



## Technical Brief

### Gelato

Ten Things About Gelato That  
May Surprise You



# Table of Contents

<b>Overview .....</b>	<b>1</b>
Scene-File Reader Plug-ins .....	1
Preview Modes .....	2
Layered Shaders.....	2
Mango Hypershade Translation.....	3
No Uniform or Varying Keywords in GSL .....	3
Image I/O Plug-ins .....	4
Vertical Bucket Order .....	4
Grid Dumps.....	4
Truly Open APIs .....	5
Great Third-Party Tools.....	5
<b>Conclusion .....</b>	<b>6</b>

The NVIDIA® Gelato™ rendering software is a complex product that offers exciting features that are buried deep. Although Gelato may look similar to other renderers, it has several functions that are very different from what you are used to, and are far superior.

Most renderers require a particular format for a scene input. For example, NVIDIA's previous renderer, Entropy, required all scene input files to be in RIB format (as do many other similar renderers). If you had a file in another format—such as .obj or some proprietary format you use in your studio—you would have had to translate that input into RIB, then have the renderer read the RIB file.

---

## Scene-File Reader Plug-ins

NVIDIA Gelato does not have a required scene file format. Instead, Gelato has scene format reader plug-ins (DSOs under Linux, or DLLs under Windows). These plug-ins make it easy to directly read any scene format you have a plug-in for. You can use the ones developed by NVIDIA, or write your own.

Gelato comes with a plug-in that reads *Pyg*, which is Python with embedded Gelato API calls. But this plug-in is not “privileged”—it required no proprietary knowledge of Gelato's internals to write it, nor is it inherently more efficient than any plug-in you may write. As an example, a third party wrote an RIB-reading plug-in that is just as efficient for getting scene input into Gelato. (Go to [http://film.nvidia.com/page/gelato\\_download.html](http://film.nvidia.com/page/gelato_download.html) for a link where you can download this plug-in, with source).

Gelato lets you mix and match formats freely within a single Gelato rendering. For example:

- ❑ You may have a top-level scene file as Pyg.
- ❑ That Pyg may read RIB files (using Pyg's `Input ()` command).
- ❑ Those RIB files may have a ReadArchive of another Pyg.
- ❑ That Pyg may `Input ()` an obj file.
- ❑ And so on....

Does your studio use an in-house format (say, for the results of your simulations, or as a common intermediate format)? If so, we encourage you to write your own scene file reader plug-in and have Gelato read it directly, rather than translating to Pyg, RIB, or any other intermediate format.

---

## Preview Modes

Gelato has a number of reduced-quality modes that are handy for quickly previewing a scene file. Go to `$GELATOHOME/examples/vasefield` to see an example of reduced-quality mode. Try rendering normally:

```
gelato -iv beautypass.pyg
```

Now, try preview mode:

```
gelato -iv -preview 0.1 beautypass.pyg
```

The 0.1 means that the `shadingquality` (the number of shades per pixel unit distance) is set to 10 percent of what it would be in a final frame rendering. (Preview mode also takes a number of other shortcuts, such as increasing the bucket size, turning down the antialiasing rate, and so on.) Increasing this number increases the fidelity of the preview; decreasing it makes a rougher, but quicker, preview. Notice that preview mode still uses your real shaders, but it evaluates them much more *roughly*.

Another mode bypasses your real shaders entirely, relying on fast hardware shading to show a plastic scene:

```
gelato -iv -shade defaultsurface beautypass.pyg
```

or

```
gelato -iv -shade keyfillrim beautypass.pyg
```

You can also combine `-preview` and `-shade`:

```
gelato -iv -shade defaultsurface -preview 0.1 beautypass.pyg
```

---

## Layered Shaders

With some renderers, you assign exactly one surface, volume, or displacement shader to an object. With Gelato you assign a *list* of shaders for each of these, and they are run in sequence. You can establish linkages by name (via the `ConnectShaders` API call) between the output parameter of an earlier layer and the input parameter of a later layer on the same primitive. This allows you to compose shader functionality from prebuilt layers, without needing to recompile shaders or even to have access to the shader source code.

