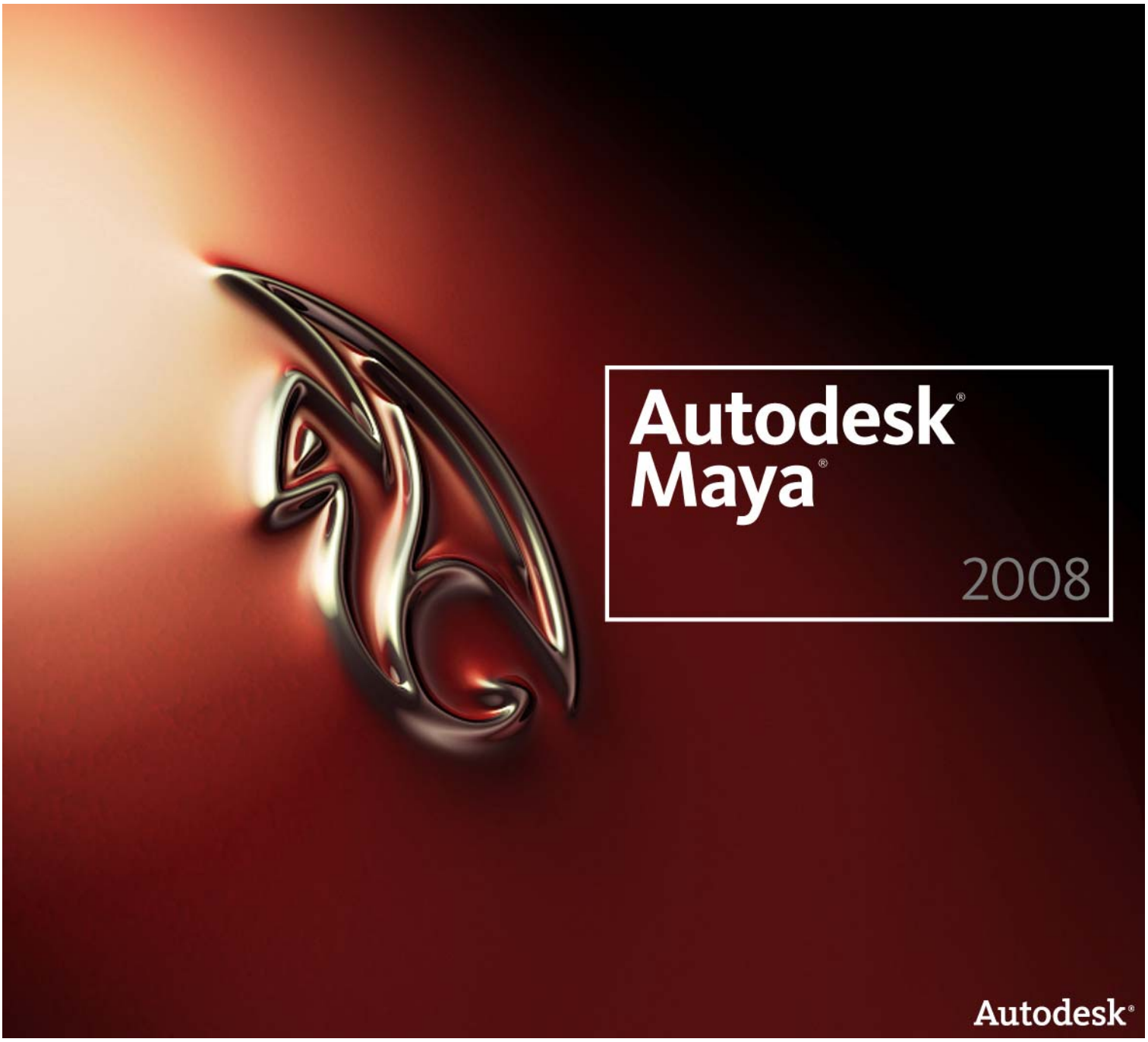

Maya 2008 Service Pack 1 Release Notes

Legal Notice

Autodesk® Maya® 2008

Trademarks

Disclaimer

Published by:
Autodesk, Inc.
111 Mclnnis Parkway
San Rafael, CA 94903, USA

Patents

Third-Party Software Credits and Attributions

Table of Contents

1 Autodesk Maya 2008 Service Pack 1 release notes 5

Licensing
Limitations 5
-lic command incorrectly described in Installation and Licensing guide 5
FLEXLM/LMTools on Microsoft Windows Vista 5

General
What's Fixed 6
Maya running out of desktop heap memory 6
Error occurs when using the Optimize Now option 6
Loading Maya 2008 on Fedora 7 6
Restoring files with Hypershade or Outliner panel in the layout 6
Lock Status not respected 6

Mac OS X
What's Fixed? 7
Production quality renders on a Mac with ATI graphics cards. 7
Marking Menu overlapping 7
Hotbox and Channel Box on Mac 7
UV Texture Editor middle mouse button move functionality on Mac 7
Marking menu offset on a Mac 7
Marking menu delay on a Mac 7
Marking menu selection feedback on a Mac 7
Mouse scrolling in the channel box on a Mac 8
Mouse wheel scroll in an empty textScrollList on a Mac 8
apiMThreadPoolTest on Power PC 8
threadTestCmd.cpp generating an error 8

API
What's Fixed? 8
rotatePivot or scalePivot expression on MPxTransform error 8
moveTool.py and invoke spMoveToolContext 8
Verifying API version 8
Maya Python API:. 8
MDGModifier::createNode() error 9
MFnMesh::split() error 9
MPxLocator draw method 9

Python
What's Fixed? 9
pymel crashes in Maya 2008 9

fileBrowserDialog fileCommand flag function in Python 9
Unable to import md5 in the Python 9

MEL and Expressions

What's Fixed? 9
MVector on systems running Microsoft products 9

Modeling

What's Fixed? 10
PolySplit Command on translated mesh 10
Middle mouse button dragging with the Polygon Split tool 10
MPxTransform and pivot rotation 10

Animation

What's Fixed? 10
Values displaying in the Component Editor (Linux) 10
Euler curve filtering (Linux) 10
Painting skin weights with Use Components on 10
Unexpected results with skin exported by FBX after using the Move Skinned Joints Tool 11
Limitations 11
Insert Joint Tool limitations 11

Dynamics and Effects

What's Fixed? 11
Particle Instancing fails when a custom attribute is used to drive the instance 11
Multiple processors for fluid rendering may result in system crash 11
Creating many nCloth objects in a scene affects rewind performance 11
IKFK connected to Hair Systems 11

Rendering and Render Set up

What's Fixed? 12
Batch rendering an AVI sequence now fixed 12

mental ray

What's Fixed? 12
Fixed mental ray for Maya limitations 12
New mental ray for Maya features and feature improvements 20
Limitations 27
Hardware renderer limitation with displaying mental ray area lights 27

1 Autodesk Maya 2008 Service Pack 1 release notes

The Autodesk Maya 2008 Service Pack 1 release notes contain information on what's fixed and limitations for this release.

