The Graphics Rendering Pipeline

Commercial game engines include: Unity, Unreal engine, and CryEngine, to name a few. Find a overflowing table of engines on Wikipedia.

See our WebGL resources page for various easy-to-use libraries for 3D display on your web browser. Since Javascript is an interpreted language, modifying, running, and sharing demos is ridiculously easy. See the three.js examples page and Lee Stemkoski's site for a start.

There are a wide range of open source renderers available. NVIDIA's Falcor gives access to DXR ray tracing capabilities, among many other features. G3D is a research-oriented renderer that offers a wide range of modern effects and works on all major PC operating systems. OpenSWR is a fast pure-CPU OpenGL-compatible renderer from Intel. Irrlicht, bgfx, OGRE, Godot, and sauerbraten are open source 3D games engines with some popularity. Open Scene Graph is a free, open source scene graph system. Coin is an open source retained mode scene graph library based on Open Inventor.

Ancient technologies can be entertaining. Source code for DOOMs and Quakes is available for download under GPL. The ioquake3 site builds upon Quake 3, adding a huge number of improvements (even ray tracing). There are also ports of DOOM to a huge number of platforms, and Wolfenstein is on the iPhone. Arcade emulators such as MAME allow you to port classic games to most any platform.

Of course, one of the best game engines is Excel.

The Graphics Processing Unit

Figure 3.14, about user-defined clipping planes, was generated with this and this live interactive three.js demo.

The series A trip through the Graphics Pipeline 2011 is an excellent in-depth presentation on all the major elements of the GPU.

The Humus 3D site has some excellent sample programs that show advanced techniques.

Guides to tessellation shaders, most with an OpenGL-bias, include this tutorial, this description, descriptions of triangle and quad subdivision, this more technical set of slides, and this reference/tutorial.

There are NVIDIA-specific extensions for Vulkan supported in their Turing architecture. NVIDIA has more information on their developer page.

Transforms

Immersive Linear Algebra is a free interactive book on linear algebra, coauthored by Tomas. This interactive demo from Eric's course shows how the set of transforms changes due to camera and model modifications. Another interactive demo shows how various common camera model parameters affect the view.

If you need to start from scratch, Wolfire's blog has a basic two-part tutorial on linear algebra, here and here.

Dave Eberly's site has useful papers and code on a wide variety of geometric operations, including quaternion interpolation. J.M.P. van Waveren discusses matrix to quaternion transforms and how to use SIMD for them. Code for rotating from one vector to another rapidly using quaternions (as described in our book) is available online from Tomas.

The Portable Game Library includes code for a Simple Geometry library.

Dual quaternion skinning offers improved quality at relatively little additional cost. The process can be done using a vertex shader. An old-but-free Maya plugin is also available.

The Graphics Gems book series contains a number of articles on transformations, with code online.

As we touch upon in our book, moving your z-buffer's near plane as far from the eye as possible is a good idea. Steve Baker has an article on this topic with a little calculator to explore the effect. Nathan Reed discusses options to improve precision in depth (we use one of his graphs in our book). An older article gives other visualizations of precision problems. Using a reversed z-buffer is one solution - this article also links to other articles on the subject.

It's possible to perform skinning in a compute shader.

Shading Basics

We drew upon Tarek Sherif's WebGL 2 Examples for our shader examples. This code is an easy way to start playing with shaders.

There are many shader tutorials out there. For example, Unity 3D and Learn OpenGL provide basics, Ronja gives some more elaborate effects. The Book of Shaders is an introduction to fragment shaders, under development. Shadertoy is at the far end of the spectrum, an interactive web environment doing everything in the fragment shader; Omar Shehata provides an introduction. Other web-based systems for experimentation include GLSL Sandbox (similar to Shadertoy), Shdr for basic inputs and meshes to play with, Shadershop for visualizing shader functions, and ISF for video DJs.

Dealing with the explosion of shaders caused by material systems is an ongoing concern. This article discusses the problem in depth and offers solutions.

Storing semitransparent textures so that the colors are premultiplied by the alphas makes compositing and blending operations much faster to compute. Tom Forsyth gives a rundown of the math and formats involved (search "Premultiplied alpha part 2").

