

# Arnold

Arnold is an advanced cross-platform rendering library, or API, used by a number of prominent organizations in film, television, and animation, including Sony Pictures Imageworks. It was developed as a photo-realistic, physically-based ray tracing alternative to traditional scanline based rendering software for CG animation.

Arnold uses cutting-edge algorithms that make the most effective use of your computer's hardware resources: memory, disk space, multiple processor cores, and SIMD/SSE units.

The Arnold architecture was designed to easily adapt to existing pipelines. It is built on top of a pluggable node system; users can extend and customize the system by writing new shaders, cameras, filters, and output driver nodes, as well as procedural geometry, custom ray types and user-defined geometric data. The primary goal of the Arnold architecture is to provide a complete solution as a primary renderer for animation and visual effects. However, Arnold can also be used as:

- A ray server for traditional scanline renderers.
- A tool for baking/procedural generation of lighting data (lightmaps for videogames).
- An interactive rendering and relighting tool.

## Why is Arnold different?

Arnold is a highly optimized, unbiased, physically-based 'Monte Carlo' ray/path tracing engine. It doesn't use caching algorithms that introduce artifacts like photon mapping and final gather. It is designed to efficiently render the increasingly complex images demanded by animation and visual effects facilities while simplifying the pipeline, infrastructure requirements and user experience.

Arnold provides interactive feedback, often avoiding the need for many render passes and allowing you to match on-set lighting more efficiently. By removing many of the frustrating elements of other renderers, Arnold fits better with your work-flow, produces beautiful, predictable and bias-free results, and puts the fun back into rendering!

## What is wrong with algorithms like photon mapping or final gather?

Such algorithms attempt to cache data that can be re-sampled later, to speed up rendering. However, in doing so, they use up large amounts of memory, introduce intermediate steps that break interactivity, and introduce bias into the sampling that causes visual artifacts. They also require artists to understand the details of how these algorithms work to correctly choose various control settings to get any speed up at all without ruining the render. Worse than that, these settings are almost always affected by other things in the scene, so it's often possible to accidentally use settings for the cache creation/use that make things worse, not better, or that work fine in one situation but are terrible in another, seemingly similar, situation.

In short, they are not predictable, other than for very experienced users, and require artists to learn way too much about the algorithms to gain any benefit. We believe that your time is more valuable than your computer's time; why spend an extra 30 minutes working with photon mapping or final gather settings, even if it saves 30 minutes render time (and more often than not it doesn't). That's still 30 minutes not spent modeling, animating or lighting.

## What are the advantages of being physically-based? Does this have implications for modeling and lighting scenes?

The advantage of being physically-based is that artists can work in a physically accurate, high dynamic range work-flow, and by using plausible values (being consistent about how you set lighting intensities and modeling scale) the end results will be more predictable. The process becomes much more akin to how a real scene is lit and photographed. It also ensures that other aspects of rendering are not broken.