To find out more, see the following sections:

- "Licensing" on page 5
- "General" on page 6
- "Mac OS X" on page 7
- "API" on page 8
- "Python" on page 9
- "MEL and Expressions" on page 9
- "Modeling" on page 10
- "Animation" on page 10
- "Dynamics and Effects" on page 11
- "Rendering and Render Set up" on page 12
- "mental ray" on page 12

Licensing

Limitations

-lic command incorrectly described in Installation and Licensing guide

In the Installation and Licensing guide, under the section Limiting the use of Maya Unlimited product licenses, the description for the `maya -lic=unlimited` should indicate that it is not applicable to the Windows versions of Maya.

FLEXLM/LMTools on Microsoft Windows Vista

For the Microsoft Windows Vista operating system, for 32-bit and 64-bit machines, you'll need an upgrade to FLEXLM/LMTools in order for Maya to be licensed properly, if you are licensing Maya using floating licenses.

This update is not required for node-locked licenses.

The version of FLEXLM that ships with Maya is 10.8. For Vista operating systems only, please download and install the 10.8.5 version of FLEXLM from the following link:

http://www.globes.com/support/fnp_utilities_download_10.8.5.htm

Download the 32-bit version of LMTools and use this on Vista 32-bit and 64-bit systems.

We only recommend this download if it is necessary for licensing (floating licenses/ Vista operating system); otherwise, use the FLEXLM version supplied with Maya.

General What's Fixed

Maya running out of desktop heap memory

Previously, Maya would crash when it runs out of desktop heap memory, for example, when you are trying to open a new window or display more UI fields. This problem has now been fixed. When Maya runs out of desktop heap memory, it will not open the new window and an error message is displayed in the Script Editor: "Cannot create UI object: <object name>". It is possible, however, that a crash may still occur, but in most cases, Maya would simply not open the new window.

When this occurs, you should close some other windows to reduce the amount of desktop heap memory being used. The windows could be from other applications you are running or they can be from within Maya. Closing Microsoft Excel is also helpful, or you may close other Maya windows.

Error occurs when using the Optimize Now option

There was an issue with selecting items in the Optimize Now menu options (File > Optimize Scene Size > ❐). This problem has been fixed.

Loading Maya 2008 on Fedora 7

Previously, loading Maya 2008 on the Fedora 7 operating system when Xinerama was inactive caused Maya to crash. This has been fixed.

Restoring files with Hypershade or Outliner panel in the layout

In previous versions of Maya, a scene that contained the Hypershade or Outliner panel would not be restored to the same layout upon re-opening of the file. This has been fixed.

Lock Status not respected

In previous versions of Maya, replacing file references caused a problem with the Lock Status. This has been fixed.

Mac OS X What's Fixed?

Production quality renders on a Mac with ATI graphics cards

In previous versions of Maya on the Mac, rendering a scene with Render Globals set to Production Quality produced distortions on systems with ATI graphics cards installed. This has been fixed.

Marking Menu overlapping

In previous versions of Maya, if you opened the Poly Brush Marking menu and selected Increase Complexity, it was difficult to return to the main marking menu. This has been fixed.

Hotbox and Channel Box on Mac

In previous versions of Maya on the Mac, if you opened the Hotbox, then pressed the left mouse button to the left of the box and kept moving the mouse to the right a menu to open or close the Channel Box appeared but did not work. This has been fixed.

UV Texture Editor middle mouse button move functionality on Mac

In previous versions of Maya on the Mac, the axis restricted move functionality using the middle mouse button in the UV Texture Editor did not work. This has been fixed.

Marking menu offset on a Mac

In previous versions of Maya on the Mac, invoking the marking menu at the edge of the screen drew the marking menu outside of the screen boundaries. This has been fixed.

Marking menu delay on a Mac

In previous versions of Maya on the Mac, there was a delay between right-clicking and the appearance of the marking menu. There was also a delay between selecting an item in a marking menu and the release of that command. This has been fixed.

Marking menu selection feedback on a Mac

In previous versions of Maya on the Mac, selection feedback was not provided at the end of a successful gesture. This has been fixed.

Mouse scrolling in the channel box on a Mac

In previous versions of Maya on the Mac, scrolling the mouse in the channel box while an attribute value was still active caused Maya to crash. This has been fixed.

Mouse wheel scroll in an empty textScrollList on a Mac

In previous versions of Maya on the Mac, scrolling the mouse in an empty textScrollList control caused Maya to crash. This has been fixed.

apiMThreadPoolTest on Power PC

Previously, there was an issue where using `apiMThreadPoolTest` on the Power PC caused Maya to crash. This problem has been fixed.

threadTestCmd.cpp generating an error

Previously, there was an issue where using `threadTestCmd.cpp` on the Mac OS X platform caused Maya to crash. This problem has been fixed.

API What's Fixed?

rotatePivot or scalePivot expression on MPxTransform error

Previously, there was an issue where applying the `rotatePivot` or `scalePivot` expression on `MPxTransform` caused Maya to crash. This has now been fixed.

moveTool.py and invoke spMoveToolContext

Previously, using `moveTool.py` and invoking `spMoveToolContext` caused Maya to hang. This problem has been fixed.

Verifying API version

An issue where users received warning messages when checking the version of Maya has been fixed.