Transparency is difficult to perform correctly in a single pass when using a Z-buffer. Steve Baker gives a good summary of the basics of the problem and traditional solutions. NVIDIA's developer site and the Humus 3D site each have sample code for using stencil routing to provide order-independent transparency. Tarek Sherif made a WebGL 2 browser demo of depth peeled transparency.

NVIDIA provides a good summary of basic techniques for transparency and some common optimizations. This article at Sketchfab gives an artist's view of transparency and using blending modes to simulate it.

The dangers of improperly accounting for alpha in cutouts is lavishly illustrated in this blog post.

24 bits of color is usually enough, but not always. Here's an image showing concentric bands only one pixel intensity value apart. Most monitors will show some banding somewhere on the image.

Here is a comparison generated using Scott R. Nelson's program of lines drawn with gamma=1.0 (note the severe roping and Moíre patterns) and properly drawn with gamma=2.2. I find this pretty convincing.

Larry Gritz's article The Importance of Being Linear is an excellent explanation of gamma correction and other monitor transforms. If that article doesn't click for you, maybe this one will.

The difference between linear, sRGB, and gamma 2.2 are shown in this Shadertoy.

The chart on this page shows different MSAA patterns used on various vendors' GPUs. Lots more on sampling pattern theory here. This FXAA vs. SMAA interactive demo lets you examine the differences. Matt Pettineo's blog post describes the problems with MLAA approaches under animation. The retrospective here gives a thorough history of the use of MLAA techniques in games.

This article on alpha coverage includes some excellent animations and figures showing various problems and solutions.

Font antialiasing via sub-pixel LCD rendering is dealt with in depth on the Anti-Grain Geometry site. GPU-accelerated 2D engines include Will Dobbie's (try the War and Peace demo, YMMV).

Code that demonstrates the MSAA and temporal antialiasing techniques used in "The Order: 1886" can be found here. The demos here are not truly temporal antialiasing, but rather progressive antialiasing, and still worth a look to see the improvement over time.

Newer techniques continue to be developed. DLSS, deep-learning super-sampling, is as it sounds, using deep learning training to create a neural net that improves an aliased image.

Texturing

Live interactive demos associated with this chapter's contents:

The three.js anisotropy demo for Figure 6.19.

Stéphane Guillitte's "Playing Marble" shadertoy for Figure 6.24.

The three.js demo showing multiple texture types for Figure 6.25.

S3TC texture compression is a standard part of DirectX, renamed DXTn texture compression. An outdated but free manipulation and compression library is available for manipulating DXTn (DDS) format textures (the package also converts heightfields to normal maps). Source code is available. The Unreal developer network has an excellent article on DXTC compression and quality comparison. DevIL is an open-source image conversion library that reads and writes DDS and many other formats.

The USC-SIPI Image Database has many classic images (Lena, Mandrill) and other texture samples for research. For free stock images, one site is Free images, among many others.

Just because a paper is ancient does not mean it's dated - math doesn't rot. Heckbert has written a worthwhile Survey of Texture Mapping and a more in-depth work, Fundamentals of Texture Mapping and Image Warping. Many classic applications of texture mapping are discussed at Paul Haeberli's site.

AMD has an old program called MeshMapper which generates normal, displacement, and ambient occlusion maps from a low and high resolution model. I know I've seen this somewhere in a newer open-source version, but can't find it... (help appreciated)

An implementation of megatexturing is available.

Shadows

Figure 7.12 was generated with this demo by Christoph Peters using the Sintel model. Another good shadow exploration tool is Matt Pettineo's sample app.

Lauritzen and all have copious notes and demo code for their Sample Distribution Shadow Maps scheme.

Light and Color

The CVRL website has a huge amount of easily downloadable primary research data relating to color. There are numerous other webpages on colorimetry and related topics, such as this overview. Poynton's color space FAQ contains much solid information on the topic.

Color spectra data for acrylic paints from one company are available for download.

Bruce Lindbloom's site summarizes color conversion equations, and includes a table for converting between the popular color spaces (see his "Math" link). There are some useful notes on correct and efficient conversion between RGB and YUV color spaces. Interestingly, the chromaticity function is not a simple triangle, as it is usually shown.

