Big on Graphics

June 2020 Edition

BIG ON GRAPHICS

Tips, Tutorials, Examples, Snippets, Pitfalls, Discussions and News

Dawn of a new era
(No more hiding but at what cost)

Fractals
Hypnotic beauty

Vulkan
Top 5 programming mistakes

Parallax Mapping

Subsurface Scattering
(Real-time Approximations)

Chaos and Randomness
(Realism in Imperfection)

1092823920326903337

Inspiring Creativity

BIG ON GRAPHICS

Kenwright

Subscribe to
Big on GRAPHICS Digital Edition Today!

Exciting comprehensive resource for all your computer graphics technology solutions.

"How to" guides, trends, latest news, and technical articles on a wide range of state of the art subjects.

EDITOR-IN-CHIEF, BIG ON GRAPHICS MAGAZINE Kenwright

BIG ON GRAPHICS

Welcome to the inaugural issue of Big on Graphics: a magazine of a new kind. With its broad scope bridging (yet not limited to) computer graphics while taking into account software engineering, artificial intelligence, applications and hardware technologies, the magazine is dedicated to challenging and questioning rather than to simply agreeing and repeating existing media. The magazine aims to stimulate debate and and raise questions for thought. This combined with educational material to help you understand how computer graphics works, how it functions, the limitations, benefits and trends (taking a look under the bonnet) - including indie, experimental and homebrew topics. Recently, Computer Graphics technologies have become an essential tool in nearly every walk of life - mobile, visualization, computation, entertainment, movie industry and more.

There are currently a number of organisations and research programs around the world that explicitly or implicitly focus on computer graphics. Yet, despite impressive successes and growing interest in the graphics domain, wide gaps continue to separate different approaches from each other necessary to rise and address some of the biggest graphical challenges of our age. While disjointed technical communities may speak different languages and pursue independent goals, at least they're pushing the limits of computer graphics. In this situation, the mission of the magazine is to foster a wider understanding of the unifying graphical topics and highlight interesting areas for concern. In doing this, the magazine will provide insights for important questions. e.g.: which is the best computer graphics architecture for real-time environments? What are the aesthetic and computational limitations for state of the art CGI models? How to give human-like emotional feeling and creativity to visual models? Both mature and new cutting edge research are welcomed by the magazine, provided they have a strong topic of interest and aligns with the magazines theme (opinion reviews, ad-hoc approaches, mathematical concepts simplified for readers and so on).

Without a double, a lot of has happens in computer graphics in recent years, some things have been really, really amazing and deserve recognition for their worth and value they have made to the industry. Of course, computer graphics is multidisciplinary subject, and can really make a difference to the world. Unfortunately, a lot of discoveries and advancements don't always make a substantial impact, usually because of the technical challenges or are difficult to accept (stuck in certain ways/old approaches). The magazine hopes to share information and help explain some of the jargon and headaches circling the computer graphics world.

Computer graphics makes most of us devout believers due to the inherent beauty that is possible - which stimulates our imagination. You might not agree, or you might say it's a bias, but how can you not love the computer graphics discipline. While the topic is challenging on many levels, the mathematics, software engineering and the artistic component, the rewards at the end are worth the effort. However, unlike some subjects, computer graphics is constantly changing and evolving (stay on your toes and you must also evolve and adapt). You can't watch a film, play a game or use your mobile phone without computer graphics getting used.

This magazine is not for financial gain, it's done for the passion of the subject. The love of code, mathematics and how they come together to create a banquet for the eyes.

THE INDEPENDENT GUIDE Big on Graphics Magazine is an independent guide to technologies in and around computer graphics. Our mission is to explore, question, explain and review computer graphics (software and hardware). Importantly, the purpose of the magazine is to remain objective and relay a variety of interesting information from facts.

CONTACTING EDITORS We welcome comments from readers. Email your comments to the editor-in-chief. We welcome articles and illustrations, however, before submitting manuscripts or material, please get in touch to discuss your proposal

No part of this publication may be reproduced, stored in a retrieval system or transmitted in any form or by any means, electronic, mechanical, photocopying, recording, scanning or otherwise, except as permitted under the Copyright Act, without either the prior written permission of the Publisher, or authorization.

Most libraries, developers and published titles are trademarks of the companies. Use of a trademark to identify a product commented upon in this magazine should be not be construed as implying the sponsorship of the trademark holder, nor, conversely, should use of the name of any products without mention of the trademark status be construed as a challenge to such status.

LIMIT OF LIABILITY/DISCLAIMER WARRANTY The publisher and the author make no representations or warranties with respect to the accuracy or completeness of the contents of this work and specifically disclaim all warranties, including without limitation warranties of fitness for a particular purpose. No warranty may be created or extended by sales or promotional materials. The advice and strategies contained herein may not be suitable for every situation. This work is sold with the understanding that the publisher is not engaged in rendering legal, accounting, or other professional services. If professional assistance is required, the services of a competent professional person should be sought. Neither the publisher nor the author shall be liable for damages arising here from. The fact that an organization or Website is referred to in this work as a citation and/or a potential source of further information does not mean that the author or the publisher endorses the information the organization or Website may provide or recommendations it may make. Further, readers should be aware that Internet Websites listed in this work may have changed or disappeared between when this work was written and when it is read.

