------= OPEN FOR SUGGESTIONS! ------

How to suggest new resources? Use comments in Google Doc!

Please be mindful of accidental edits - I see many erroneous edits (spaces added, returns, deletions…).

- Tinkering & Practical > Theoretical & Boring :)
- Projects, books, videos, whatever…
- Quality > Quantity. Let's try to find a "minimal" set :)
    - The internet is -full- of information, in order to help one has to be concise, sift through the noise.
    - Same for frameworks & programs. Lots of choice. Prefer things that come/have great projects a beginner can go through.
- Both Real-time and Offline Computer Graphics are covered.

Angelo Pesce - c0de517e.com - @kenpex

3D Computer Graphics Programming for *Beginners*

A selection of resources to **learn** and **tinker** with.

---

1 - Have **fun & tinker** with small projects.

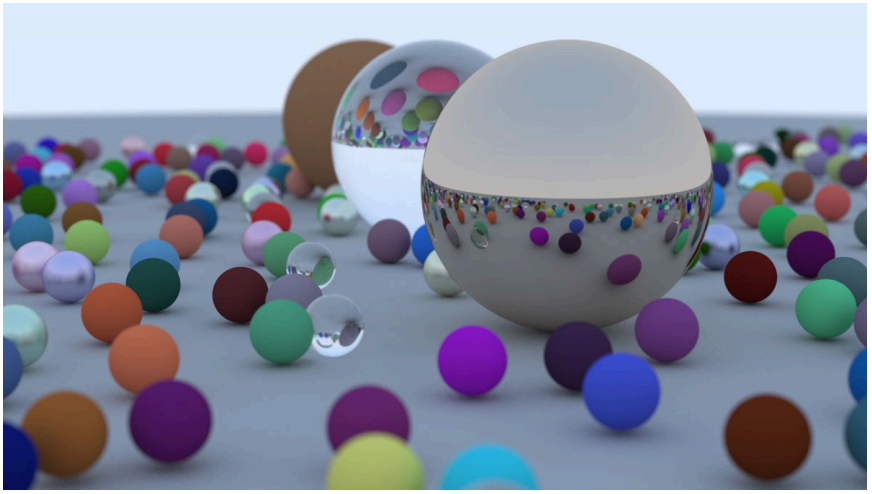
Pick any of these as a starting point… then graduate to some reference materials, books… Keep continuously learning via conferences, blogs, articles etc…

Three tracks:

- **[OFL]** Offline Rendering - What's used in movies! Non-real-time, we have a lot of time to generate an image. Theory of lights, scattering, materials. Monte-carlo solutions of the rendering equation: Ray tracing, Path tracing.
- **[RTR]** Real-Time Rendering - What's used in video games! Some of the same theory as offline, but the practical implementation has to deal with GPUs, rasterization, shaders… On this track, one can start from the "low level" and learn a 3D API, shaders etc, writing a renderer from scratch, or start from a 3D engine (e.g. Unity, Unreal…), learn higher level concepts and then graduate to understand how things work at a lower level.
- **[ART]** Fabrication, generative, digital art and so on. Can use both real-time and off-line rendering, and other techniques for procedural generation of geometry, patterns etc…

[OFL] First steps:

[OFL] Ray Tracing in one weekend



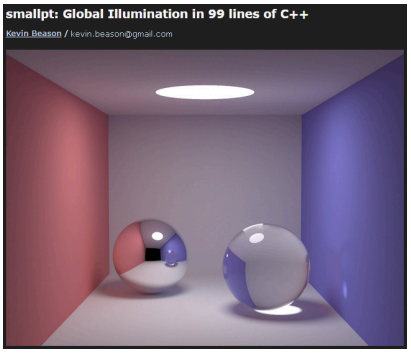
A fun and cheap booklet by Peter Shirley who authored many other (longer) books on computer graphics. It guides through a weekend project, building a simple offline renderer. There's nothing like raytracing to get hooked on 3d graphics programming!

https://raytracing.github.io/

Take this project if you're interested in generating photorealistic images, and want to understand the physics of light transport. From here, you can graduate both towards the intricacies of how to make

path-tracing practical for complex scenes and lighting scenarios - or towards a deeper understanding of the physics of light and materials - which is relevant also for real-time (GPU, rasterization-based) rendering.

