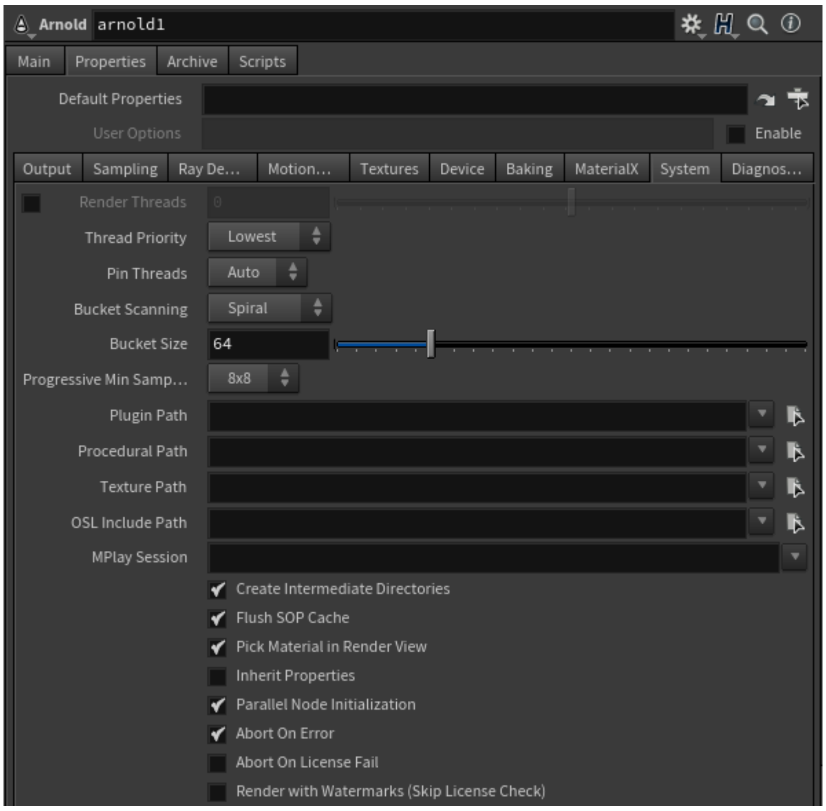
System


Arnold arnold1

Main | Properties | Archive | Scripts

Default Properties

User Options ☐ Enable

Output | Sampling | Ray De... | Motion... | Textures | Device | Baking | MaterialX | System | Diagnos...

☐ Render Threads 0
Thread Priority: Lowest
Pin Threads: Auto
Bucket Scanning: Spiral
Bucket Size: 64
Progressive Min Samp...: 8x8
Plugin Path
Procedural Path
Texture Path
OSL Include Path
MPlay Session
✓ Create Intermediate Directories
✓ Flush SOP Cache
✓ Pick Material in Render View
☐ Inherit Properties
✓ Parallel Node Initialization
✓ Abort On Error
☐ Abort On License Fail
☐ Render with Watermarks (Skip License Check)

Render Threads

The number of threads used for rendering. Set it to zero to autodetect and use as many threads as cores in the system. Negative values indicate how many cores not to use, so that -3, for instance, will use 29 threads on a 32 logical core machine. Negative values are useful when you want to reserve some of the CPU for non-Arnold tasks.

Negative numbers are also allowed. If specifying 0 threads means use all cores on a machine, then negative numbers can mean use all but that many cores. For example, threads=-2 means use all but 2 cores, while threads=2 means only use 2 cores. This is useful when you want to leave one or two cores for other tasks. One example of this is so that Houdini can be more responsive while Arnold is rendering in the Render View.

Thread Priority

Adjusts the priority of the render threads.

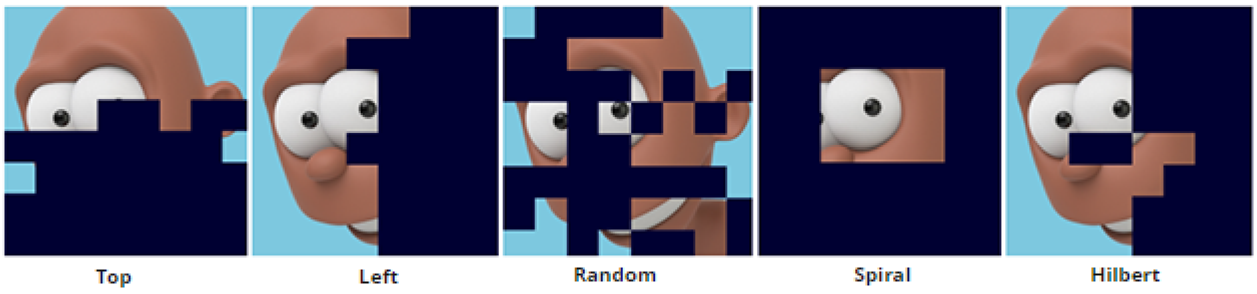
Pin Threads

Arnold can pin threads on Linux, so they don't jump between multiple processors. This can improve scalability in modern machines, multiple processors. It can be set to *off, on*, or *auto*. By default is set to *auto*, meaning that if the number of threads is more than half the number of logical cores on the machine, Arnold will pin the threads.

If client code, for instance, a custom shader, spawn their own threads manually (with pthread_create or similar), these threads will inherit the same thread affinity, which totally breaks the point of spawning threads; in these situations, they can either set options.pin_threads to off, or they can create their threads with the Arnold API AiThreadCreate() which will un-pin the created thread.

Bucket Scanning

Specifies the spatial order in which the image buckets (i.e. threads) will be processed. By default, buckets start in the center of the image and proceed outwards in a spiral pattern.

