



# Technical Brief

**Gelato**

Production Rendering for Film



# Overview

NVIDIA® Gelato™ rendering software is a no-compromises, production-quality renderer. Gelato's properties make it ideal for rendering final frames in motion picture production. Visual effects and feature-length animated films are the most demanding users of renderers, so success in this domain verifies, once again, that NVIDIA is on the leading edge of graphics innovation.

Just what is a “no-compromises” renderer? What does it take to render for film? What does “production quality” really mean? And how does this differ from a game engine, real-time graphics, or other types of rendering?

---

## How Film Rendering Is Different

It is important to understand the critical differences between games (and other real-time graphics applications) and final-frame rendering for film and other quality-intensive graphics applications.

## Image Quality vs. Rendering Performance

The film and games markets assign different values to the equation of image quality versus rendering engine performance. In film, images are precomputed, so a premium is placed on the quality and complexity of the final rendered image. An increase in quality improves the audience's experience, while a simple increase in rendering speed is unnoticed by the theatergoer. The opposite applies in games; time spent rendering a frame detracts from the gamer's experience. Therefore, film renderers like Gelato are designed primarily to gracefully handle massive scene complexities, rather than real-time performance.

Both in games and films, scenes are organized into groups of temporally contiguous frames, and rendered consecutively with mathematically continuous camera positions and with a common set of art assets. For games, these sets of similar frames are called “levels,” and they last for many minutes of game play. For films, the sets are called “shots,” and typically they last for only a few seconds of screen time.

Because of this distinction, game engines and film production amortize rendering overhead differently. In a game rendering engine, textures are loaded over tens of thousands of frames, and this task is usually performed before the first frame is drawn—thus “not counting” in the computation of rendering speed. In contrast, most film rendering is done a single frame at a time. But even if entire shots were rendered simultaneously to maximize coherence (and scene complexity makes this

prohibitively difficult), overhead could only be amortized over the few hundred frames (at most) composing the shot.

## Use of Texture Maps

Another key difference is how each medium uses texture maps. The texture files that film production uses consist of many gigabytes, whereas those used in games are perhaps tens of megabytes. Just the time to read a film frame's textures from disk or a network can take several minutes. Similar costs apply to reading, culling, sorting, and dicing the gigabytes of scene geometry.

No matter how fast the graphics hardware is, and no matter how much is performed with the graphics hardware instead of the CPU, the data manipulation just to prepare to render a film frame precludes real-time rendering. This is no surprise—games would not be real time either, if new game levels had to be loaded every 2 to 10 seconds of play (let alone every frame). The lower limit of film's final frame-rendering speed is dictated by disk transfer and other factors, not by transformation, fill, or even shading speed.

## Blinn's Law

A film render needs to take into account Blinn's Law, which states that when there's an improvement in rendering speed, the artist will increase the complexity so as to negate that speed improvement. In other words, the time to render a scene is a constant—usually from 45 to 90 minutes, depending on the studio's tolerance for wait time. Studios respond to increased rendering resources by constructing more complex scenes with more geometry, more lights, more complex shaders, and more expensive realistic lighting—never by simply taking advantage of the speedup. Increased complexity is the obvious choice for film studios, because the final images are precomputed and delivered on film.

The exception is interactive lighting, where production needs interactive speeds for many successive similar renderings. In interactive lighting, the geometry, camera, and shaders are fixed, and only the lighting parameters change from frame to frame. Filmmakers prefer to see results produced by the final frame renderer instead of by different previsualization tools so that they can accurately assess what the final frame will look like. In this case, a degree of reduced quality is acceptable. NVIDIA plans to offer an interactive lighting tool for Gelato in the near future.

## Development and Product Cycles

Another difference between games and film is their development and product cycles. Games are played by millions of people, and for every invocation, hundreds of thousands of frames may be rendered. Image quality, code reusability and clarity, and artist/technical director time are happily traded for guaranteed frame rates.