For example, Gelato ships with a “plastic” shader that does not do any texture mapping, but does use the `C` attribute for surface color and has a `Ks` parameter controlling specularity. It also doesn’t do any reflections.

Gelato also ships with a “texmap” shader that does a basic texture lookup and stores its results in its “`Cout`” and “`kout`” outputs, and a “`postgloss`” that adds reflections to whatever is already stored in “`C`.” These three shaders can be combined into a group that has a plastic reflection model, with texture-mapped color and specularity, plus reflections (also controlled by the specularity map), as follows:

```

ShaderGroupBegin()
Shader ("surface", "texmap", "layer1", "string
texturename", "colormap.tx")
Shader ("surface", "texmap", "layer2", "string
texturename", "specmap.tx")
Shader ("surface", "plastic", "layer3")
Shader ("surface", "postgloss", "layer4", "string envname",
"reflections.env")
ConnectShaders ("layer1", "Cout", "layer3", "C")
ConnectShaders ("layer2", "fout", "layer3", "Ks")
ConnectShaders ("layer2", "fout", "layer4", "Kr")
ShaderGroupEnd()

```

This is accomplished at the API level, without the having to access the shaders source code.

---

## Mango Hypershade Translation

The NVIDIA Mango™ software—NVIDIA’s Alias Maya plug-in that exports from Gelato and comes bundled with Gelato—translates your existing Maya Hypershade networks into layered Gelato shaders. If you look in `$GELATOHOME/mango/shaders`, you’ll see Gelato layers for many Maya shading nodes. Any Hypershade shader network that only consists of the supported nodes will be translated properly.

NVIDIA is working on translating the remaining nodes, and will include them in future patches to the Gelato/Mango software. If some of these nodes are especially important to you, please let NVIDIA know and their translation will be expedited.

When you render with Gelato from Mango, go ahead and view the Pyg that Mango generates (`gelato.pyg` in `/usr/tmp` on Linux, or `C:\temp` on Windows). It is instructional to see how Mango strings them together into layered shaders.

Of course, you can also attach your own custom-written Gelato shaders within Maya; you are not required to use the Hypershade translation.

---

## No Uniform or Varying Keywords in GSL

Unlike other shading languages, Gelato Shading Language (GSL) does not require you to declare variables as either `uniform` or `varying` in the shader source code. Gelato automatically performs the optimizations that other renderers do for uniform values. In fact, uniformed optimizations are done dynamically. As the shader executes, it performs optimizations that the shader author couldn’t accomplish by marking certain variables as `uniform`. For example, a particular variable might be treated as `uniform` for some computations and as `varying` for other computations, within the same shader.

Because you do not need to pre-declare `uniform` or `varying`, you can attach geometric primitive variables (automatically interpolated across the primitive surface) to any shader parameter, not just to those you previously declared as `varying` when you wrote the shader. This is especially handy when you're using layered shaders.

---

## Image I/O Plug-ins

Some renderers have clunky plug-in APIs that let you write out image files in arbitrary formats. Gelato's Image I/O plug-ins allow you to both write and read arbitrary image formats with a really clean and easy API.

That means that you can get Gelato's image viewer program, *iv*, to display any image format for which you have an Image I/O plug-in. You can use Gelato's *maketx* program to create a tiled MIPmap texture from any image format. Plus, any formats that support multiresolution tiled access can be used directly as Gelato texture maps, without converting them. For example, Gelato's OpenEXR plug-in allows Gelato to directly use OpenEXR texture and environment maps (without translation, even if they use 16-bit float data).

Go ahead and use Gelato's Image I/O protocols in your own software. Contact us if you need help.