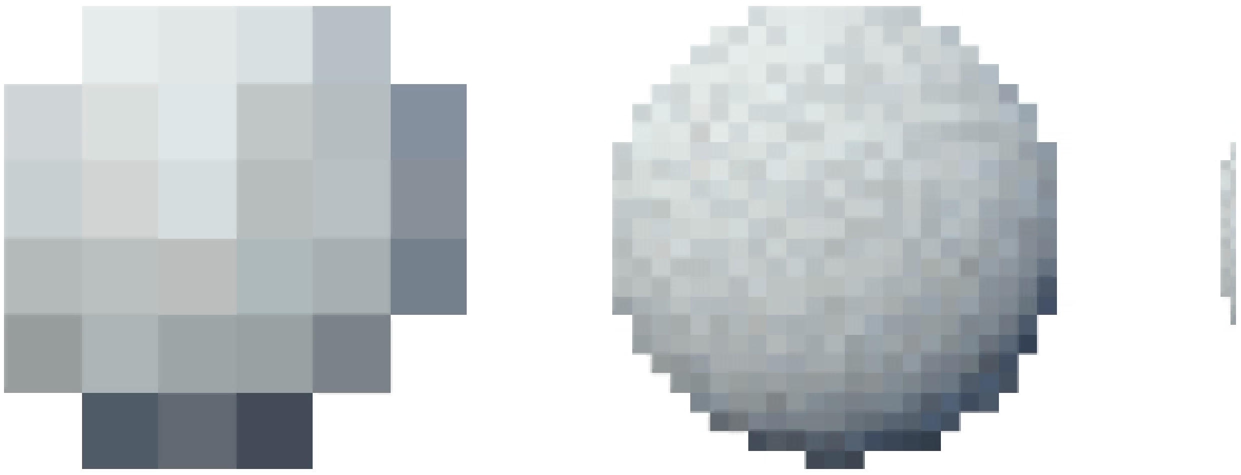


Bucket Size

The size of the image buckets. The default size is 64x64 pixels, which is a good compromise; bigger buckets use more memory, while smaller buckets may perform redundant computations and filtering and thus render slower but give initial faster feedback.

Progressive AA

The initial AA samples used for the first progressive render. Negative values sub-sample the render, allowing faster feedback in the render window.



Plugin Path

Defines a location to search for plugins such as shaders, procedural plugins that create new node types, and volume plugins.

For example, if you load ass files from a different plugin, you may need to set the *plugin_searchpath* to load plugin-specific shaders (such as *mtoa_shaders*).

Procedural Path

Defines a location to search for procedural nodes that load ass files (or obj or ply files).

Texture Path

Defines a location to search for textures.

OSL Include Path

Allows additional include search paths to be provided to OSL when compiling shaders. This option is a single string that can contain multiple search paths separated by a colon (`:`) or a semi-colon (`;`).

MPlay Session

Specify an MPlay Session label (useful with multiple workspaces).

Create Intermediate Directories

Option to create intermediate directories for output images and ASS files.

Flush SOP Cache

Flush SOP cache after each frame to reclaim memory.

Pick Material in Render View

You can use Ctrl+LMB in the *Render View* to pop up a panel with the material that shaded the pixel that was clicked.

Inherit Properties

Enable properties inheritance for geometry at the expense of translation time.

Parallel Node Initialization

Initialization and update of scene nodes can be multithreaded to lower the scene preparation time in complex scenes with many objects, shaders or lights.

Bucket Work Sharing

Share work among all threads for the last remaining buckets.

Abort On Error

Rendering will abort as soon as an error is detected. This is the recommended setting. In general, you should not ignore important error messages, or you'll risk crashes, rendering artifacts, and undefined behavior.

Abort On License Fail

If set, rendering will stop if the license is not detected when the render begins.

Render with Watermarks (Skip License Check)

Switch this on to avoid taking an Arnold license from the license server. This will always render your image with a watermark.

Color Management

A color manager is a connection between Arnold and an external color management library like OpenColorIO or synColor. Color managers hold information about the availability of different color spaces and also transform RGB colors to and from the rendering color space.

Auto Color Space

You can set the input color space or the output driver's color space to *auto*. If you do this, Arnold will use whatever you have set in the *Auto Color Space* option.

If it exists in the OCIO config, this should be set to the name of the 'sRGB Gamma' color space. This is used internally for input and output color spaces in '**auto**' mode. If set by the user, this color space is also used as a reference to detect the rendering color space gamut and white point.

Rendering Color Space

Using specific implementations of a color manager it is possible to specify Arnold's rendering color space. Arnold's default rendering color space is Linear sRGB. For other color spaces, Arnold will try to auto-detect chromaticities and illuminant, and will expect all data (textures and .ass file parameters) in that same color space. An important note is that different rendering color spaces will yield different render results when compared with the default sRGB linear. These color and intensity shifts will be more visible with transmission, but will also affect illumination and subsurface. Because of this, shader adjustments and texturing should happen in the same color space as the rendering one.

This is the default linear color space that Arnold will use as its rendering color space. Arnold's default color space is 'sRGB linear, ' but this can correspond to any linear color space if needed. If chromaticities for this linear color space can be guessed or are user specified certain spectral effects will take them into account, but not other adaptations for albedos, transparencies, etc. are applied.

You must set the OCIO environment variable to the config.ocio color profile path before starting Houdini. Houdini will then pick up the color spaces in the config and make them available as options for rendering color space, texture input color space and driver output color space.

Pepe model by Daniel M. Lara (Pepeland).