Vulkanised

Feb 8-10, 2027

Place:

Kortrijk, Belgium

Title:

Scientific Visualization with Vulkan in WPF

Author:

Sven Axmann

Abstract:

Scientific visualization applications often rely on existing rendering libraries that abstract away all the underlying complexity of GPU programming. But what if the requirements regarding visualization and efficient data handling are too specialized to fit those abstractions efficiently?

We faced this problem while developing a new rendering backend for ArtemiS Suite, an analysis, processing and recording application for audio and vibrations. It requires real-time curve, line, and spectrogram plotting, alongside 3D models and text, with many unique variations. At the same time, the renderer must efficiently visualize and present live data from hundreds of sensors across dozens of independently updating render areas.

After evaluating existing solutions, we decided to build our own rendering engine in C++ with a Vulkan backend. Choosing Vulkan introduced another challenge: ArtemiS Suite is a WPF application, which means that the renderer must coexist with WPF's DirectX backend, obey its threading constraints, and avoid the airspace problem associated with embedding separately rendered surfaces.

We started without previous Vulkan experience and worked our way from initial prototypes to a production-ready renderer using Vulkan 1.3 features such as dynamic rendering and buffer device addresses, multithreaded command-buffer recording, indirect drawing and VK_EXT_external_memory_host. The latter allows Vulkan to render directly into the backbuffer of WPF's WriteableBitmap control without conventional Vulkan presentation and without additional copy overhead.

During development and testing, profiling with RenderDoc, NVIDIA Nsight, and Tracy led us through several architectural and performance optimizations. The resulting system can sustain 30-60 FPS with up to 100 independently rendered areas in a single WPF window, using our swapchain-like mechanism to coordinate rendering and presentation through WPF.

This talk presents the architecture behind the integration, the Vulkan techniques, performance considerations, and the struggles and lessons we learned along the way while integrating a renderer into a UI framework that predates Vulkan by roughly a decade and was not designed for this kind of interoperability.

Find more event abstracts in our >> abstracts archive <<

HEAD acoustics GmbH
Ebertstraße 30a
52134 Herzogenrath, Germany

Sven Axmann
Mail: info@head-acoustics.com
Telephone: +49 2407 577-0