Maya Python API: MSceneMessage::addCheckCallback error

An issue where `MSceneMessage::addCheckCallback` was not working in the Maya Python API has been fixed.

MDGModifier::createNode() error

`MDGModifier::createNode()` failed to work in Maya 2008; this has now been fixed.

MFnMesh::split() error

An issue where using `MFnMesh::split()` caused Maya to crash has been fixed.

MPxLocator draw method

In Maya 2008, drawing a `GL_QUADS` within the `MPxLocator` draw method did not work. This has been fixed.

Python What's Fixed?

pymel crashes in Maya 2008

Using the `pymel` command causes a crash in Maya 2008. This has been fixed.

fileBrowserDialog fileCommand flag function in Python

Previously in Maya, the `fileBrowserDialog` `fileCommand` flag did not function in Python. This has been fixed.

Unable to import md5 in the Python

Previously, the `md5` format could not be imported in Python. This has now been fixed.

MEL and Expressions What's Fixed?

MVector on systems running Microsoft products

In Maya 2008, using MVector (or any other expressions that refer to the MS namespace) on a system with Microsoft. Net libraries installed caused an preprocessing error. This has been fixed.

Modeling What's Fixed?

PolySplit Command on translated mesh

In previous versions of Maya, using the PolySplit command with Split only from edges turned off on a mesh that was not centered at (0,0,0) did not split the polygon the same way the Polygon Split Tool did. This problem has been fixed.

Middle mouse button dragging with the Polygon Split tool

In pervious version of Maya, dragging a point of a Polygon Split that wasn't on an edge using the middle mouse button caused Maya to display an incorrect path of edges. This problem has been fixed.

MPxTransform and pivot rotation

In previous versions of Maya, modifying the rotation pivot or scale pivot of an MPxTransform node with an expression resulted in the no change to the pivot (the pivot reverted to its default value each time). This has been fixed.

Animation What's Fixed?

Values displaying in the Component Editor (Linux)

An issue that caused incorrect values to display in the Component Editor for objects with greater than 70 000 faces has now been resolved. You can now set up skin weighting for an object, then select vertices, and view the correct values in the Component Editor as expected.

Euler curve filtering (Linux)

A problem that prevented the filterCurve command from finding Euler anim curves specified directly in the command line, or if the attribute was locked, has been fixed.

Painting skin weights with Use Components on

A problem that occurred as you painted influence weights on a mesh with the skinCluster Use Components option turned on has been fixed. You can now paint weights for a mesh with the Use Components option turned on without unexpected deformation on the mesh.

Unexpected results with skin exported by FBX after using the Move Skinned Joints Tool

A problem that caused inverted worldMatrix values to be stored in the pose node after joints were moved with the Move Skinned Joints Tool has been corrected. You can now move joints and export the skinning data using the FBX format as expected.

Limitations

Insert Joint Tool limitations

As you are using the Insert Joint Tool, you may need to tumble or otherwise adjust the camera angle in order for joints to be inserted. In addition, to ensure that the tool works as expected, you must turn on the display of pivots in the Show menu (Show > Pivots). If you prefer to work with pivot display turned off, you can also change the selection mask to include joints.

Dynamics and Effects

What's Fixed?

Particle Instancing fails when a custom attribute is used to drive the instance

Particle instancing no longer fails when a custom attribute is used to control the attribute(s) that are driving the instancing.

Multiple processors for fluid rendering may result in system crash

Maya no longer crashes when using a multi-processor/multi-core machine to render fluids.

Creating many nCloth objects in a scene affects rewind performance

Creating many nCloth objects (for local or world space) in the same scene no longer affects rewind performance.

IKFK connected to Hair Systems

Previously, a scene with IKFK connected to Hair Systems would cause issues. This problem has been fixed.

Rendering and Render Set up

What's Fixed?

Batch rendering an AVI sequence now fixed

Previously, batch rendering an AVI sequence may create a NULL AVI file and you would need to switch to a different compressor or select the uncompressed setting instead. This has now been fixed.

mental ray

What's Fixed?

Fixed mental ray for Maya limitations

The following lists the mental ray for Maya rendering limitations that have been fixed for Maya 2008 Service Pack 1.

Fixed BSP/BSP2 acceleration limitations

- For BSP acceleration used in combination with segmented or sorted shadows, fixed missing geometry normal transformation for polygonal objects without vertex normals.
- For BSP2 acceleration, fixed possible crash when rendering subsequent frames for scenes with motion blur.
- For BSP2 acceleration used in combination with the rasterizer or scanline, fixed possible self-intersection artifacts for scenes with multiple-instanced objects.
- For BSP2 acceleration, fixed possible artifacts produced by `mi_trace_continue` function.
- Fixed a potential crash for BSP2 acceleration used in combination with assemblies.
- For BSP2 acceleration, fixed possible failure to trigger motion oversampling.
- Fixed a crash for lightmaps computed with BSP2 acceleration.
- Fixed a crash if a combination of the BSP2 acceleration, rasterizer (or detailed shadowmaps) and hair was used.
- Fixed missing shadows if a light sample loop was called from a volume shader and BSP2 acceleration was used.
- For BSP2 acceleration used in conjunction with segmented or sorted raytraced shadows, fixed a possible infinite self-intersection loop which may lead to out-of-memory total. This underlying precision issue has only been observed on the 32-bit Linux platform.
- For BSP2 acceleration used in conjunction with motion blur and raytraced shadows, fixed possible failure to detect motion and trigger motion blur oversampling.
- Fixed a crash for lightmaps computed with BSP2 acceleration.