Physically Based Shading

The Advances in Real-Time Rendering SIGGRAPH course notes for the past few years are available for download, with talks discussing many areas. Stephen Hill's blog is an great guide to SIGGRAPH courses and other resources.

The Learn OpenGL site has a surprisingly detailed page on microfacet theory and other elements of physically based rendering.

Filament is an open-source rasterizer. It includes an extensive and impressive amount of documentation and theory about physically-based materials.

Morgan McGuire's Graphics Codex is a great compendium of short, readable articles on advanced rendering techniques and much else. It can be accessed on the web or on an iPad.

Andrew Glassner's classic Principles of Digital Image Synthesis is free for download; a bit old, but physics and math don't change that much.

BRDF data is available from Cornell and Columbia-Utrecht Universities. FreeSnell has the refractive indices and coefficients of extinction for many materials, as well as a thin-film simulator.

If you have only a minute to learn about modeling layered materials, watch this. You might then be tempted to watch his free series of lectures.

Local Illumination

Figure 10.24 is from this interactive three.js demo, with a few of our own modifications. Figure 10.32 was derived from this Shadertoy by Nimitz.

A great history of reflection mapping is available from Paul Debevec's site. Some normally difficult to obtain early papers and videos can be found here. High dynamic range environment map image data is available at this site, along with 8 bit/channel spherical map images.

Humus has a large set of cube maps available for experimentation.

The IVRPA is a good place to see panorama images and learn about how to make them. 360 Cities also has panoramas.

Global Illumination

Figure 11.27 showing light baking was generated using content from this live interactive three.js demo.

Phil Dutre's old Global Illumination Compendium, mentioned in "Further Resources", has much useful information on BRDFs and other facets global illumination theory.

A pleasant visualization of how the coffee-cup caustic is formed is available.

A gallery of game screenshots shows the evolution of how water is rendered in games - click through the images near the bottom.

Image-Space Effects

Matt Pettineo's BakingLab demo generated Figure 12.16, and is a good way to play with depth of field, tone mapping, and many other effects.

Tarek Sherif's depth of field demo runs in the browser, so is just a click away. It is part of his set of WebGL 2 examples.

Matt Pettineo's Baking Lab demo lets you explore depth of field, tone mapping, and other effects. G3D takes more work to get going, but also include depth of field, motion blur, and many other effects.

Beyond Polygons

The particle system images shown in Figure 13.4 of the book are better enjoyed as animated programs that run in your browser (i.e., just a click away):

"The Spirit" by Edan Kwan (more demos).

"Fluid Particles" by David Li.

"Southern Delta Aquariids meteor shower" by Ian Webster (more demos).

If you want to play with source, start with this three.js demo, one of many particle demos in that codebase. Tarek Sherif's WebGL 2 particle demo is another starting spot.

Unsurprisingly, point sprites are more efficient than particles generated using the geometry shader.

There are any number of presentations about using particles and other effects in games, such as in Unity.

Potree is a great open-source point cloud rendering system with a number of worthwhile features. Source code and a demo for the ancient point rendering system QSplat is still available for download.

The OpenEXR image format, developed by ILM, allows higher precision formats to be written and read, including support for the 16-bit floating point "half" format used in NVIDIA's Cg format. It is an extensible format that allows arbitrary buffers of data.

Volumetric and Translucency Rendering

Figure 14.37, refractive glass angels, is from a tweaked version of this live, interactive three.js demo and from Lee Stemkoski's Bubble demo at his site.

Figure 14.40, water ripples, is from Evan Wallace's live WebGL demo, explained here.

If you want to know just a bit about volume rendering, Kyle Hayward's old 101 and 102 tutorials are worth a look.

Compared to polygon rendering, volume-related test data is relatively rare. Disney provides a volumetric cloud dataset at different resolutions. The USC-HairSalon is a set of 343 hair models.

Sébastien Hillaire has made his HLSL spherical harmonics code available for use as a submodule, used for example in directional ambient occlusion for clouds

Non-Photorealistic Rendering

See the Non-Photorealistic Animation and Rendering Proceedings for the latest on NPR research.

