The ryg blog
When I grow up I'll be an inventor.

A trip through the Graphics Pipeline 2011: Index

July 9, 2011
Welcome.

This is the index page for a series of blog posts I'm currently writing about the D3D/OpenGL graphics pipelines *as actually implemented by GPUs*. A lot of this is well known among graphics programmers, and there's tons of papers on various bits and pieces of it, but one bit I've been annoyed with is that while there's both broad overviews and very detailed information on individual components, there's not much in between, and what little there is is mostly out of date.

This series is intended for graphics programmers that know a modern 3D API (at least OpenGL 2.0+ or D3D9+) well and want to know how it all looks under the hood. It's *not* a description of the graphics pipeline for novices; if you haven't used a 3D API, most if not all of this will be completely useless to you. I'm also assuming a working understanding of contemporary hardware design – you should at the very least know what registers, FIFOs, caches and pipelines are, and understand how they work. Finally, you need a working understanding of at least basic parallel programming mechanisms. A GPU is a massively parallel computer, there's no way around it.

Some readers have commented that this is a really low-level description of the graphics pipeline and GPUs; well, it all depends on where you're standing. GPU architects would call this a *high-level* description of a GPU. Not quite as high-level as the multicolored flowcharts you tend to see on hardware review sites whenever a new GPU generation arrives; but, to be honest, that kind of reporting tends to have a very low information density, even when it's done well. Ultimately, it's not meant to explain how anything actually *works* – it's just technology porn that's trying to show off shiny new gizmos. Well, I try to be a bit more substantial here, which unfortunately means less colors and less benchmark results, but instead lots and lots of text, a few mono-colored diagrams and even some (*shudder*) equations. If that's okay with you, then here's the index:

From → Coding, Graphics Pipeline

38 Comments

1. **nordicsavage permalink**
   Don't forget a chapter on Multi-sampling and the difference between the various AA techniques out there.. :) Am loving this series though :)

   Reply
   - **fgiesen permalink**
     Unlikely. I'm sticking with the basic D3D11 pipeline, and even there I'm dropping some subjects. Basic MSAA (2x, 4x, 8x) must be supported by every D3D11 device, all fancier stuff is strictly optional. And there's even bits in core D3D11 that I'm mostly ignoring – point and line primitives (and their setup+rasterization), the finer points of cube map filtering, the various trilinear filtering "optimizations" that GPUs do (not, ahem, strictly in accordance with the spec…), and so on…

     All that's there, but I have enough material to write another 7 parts already; I do plan to finish this series eventually :)

     Reply
2. **Manu permalink**
   Good job man!
   Maybe you should write a book or something. Many people will buy it for sure.

   Reply
   - **doctor_shim permalink**
     that would negatively impact the size of the readership, in addition to reducing exposure!

     Reply
     - **fgiesen permalink**
       Cleaned up, expanded PDF version is in the works. You're gonna have to print it yourself though. :)

3. **Brandon Furtwangler permalink**
   This is a great series of articles. Thanks for making them. Can't wait for the part(s) on compute shaders.

   I'd love to hear how you think the pipeline could/should evolve in the coming years.

   Reply
4. **Francis Boivin permalink**
   Are you considering talking about constant buffers? They are an important part of d3d10+. At least, they are from an API perspective – do driver actually still care about this optimized representation of shader parameters? I'm not an OpenGL guy so I don't know if it went with a similar API or if constants still set using glUniform[…]?

   Reply
   - **fgiesen permalink**
     They're an important part of the API, but on the GPU side they're really just chunks of memory that the shader units can access.

     Originally, the constants used to be an actual special register file on the chip. D3D10 increased the limits too far for that to be practical: up to 16 constant buffers per stage – 15 API-accessible and one for immediate constants – with up to 4k elements of up to 16 bytes each; so up to 64k per CB and up to 1MB total, *per stage* – and you can have a lot of them active at once (worst case, five of them: VS, HS, DS, GS, PS). So now CBs are just regular buffers in GPU/host memory (like VBs, IBs or Textures) and can be accessed as such. One option is to still have some (smaller) amount of fast memory reserved for constants, and try to pack the CBs that will fit in there. But this generation of GPUs has (this is fairly new!) a regular fast cache between shader units and memory. With a cache, I'd just leave the CBs completely memory-mapped and let the cache deal with it! This can adapt to the dynamic behavior of shaders, rather than having to rely on some heuristic to pick which CBs to pack into fast memory.

     Reply
5. **ikrima permalink**
   Dude, this is amazing. I took a sabbatical out of gfx/vfx for 4 years and been playing catch up over the last 7 months. Thanks for the awesome in-depth articles for an easy quick dive into how things have changed.

   Reply

6. **sinistraldexter permalink**
   Reblogged this on sinistraldexter and commented:
   one of the best work w.r.t the graphics pipeline

   Reply
7. **nandu permalink**
   Hi, intern@ Nvidia. Really helpful to get started.

   Reply
8. **Samuel Egger permalink**
   An amazing series! Thank you. Although I am quiet curious where one learns all these things?

   Reply
   - **fgiesen permalink**
     I was working on a GPU at the time I wrote the series.

     Reply
9. **Suso permalink**
   I do you plan to review the new mesh and amplification shaders?

   Reply

Trackbacks & Pingbacks

1. Real-Time Rendering · Seven Things for July 24th, 2011
2. Viaje alucinante por un pipeline grafico « martin b.r.
3. A trip through the Graphics Pipeline | Light is beautiful
4. A Very Good Technical Guide to the 3D Graphics Pipeline
5. A trip through the Graphics Pipeline 2011, part 1 « The ryg blog
6. A trip through the Graphics Pipeline 2011, part 5 « The ryg blog
7. A trip through the Graphics Pipeline 2011, part 4 « The ryg blog
8. A trip through the Graphics Pipeline 2011, part 6 « The ryg blog
9. A trip through the Graphics Pipeline 2011, part 7 « The ryg blog
10. A trip through the Graphics Pipeline 2011, part 8 « The ryg blog
11. A trip through the Graphics Pipeline 2011, part 3 « The ryg blog
12. A trip through the Graphics Pipeline 2011, part 2 « The ryg blog
13. A trip through the Graphics Pipeline 2011, part 9 « The ryg blog
14. A trip through the Graphics Pipeline 2011, part 10 « The ryg blog
15. A trip through the Graphics Pipeline 2011, part 11 « The ryg blog
16. A trip through the Graphics Pipeline 2011, part 13 « The ryg blog
17. A trip through the Graphics Pipeline 2011, part 12 « The ryg blog
18. Programming | Pearltrees
19. Túra a grafikus csővezetékben | cikksorozat | szimpatikus.hu trackback proxy
20. Xbox / PC, early-Z and early stencil in XNA « IceFall Games
21. Order and types of depth testing « Interplay of Light
22. HPG 2013 | dickyjim
23. A trip through the Graphics Pipeline | The blog at the bottom of the sea
24. What's the big deal with Apples Metal API? | RenderingPipeline