- Some of the memory allocations in the BSP2 acceleration were not encountered in mental ray memory statistics; this has now been fixed.
- Fixed a crash that occurs when photon emission with `photon autovolume` mode enabled is used in combination with BSP2 acceleration.
- For BSP2 acceleration, fixed a possible crash that occurs when a mix of static and moving geometry is used.
- For BSP2 acceleration used in combination with multi-segment motion, fixed a severe bug with the computation of internal boxes during BSP2 construction. This also fixes possible artifacts and significantly improves the performance for such scenes.
- For BSP2 acceleration, fixed possible artifacts due to missing motion (time contrast) oversampling.
- For BSP2 acceleration used in combination with the rasterizer and with motion samples option set to 1, build static (no motion) acceleration. This dramatically reduces memory consumption and improves performance for that option combination.
- Fixed missing echo for the BSP2 acceleration.
- For BSP2 acceleration, fixed possible missing geometry for objects with multiple instances in the scene. Fixed possible self-shadowing artifacts for such scenes.
- Fixed severe artifacts if multiple-instanced assemblies are used in combination with BSP2 acceleration.
- For BSP2 acceleration, previously, `miState::dist` was not set to zero if no geometry is hit. This has now been fixed.
- For BSP2 acceleration used to render scenes with assemblies, fixed unnecessary loading of assemblies. In addition to lower memory consumption, this fix results in significant speedup for scenes with a large number of assemblies due to better parallelization of assembly loading.

Fixed hair rendering limitations

- For hair rendering, some memory allocations were not included in the memory statistics; this has now been fixed.
- For hair rendering, fixed possible shadow artifacts.
- For hair rendering with the rasterizer, fixed possible artifacts due to improper `miState::normal_geom` initialization for hair primitives.

Fixed potential crashes

- Fixed a potential crash if one of the IPv4 or IPv6 interfaces for localhost is not available.
- Fixed a crash if a state shader called in charge of a geometry shader accessed `miCamera::buffertag`.
- For OpenEXR files, fixed a crash if more than 8 images are stored in one file. The fix is in the `mi_openexr.so` library.
- For `ccmesh` objects, fixed possible crash for mixed tri-/quad-meshes.

- For texture caching, fixed a crash that occurs during image conversion when a color image is opened as a scalar texture.
- Fixed a possible crash if final gathering is called from a volume shader and BSP acceleration is used.
- Fixed a possible crash if the rasterizer with a lens shader is used in combination with the BSP diagnostic mode.
- Fixed a possible crash when using displacement shaders in assemblies.
- Fixed a possible crash if the `miQ_LIGHT_NORMAL` mode of `mi_query` has been used on a cylinder light.
- Fixed a crash that occurs when both contour rendering and a shader modifying `miState::raster_x/raster_y` are used. The contours drawn outside of the currently rendered tile are ignored.
- Fixed possible crash for the raytraced shadows if the simple `shadow on` mode is used in combination with BSP acceleration, and the shadow shader called ray or shadow tracing functions.
- Fixed possible crash for large scenes using BSP2 acceleration in combination with a large number of assemblies.

Fixed platform specific limitations

- On Mac OS X 64-bit PPC machines, fixed broken `mi_openexr.so`.
- For Mac OS X, fixed possible tessellation artifacts in the edge merging.
- For Mac OS X, fixed the number of CPUs detection for dual-core machines.
- Fixed possible parse errors for non-tagged primlist objects on platforms using a GCC4 compiler.

Fixed shader limitations

- For the architectural sky shader (`mia_physicalsky`), fixed missing Pi factor in the computation of Zenith illuminance.
- For the architectural physical sun shader (`mia_physicalsun`), the illumination mode was not initialized in the shader implementation; this has now been fixed.
- For the architectural physical sun shader (`mia_physicalsky`), the illumination mode was not initialized in the shader implementation; this has now been fixed.
- For the paint shader (`mi_car_paint_phen`), added missing `edge_factor` parameter to the declaration.
- For the architectural physical sun shader (`mia_physicalsky`), added parameters that allow you to set direct normal illuminance explicitly.
- For the architectural material shaders (`mia_material/mia_material_x`), fixed possible dot artifacts in rendering of indoor scenes with bright environment and glossy interpolation used in combination with Use Detail Distance.
- Ensure that, for shadow shaders, `miState::scanline` is 0 as specified.

- Extended architectural `mia_physicalsky` shader with the explicit Perez model.
- For the architectural physical sun shader, fixed the `automatic_photon_energy` mode.
- For the architectural material shaders (`mia_material/mia_material_x`), improved performance by ignoring bumps for finalgather rays.
- For the architectural material shaders (`mia_material/mia_material_x`), fixed the code which removes specular highlights for finalgather rays.
- For the architectural material shaders (`mia_material/mia_material_x`), fixed wrong reflections of some volumetric effects.
- For the architectural material shaders (`mia_material/mia_material_x`), fixed possible artifacts caused by the Use Detail Distance feature when no environment shader is used.
- For the architectural output shaders, fixed a minor issue in triggering *once per render* error messages.
- For the base lightmap shader (`mib_lightmap_write`), added a new `fg_mode` parameter that controls the precomputing of final gather points on the lightmaps. Possible values for the parameter are: -1 for no precomputation (compatibility behavior), `0` for every mesh vertex, and `N>0` for a single finalgather point per N pixels. The default value is 3.
- For the architectural material shaders (`mia_material/mia_material_x`), if the mental ray kernel trace depth limit is reached, the black color is returned instead of the environment color.
- For architectural material shadow shaders, significantly improved performance for opaque materials.
- For the architectural round corner shader (`mia_roundcorners`) used in combination with bump mapping, fixed a crash that occurred if the round corner shader was called as a shadow shader. For scenes using such shader combinations, performance has been significantly improved.
- For the architectural material shaders (`mia_material/mia_material_x`), fixed possible "glowing object" artifacts in scenes using final gathering with low final gather trace depth limit and HDR environment shaders.
- For the subsurface scattering shader, enabled the support for instance light lists.
- For the physical light (`physical_light`) shader, fixed incorrect energy scaling for visible rectangle and cylinder lights. This fix affects the color of ray intersections with lights only; the illumination of scenes is unchanged.
- For architectural material shaders (`mia_material/mia_material_x`), disable reflection and refraction interpolation if contour rendering is enabled.
- Some of the contour shaders were not exported on Windows platforms; this has now been fixed.
- For the architectural material photon shader, fixed possible tracing of photons in an uninitialized direction.
- Improved embedded strings for shader library date/version.