One important new area in NPR that we mostly ignore in the book is style transfer, mostly because it's much more related to image processing and deep learning. This is the seminal paper, and there's now even an app, Prisma.

Aaron Hertzmann has a recent three-part series on image stylization. Here is part 1, part 2, and the third part to appear.

The old Stylized Rendering in Games SIGGRAPH 2010 course materials have a number of worthwhile presentations.

A dated but worthwhile NPR resources page has been put together by Craig Reynolds.

Even gibbets can be stylized, see NPRQuake.

Polygonal Techniques

MeshLab is an open source system for manipulating meshes. It has a huge number of meshing operations available. The Meshlab blog has worthwhile articles, including a rundown of experiments performed comparing three different vertex normal computation techniques.

The McGuire Graphics Data page is a curated collection of a variety of model types, extremely useful for research. The Stanford 3D Scanning Repository contains the famous bunny model, happy buddha, dragon, armadillo, and other dense polygonal meshes. The relatively new ORCA repository has a few complex models that see use in research papers. See our portal page for more model sources.

Unity provides a free 96-page guide to using photogrammetry.

CGAL has a computational geometry bias, but supports many operations on polygonal models. If you need meshes with various constraints (e.g., avoiding long, thin polygons), try Jonathan Shewchuk's Triangle software.

For translating various file formats, we recommend Assimp. For file format information, start at Wikipedia or the Graphics File Format Page. glTF is a new file format that is a tight match with how GPUs store and display 3D data. In the film world, Pixar's Universal Scene Description (USD) software is gaining traction for asset interchange.

Gavin Bell describes a bit more about how to get the normals to point outwards, along with sample code.

GTS is an (ancient) open-source, LGPL polygonal manipulation library that does VIPM, stripification, hierarchical bounding box generation, and more.

Mark Duchaineau's free LibGen has simplification code buried in it (see the "surf" library and "surftools" commands).

The Virtual Terrain Project has a huge amount of useful information about terrain storage and rendering, large terrain datasets, as well as source code.

Curves and Curved Surfaces

Figure 17.32 was generated using Tamy Boubekeur's Phong tessellation Windows demo. Figure 17.37 was generated from Iñigo Quilez's shadertoys rainforest and Snail.

Pixar's Open Subdiv open source libraries site includes a tutorial on subdivision surfaces.

Paul Baker has an old demonstration program with source that tessellates and renders metaballs.

We do not cover NURBS in our book, but these are important in CAD. The verb library is a relatively recent open-source system for manipulating these. Dean Macri has an article on using NURBS in real-time applications at Gamasutra's site. The source code for the book An Introduction to NURBS is available online. There is also an older Sourceforge NURBS manipulation library.

A short history discusses the most famous spline surface model, the Utah Teapot; here's a video about it that Eric made; a C program to generate it is available for download.

Pipeline Optimization

This beginner guide to GPU optimization does a nice job of discussing some of the major performance bottlenecks and tools used to discover these.

We list a number of tools for debugging in the book, here are the links:

RenderDoc - a high-quality Windows, Linux, and Android debugger for DirectX, OpenGL, and Vulkan, originally developed by Crytek and now open source.

AMD's GPU PerfStudio - AMD's suite of tools for their graphics hardware offerings, working on Windows and Linux. One notable tool provided is a static shader analyzer that gives performance estimates without needing to run the application. AMD's Radeon GPU Profiler is a separate, related tool.

NVIDIA Nsight - a performance and debugging system with a wide range of features. It integrates with Visual Studio on Windows and Eclipse on Mac OS and Linux.

Microsoft's PIX - has long been used by Xbox developers and has been brought back for DirectX 12 on Windows. Visual Studio's {\em Graphics Diagnostics} can be used with earlier versions of DirectX.

GPUView - from Microsoft; uses Event Tracing for Windows (ETW), an efficient event logging system. GPUView is one of several programs that are consumers of ETW sessions. It focuses on the interaction between CPU and GPU, showing which is the bottleneck.

