Release Notes

Ver 6.20.07, API Version 4.04.00

Date - Apr 11, 2024

Highlights

- Partial USD Support (see below).
- GeomScatter plugin and GeomUtils.readScatterData() function.
- New VFB Layer Manger API.
- Added Python 3.11 binding.
- Added Node.js 18, Electron v23 and Electron v27 binding.
- Updated copyPlugins between renderers API (see below).

Features

- USD support in "read-only" mode via the VRayScene plugin,
  Experimental, partial support, subject to change:
  USD files can be loaded with VRayRenderer.load/loadScene(filename).
- Updated Util.copyPlugins():
    - Added new signatures with more informative pre- and post-create callbacks;
    - Added new signatures which allow copying plugins of a specified type;
    - Deprecated the function with the old signatures;
    - Bug fixing.
- [Python] Added a new TransformList type (implemented as a dynamic array of Transforms). Used it in GeomUtils.readScatterData()
- [Node.js] Added a new non-resizable TransformList type backed-up by FloatList/Float32Array (accessible via the 'floatList' property).
  Used it in GeomUtils.readScatterData()
- [Python] Added a new PluginList type (a dynamic array of Plugin instances).
  Now getInstnces() returns a PluginList instead of python list.
- [Python] Added missing equality comparison to Plugin classes and categories.
  (e.g. plugin.getClass() == renderer.classes.Node would work correctly now)
- [Node.js] Added a new array-like resizable PluginList type, a descendent of Array. Now getInstnces() returns a PluginList instead of an array of plugins but this should be transparent to the user in most cases.
- Read OpenColorIO data via OCIOConfigurationData type.
- [C++, .Net] Added a VRayProfilerWrite callback/event for profiler output.
- [.Net, Node.js] Added missing UIGuides string to propertyMeta.
- Added methods/properties to VRayRenderer for enabling the Visual Debugger.
- New auto-exposure and auto white balance algorithms implemented in the SettingsCamera plugin.
- VFB zoom camera control with keyboard and mouse now works with CameraPhysical as well.
- Added examples for vrscene export and pack assets, vrdata and vrfiles.
- Exposed flags in RendererOptions for showing the VFB Pause, Update IPR, CopyToHostFrameBuffer, AddLightMixREToScene and AddDenoiserREToScene buttons. Added the ability to set user callbacks/event handlers for them.
- Removed padding between BitmapInfoHeader and actual bitmap pixel data in in-memory BMPs created by compress/convertToBmp() function. Now the result is the same as directly writing to a file. Also in-memory BMPs are now bottom-top with positive height instead of top-bottom with negative height.
- Added GeomScatter, GeomEnmesh, Create-proxy-with-instancer-data and V-Ray Profiler examples.
- [C++] Added PluginMeta.isValid() method.
- Added an example to list GPU support of the V-Ray Plugin types and their parameters.
- [C++] Added PluginTypeId. Now in addition to a name string, plugin types can be referenced by an integer TypeId as well.
- [C++, .Net] Updated the ScenePreview class:
    - Added GeomInstancer data type,
    - Added getInstancerParticles() and a Particle struct,
    - Added getWorldBoundingBox() for all Object types,
    - Added getDecalSize() and getClipperSize().
- [.Net] VRayRenderer.GetPlugin<T> now works with any parent type of the actual plugin type or interface implemented by the plugin.

- [.Net] IPluginRef<T> interface now implements IPluginRef thus making it implicitly convertible to IPluginRef.
- [.Net] Added two Box.Expand() methods to make the Box struct consistent with C++.
- Added VRayRenderer.isFrameSkipped() helper function to check if the user has clicked on the Stop button in VFB while rendering animation.

Bugfixes

- Fixed a bug in consecutive calls for incremental vrscene file export where if any of the calls had a non-empty list of explicit plugins, the list wasn't cleared for the next export.
- [C++] Updated VRayServerInit to use VRAY_APPSDK_BIN and VRAY_SDK environment variables to locate VRaySDKLibrary and V-Ray binary files.