Images for film, on the other hand, are only rendered once at full resolution. The machine time spent rendering an object is often dwarfed by the amount and cost of the time it takes the technical director to write the shader. This has many implications for API development, and explains why APIs and languages that are appropriate for real time (DX, OGL, Cg) are not always appropriate for offline rendering, and vice versa.

---

# Essential Properties of Film Production Rendering

## Geometric Primitives

Games, real-time graphics, graphics hardware, and even “low-end professional” packages rely on polygons (primarily triangles) to create objects. But film work is almost always based on bicubic patches, NURBS (often trimmed), and, increasingly, subdivision surfaces. Large collections of polygons are rare in film frames; it is NURBS and subdivision surfaces that need to be optimized.

Points (for particles) and curves (for hair) are also important, and must be rendered efficiently by the millions. Furthermore, all curved primitives must dice adaptively based on screen area and curvature, so as to *never* show tessellation artifacts.

## Geometric Complexity

Geometric input in a single film frame can easily consist of several gigabytes—25 GB is not unheard of. As a primary design goal, a film renderer must be able to handle more geometry than could fit into RAM at once.

Gelato uses a variety of strategies for accomplishing this, including bucketing, sorting, aggressive culling, occlusion testing, procedural geometry, and caching to disk.

## Texture Complexity and Quality

A film frame often requires hundreds or thousands of textures, totaling tens of gigabytes of storage. Successful film renderers use a caching scheme where individual coherent tiles are paged from disk, as needed. It’s impossible to read all the textures into memory before rendering. And, because texture names can be generated dynamically by the shader, the filenames of the textures are often not known until the shader has begun executing.

Texture map lookups must have the highest fidelity. Texture, environment, and shadow maps must blur without artifacts, seams, or noise. And in many cases, lookups must have a better filter than trilinear mipmap.

## Motion Blur

Film renderers must be able to produce high-quality motion blur, with no strobing and little noise. Plus, the blur must have transformation blur (changing position, orientation, and scale), as well as geometric deformation blur. The best of this class of renderer, like Gelato, support multi-segment motion blur for an arbitrary number of knots.

Depth of field effects are also important.

## Antialiasing

No amount of visible aliasing artifacts are acceptable in film. This requirement is usually achieved by taking many point samples per pixel. For scenes with hair or fur, 64 to 100 samples per pixel is common. The renderer must reconstruct this subpixel data into pixels with a high-quality, user-specified filter with support larger than one pixel.

## Image Size, Depth, Format, and Data

Film requires arbitrary image resolutions (4 K for final frames, and 8 K or higher are not uncommon for shadow maps); flexible bit depths (8, 16, half, float); and useful image formats (Gelato supports TIFF, plus an easy API for user-supplied image writing routines).

It is also critical that the renderer output not just color, alpha, and depth, but any data computed by the shaders. This last property is called “arbitrary output variables,” or AOVs, in film, although in the real-time graphics world it is often called “Multiple Render Targets” or MRTs. Frames are saved to disk; rendering directly to the frame buffer is not important.

## No Limits

Arbitrary limits are not tolerable in film production. Arbitrary limits consist of number of textures, size of geometric database, resolution, number of output channels, and number of lights.

## Global Illumination

Modern film renderers, like Gelato, support ray-traced reflections and shadows, indirect light transport, caustics, environment lighting, ambient occlusion, and subsurface scattering.

## Displacement

Adaptively diced, subpixel frequency displacements are required in a film renderer and features, and should use no more system resources than undisplaced geometry.

## Flexible Programmable Shading

It used to be enough to simply say that film-quality rendering needed programmable shading (which was rare in software and impossible in hardware). Now that programmable shading is common—even in hardware—it’s helpful to enumerate some features that distinguish programmable shading for film work from those that are suitable for real time rendering:

- ❑ Rich data types, such as vector, color, point, normal, matrix, and their arbitrary-length arrays.