Of course, it's impossible not to mention **"PBRT"** https://www.pbrt.org/ here… and we will, but in the "reading material" section below - as I feel like PBRT goes beyond "tinkering" - it requires some serious commitment!



Related: Tinker with an educational path tracer.

SmallPT is a tiny program (99 lines of C++ code!) which generates photorealistic images. Another great way of getting started with offline rendering! http://www.kevinbeason.com/smallpt/ …before "Raytracing in one weekend" it was probably the best starting point to tinker with offline rendering.

In its "raw" form it's not too easy to understand (but fun to tinker with), but on the homepage there is a slide deck by David Cline that does a great job at explaining each single line. https://drive.google.com/file/d/0B8g97JkuSSBwUENiWTJXeGtTOHFmSm51UC01YWtCZw/view

There are a few other path tracers that are similarly made with simplicity in mind. For example: **MiniLight** https://www.hxa.name/minilight/, http://www.smallvcm.com/ (which implements a quite advanced light transport algorithm - thus - not the easiest to play with) and **Nori** https://www.cs.cornell.edu/courses/cs6630/2012sp/nori/ - http://rgl.epfl.ch/courses/ACG17

[OFL - mostly] Scratchapixel.

Part beginner's textbook, part guide for practical exercises, Scratchapixel has some truly **great** content.
The downside of this website is that it's still work-in-progress, and certain chapters are not as good as others.

Take this path if you want a bit more of a structured and theoretical introduction to the topic - more akin to reading a book. You'll probably want to write your own small project along the way, and use the resources in "continuous learning" to fill gaps.

Focuses on offline rendering, but the theoretical basis are the same for real-time… https://www.scratchapixel.com

[RTR] First steps:

[RTR] Play with WebGl2!

Sadly perhaps, WebGl is today one of the few more "beginner friendly" APIs that are not discontinued…

This interactive web book provides a good introduction and all the code is tinkerable! https://webgl2fundamentals.org/

Take this path if you want to understand how GPUs are used, the lower-level details of images and 3d objects, and how to build your own real-time 3d engine. This is often done by people who are interested in videogame programming, but real-time rendering is also important for interactive/responsive visualization in general. Typically, this path leads eventually to C/C++ and using modern graphics APIs such as Vulkan and DirectX 12 - to get the maximum performance.

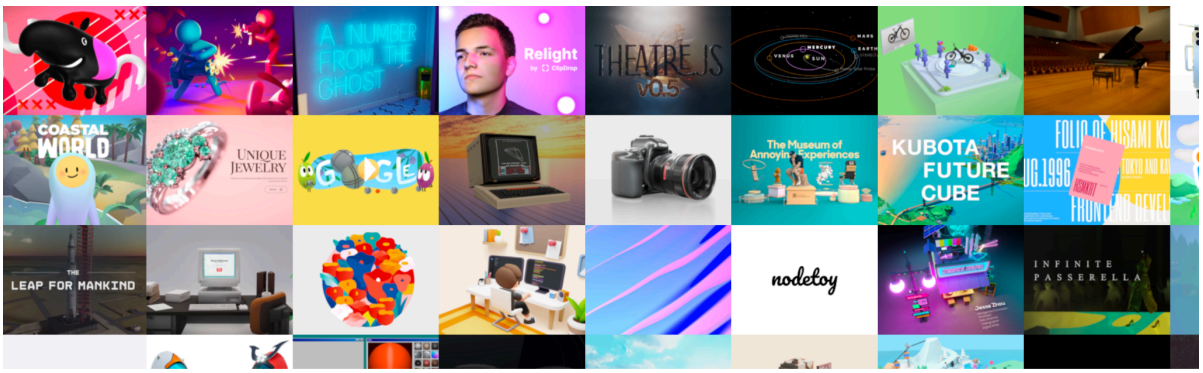
Alternatives:
http://www.webglacademy.com/ is very good and interactive, it also goes into some advanced topics.
https://github.com/stackgl/shader-school
http://learnwebgl.brown37.net/

I would not necessarily use WebGL directly btw - there are wrappers that make programming it much more pleasant.