- For fast and physical subsurface scattering shaders (`misss_*`), added instance light list support if the `mode` parameter is set to 4.
- For the architectural portal light shader (`mia_portal_light`), fixed a small darkening artifact at the midpoint of the edge of the portal light if it is very close to a surface.
- For the architectural round corner shader (`mia_roundcorners`), improved the quality of edge rounding.
- For the architectural material shaders (`mia_material/mia_material_x`), fixed possible artifacts in Fresnel reflection (e.g. glass) in combination with bump mapping and rounded corners.
- Added a `mib_fast_occlusion` shader based on the build-in ambient occlusion.

Fixed limitations for openEXR images

- Fixed the null tag fatal in direct creation (not output of rendering framebuffers) of OpenEXR image files, used in particular for lightmap textures.
- For multi-imaged OpenEXR output files, removed unnecessary conversion to color for depth, normal and motion framebuffers which caused information loss.
- For OpenEXR files containing multiple framebuffer outputs, corrected the output for user framebuffers.
- For OpenEXR output files containing multiple images, the `-file_dir` command line option has not been used properly; this has now been fixed.
- For OpenEXR output files containing multiple images, fixed a possible crash if the image type conversion for some framebuffer images is required.
- For OpenEXR image files, removed a limit on the string length of the qualified file name.
- For OpenEXR output images, do not write the frame buffer as a name channel for the Z depth buffer; instead, use channel name "Z".
- For OpenEXR images, fixed a crash for single channel images containing only the `"A"` channel.
- Fixed wrong compression options used for OpenEXR output images.

Fixed mental ray standalone limitations

- Fixed the non-functional `-fb_dir` command line option.
- Fixed incorrect command line help for several command line options.

Fixed miscellaneous limitations

- For the rasterizer, fixed possible crash if mi_query miQ_NUM_TEXTURES mode is called from a volume shader.
- Fixed missing raytraced hair in assemblies.
- Fixed a tag leak for assembly filenames.

- For database dumps, fixed a name for the assembly tag type.
- For integrators, fixed a typo in the `mi_par_aborted()` check, which prevented the application callback from been called.
- Fixed a tag leak if an assembly tag has been deleted with the `mi_api_delete` function.
- For `ccmesh` objects, fixed possible texture seams artifacts.
- Fixed motion transformation inheritance for assemblies attached to a moving instance.
- For raylib integrations, fixed incorrect thread count in out-of-memory emergency exits.
- An application may register a memory callback, which is called when the mental ray kernel is not able to free a sufficient amount of memory by itself. If this callback is called and the application is aware that it is not able to free a significant amount of memory, it may decide to call the `mi_par_abort()` function with a `miABORT_RAY_FREEZE` flag. In that case, the mental ray kernel would hang all rendering threads and delete the database elements which are currently not being accessed, freeing a moderate amount of memory. After a short delay, the `mi_rc_run` call will return. The application is supposed to save user work and exit. mental ray library functionality should no longer be used, and in particular the `mi_raylib_exit` should not be called.
- For multi-hosted rendering, fixed possible crash if multiple instances of trilist objects are used.
- For the scene echo, do not print instance light lists if echoing of lights or light instances is disabled.
- For the command line option `-ip`, fixed the default value to `off` if the option is given.
- Fixed motion transformation inheritance into assemblies instances.
- Removed false error message when editing objects during rendering. The operation is legal for assemblies.
- For `ccmesh` objects, fixed texturing on seams.
- Fixed a string options tag leak.
- Fixed a memory leak and possible crash in the `Framebuffer::remove` method.
- In final gather precomputing statistics, do not print out the number of light shader calls.
- Fixed a parser error for lightprofiles attached directly to `miLight` (in other words, not used as a light shader argument). For the Framebuffer interface, `get_index` no longer modifies the index value to -1 if a buffer with the given name is not found.
- For scenes with assemblies, fixed possible wrong motion and missing scene parts due to an erroneous assembly motion bounding box.
- For scenes with assemblies, fixed a possible crash if the main part of the scene contains no geometry or geometry with no texture spaces only.

- For trilist objects with displacement, fixed echoing of the displace approximation.
- Fixed possible framebuffer counting problem resulting in "failed to create frame buffer" error message.
- Fixed possible artifacts if a single eye ray spawns a very large number of rays in the shading subtree.
- Fixed a crash for `mi_query` mode `miQ_PRI_INDEX` used on a `miState` with `state->pri == 0`.
- For `ccmesh` objects with displacement, added clipping of negative displacement values smaller than -maxdisplace. Added a warning for NaN displacement values returned by displacement shaders.
- For non-`ccmesh` subdivision surface objects, fixed missing displacement in the immediate tessellation called by a geometry shader.
- For the `ccmesh` objects, fixed possible crash for large scenes not fitting into physical memory.
- For `ccmesh` objects, fixed instance and instance override approximations which were not taken into account. For `ccmesh` objects, fixed possible wrong subdivision level selected for the immediate tessellation called by a geometry shader.
- Fixed possible view-dependent tessellation quality artifacts for NURBs when the orthographic camera is used.
- For the framebuffer C compatibility interface, fixed the following:
  - unintentional coping of the primary framebuffer color to user framebuffers
  - broken output of user framebuffers with index 8 and larger
  - echoing of user framebuffers
- For the pass file rendering used in combination with non-primary ("user") framebuffers, fixed an error in writing out the first rendered tile of the image.
- For multi-hosted rendering, fixed a rare crash observed for scenes using large BSP acceleration.
- For multi-hosted rendering, fixed possible database statistics and dump reporting tags in the NULL module.
- For rasterizer used in combination with volume shaders, fixed a possible crash for shaders setting `state->pri` to 0, such as the physical `parti_volume` shader.
- Fixed possible artifacts for sphere and cylinder area lights originating from an incorrect normal transformation.
- Fixed void photon emission from object lights.
- For texture caching, removed an erroneous info message that conversion on the fly from one image type to an identical one is required.
- Fixed possible artifacts for `ccmesh` objects: if multiple regions are created, wrong materials from other regions on the same surface may be used.
- Fixed minor memory leaks in assembly loading.

