in our tests, although in theory the memory savings could be even larger. We have also saved 64 bytes per polymesh node, 72 bytes per curves node and 56 bytes per all other geometric objects, which helps when rendering very complex scenes with millions of instances, polymeshes or procedurals. (#3675, #3707, #3713, #3715, #3724, #3726, #3733)
- **Faster binary .ass reading**: We can now read b85-compressed integers in large meshes slightly faster. Assuming the .ass file is already in the OS file cache, we've seen up to a 1.3x speedup for .ass files with large meshes. (#3710)
- **Faster Perlin noise**: AiPerlin2() and AiPerlin3() noise is now about 1.3-1.5x faster. (#3746)
- **Faster gobo shader**: The overhead of the gobo shader used for slidemap projection in spotlights has been reduced. This was done at the cost of disallowing shader links in all parameters other than slidemap. (#3712)
- **Blending of RGBA AOVs**: RGBA AOVs now support opacity blending in shaders that have properly declared them to be composable. Note that this will only allow the RGB components of the RGBA value to accumulate through the various semi-opaque layers, since there is currently no way for shaders to define the opacity of the A component of their RGBA result. (#3007)
- **Automatic simplification of motion keys**: Remove motion blur from cameras and lights which have identical motion keys. This is done for consistency with the processing already applied to geometry. This was causing lots of information to be calculated per sample instead of per light. (#3721)
- **Automatic estimation of custom camera ray derivatives**: Developers writing custom camera nodes can leave the dDdx, dDdy, dOdx, and dOdy fields in AtCameraOutput to zero, and an accurate estimate will be computed for them automatically. This prevents extreme texture IO degradation and makes it possible to implement custom cameras without having to worry about the math for computing ray origin and direction derivatives. (#3652)
- **Custom EXR metadata**: An array of strings custom_attributes has been added to the EXR driver which lets users write their own metadata (#2153). Supported types are int, float, point2, matrix16 and string. Each string has the format "type_name attr_name value[s]". Some examples:

```
driver_exr
{
 filename output.exr
 custom_attributes 3 1 STRING
  "float mycustomfloat -23.23"
  "point2 mycustompoint2 -23 -23"
  # will create an EXR string attribute containing "this is my string"
  "string mycustomstring     this is my string"
 ...
}
```

- **Raw drivers now can accept input from user defined AOVs**: When declaring output AOVs in options.outputs the syntax for regular drivers is now also allowed for raw drivers. This means users can add new AOVs at runtime for a raw driver to work with. If needed, this information can be queried through the usual AiOutputIterator* calls. (#3750)
- **OpenEXR2 deep output**: A new driver driver_deepexr has been added to output to deep images. It can be declared in options.outputs like a regular driver. Volumetric samples are not yet supported. For now, this driver is only available in Linux and OSX. A brief explanation of the available attributes: (#3756)

BOOL tiled false	Write out tiled or scanline deep images; Nuke only supports scanline deep images
BOOL subpixel_merge true	Nearby subpixel samples will be merged
BOOL use_RGB_opacity false	Write out RGB opacity, rather than just alpha; Nuke can read these images but cannot display them
FLOAT alpha_tolerance 0.01	Alpha tolerance over which samples will not be merged together
FLOAT depth_tolerance 0.01	Depth tolerance over which samples will not be merged together
BOOL alpha_half_precision false	Use 16-bit floats for alpha layer
BOOL depth_half_precision false	Use 16-bit floats for depth layer
FLOAT[] layer_tolerance (empty)	A list of tolerances that will prevent merging for each AOV in options.outputs; if it is a single value it will apply to all layers
BOOL[] layer_enable_filtering (empty)	A list of booleans enabling or disabling filtering for each AOV in options.outputs. Integers, vectors or points are not filtered by default
BOOL[] layer_half_precision (empty)	A list of booleans enabling or disabling 16-bit floats for each AOV in options.outputs. Integers are always full precision