Ver 6.10.07, API Version 4.02.00

Date - Apr 26, 2023

Highlights

- Added new functionality to copy plugins from one renderer to another (or to the same one). See below.

Features

- Added Util.copyPlugins() smart and versatile helper function with support for optional include and/or exclude lists and pre- and post-create callbacks. Added examples.
- VFB Debug shading (limited functionality)(WIP): Global debug shading modes and Isolate selected mode for Node plugins and Lights.
- Updated V-Ray Profiler: Added custom metadata support and the ability to generate html files.
- Added localization (vfbLanguage) support.
- Added VRayImage.load(filename) and VRayImage.loadSize(filename) static methods to create a VRayImage from file or get the image dimensions contained in a file.
- Added VRayImage.resize() method that suppercedes the old VRayImage.downscale(). fitIn(), fitOut() and cutIn() methods adapted to perform both down- and up-scale.
- Added VRayImage.mulColor() method.
- [Node.js] VRayRenderer, VRayServer or ScannedMaterialInfo will now throw if accessed after being closed instead of doing nothing.
- [C++, .Net] Added asynchronous VRayRenderer.abort() method.
- [C++] Added macros indicating V-Ray and API versions.
- [Python] Changed the buffer/memoryview of typed lists (IntList, FloatList, ColorList and VectorList) to be a simple byte array because it is much more useful in exporters (file readers/writers, etc.). memoryview.cast() can be used if other types are needed.
- [Node.js] Added VRayRenderer.keepAlive() function to prevent the renderer from premature garbage collection in some rare cases.
- Added options for exporting .vrdata and .vrfiles
- [C++, .Net] Added Box.expand() methods.
- Allowed acquiring compute devices lists without a renderer instance.
- Added the "Test Resolution" function of VFB.
- Additional Color and AColor methos implemented.
- [C++, .Net] Added PropertyMeta.getElementType().
- Made plugin list properties in .Net more strictly typed.
- [TypeScript] Fixed auto-generated plugin property types.
- [C++, .Net] Added VRayRenderer.abort() method.

V-Ray Proxies support

- Added numFrames / hasVelocityChannel / hairVelocities / particleVelocities to ProxyReadData.
- VRayRenderer.createProxyFile() extended with support for instancer types and Node as input.

Experimental (subject to changes)

- Added VRayImage.calculateWhiteBalanceMultiplier() method to allow alternative white balance corrections as image post-processing.

Bugfixes

- [Node.js] Fixed a serious bug which resulted in a crash if async stuff (events, callbacks) were used in a worker thread!
- [Python] Fixed a bug in the slice getter of typed list (IntList, FloatList, ColorList and VectorList) with step!=1 where instead of copying the items from the source to the resultant list, items were copied back to the source thus destroying it.
  (Note that slice setters and bulk delete are not implemented yet.)

Ver 6.00.30, API Version 4.00.00

Date - 2 Nov, 2022

Highlights

- New easier to use Animation API
- Added Electron v16 binding
- Added Python 3.10 binding
- Added V-Ray Profiler

Features

- All language bindings now have Plugin property setter and getter methods with time parameter to easily set animation frames. The old API works as well
- If VRAY_APPSDK_BIN environment variable is set, use it to locate VRaySDKLibrary and V-Ray binary files (Then fallback to %VRAY_SDK%\bin and finally fallback to OS paths.)
- Added stripAlpha and flipImage parameters to VRayRenderer.getImage()
- Plugin property default values are now returned as native types instead of strings in meta info in the corresponding languages
- Added the ability to log messages into VFB
- Added getAliases() to PropertyMeta and PropertyRuntimeMeta that return property alias names
- Added Matrix(Vector) constructor to easily construct diagonal matrices
- [Node.js] Plugin classes can now be compared with operators == and === (previously only plugin instances could be compared that way)
- [C++] Implemented equality and inequality operators of Value, Color and AColor
- [Python, Node.js] Implemented single scripts to load binding modules
- [Node.js] Added for-of loop iterator to VRayRenderer.classes
- [Python] Changed Plugin string representation. Now it consists of "name", "type" and "value" properties and the string can be eval-ed
- Added VRayImage.toIntList() method and examples that convert BitmapBuffer to RawBitmapBuffer in vrscene files that use it
- Added toSRGB() and fromSRGB() helper functions to Color and AColor
- [GPU Device selection] Low priority can now be set for each device
- Added new functions to select devices to be used for denoising
- Added static version of getIESPrescribedIntensity to Util namespace
- Changed the default value of progressiveImageUpdatedHasBuffer and bucketReadyHasBuffer default value from true to false. Now ProgressiveImageUpdated and BucketReady events/callback won't contain images by default. You can use VRayRenderer.getImage() inside the callbacks (now this is recomended) or set the corresponding flag to restore old behavior
- Added VRayRenderer.setVRayProfiler(VRayProfilerSettings) function