- ❑ String variables; string manipulation; and the ability to refer to textures, coordinate systems, and external data by name rather than by handles or explicit passing of matrices as parameters.
- ❑ Hiding of hardware details and limits. Technical directors who write shaders should not need to know about reduced precision data types, memory and instruction limits, or the names and details of hardware registers. The renderer should never create artifacts or lose precision because of these hardware limits.
- ❑ The ability to call out to user-supplied code on the host, *DSO shadeops*.
- ❑ Data-dependent looping, especially for applications such as ray marching for volumetric effects, or numerical integration inside shaders.
- ❑ Separate compilation of different types of shaders, especially light shaders.
- ❑ Precision calculation of derivatives and antialiasing. Derivatives must use central differencing and extrapolation at grid edges. Simple forward-differencing leads to unacceptable artifacts.
- ❑ A sequence of shaders on a surface, with a flexible means of layering and connecting inputs and outputs. In allowing for this, Gelato pushes the state-of-the-art.
- ❑ Input and output of arbitrary name, data type, and number of parameter arguments to and from shaders.

## Appropriate APIs and Formats

A film renderer needs to fit into a film production pipeline. This means, at a minimum:

- ❑ A clean procedural API (C or C++). OpenGL and DX expose too many hardware details; have limited support for higher-order surfaces and for shader assignment and binding; and are burdened with large portions of the API dedicated to interactive functionality that is not used in film.
- ❑ A scene archive format for “baking” geometry. Gelato unifies a scene archive format with the means to write procedural primitives by using the Python scripting language. It is also possible to write Gelato plug-ins that directly read any scene format, without translation into an arbitrary intermediate format.
- ❑ Procedural geometry as a first-class concept—for example, archive files read; geometry-producing programs run; or API-calling DSOs executed lazily, as demanded by the renderer.
- ❑ A shading language that incorporates all the requirements demanded by film. Gelato has its own shading language (GSL) to satisfy this need.
- ❑ Flexibility to read a variety of formats used by film studios. Gelato has a flexible API that allows it to accept input in virtually any format, allowing Gelato to be easily tested, and studios to transition their existing pipelines.
- ❑ Plug-ins for major animation packages. Gelato has a plug-in to export data from Maya to Gelato; export plug-ins for other modeling systems will soon follow.

## Appropriate Platform

Most major studios use Linux, with Windows a major secondary choice. Gelato supports both, and will support Mac OS X if the film industry demands it.

---

## Conclusion

Film is a different application than the real-time use of graphics in games. It makes different demands and puts a premium on different values. It is only natural, then, that filmmakers demand a specially designed tool.

NVIDIA offers a unique tool that meets the market's needs. With products like NVIDIA Gelato, NVIDIA demonstrates the same commitment to the world of film that it is known for in the real-time world of games.



## **Notice**

ALL NVIDIA DESIGN SPECIFICATIONS, REFERENCE BOARDS, FILES, DRAWINGS, DIAGNOSTICS, LISTS, AND OTHER DOCUMENTS (TOGETHER AND SEPARATELY, "MATERIALS") ARE BEING PROVIDED "AS IS." NVIDIA MAKES NO WARRANTIES, EXPRESSED, IMPLIED, STATUTORY OR OTHERWISE WITH RESPECT TO THE MATERIALS, AND EXPRESSLY DISCLAIMS ALL IMPLIED WARRANTIES OF NONINFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A PARTICULAR PURPOSE.

Information furnished is believed to be accurate and reliable. However, NVIDIA Corporation assumes no responsibility for the consequences of use of such information or for any infringement of patents or other rights of third parties that may result from its use. No license is granted by implication or otherwise under any patent or patent rights of NVIDIA Corporation. Specifications mentioned in this publication are subject to change without notice. This publication supersedes and replaces all information previously supplied. NVIDIA Corporation products are not authorized for use as critical components in life support devices or systems without express written approval of NVIDIA Corporation.

## **Trademarks**

NVIDIA, the NVIDIA logo, and Gelato are trademarks or registered trademarks of NVIDIA Corporation. Other company and product names may be trademarks of the respective companies with which they are associated.

## **Copyright**

© 2004 by NVIDIA Corporation. All rights reserved.



**NVIDIA.**

NVIDIA Corporation  
2701 San Tomas Expressway  
Santa Clara, CA 95050  
[www.nvidia.com](http://www.nvidia.com)