Graphics Performance Analyzers (GPA) - a suite from Intel, not specific to their graphics chips, that focuses on performance and frame analysis.

Instruments - for Xcode on OS X; has several tools for timing, performance, networking, memory leaks, and more.

OpenGL ES Analysis - detects performance and correctness problems and proposes solutions

Metal System Trace - provides tracing information from the application, driver, and GPU for Metal applications.

For profiling, this tool allows you to see how the waves (warps) are executed; the documentation explains various stalls and other problems that can happen. The Kernel Occupancy Viewer is aimed at determining occupancy, how many warps are available for processing. This archived page on occupancy describes how this can be measured and expressed.

There are a fair number of in-depth articles on how various engines perform rendering:

GTA V - Graphics Study

DOOM (2016) - Graphics Study

How Unreal Renders a Frame

Unreal Engine 4 Rendering

World of Tanks: Graphical Update Technical Overview

Godot 3's Renderer Design Explained

Mud and Water of Spintires: MudRunner

Reverse engineering the rendering of The Witcher 3

Tom Forsyth gives his view on renderstate change costs (search "Renderstate change costs").

In these older articles, Christer Ericson discusses how to order draw calls around for efficiency (this article is pretty popular) and how to optimize particle systems.

Noel Llopis explains the basics of data alignment and what to know to improve efficiency.

Concurrency now needs to be designed into rendering systems from the start. Herb Sutter's classic article is a good start as to why this is now so. One popular tool for programming in parallel is Intel's Threading Building Blocks.

YCoCg compression can be explored with this interactive WebGL demo, explained here.

Matt Pharr describes the challenges of reading an extremely large dataset, the Moana island scene. While focused on his ray tracer's performance, the issues of profiling and finding the bottlenecks and memory wasters affect all graphics applications.

Acceleration Algorithms

The Windows demo used to generate Figure 19.33 still works and can be found here. Quick to download and run, and to the point.

The old-but-good book Level of Detail for 3D Graphics covers many aspects of level of detail algorithms in depth. This book has a companion web site.

Some excellent examples of LOD popping in games are available; move your mouse in and out of each image to see the effect.

See this demo, which runs in your browser, for a straightforward visualization of occlusion culling's effect.

Efficient Shading

Matt Pettineo has an in-depth rundown of his and others' experiments with various light deferred shading techniques, and provides a demo for testing. He also has an article and code for efficient rendering of deferred decals.

Other demos and code for efficient shading of many lights are available, such as:

Ola Olsson's clustered forward code, and his older tiled shading code.

Andrew Lauritzen's

Marc Fauconneau Dufresne's

Emil Persson's

Jeremiah van Oosten's

Sebastian Sylvan gives an overview of Drobot's approach and explains why clustering is so useful. Eidos provides a summary of a deferred+ system they developed, and explain further in GPU Zen.

If you want more on micropolygons, these lecture slides from CMU describe the challenges of using this technique on the GPU.

Virtual and Augmented Reality

There are many sources of information on new VR and AR technologies and applications. These include the Oculus blog, SteamVR, NVIDIA's GameWorks blog, and the Google AR and VR blog. Some of the projects posted in the Made with ARKit are magic. There are also way too many other blogs tracking these fields.

Keep an eye on the OpenXR working group, whose goal is to set an open standard for VR and AR applications.

Intersection Test Methods

We created a 3D Object Intersection page, giving references and pointers to code for a wide variety of object/object intersection tests.

Dan Sunday's GeometryAlgorithms.com (defunct, but archived) has some good summaries of algorithms for making bounding containers for various geometric primitives.

Graphics Hardware

Wikipedia has some excellent articles on hardware-related topics, such as this one on color depth. The comments on this post on our blog are great for why there are high refresh rates.

Humus gives a rundown of the various ways of computing and storing z-depths.

Valve's Steam hardware survey tracks what is used by their subscribers; incredibly valuable for knowing what is out there.

Tech Power Up has an up-to-date summary of the clock speed, memory size, and other characteristics for every major consumer PC GPU. Ars Technica sometimes covers GPU architectures.