- For multi-hosted rendering, fixed a crash in animation rendering. For some tags, the old content was re-transmitted over the network for the second and subsequent frames.
- For rendering of scenes that do not fit into physical memory even with the use of a combination of assemblies and BSP2 acceleration, fixed possible unknown tag access fatal error.
- For the rasterizer, fixed a performance drop caused by the increased shading rate due to the unnecessary increase of tessellation for objects with slow motion.
- For multi-hosted rendering, fixed possible freed memory access after a frame end.
- For framebuffers, added a missing default file format.
- Fixed Color + Z combined output for Iff and RLA files using C-style buffer interface.
- Fixed a possible crash with multi-hosted rendering when a slave is removed from the list of the rendering hosts shortly after rendering finishes.
- Fixed possible texture seam artifacts in `ccmesh` tessellation.
- For some combinations of output statements and non-primary (user) framebuffers, correctly detect the bit-per-pixel value.
- For the pass file rendering used in combination with non-primary ("user") framebuffers, fixed an error in writing out the first rendered tile of the image.
- For the rasterizer used in combination with volume shaders, fixed a possible crash for shaders setting `state->pri` to 0, such as the physical parti_volume shader.
- For `mi_query` mode `miQ_PRI_INDEX`, fixed crash that occurred if `state->pri` is 0.
- In rasterizer, fixed initialization of `state->org` and `state->dist` if lens shaders were used and no geometry was hit.
- mental ray rendering of softbody dynamics no longer creates artifacts.

  Previously, artifacts may appear when mental ray is used to render softbody dynamics. This has now been fixed.
- Maya crashes when mental ray is used to render particles as blobby surfaces.

  Maya no longer crashes when mental ray is used to render particles as blobby surfaces.
- mental ray for Maya now queries light links at the object level rather than at the component level. This method resolves some issues that arise when light linking is interpreted incorrectly. To revert to the old behavior, use this mel script:

```
addAttr -at bool -ln "useComponentLinking" mentalrayGlobals;
setAttr mentalrayGlobals.useComponentLinking 1;
```

- mental ray for Maya has now improved support for bump mapping with multiple UV sets.

Use the following mel script to enable this behavior:

```
addAttr -at bool -ln "exportMultiBumpBasis" mentalrayGlobals;
setAttr mentalrayGlobals.exportMultiBumpBasis 1;
```

Note: Enabling this behavior also increases the amount of data translated by mental ray for Maya, as each UV set now carries extra information.

- Command line render to .iff or .psd no longer results in two separate files being rendered.

  Previously, a command line render to .iff or .psd, with Depth channel selected under Renderable Cameras, produced two separate files, one for z-depth and one for RGB. This has now been fixed and only one file is created.

- Maximum values are increased for mental ray Rasterizer Quality attributes.

  In the Render Settings window: mental ray tab, under the Anti-Aliasing Quality section and the Rasterizer Quality sub-section, the maximum for the Visibility Samples attribute has now been increased to 30 and the Shading Quality attribute does not have a maximum limitation.

- Load of phenomena into Node Factory now delayed.

  Previously, if you choose to autoload your phenomenon file via the MI_CUSTOM_SHADER_PATH environment variable, and your phenomenon references another shader package, an error may occur if the shader package is loaded into the Node Factory after the phenomenon file.

  This has now been fixed and mental ray for Maya waits until all shader libraries have been loaded and registered before loading your phenomenon.

  To disable this feature, add a new MEL function to `mentalrayFactory.mel`:

```
global int miCustomNodesLatePhenomena() which should return '0'.
```

- Maya now builds the Attribute Editor properly for custom shaders and phenomenon.

  Previously, Maya would not build the Attribute Editor properly when a shader or a phenomenon has struct parameters with certain types of member parameters as follows:

  - If the struct has two or three members of either integer or scalar type parameters.
  - When integer or vector type parameters are members of a struct.
  - A struct is nested inside of a struct.

  This problem is now fixed.

New mental ray for Maya features and feature improvements

New global illumination technique: irradiance particles

A novel global illumination technique, *Irradiance Particles* (short: IP), has been introduced. It is, in many cases, superior to final gather and/or photon mapping in terms of image quality, usability and performance.

A short description of the technique may be given as follows: before rendering, importons are shot to the scene from the camera. Their hit positions with information on the amount of direct (and possibly indirect) illumination coming at their position (hence the name "irradiance particles") are combined into a map. Optionally, one or more passes of indirect illumination can be computed. The nature of the algorithm is that the computation is importance-driven. During rendering, Irradiance Particles are used to estimate the irradiance for every shading point; if only direct illumination is collected for irradiance particles, then this is equivalent to one bounce of indirect lighting. Irradiance can also be interpolated from precomputed values at particles' positions.

Irradiance Particles cannot be used in combination with global illumination and final gathering (the options are automatically adjusted on an attempt to do so). However, Irradiance Particles are compatible with caustic photons.

The following string options can be used to control irradiance particles.

- `"irradiance particles" <boolean>`: this string option enables the Irradiance Particles. The default is off.
- `"irradiance particles rays" <int>`: this string option controls the number of rays shot while estimating the irradiance. The number is similar to the number of rays used for final gathering, but specifies the maximum and delivers better quality compared to final gathering used with the same number of rays. The minimum is 2. The default value is 256.
- `"irradiance particles indirect passes" <int>`: the number of possible passes of indirect lighting. If this number is greater than 0, then a sequence of passes is computed to collect the irradiance coming from multiple indirect illumination bounces. Irradiance Particles would have both direct illumination and indirect illumination information in this case. If this number is 0, then Irradiance Particles will have only direct illumination information. The default value is 0.
- `"irradiance particles scale" <float> | <color>`: this is the global scale factor applied to the intensity of the irradiance during rendering. Values different from the default do not lead to a physically correct rendering but are useful for artistic purposes. A single value or a color may be specified. A single value is expanded to a color having the same R, G and B components. The default value is 1.0.
- `"irradiance particles interpolate" <int> | <string>`: this string option is used to control the use of interpolation. It can either be a numeric value or a string. `0` means no interpolation, `1` means interpolate always, `2` means interpolate only for secondary rays (that is, no interpolation for eye rays and interpolation for reflections, retractions, and so forth). Alternatively, strings `"never"`, `"always"` or `"secondary"` may be used. The default value is `"always"`.
- `"irradiance particles interppoints" <int>`: the number of Irradiance Particles used for the lookup interpolation. The default value is 64.
- `"irradiance particles env" <boolean>`: this flag enables the use of the environment maps for irradiance computation. In this case, a separate particle map is built for the environment (if an environment shader is present) and used during rendering for image based lighting. The default value is on.