Ver 5.20.20, API Version 3.10.00

Date - 3 Feb, 2022

Highlights

- Native Arm M1 and Mac OS BigSur support
- Added the Material Library
- New UV Texture Sampler
- Added Node.js v14, v16 and Electron v11 bindings. All are now context aware
- Added Python 3.9 binding

Features

- If VRAY_SDK environment variable is set, %VRAY_SDK%\bin is used to locate VRaySDKLibrary and V-Ray binary files by default
- Added BinUserAttributesWriter
- [Node.js] Added for-of loop iterator to VRayRenderer.plugins
- [Node.js, Electron] Binding modules are now context aware
- Added VRayRenderer.waitForVFBClosed() function
- Updated ScannedMaterialParams and ScannedMaterialPresets; removed their setDefaults() and reset() methods
- Added a confirmation dialogue for the VFB 'Clear image' action as well as functions to enable/disable it and get its state
- Added clearPropertyValuesUpToTime(double time, PluginCategories categories)
- [C++] Added Plugin value getters to the optional V-Ray SDK interoperability layer
- Added vfb.setProgressTextAndValue()
- [C++, .Net] Added setGUIMessageProcessing() to allow disabling internal VFB message pumping
- Added VRayRenderer::clearPropertyValuesUpToTime(double time, PluginCategories categories)
- Added VRayRenderer::getChangeIndex()
- Added useVfbLog flag to RendererOptions
- Added RenderElement::getMetadata()
- UV Texture Sampler: [C++ and C#] Added a new UVTextureSampler class. It can be used for sampling VRay texture plugins at different u-v points. Also the sampling does not require a rendering license as it does not start the whole rendering process. The sampler class is lightweight and fast to set up but note that it might not always return satisfying results when used with more complex texture plugins.
- Improved V-Ray Proxies support:
    - Export hair and particle voxels from Plugin and MeshFileData input,
    - Read and export the edge_visibility data and the voxel info channel to a mesh file,
    - Faster execution speed of readFullData().

Version 5.10.01, API Version 3.00.00

Date - 5 Mar, 2021

Highlights

* New V-Ray VFB (codenamed VFB2).
* Share VRayRenderer instances among programming languages.
* [C++] Added move semantics to the Value type and the Plugin property setters.
C++11 is now required to use AppSDK in C++.
* Added Python 3.8 binding.
* Added .Net Core builds.

Features

* [VFB2] Added functions to get/set VFB2 settings as JSON.
* [VFB2] Added an update notification event/callback to VRayRenderer.
* [VFB2] Added LightMix Transfer To Scene event/callback to VRayRenderer.
* [VFB2] Added a flag to disable the VFB2 Transfer To Scene button to RendererOptions.
* [VFB2] Window handle functions:
- Added getNativeHandle(), getQWidget(), setNativeParentWindow(),
setQWidgetParent() and getMinimumSize();
- Marked getWindowHandle() and setParentWindow() as deprecated/obsolete.
* Added the ability to share the same VRayRenderer instance among different

