RenderMan 22.0

Star Wars: The Last Jedi

Welcome to RenderMan 22.0!

Welcome to RenderMan 22. This release introduces improvements to the previous RenderMan in very significant ways.

Please dive right into the release notes below for more detailed information on the latest version of your favorite renderer!

New Features in 22.0

Revolutionary New Workflow

RenderMan 22 brings truly interactive rendering that works with the artist from concept to completion.

Artists can continuously render while modeling, animating, texturing, shading and more! Previously artists may have had to restart renders to see their changes to their scene during live or interactive rendering. Many of these roadblocks have been removed so that you can continue to work while seeing the image refine. Importing and exporting assets are now visible in-render as are changes to geometry while modeling, updates to animation, object placement, material and light parameter tweaks, even light path expression authoring!

Examples include:

- Rendering playblasts in RenderMan for client approval on animation
- Placing textures interactively
- Refining light position and parameters
- Authoring Light Path Expressions to capture the right output for post enhancements
- Importing and replacing assets while rendering
- Grooming hair for the perfect look
- Trying new camera angles or framing for the perfect composition

True Interactive Rendering — Rely on RenderMan like never before with the ability to continuously render while you work. We no longer use the intermediate step of rendering to RIB in interactive sessions. See changes immediately and make decisions sooner while refining your artistic choices. RenderMan has the ability to update while artists model scenes, complete layout, perform look development, and much more. We've removed many of the restrictions that require users restart a render or wait for feedback.

All New Curve Representation — We introduce new curve rendering techniques designed around ray tracing large amounts of curves while editing grooms and looks without needing to re-render to see changes. Now you can comb, style, and dye without having to wait!

Lighting Improvements — Changes to our light selection techniques improve convergence on scenes with many lights and many types of lights at the same time. Move lights and change parameters with immediate results.

UI Enhancements — Bridge products enjoy streamlined workflows to enhance our interactive rendering capabilities. Improved exposure of RenderMan Attributes and Options help users make decisions and see results at all stages of production.

Rewritten RenderMan for Bridge Products — RenderMan for Maya and Katana have been redesigned. Better integration means less looking for options, fewer clicks, and faster images. Render directly to Viewport 2.0 in Maya and see your changes as they happen. Time to first pixel is greatly reduced and your work is uninterrupted.

A New Ray Tracing Core — To compliment our interactive always-on workflow, we've rewritten the ray tracing core to improve speed and quality. Scenes interact quickly and converge faster than ever before!

Opacity from Presence — We now support partial presence values (values between 0 and 1) for true opacity. This changes how we calculate presence on *camera rays* and results in noise-free partially opaque effects up to a user-specified depth in the integrator.

OpenVDB Updated — PxrVolume provides more sampling options to enhance performance as well as new support for OpenVDB 4.0.

Improved Sampling Stratification – This improves noise reduction, time to converge, and preview renders. Please see the paper on Progressive Multi-Jittered Sample Sequences.

New Adaptive Sampler Option – In addition to the new stratification, we include a new experimental adaptive sampling technique as an option. You may find better convergence at higher Variance thresholds.

Important Differences

- To achieve interactive rendering including geometry edits, RenderMan 22 has a high performance interface for bridge products. However, the Ri interface is still supported, including Ri Procedurals. In particular, Ri editing features are not supported anymore.
- **RtToken has been replaced**, any string that is passed to the renderer must be a RtUString. Any customer code will need to implement this change.
- Look differences will happen between older layered PxrSurface materials and RenderMan 22, especially subsurface scattering. This is due to the improved layering logic.
- Lights are no longer visible by default in refraction, you must use "int visibleInRefractionPath" [1]
- The default coordinate system known as **"current"** has changed. **"Current"** space is context dependent and subject to change
    - Displacement is changeable from Object Space (current space) or World Space in shaders, it will not be possible to exactly match previous displacement results.
    - Bxdf is in World Space
    - Analytic Lights are in Object Space
    - Light Filters are in World Space
    - Integrators are in World Space
    - Sample Filters are in World Space
- All geometry is converted to instance geometry, i.e. everything is now an instance within the core of the renderer
- Attribute identifiers are now 64-bit integer instead of float
- Note that the light cone angle parameter on lights has been corrected to use the correct measurement in degrees from the edges. This will alter the look of older scenes. To match older scenes there's a bit of math involved: newTheta = 90 - atan(2.tan(90 - oldTheta)) (Note that this is radians)
- We do not include the librix standalone library, RixInterfaces is only available through libprman.
- Always camera-facing curves are not supported anymore. Curves without explicit normals are now rendered as cylinders.
- Dtex format is deprecated, use Deep EXR instead and will be removed in the near future.
- Some previous materials have been deprecated or removed
    - LM Materials
    - PxrGlass
    - PxrSkin
- The PxrUPBP integrator has been removed.