One reason little is published about commercial graphics hardware architectures is that there are trade secrets and possible patent infringement involved. The Patent Arcade site tracks patent infringement, copyright infringement, and other videogame related legal issues. Knowingly violating patents causes triple damages, so you've been warned.

24 bits of color is usually enough, but not always. Here's an image showing concentric bands only one pixel value apart. On most displays some area of the image will exhibit banding.

A good resource on graphics hardware architectures is the course notes for the Beyond Programmable Shading course at SIGGRAPH.

NVIDIA's list of technologies gives some information about each. They also have a light overview of conservative rasterization.

The UserBenchmark site provides benchmarking, along with GPU ratings, and can find performance problems on systems. FurMark is a free OpenGL stress test.

Already mentioned in "Pipeline Optimization" but worth a repeat: This tool allows you to see how the waves (warps) are executed; the documentation explains various stalls and other problems that can happen. The Kernel Occupancy Viewer is aimed at determining occupancy, how many warps are available for processing.

A recent paper by Zhe Jia et al. dissects various elements of the NVIDIA Volta and compares it to its predecessors.

Matt Pharr's The story of ispc discusses the failure of Larrabee, and his subsequent project of making a better compiler for SIMD. Along the way he describes the various political factors that went into decisions. A worthwhile read on a number of levels.

Two software-only solutions for making movies of interactive programs are FRAPS and HyperCam. FRAPS also measures and displays the frame rate of any 3D application.

There are many little utilities for checking various hardware capabilities, mostly for overclocking but also just educational to examine. GPU-Z displays the GPU's capabilities and monitors temperatures and voltages of various components.

The Accelenation site has an excellent history of the early years (1995-2002) of consumer graphics cards. Maximum PC has an extensive visual history of the GPU boards from 1995 on. For a general history of computer graphics, see Wayne Carlson's site. Steve Collins has a fascinating look at ancient consoles from a programmer's perspective. Emulators for many old machines can be found at the Emulator Zone.

The Future (resources)

Our still image from Claybook doesn't capture its dynamic nature, so check out the preview.

GPUs have improving in speed faster than Moore's Law, and this brief article summarizes some reasons why.

For a glimpse at what goes into a AAA title game, this short (and old, 2013, but worthwhile) video shows some of the many elements involved.

Inspired to go write some code? Consider studying the Unreal Engine (some free assets) or Unity, each for free.

Almost last mention: don't forget our portal for a list of some of the best resources.

Collision Detection

The Physics Simulation Forum has many threads about collision detection and physical simulation.

One related hardware product is NVIDIA's PhysX processor (they purchased Ageia), a dedicated physics action accelerator.

A number of free collision detection packages are available on the Web. These include source, and most have limitations on commercial reuse:

Bullet Physics Library - library for performing rigid-body collision detection and response. Open source and free for commercial use, and is integrated with Blender.

SOLID - Software Library for Interference Detection. Now a commercial product, and GPL'ed with source.

ColDet - a free collision detection library for generic polyhedra.

Havok - a popular commercial library for games

Related to collision detection, Qhull implements the Quickhull algorithm for finding convex hulls quickly. The Stony Brook Algorithm Repository has convex hull and other code in its computational geometry section.

Real-Time Ray Tracing

See our Ray Tracing Resources page.

Linear Algebra

The interactive Immersive Linear Algebra book is a great way to build up your intuition on the geometric interpretation of various operators and elements.

Flipcode has a 3D geometry primer online.

Wolfram MathWorld is an incredible resource for (sometimes dense) mathematical definitions. You can find a collection of math-related definitions at Cut the Knot.

Christer Ericson has a nice presentation on the scalar triple product, a way to compare the orientation of one line compared to another. He also has a followup article.

Maxima is a symbolic computation program, like Mathematica and Maple: you define equations and can easily combine them, integrate, take the derivative, etc. Maxima is GNU source now, and free.

Trigonometry

Trig formulas, tables, and other mathematical reference material can be found at Dave's Math Tables.

Bibliography

You can access the bibliography of the book, with many of the articles having hyperlinks.

One last mention: our portal is where we list all the best ways to find more information, including Ke-Sen Huang's great conference article site.