programming languages:
- Added VRayRenderer.getInstanceHandle() method
(InstanceHandle read-only property in C#);
- Added VRayRenderer(instanceHandle) constructor.
* Added new checkDRHost(hostString) function that checks whether a VRayServer instance
is listening on a given host/port and returns info about it if it is.
* Filter the available CUDA and Optix devices in their corresponding get and set
functions;
* Exposed CUDA Compute Capability info.
* Added vfb.loadBackgroundImage() to load VFB CC background image.
* [Python] Completely removed old list wrapper type.
* denoiseNow() now accepts an optional "ready" callback function.
* [C++, .Net] Added two new methods get/setRenderSizes which get/set all ImageSize,
CropRegion and RenderRegion parameters and the state of the VFB region button state.
* Added VFB::clearImage() and VFB::resetState() methods.
* [C++] Added overloads with move semantics to loadAsText/appendAsText functions.
* [.Net] Added ToString() to Plugin and PluginRef.
* [.Net] Added GetHashCode() and Equals() methods, and operator== to PluginRef and PluginLink.
* Disallowed invalid characters in plugin names; Added allowInvalidCharactersInPluginNames
flag to RendererOptions to force old behavior and allow them.
* [.Net] Throw exception if creating a new plugin fails. In most cases it would happen
because of the new bad plugin name checks.
* [Node.js, Python] Changed vfb.saveImage to use the default ImageWriterOptions
if the user hasn't passed any options at all as it is in C++ and .Net
* Changed ImageWriterOptions in all languages to always set IWI_MULTI_CHANNEL
(i.e. produce a single multi-channel file if the file format supports it)
if the user hasn't passed any options at all as it is in C++ and .Net
* [Node.js, Python, Windows] Fixed setSDKLibraryPath() in Node.js and Python didn't work
with relative paths. Now it works with both absolute and relative paths.
Also full VRaySDKLibrary path is now returned by getSDKLibraryErrorString()
if the file has been found and loaded.
* Removed gpuPluginPath member from RendererOptions.
* [C++, .Net] Added callbacks for color corrections and compositing layer changes in VFB.
* [C++] Added the missing setValue(double) and getDouble methods to Plugin class.
* Added returning default values of TYPE_TEXTURE* plugin property types.
* Added Object Select render element.
* Added cameraName setters/getters to easily switch cameras which has the scene_name
property.
* [.Net] Replaced VRayImage.ToBitmapImage() with VRayImage.ToBitmapStream() in order
to avoid GDI+ dependency.
* Changed the current frame and time logic when rendering a sequence. Now the current
frame and time are changed before the wait for sequence continue is started.
VRayRenderer::getCurrentTime() and VRayRenderer::getCurrentFrame() now return
the time or frame number of the next frame that will be rendered in the sequence.
Also aborting the renderer while it is waiting to continue the sequence will
leave the next frame number as current time of the renderer.

Improved V-Ray Proxies support

* readFullData() and readPreviewData() added reading of the following data channels:
MeshObjectInfo, HairObjectInfo, ParticleObjectInfo, Hair geometry, Particle geometry,
GeometryVoxelsBBoxes, HairVoxelsBBoxes, ParticleVoxelsBBoxes
* New interfaces with ProxyReadParams input (which don't require creating a VRayRenderer)
for readFullData(), readPreviewData(), readPreviewGeometry(), readPreviewHair() and
readPreviewParticles()
* New interfaces with MeshFileData input (which don't require creating a VRayRenderer)
for createMeshFile() and createCombinedMeshFile()
* New interfaces with AnimatedTransform input for createMeshFile() and createCombinedMeshFile()
(transformations at corresponding keyframes which are written to the created file)
* readFullData() and readPreviewData() now contain start indices data,
indicating where the data belonging to the individual geometry / hair / particle voxel starts.
* readFullData() and readPreviewData() now return ObjectInfo containing the range of voxels
included in the corresponding geometry / hair / particle object.

Bugfixes

* [C++] Made callback setters and callback execution thread-safe.
* Fixed reading heterogeneous lists at different times when they were set from code.
* Fixed "progressiveImageUpdated" events not fired for lightcache updates in bucket

mode with render region set.
* When possible, preserve the type of plugin property numerical values
(bool/int/float/double) when they are set and read back from code.
* [.Net] Fixed a bug in Matrix.Transpose()
* [.Net] Fixed wrong output sizes of output downscaled images returned from some methods.