PicoGL (a WebGL2 wrapper) is **amazing**. It's truly a nice API to use, but to my knowledge there are no tutorials that use that wrapper https://tsherif.github.io/picogl.js/ - seems a bit abandoned, but not everything requires continuous improvement. I imagine most web graphics fervor these days is around webGPU.

Alternatives: SwissGL is also interesting
https://github.com/google/swissgl and the same is for twgl https://github.com/greggman/twgl.js and regl http://regl.party/ - all WebGL2, but I prefer picogl to these.

[RTR] Three.js



If WebGL is a good starting point to learn a GPU 3d graphics API today, three.js is a great first 3d engine to begin with.

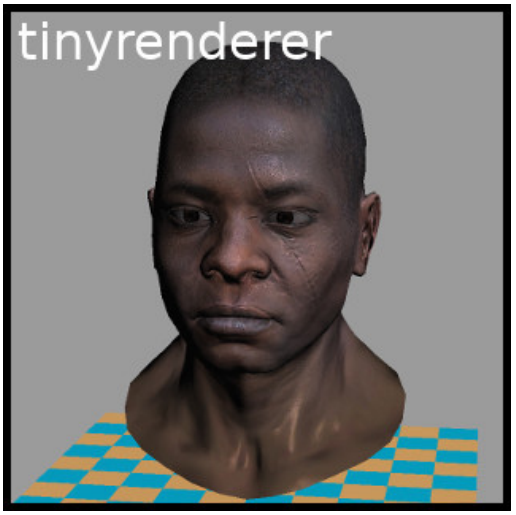
This means it's more of a high-level introduction than working with a GPU api, directly working with scenes, objects, lights, textures etc… https://threejs.org/docs/#manual/en/introduction/Useful-links

After this, one can graduate to engines like **Unity, Unreal or Godot**, or learn a 3d API and write an engine from scratch…

Interactive 3d Graphics, an udacity course from Eric Haines (one of the authors of Real-Time Rendering) goes through the basics using WebGL and three.js.The course is a bit lengthy: https://www.udacity.com/course/interactive-3d-graphics--cs291

The three.js website also has its own list of resources, and the documentation itself is quite nice
https://threejs.org/manual/#en/fundamentals

[RTR] TinyRenderer



Similar in spirit to Raytracing in one weekend, but I don't think it's quite as easy / great first project, unfortunately. Still, worth mentioning.

The idea is to write a software rasterizer (the algorithm behind modern GPU rendering)

https://github.com/ssloy/tinyrenderer/wiki

If you want to start from software rasterization - perhaps a better, more guided path, is to subscribe to one of Pikuma's courses: https://pikuma.com/courses

3D Computer Graphics Programming

37 hours

Learn to create a 3D software renderer from scratch using the C language.

Alternatives:
https://tayfunkayhan.wordpress.com/2018/11/24/rasterization-in-one-weekend/
https://www.gabrielgambetta.com/computer-graphics-from-scratch/00-introduction.html
https://lisyarus.github.io/blog/posts/implementing-a-tiny-cpu-rasterizer.html

[ART] First steps:

[ART] The Nature of Code / Processing

Processing is a language developed by MIT's Aesthetic and Computation Group to ease generative graphics programming.

While it won't teach you much about the theory of computer graphics, it provides a fun and easy framework to play with - and it's a great tool for generative art!

https://processing.org/

Good books on Processing are listed here:
https://processing.org/books/
I would recommend **“The Nature of Code”** https://natureofcode.com/

Other resources:
https://thecodingtrain.com/tracks/learning-processing/processing
https://inconvergent.net/generative/
https://p5js.org/ javascript version, has an online editor. See also: https://www.openprocessing.org/

Alternatives: An OSX-only, Python based alternative is **DrawBot**
http://www.drawbot.com/

[RTR/ART] Artistic shader coding

The Book of Shaders

by Patricio Gonzalez Vivo and Jen Lowe

This is a gentle step-by-step guide through the abstract and complex universe of Fragment Shaders.