There have also been *many* subtractions designed to improve the user experience through simplification and streamline performance options. In many cases they were a duplicate function and valid options and attributes can be found in the Developer Section of the documentation. These changes may impact your scripts, plugins, pipeline, and more.

Miscellaneous Changes

- A change in the display driver channel order now outputs **RGBA instead of ARGB**
- The default (current) coordinate space is different. We now render in world space. Previously this was all in camera space. This affects features like the P AOV
- RenderMan can now use more than 64 logical cores on Windows
- Many shading operations are now in object space (displacement is particularly affected and will likely require lookdev or shader changes)
- Any string that is passed to the renderer must be a "UString" or "unique string". Refer to the RtUString Class in the Developer's Doxygen Guide
- Curves with normals are supported, as are round curves. But curves with no normals as a notation for desiring always camera-facing curves is not supported. as such, orientation and reverse orientation for curves is no longer supported
- For subdivision surfaces, face edits higher than depth 1 are not supported
- dPdTime and dPdCameraTime have changed
- All functions in librx have been removed, including RxGetContext
- RixTransform API is deprecated and will be removed soon.
- Added circular bucket/tile order
- A new InvalidateTexture call has been added to RixRiCtl.
- A new mechanism has been added to RixRiCtl to cancel a render
- Default cache limits have been increased to better reflect modern machines
    - Texturememory and Geocachememory limits are now 4194304
    - Ptexturememory limit is now 32768
    - Opacitycachememory is 2097152

- Added motion blurred FOV capabilities and a detail modifier to the PxrCamera plugin

- Camera options such as depth of field and FOV must be specified through a projection plugin.
- Alpha and RGBA color sets are now supported. RGBA color sets are emitted as "color colorSetName" and "float colorSetNameA" to allow for ease of shader binding.
- Increased default precision for RIB export to 9
- Added Attribute "displacementbound" "int offscreen", please see the Primitive Variables section for an explanation
- Improved adaptive sampling when using checkpointing
- JIT compilation of OSL shaders now executes in parallel
- Added support for interactive arbitrary clipping planes
- Volume instancing is fully supported, however, density (and motion blur) cannot be altered per instance.
- Removed the PxrRollingShutter plugin, the functionality exists in PxrCamera
- PxrHairColor allows for either rec709/sRGB or ACEScg for Physical Mode
- Improved texture filtering results, we improved mipmapping level selection at grazing angles.
- Crop windows are no longer limited to the (0,0) to (1,1) square. Going outside the unit square will add overscan
- Cryptomattes now respect the crop window rather than adding black padding. Now they are always autocropped which may reduce size further.
- Cryptomatte now defaults to scanline EXR (ZIPS) output. This improves performance in Nuke at the cost of larger files.
- PxrCryptomatte now supports checkpointing
- Improved transparency handling in PxrCryptomatte
- Improvements in point rendering, more speed and less memory usage reported in many scenes. Note point mesh lights are not supported and points cannot be used as closed volumes
- Improved shadow tracing performance, especially on scenes where lights may not contribute to a point being shaded
- Alembic geometry is treated as instances when "-instanced" flag is specified
- Better handling of OpenVDB containers with zero density
- PxrSurface can now report its IOR when using nested dielectrics, removing the need for users to manually specify the attribute. The attribute is deprecated.
- Bump Adjust Amount is now defaulted to 1 (it was 0) to remove possible artifacts preemptively when a normal is below the horizon
- Bump mapping has been improved to avoid light leaking
- The exrinfo utility now automatically truncates the display of very long strings from the image header. Use its new -full option to show these in full without truncation
- The "-d type" command line option now overrides the display type for all displays, not just the primary one
- Improved the calibration of timing statistics on Linux
- Methods in the the RixRNG.h header have been changed to be more typesafe
- Various fixes to volume rendering in 22 will result in different looks in volume rendering from pervious versions, they may appear (much) brighter depending on lighting circumstance.
- Added a new Primvar for points, falloffpower, float. I f not supplied, or set to zero, the points will have a hard edge. A value of 1 is a "reasonable" value that emulates the usual cosine based falloff; this will likely be the goto value for most people doing volumetric particle effects. Values between 0 and 1 makes the falloff faster, eroding the point faster - point has "less presence". Values higher than 1 (up to infinity) makes the falloff slower to the point of being non-existent.
- Defaulted the following patterns to Object Space:
    - PxrPrimvar
    - PxrVariable
    - PxrManifold3DN
    - PxrRoundCube
    - PxrManifold3D
- Introduced a new Hider option: "int sampleoffset". This allows several images to be rendered in parallel (with different sampleoffset) and then combined. (Essentially: Users can separate a single render across multiple nodes to share the work and then combine them when complete using a post process.)
    - With non-adaptive sampling: Let's say you render four images with 256 samples each, with sampleoffsets 0, 256, 512, and 768. If you combine those four images, you'll get exactly the same image as if you had rendered a single image with 1024 samples.
    - With adaptive sampling: Let's say you again render four images, each with "maxsamples" 256, with sampleoffsets 0, 256, 512, and 768. Let's say that due to adaptive sampling, some given pixel only gets 64 pixel samples in each of the four images. Then the combined image has been rendered with sample numbers 0-63, 256-319, 512-575, and 768-831. Due to the stratification of the samples, this is not quite as good as if you had rendered a single image with 256 consecutive samples. (However, it is, of course, quite a bit better than rendering a single image with only 64 samples.)
- Support lights in procedurals. This change allows for:
    - lights to be inside of groups and obey the usual instancing rules (one light in a single group master will yield multiple 'lights' when the group is instanced multiple times)
    - lights to be inside procedurals
    - lights to be inside ObjectBegin/ObjectEnd blocks, included nested ones