Table of Contents

BIG ON GRAPHICS

Big on Graphics

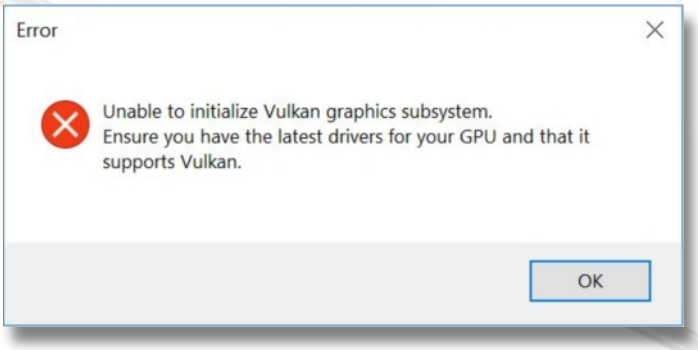
WHY IS VULKAN SO COMPLICATED?

Does the Vulkan API trade simplicity for control?

For seasoned software developers, who've been exposed to a variety of API and programming languages, they will argue, that we need Vulkan, that the trend in recent years has been to 'simplify' (or spoon feed programmers, which is wrong! Hiding away the complexity, such as, memory management and simplifying/limiting the API, has hindered the developers abilities (it's like developers can't be trusted to write safe, robust, reliable code).

People who have mastered core mathematical concepts and low-level programming principles, find the transition to new languages and API relatively easy (Vulkan is just another API).

In some cases though, developers find Vulkan difficult, as they do not have a solid understanding of the underlying computer architecture (e.g., how memory works, stacks and low-level registers).



While many new API and programming languages are designed to be more 'user friendly', this comes at a cost. For example, you do not have complete control, such as, defining and allocating your own memory and data structures. You might think that someone very smart is doing it for them under the bonnet. But this is not always the truth. As you'll be aware, there are many times, when you launch and application or a library only to be left waiting, and wondering, why you have a high-end multicore computer, yet your library is running slow and inefficiently.

Previous Graphical API where based on old ways of thinking (stacks and single-core systems). Many people consider this approach sufficient. Vulkan on the other hand is not limited by this way of thinking. Personally, Vulkan's complexity should not be seen as a disadvantage. The long term benefits outweigh the initial time and commitment necessary to learn how Vulkan works.

The one thing that Vulkan lacks, is any 'default' options. The majority of the time, when you're writing a Vulkan application, is you need to do all the work.

Then there is the Vulkan design philosophy: You never pay for what you don't use.

Vulkan means you don't have to worry about excess bagage. The legacy of backward compatibility, both with earlier versions (i.e., OpenGL).

As a bonus, the Vulkan API standard library is lean, and adding to it is a conservative process, so sometimes you need to go outside of the standard library for what you need, and that means dealing with either your own libraries, or third party libraries of inconsistent style, quality, and complexity.

Right Tool for the Right Job

Although Vulkan is harder to use, it gives you absolute control over the machine resources, and if you understand your hardware architecture and you're a good coder that usually means noticeably higher speeeds: There are tests were Vulkan is multiple times faster than previous generation API.

Novel, easy to use API (OpenGL) are good for simple cases, like the small practices in a computer science course. However, real world applications (triple-A video games, real time systems, compute vision, profesional software development, operative system development, driver development) have extreme requirements simple API can't meet.

You start coming across such cases when you try to develop hardcore graphical demos/programs. Ever tried to make a realistic interactive 3-dimensional scene? It's too slow to work, unless you start thinking about optimization. Ever tried to create heavy graphical algorithms that need millions of iterations to work? You don't want your graphics card or program to 'stall'. Think of games, such as, Halo, Call of Duty and other AAA games and the standard/quality expectations. Those expectations were yesterday, tomorrows are even higher! There's a reason why successful AAA games are 'successful'. It's laughable how some graphical programs used for doing biology-related simulations take days to run (in a distributed processing environment) when the same solution would take hours if optimized for the Vulkan API (real life story).

None of this would be possible if the API took care of all the memory-related work for you, if it had the need of a garbage cleaner, if it needed you to have a framework or runtime machine to interpret operations in the background, or if the program wasn't compiled to binary before running. **Vulkan is hard because you need to do stuff that previous API did under the hood**. Vulkan is hard because it's similar to the way machines speak, not to the way humans speak. Vulkan is hard because the result should be easy to compile, instead of easy to debug/mantain.

It's a trade, and it's the Vulkan API that was used to build the Graphical/Compute systems of the future, the latest vidoe games, graphical design packages, modelling packages are probably already using or moving over to Vulkan. It works when you understand it and the world is full of capable Vulkan API developers. Mastering Vulkan is akin to riding a bike or learning mathematics, once you get over the curve, it's not so bad.