The “book of shaders” is truly a marvel, an interactive online book going through the basics of pixel shaders and showing how to make art with them.

http://thebookofshaders.com/

It’s not a full introduction to 3d real-time rendering as it only covers pixel shaders to directly generate images, no 3d geometry involved, but it’s a good read to familiarise with the shading language that is used for real-time rendering.

Once you're done, you can graduate to **ShaderToy**. Shadertoy is the best way to start tinkering with GPU programming. Made by Inigo Quilez and Pol Jeremias, masters of procedural 3d graphics, it quickly grew into a vibrant community of shader authors.

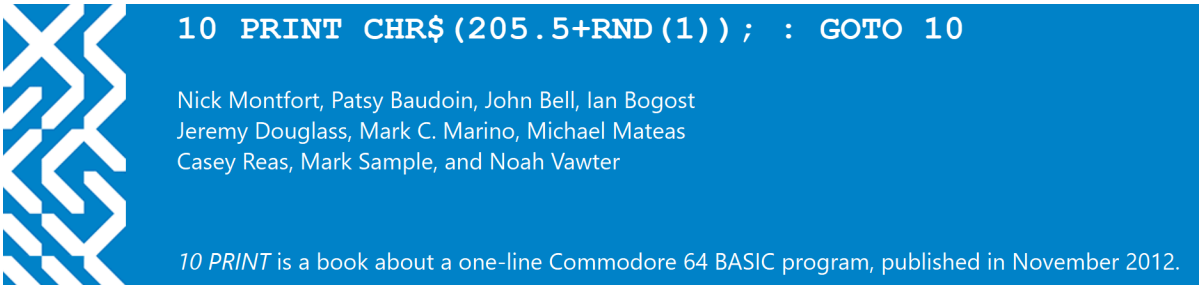
As for "the book of shaders" here shaders are directly used to generate procedural images, without 3d meshes (geometry). This is not how 3d real-time rendering usually works, but it is a great introduction to shader and GPU programming.

One really great introduction is this YouTube video by kishimisu https://www.youtube.com/watch?v=f4s1h2YETNY

See also: https://gamedevelopment.tutsplus.com/tutorials/a-beginners-guide-to-coding-graphics-shaders--cms-23313 http://pixelshaders.com/

When you feel confident of the basics, Inigo's website is a treasure trove of advanced procedural shader stuff! http://www.iquilezles.org/www/articles/raymarchingdf/raymarchingdf.htm

[ART] Book: 10 PRINT CHR$(205.5+RND(1));:GOTO 10.



I haven't read this book yet - so I can't comment on it, but it looks interesting.

Best thing, there is a free version available online! https://10print.org/

Vintage vibes also at: https://www.bbcmicrobot.com/ and https://tixy.land/

Useful software:

These are not small projects / do not have (or I could not find) accompanying materials that use them in a "tinkering" way for beginners - but they are still great to play with especially when starting.

[RTR] Sokol is a great, easy wrapper of modern 3d APIs - afaik no tutorial uses it, but it's a great starting point once one learns the basics (and it stays relevant… forever)
https://github.com/floooh/sokol

[RTR] RenderDoc is a must-have tool https://renderdoc.org/

[RTR] CToy. A fun, graphic-oriented, C99 live-coding framework. Not a project or tutorial per se, but it can be used to tinker with graphics "easily" - and because it's live-coding, one can start with the examples and modify the code…
https://github.com/anael-seghezzi/CToy

[RTR] Unity and Godot are by far the easiest "real" 3D engines to start tinkering with.

[RTR,ART] KodeLife - similar to ShaderToy, but offline and allows you to experiment with all kinds of shaders.
https://hexler.net/software/kodelife

[OFL,RTR] Yocto single-header libraries
https://github.com/xelatihy/yocto-gl

---

2 - Continuous Learning.

Youtube & al.