- "irradiance particles env scale" <float> | <color>: this is a global scale factor applied to the irradiance contribution of the environment. Like the "irradiance particle scale" string option, a single scalar or a color may be specified. The scaling factor is relative because it applies to the environment irradiance only; the environment irradiance could be further scaled (multiplicatively) if the user specifies a global scaling factor with the "irradiance particle scale" string option. The default value 1.0.
- "irradiance particles env rays" <int>: the number of rays used for the computation of irradiance coming from the environment map. This number can be greater than the number of rays used for normal irradiance computation, especially if most of the environment is covered by scene geometry (typical case is a room with just one or two windows). For outdoor scenes, it should work fine with a smaller number of rays. Default value is equal to the number of rays specified with the "irradiance particles rays" string option.
- "irradiance particles file" <file_name_string>: specifies the file name for the Irradiance Particle map. If the file exists, mental ray will try to read the Irradiance Particle map from it (unless the "irradiance particles rebuild" string option is set on). If the Irradiance Particle map is not (successfully) read from the file, mental ray will compute it and save it to a file with the given name. This is consistent with the option for photon maps. The default value is no name, meaning that Irradiance Particle is not associated with a file.
- "irradiance particles rebuild" <boolean>: if set on, mental ray will compute the Irradiance Particle map even if a file with a name given with the previous string option exists. If it is off, then mental ray will either read it from file (if the "irradiance particles file" string option is used) or it will reuse the Irradiance Particle map coming from the previous frame rendered. The latter can be useful in animations, which are flicker-free. However, the Irradiance Particle map may lose quality if the objects and camera are moving, so this can actually be done only for fly-throughs. In addition, since the particle map is view dependent, inaccuracies might show up on the frame image borders. The default value is on.

  The following command line options are equivalents of the strings options listed above: -ip, -ip_rays, -ip_indirect_passes, -ip_scale, -ip_interpolate, -ip_interppoints, -ip_env, -ip_env_rays, -ip_env_scale, -ip_file, -ip_rebuild.
- Added Irradiance Particles entry to the mi_query mode miQ_STAGE.
- Fixed possible crash in evaluation if importance was close to zero.
- Fixed computation of surface translucency.
- Extended progress strings enum with Irradiance Particles entry.

Framebuffer image caching

- Framebuffer image caching. This option may save a significant amount of memory for rendering of high resolution images. If enabled, only a small fraction of each resulting image is present in memory: newly rendered tiles and tiles recently accessed by the display callbacks, output shaders, writing out final images into files and kernel in some diagnostic modes.
- Framebuffer image caching is superior to the virtual framebuffers (`-fb_virtual on` command line option), as, for virtual framebuffers, an output image could be memory mapped or unmapped as a whole; that is, the advantage is only present if multiple framebuffers are rendered.
- Framebuffer cache may be enabled with the command line option `-fb_virtual cache` or by setting the `miOptions::fb_virtual` value to `2` in the integration version. The type of the `miOptions::fb_virtual` has been changed to integer. Legal values are: `0` for in-memory framebuffers, `1` for memory-mapped (virtual) framebuffers, `2` for cached framebuffers.
- Note that output shaders operation on usage framebuffers should use `mi_img_*` functions and avoid `miIMG_ACCESS` and `miIMG_LINEACCESS` macro. Output shaders should also not access `image->height` and `image->width` directly, but use `mi_img_get_width` and `mi_img_get_height` functions instead.

New miscellaneous features

- The following registries are filled in by the mental ray kernel:
  - `{_MI_RAY_VERSION}` contains the mental ray version
  - `{_MI_RAY_PLATFORM}` contains the platform name
  - `{_MI_RAY_DATE}` contains the mental ray date
  - optional `{_MI_RAY_REVISION}`, if set, contains the internal repository version from which the executable has been build.
  - `{_MI_RAY_MAJOR_VERSION}` contains the mental ray major version (that is "3" for this release)
  - `{_MI_RAY_MINOR_VERSION}` contains mental ray minor version (that is "3.6" for this release).
  - `{_MI_RAY_BRANCH}` contains the branch version (for example, "3.6.50").
  - `{_MI_RAY_SYSTEM}` contains a short string for the current system class (for example, "windows").
  - `{_MI_RAY_OEM_ID}`, if set, contains an ID of the OEM partner.

  These registries may be used to setup multiple mental ray versions and platforms (like 32-bit vs. 64-bit shaders) on the same machine.

Feature improvements

New functions

- Added new function `mi_phen_call` and a set of macros
  - `mi_call_boolean(param)`

- `mi_call_integer(param)`
- `mi_call_scalar(param)`
- `mi_call_vector(param)`
- `mi_call_transform(param)`
- `mi_call_color(param)`
- `mi_call_tag(param)`

These functions and macros are equivalent to `mi_eval`, but do not use a caching mechanism. If a shader calls this function on a parameter which is connected, then all shaders in the sub-shader chain (only) will be forced to re-compute and the resulting value will be returned. The original cache is not affected nor destroyed by this function, meaning that any previous or following `mi_eval()` call on the same parameter will use the original cache. These functions and macros are handy for evaluation shaders with temporary modifications to miState variables, such as normal or texture coordinates.

- For raylib integrations: added two new functions `mi_disp_fbmap_name` and `mi_disp_fbunmap_name`. These functions may be used by DISP callbacks and are functionally equivalent to `mi_disp_fbmap` and `mi_disp_fbunmap`, but take a name of the framebuffer instead of the index as an argument.

BSP/BSP2 Acceleration

- New caching mechanism for building of the BSP2 tree with motion is enabled. The BSP2 construction time for large scenes is significantly reduced.
- Construction time for the BSP2 acceleration used in combination with motion blur has been decreased dramatically. In the current version, some additional memory is allocated temporarily for the construction process. The amount of that memory will be reduced in upcoming releases.
- Performance of the BSP2 in combination with motion blur has been improved.
- Improved overall BSP2 acceleration performance.
- For the BSP2 acceleration, improved parallelization if placeholder objects are used.
- Use more intuitive names in the BSP2 progress messages.
- For the BSP2 acceleration, significantly improved performance and reduced memory consumptions for scenes with motion.
- For the BSP acceleration, removed unused BSP memory control from the interface. If parsed from a `.mi` file, a warning is printed.