Known Rendering Differences

- Hair will have more contrast between lit and unlit areas as an incorrect highlight was corrected in 22
- Incorrect endcaps were fixed on hair and removed for tubes. This has the effect of making the curve slightly shorter by a factor of nearly **curveWidth/2** at both the starting point and endpoint of the curve. This makes a bigger difference to short and fat curves
- High frequency bump maps may be different as an issue with filtering was corrected
- Change from "camera" to "world" space changes anisotropy, this should be more correct, objects with specified anisotropic direction should be unaffected
- Mesh lights are half as bright as double contribution was fixed
- Particle Ng was incorrect in previous versions, this alters their illumination which is now correct
- All lights now default to off for Visible in Refraction (initial releases)
- Mesh lights illuminate now even if invisible to all ray types

Known Limitations

Interactive/Live Rendering Limitations

- Crop window edits are restricted to fall inside the original crop window
- Bucket order or size cannot be changed during live rendering
- min and max samples settings cannot be altered during live rendering
- Changes to Presence do not update when using the opacity cache option
- Motion Blur will disappear during interactive rendering with scene changes
- Displacement does not update on changes
- Objects are not re-diced during interactive sessions
- Mesh lights cannot be interchanged as geometry without restart.

RenderMan Pro Server

- PxrUnified integrator is currently experimental as it does not yet support all the standard rendering features.
- Meshlights cannot be instanced
- Load on demand procedurals are not supported anymore, all procedurals are now loaded immediately
- We do not read point data from OpenVDB files
- PxrSurface back diffuse color is not output to the albedo color AOV
- Analytical lights placed inside volumes may yield artifacts when made visible to the camera. As a work around, the light camera visibility should be turned off, and a geometry with a similar shape should be used (visible to camera, invisible to transmission and indirect rays), with the proper emissive bxdf.
- Using the '**.**' character in the handle for an OSL shader could cause unpredictable results during re-rendering.
- Per-Instance baking is not supported, only the reference instance.
- 3d baking: no direct bake-to-ptex support.
- PxrBakePointCloud cannot directly render ptex.
- Sample/Display filter plug-ins do not have access to lighting services for light dependent effects, e.g. lens flare.
- Adding new mesh light on existing geometry during IPR results in double geometry.
- Motion blurred polygons do not motion blur normals when deformed. Use Subdivision meshes instead.
- PxrUPBP is no longer supported
- When attempting to access an array primvar, you must first check the size of the array primvar and allocate the appropriate space. Not doing so may lead to a crash.
- Points and curves cannot be used as geometric lights.
- Deformation motion blurred volumes don't currently work with densityFloatPrimVar or densityColorPrimVar. You will need to use a PxrPrimVar node connected to densityFloat and densityColor instead.

The Centos KDE style "Oxygen" installs a version of Qt and sets the user's environment variable QT_PLUGIN_PATH forcing "it" and LocalQueue to attempt to load an incompatible Qt library. Either avoid installing the Oxygen theme or unset QT_PLUGIN_PATH before running "it" or LocalQueue. Other KDE styles may also install this theme.