API additions

- **AtTextureParams channel control**: A new field start_channel has been added to AtTextureParams which allows a shader writer to choose the starting channel in the texture to sample from, where consecutive channels will be used (1, 3, or 4 depending on single-channel, RGB, or RGBA sampling). This can be useful for e.g. multi-channel textures with more than just RGBA channels. In order to help identify the different channels, a new API function has been added: (#3749)

```
AI_API const char* AiTextureGetChannelName(const char* filename, unsigned int channel_index);
```

Incompatible changes

- **Changed type of subdiv_iterations**: The data type of the polymesh parameter subdiv_iterations has been changed from int (32-bit) to byte (8-bit) to save memory. Applications/plugins that set this parameter with the C or Python API must now call AiNodeSetByte instead of AiNodeSetInt. (#3707)
- **Changed default autobump_visibility**: The default autobump_visibility has been changed so that glossy rays by default do not perform autobump. This results in faster renders. Usually this change should not be noticeable because of the blur inherent in glossy reflections, but if it causes unwanted artifacts, it can be reverted by setting the glossy ray bit in autobump_visibility back to 1 (the integer value would be 223). (#3739)
- **AiNodeEntryGetFilename() can return NULL**: Client code using the AiNodeEntryGetFilename() API function should make sure it checks for theNULL pointer, which is used to indicate a built-in node. (#3715)
- **AtTextureParams new field**: The new field start_channel in AtTextureParams will be initialized to zero by default when usingAiTextureParamsSetDefaults(). But if your shaders are initializing the fields of that structure manually, you will need to set this new field to zero or else you may get random start channels when sampling textures. The size of the structure is unchanged, so this should be the only side effect of the new field. (#3749)

Bug fixes

Ticket	Summary	Component	Owner	Priority	Version	Created
#2727	inconsistent emission in standard shader when bounce_factor=0	arnold	borja	major	3.3	21 months
#2789	crash in gobo shader with empty rotate array	arnold	marcos	major	3.3	20 months
#3006	crash when firing shadow rays when shading shadow rays	arnold	ramon	major	4.0	15 months
#3007	Support RGBA aov blending through auto transparency	arnold	alan	major	4.0	15 months
#3351	Ward-Duer BRDF in standard shader crashes with infinite dPdu or dPdv values	arnold	thiago	major	4.0	8 months
#3574	per-component linking not working within arrays	arnold	thiago	major	4.0	3 months
#3632	crash in procedural with lights but no geometry	arnold	angel	major	4.0	6 weeks
#3694	metadata errors should be warnings	arnold	angel	major	4.0	4 weeks
#3696	Crash when re-rendering a texture-mapped quad_light	arnold	mike	major	4.0	3 weeks
#3701	rotation matrix decomposition crash	arnold	thiago	major	4.0	3 weeks
#3702	Fix instancing stats for nurbs	arnold	mike	major	4.0	3 weeks
#3703	Replacing shader in an object does not affect its instances	arnold	angel	major	4.0	3 weeks
#3705	Avoid vertical aliasing with rolling shutter	arnold	oscar	major	4.0	3 weeks
#3717	rare crash in malformed .ass file with unmatched square bracket	arnold	angel	major	4.0	2 weeks
#3722	allow '(' and ')' characters in user-data parameter names	arnold	marcos	major	4.0	2 weeks
#3723	bump3d does not work correctly with transformed geometry	arnold	ramon	major	4.0	2 weeks
#3729	bump3d can generate erroneous normals pointing below the surface	arnold	ramon	major	4.0	13 days
#3730	don't write 'threads' and 'ignore_list' options in .ass files	arnold	marcos	major	4.0	13 days
#3734	Memory growing too much with many mesh lights	arnold	ramon	major	4.0	11 days
#3740	Shader override broken for multiple levels of instancing	arnold	angel	major	4.0	8 days
#3742	Visibility override broken for multiple levels of instancing	arnold	angel	major	4.0	7 days
#3745	subdivision creates nans in mesh's dPdu/v	arnold	thiago	major	4.0	7 days
#3757	Procedural override of "transform_time_samples" parameter	arnold	angel	major	4.0	2 hours
#3695	kick -repeat concatenates kick command line in logs	arnold	oscar	trivial	4.0	3 weeks
#3700	Amend AiASSWrite() in the Python API	arnold	oscar	trivial	4.0	3 weeks