Framebuffers

- For the new framebuffer interface, added a treatment of compatibility-style file types with sign, like `"+rgba"`.
- Added `remove()` method to the `Framebuffer` class to allow removal of a framebuffer.
- For the Framebuffer interface class, added methods `get_buffercount` and `get_buffername`, which return the number of framebuffers and the name of a framebuffer with a specific index.

- For the framebuffers, added a new `autofill` flag. If set, the content of the framebuffer is initialized by the mental ray kernel according to the framebuffer type (i.e. color, normal, z depth etc.). The default value for the `autofill` flag is true. The `autofill` flag supercedes the `useprimary` flag effective for the color framebuffers only. The `useprimary` flag is still supported for compatibility, but integrators are encouraged to use the new `autofill` flag instead.
- For framebuffers, added a boolean `user` flag, because the name is more intuitive. Its semantic is the opposite of the `autofill` flag. If set, the content of the framebuffer is not filled by the mental ray kernel. The `autofill` flag can still be used as an antonym, but is deprecated.
- For the framebuffers, added a new `useprimary` keyword for backward compatibility with the C-style interface for color + Z files.
- For the framebuffer interface, deduce the file type from the image file extension if it has not been specified.

Miscellaneous

- Platform specific: on 32-bit linux platforms, compile using `_FILE_OFFSET_BITS=64` define in order to allow files large than 2Gb.
- For integrators: added a new function `mi_mi_parse_assembly`. This function accepts full `.mi` syntax and may be called for parsing of assembly files from *procedural assembly callbacks* only. The `mi_mi_parse` function accepts only a subset of the `.mi` syntax for the OEM raylib versions.
- Added a new string option `"importon emitted"`. It specifies the total number of importons to emit and overrides the `"importon density"` string option. The options are similar, but `"importon emitted"` specifies the value which is independent of the image resolution and thus is better for tuning.
- For applications integrating raylib: added a new type for progress message in the ambient occlusion computation pass.
- Added IPv6 support for mental ray multihosting rendering. In the `.rayhosts` file, as well as on the command line, IPv6 addresses may be specified. Host names and IPv4 addresses are still supported. IPv6 addresses need to be put into the `][ ]` brackets following the common URL convention. Note that if a slave host is specified on the command line, the `[ ]` brackets may need encapsulation depending on the shell used.

  mental ray master host tries to establish connections to slaves using either TCP or TCPv6 protocol, whatever is available and applicable to a slave host (if a slave host is given by name, both TCP and TCPv6 protocols are tried out, the order depends on operating system used and OS configuration).
- Added IPv6 support for the socket layer of mental ray DISP protocol (immediate display of tiles as they rendered). mental ray would listen for connections done with either TCP or TCPv6 on the same port id. The `imf_disp` tool would try to get the connection with any of the TCPv6 and TCP protocols.
- For ambient occlusion cache, decreased reaction time on the aborting of rendering.

- Do not print an API warning on a shader redeclaration if the redeclaration is done in the namespace scope, such as an "anonymous" namespace for an assembly or geometry shader. This may be used for including shader declarations to assembly files.
- For the `mi_geoshader_echo_tag` function, echo link statements if the `nolinkcommand` echo option has not been set.
- For the `mi::shader::Options::get(const char , int)` interface function, recognize the boolean option value and convert it to 0 or 1. This fixes a non-intuitive boolean option lookup failure if a variable of type `miBoolean` was used.
- For the architectural material shaders (`mia_material/mia_material_x`), significantly improved performance if the segment shadow mode is used.
- For OEM integrations of the mental ray library, file assemblies are enabled.
- For `ccmesh` objects with n-sided polygons (not only triangles and quads), the new fast subdivision surface implementation has been enabled. In addition to increased performance, the quality of texture mapping for such objects has been improved.
- Added new `mi_query` mode `miQ_PRI_ASSEMBLY_INST_TAG`. For a valid intersection state, the assembly leaf instance for the hit primitive is returned. If the geometry does not belong to an assembly, the null tag is returned. This query mode may be used by shaders to implement custom attribute inheritance from the main scene file.
- For OpenEXR output files containing *multiple* framebuffers, changed the channel naming scheme. The channels names used for color frame buffer `buffername` are `buffername.R`, `buffername.G`, `buffername.B`, `buffername.A`. For the primary framebuffer, the channel names `R`, `G`, `B` and `A` are still used.
- For scene echoing, the `object space` requirement for irrelevant declaration types has been removed.
- For the rasterizer, improved the statistics output.
- For the rasterizer, added transparency/opacity handling to "userbuffers" (i.e. non-primary color buffers). In the new framebuffer API, the `useopacity` boolean modifier can be set on a framebuffer. For the integrated version, the `"useopacity"` attribute can be used.
- It is also possible to use the boolean string command `"rast useopacity"` or the command line option `-rast_useopacity` to force the behavior for all color framebuffers. This way the new feature may be used in combination with the C-style framebuffer API.
- For free form surfaces, the performance of derivative evaluation has been improved.
- Added some non-null checks to the `mi_shader_if.h` headers. The change is relevant only for products built without the rendering component, not for mental ray and raylib.

Limitations

Hardware renderer limitation with displaying mental ray area lights

When displaying mental ray area lights, the hardware renderer does not provide the same results as an actual render, but instead produces a very rough approximation that can be viewed in High Quality mode.

The Maya hardware renderer supports the following configuration for mental ray area lights:

- A Maya area light must be used.
- Only the Rectangle mental ray area light shape is supported.
- Basic light parameters such as Color, Intensity, Decay Rate are supported.
- An approximation of shadows is supported.
- A fixed point sampling rate is used for both diffuse and specular highlights. Sampling artifacts may occur, especially for specular highlights where the area of the light is large and the light is close to the surface.
- Other light shapes and options such as High Samples, High Sample Limit and Low Samples are not supported.