[RTR, ART] Freya Holmer's YT channel.
https://www.youtube.com/@Acegikmo Lots of amazingly well produced tutorials, mostly around "math-y" subjects.

[RTR, OFL] Cem Yuksel's YT channel.
https://www.youtube.com/@cem_yuksel amazingly well produced courses from the University of Utah!

[OFL] Pixar In a Box.
A series of videos on Khan Academy illustrating various aspects of how Pixar makes movies. There is a section dedicated to rendering.
https://www.khanacademy.org/computing/pixar/rendering

[RTR] Handmade Hero (YouTube) is an amazing resource on how to build an engine from scratch https://handmadehero.org/

I didn't put it in the "tinkering" section because albeit it starts from zero, so it is very beginner friendly, it's not a short project to follow/undertake.
The "network" is also very good https://handmade.network/projects

[RTR] A DirectX 11 engine tutorial, also built from scratch, by **ChiliTomatoNoodle**
https://www.youtube.com/playlist?list=PLqCJpWy5Fohd3S7ICFXwUomYW0Wv67pDD

[RTR] A modern Vulkan renderer, from scratch, by Arseny Kapoulkine
https://github.com/zeux/niagara

[RTR, ART] Ferris Streams Stuff
https://www.youtube.com/channel/UC4mpLlHn0FOekNg05yCnkzQ
Ferris is AMAZING. Demoscene and more!

[RTR, OFF] Two minute papers.
Explaining research papers simply. Both recent publications and some of the seminal works.
A lot of this is dedicated to computer graphics as the author is a researcher in the field. It won't allow you, in most cases, to replicate results in practice, but it's just nice to be exposed to the concepts.

The youtube channel also hosts the lectures for an **offline graphics course**:
https://www.youtube.com/watch?v=pjc1QAI6zS0&list=PLujxSBD-JXgnGmsn7gEyN28P1DnRZG7qi

Sebastian Lague's channel is not dedicated exclusively to computer graphics https://www.youtube.com/@SebastianLague/videos but often his projects are about 3d rendering. Each video is on a different project.

Similar is **Acerola's channel**, a technical artist illustrating different projects in each video: https://www.youtube.com/@Acerola_t - https://www.youtube.com/watch?v=O-2viBhLTqI

Websites

[RTR] Learning Modern 3D Graphics programming
https://paroj.github.io/gltut/
Uses OpenGL, which is still a wise choice for beginners, as OpenGL CAN be friendly.

Unfortunately, it also has a lot of legacy mixed with modern concepts. The legacy stuff is easier, the modern parts are better/faster, in general it's a bit of a mess and it's being phased out on some systems (e.g. Mac).

There are many other websites dedicated to OpenGL, but I highlighted the one above because it tries to be a bit more general than websites that really focus on the API itself.

Alternatives (all OpenGL-based):
https://learnopengl.com/ https://antongerdelan.net/opengl/
http://www.opengl-tutorial.org/ https://open.gl/
https://ogldev.org/index.html
https://github.com/bartvbl/A-Hitchhikers-Guide-to-OpenGL
http://nehe.gamedev.net/
https://duriansoftware.com/joe/an-intro-to-modern-opengl.-table-of-contents

[RTR] Learn {Vulkan,DX12} from code samples
Still WIP, but promising:
https://github.com/PAMinerva/LearnVulkan
https://github.com/PAMinerva/LearnDirectX

See also: https://www.3dgep.com/learning-directx-12-1/

[RTR] Modern Real-time Graphics with Javi Agenjo
https://tamats.com/learn/realtime-graphics/

[RTR] Game Shaders for Beginners.
Very well explained collection of "popular" 3d effects and how to code them. https://github.com/lettier/3d-game-shaders-for-beginners

[RTR] UCSB CS291: Real-Time High Quality Rendering
There are (luckily!) still many universities teaching Computer Graphics with content available online… I am highlighting this course though because it is the only one I've seen that is not only about real-time rendering (which itself is rare, most courses cover offline), but specifically aims at teaching some more modern and advanced techniques.
https://sites.cs.ucsb.edu/~lingqi/teaching/cs291a.html
Content available here:
https://sites.cs.ucsb.edu/~lingqi/teaching/games101.html
https://sites.cs.ucsb.edu/~lingqi/teaching/games202.html