---

## Vertical Bucket Order

Have you ever been rendering a big crowd scene in 2.35:1 aspect ratio, only to find that your renderer comes to a screeching halt when it gets to the scanlines that intersect all 10,000 of your characters?

Perhaps you have tricks where you turn the image on its side by modifying the input files to alter the camera matrix, render, and then rotate the image when finished?

That's unnecessary in Gelato. Just put the following in your file:

```
Attribute ("string bucketorder", "vertical")
```

---

## Grid Dumps

Using the various grid dump attributes, Gelato can dump the shaded grids (just before hiding) to a file. In fact, the grids are dumped as Pyg primitives—specifically as `Patch("linear")` with attached colors and other variables—which means you can easily inspect, read, filter, alter, or add to these files.

You can also read grid files back—any `Patch("linear")` primitives that use `null` as their surface shader and have no displacement shader are automatically interpreted as pre-shaded grids. These go straight to the hider, bypassing the shading engine.

Among other things, this means you can:

- ❑ Use Gelato as a tessellation-only and shading-only engine, dumping the diced, shaded grids to a format that you can massage or use as input to another renderer or program.
- ❑ Produce shaded geometry from any source, including another renderer, and give it to Gelato for hiding (directly, without reshading), including combining it with normal Gelato geometry.

---

## Truly Open APIs

All of Gelato’s APIs and formats (the C++ API, Pyg, grid dump files, and so on) are fully documented. We have no secret or proprietary formats.

Gelato’s APIs are also truly “open.” The APIs and how they are used, the header files, and all the example scenes, shaders, and source code that we ship with Gelato are covered by the BSD License. With the exception of the trademarks on the names *NVIDIA*, *Gelato*, and *Mango*, you are free to use all the header files and examples, copy them, modify them, redistribute them, and extend them. You can also write, distribute, or sell the readers and writers and compatible tools, including renderers.

---

## Great Third-Party Tools

There are some great third-party Gelato add-ons available.

- ❑ Open Source scene-reader plug-in (what Gelato calls a *generator*), which lets Gelato directly read RIB files.
- ❑ RenderMan shading language-to-Gelato shading language source-to-source translator, currently distributed as a binary, but still free.
- ❑ *Amaretto* (a 3D Studio Max plug-in that exports to Gelato), which Frantic Films has been showing. *Amaretto* has a very slick Max-to-Gelato material translation, similar to what *Mango* does, but probably even more polished. *Amaretto* is not currently distributed, but check on Frantic Films Web site for details: <http://software.franticfilms.com/>

For updated links to third-party plug-ins and utilities, go to [http://film.nvidia.com/page/gelato\\_download.html](http://film.nvidia.com/page/gelato_download.html)

NVIDIA encourages third-party development through the company developer program where you can get a free Gelato license, discounts on NVIDIA Quadro® boards, and preferential treatment at marketing events. For details, see [http://film.nvidia.com/page/gelato\\_developers.html](http://film.nvidia.com/page/gelato_developers.html).

To assist third-party developers, NVIDIA continually revises and adds features and easy plug-in APIs into Gelato. Another example of our commitment is that Gelato’s shader compiler, *gslc*, has an option to produce encrypted compiled shaders. This lets you safely distribute shaders without exposing the source code (or allowing the object code to be examined or decompiled). If you have suggestions for encouraging third-party development, please let NVIDIA know.



## Conclusion

Having ten fingers, we humans tend to like lists of ten things. But ten items don't even scratch the surface of unique and advanced features of Gelato that you won't find in other renderers. So we hope that this list will inspire you to take an even closer look at our product to uncover the dozens or hundreds of little details that will make your life better as a renderer user, if you use Gelato.



## **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, Gelato, Mango, and NVIDIA Quadro, are trademarks or registered trademarks of NVIDIA Corporation in the United States and other countries. Other company and product names may be trademarks of the respective companies with which they are associated.

## **Copyright**

© 2004 NVIDIA Corporation. All rights reserved.



**nVIDIA.**

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