Other university courses - most often focus on offline rendering or cover real-time (rasterization) only theoretically:

[OFF] CMU Computer Graphics Lectures
http://15462.courses.cs.cmu.edu/fall2020/
[OFF, RTR] MIT 6.837 Lectures
https://www.youtube.com/playlist?list=PLQ3UicqQtfNuBjzJ-KEWmG1yjiRMXYKhh
[OFF] Utrecht University
https://www.youtube.com/playlist?list=PLDFA8FCF0017504DE
[OFF, RTR] UC Davis
https://www.youtube.com/playlist?list=PL_w_qWAQZtAZhtzPI5pkAtcUVgmzdAP8g

[RTR] CatlikeCoding Unity tutorials: An excellent series on Unity
https://catlikecoding.com/unity/tutorials/

[RTR] Game Engine Architecture. This one, by Nikola Dimitroff http://nikoladimitroff.github.io/Game-Engine-Architecture/ is great, but more advanced. For Unreal.

[RTR] GPU Architectures by Maurizio Cerrato.
https://drive.google.com/file/d/12ahbqGXNfY3V-1Gj5cvne2AH4BFWZHGD/view

[RTR] A trip through the graphics pipeline by Fabian Giesen**.**
https://fgiesen.wordpress.com/2011/07/09/a-trip-through-the-graphics-pipeline-2011-index/
Both are great read on how GPUs work!

https://betterexplained.com/ good reference for many mathematical concepts, for rendering what you want to look up is their introduction to linear algebra.

http://immersivemath.com/ila/index.html Another GREAT reference for math / linear algebra!

Also a must read is **https://ciechanow.ski/archives/** - few articles but excellent, see https://ciechanow.ski/lights-and-shadows/

Aras' has a great blog, but it's generally more for experts. His series on writing a Path Tracer though could be useful here **https://aras-p.info/blog/2018/03/28/Daily-Pathtracer-Part-0-Intro/**

Blogs

Not many blogs are beginner friendly, but these are good resources:

[RTR] Fabien Sanglard's blog & books.

Fabien does an -awesome- work reviewing, dissecting and explaining old (public) source-code of 3d engines.
Very instructive! http://fabiensanglard.net/

He even recently published a series of books, dissecting the original Wolfenstein 3D game and DOOM:
https://fabiensanglard.net/three_books_update/index.html

Related (book): The Black Book of Graphics Programming. The DOS, x86 and VGA sections were dated even by the time the Special Edition came out, but the mindset is still valuable, and the Quake work is mostly still relevant.
Could be considered a "companion" to Sanglard's website. Now free and available online as PDF.

[RTR] Various "graphic analysis"/reverse engineering blog posts.
- Adrian Courreges keeps a list here, of his own work and other's:
https://www.adriancourreges.com/blog/2020/12/29/graphics-studies-compilation/

Similar to Fabien's website, this blog features analysis of shipped videogames.
In this case, it's not about source code, but reverse-engineering graphics techniques by looking at frame captures (via the excellent RenderDoc debugger).

Other similar blogs:
- "Simon's old VFX" (same concept, but for older games)
http://simonschreibt.de/gat/simons-old-vfx/
- **Froyok** https://www.froyok.fr/articles.html
- **Behind the pretty frames**, like Adrian's series above, dissects rendering of commercial games
https://mamoniem.com/category/behind-the-pretty-frames/
- Related: **"how Unreal renders a Frame"** on **"interplay of light"**:
https://interplayoflight.wordpress.com/2017/10/25/how-unreal-renders-a-frame/,

[RTR] https://www.jendrikillner.com/tags/weekly/ chronicles what's happening in the real-time rendering world and…
[RTR] https://halisavakis.com/category/technically-art/ does the same for technical art.

[RTR] https://blog.demofox.org/ and **https://erkaman.github.io/articles.html** write of miscellaneous topics around computer graphics, but their articles are usually clear and beginner friendly.

[RTR] Filament - technically it's a software (library) but what we are interested in here is its documentation, which is an amazing reference to the world of Physically Based Rendering: https://google.github.io/filament/Filament.md.html

Books

[RTR] Real-Time Rendering.
Probably the most well-known "encyclopaedic" book for 3D real-time rendering.
It is theoretical, but not hard to read. It's great both as a reference text and to read end-to-end.
It will not guide you through practical projects though, and it will not delve deeply into any subject, it's meant to provide the necessary background to understand more advanced papers and presentations.

[RTR, OFF] Game Math (free, online).
https://gamemath.com/book/

[RTR, OFF] Graphics Codex.
This is another great reference text, very well done. In contrast to Real-Time Rendering, this has an offline rendering slant, but lots of the underlying math and physics are relevant to real-time computer graphics as well.
The author keeps also a page of small projects and exercises on http://graphicscodex.com/projects/projects/

[OFL] Physically based rendering: from theory to implementation.
https://www.pbr-book.org/ Known as "**PBRT**", this one is not about real-time rendering but offline path tracing.
Most of the theory applies to physically based shading in games as well, and the big perk of this book is that it's written as "literate programming", the book is itself the source-code of a raytracer. It is not a quick read, but by the end of it you'll have a nice photorealistic renderer!

After PBRT, you might be ready to have a look at "research" oriented path tracers like Mitsuba https://www.mitsuba-renderer.org/ and Lightmetrica http://lightmetrica.org/

[RTR, OFF] Computer Graphics, Principles and Practice.
Another very notable "encyclopaedic" book, tends to be a bit more theoretical than RTR, but it's a great read…

[RTR] Game Engine Architecture.
One of the very few books written by an industry professional (Real-Time Collision Detection is another example, as the book below).
Recommended. Covers also workflows, debugging, using visual studio etc. https://www.gameenginebook.com/

[RTR] Practical Rendering & Computation with Direct3d 11.
This book guides through Direct3D 11 and it's written by industry experts (Pettineo is the rendering lead at Ready at Dawn).
It also comes with / builds a 3d rendering engine.
The only downside is that it might be a bit overly specific delving into lots of details specific to DX11.

[RTR] Mastering Graphics Programming with Vulkan. This book is a must have. It does assume familiarity with Vulkan as a prerequisite and from there builds a fairly modern (albeit still "educational") engine around it!
https://www.packtpub.com/product/mastering-graphics-programming-with-vulkan/9781803244792

Eric Lengyel's books.
http://www.terathon.com/lengyel/ He has been recently working on a series of books on the fundamentals of 3d engines.
His writings are known for their precision and quality!

Honourable mentions

These are also important resources for beginners, but I've created a separate section because the contents might only partially apply.

Real-Time Collision Detection.
This book, as the title implies, is not related to Computer Graphics.
Nonetheless it's -really- well written and it's more general than it might seem. The first few chapters go through essential math in a very clear way, while end ending talks competently about optimizations and numerical issues. Also, most of the spatial subdivision structures discussed also apply to CG (visibility).

[RTR] Game Programming Patterns.
Chapter 6 only, it explains the basics of modern CPU performance quite clearly. For the rest, I **do not** endorse design patterns. Especially avoid chapter 2.
Available for free here: http://gameprogrammingpatterns.com/

Visualising Quaternions.
…is of course, as the title says, on a very specific topic, but it's a quite handy guide and it is written extremely well.
https://shop.elsevier.com/books/visualizing-quaternions/katsaropoulos/978-0-12-088400-1

Glassner's Principles of Digital Image Synthesis.
https://www.glassner.com/portfolio/principles-of-digital-image-synthesis/
He also has a book on deep learning, which is a great introduction to the topic, very well written, easy.
https://www.amazon.ca/Deep-Learning-Approach-Andrew-Glassner/dp/1718500726

[RTR] Metal by Example - looks like a solid book, and the blog is useful as well https://metalbyexample.com/the-book/
Related: The Vulkan Tutorial is good too, but Vulkan is definitely not beginner-friendly https://vulkan-tutorial.com/