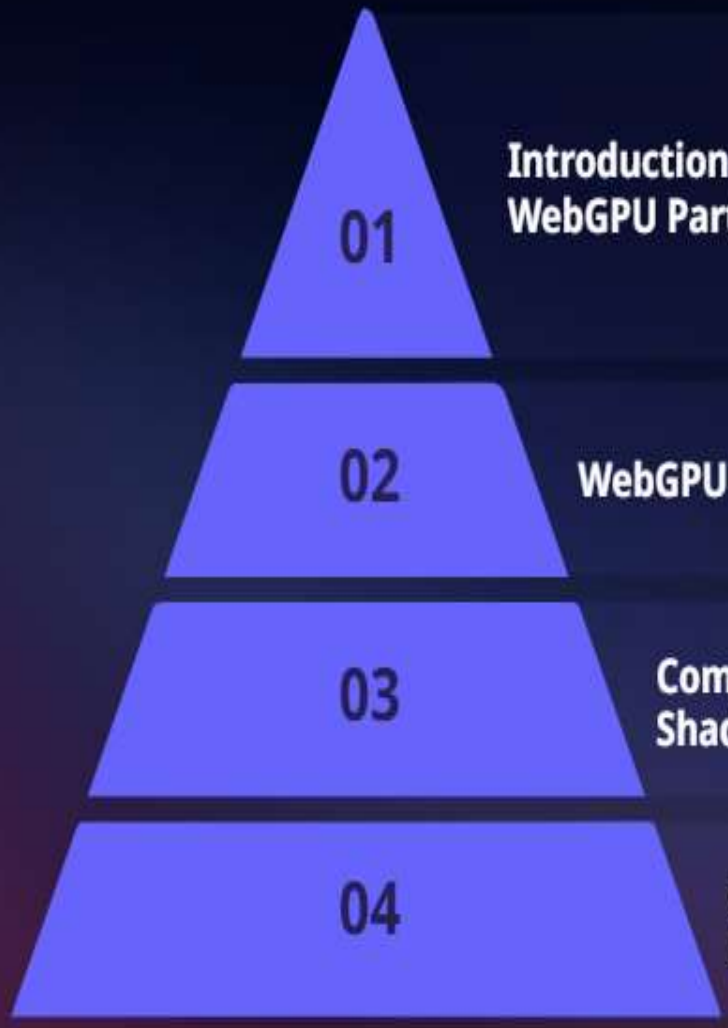
WebGPU Particle Systems Unleashed

Performance Through Compute Shaders and Render Shaders

Konstantin Ljakoski

WebGPU Fundamentals

WebGPU particle system overview



Introduction to WebGPU Particles

WebGPU enables high-performance particle systems in browsers, utilizing GPU acceleration for rendering thousands of particles simultaneously with minimal performance impact.

WebGPU Fundamentals

Modern graphics API providing low-level GPU access in browsers. Offers superior performance compared to WebGL through reduced CPU overhead and efficient memory management.

Compute vs Render Shaders

Two primary shader types in WebGPU. Compute shaders process data in parallel, while render shaders transform geometry and apply visual effects.

Key Shader Differences

Compute shaders operate on arbitrary data structures, while render shaders focus on vertex/fragment processing. Compute offers more flexibility for non-graphics tasks.

Shader types explained

01

Compute vs. Render Shaders

Two primary shader types in WebGPU with distinct purposes. Compute shaders process data in parallel while render shaders handle visual output.

02

Compute Shader Capabilities

Optimized for general-purpose parallel computation. Processes large datasets efficiently without graphics pipeline overhead, ideal for particle simulations.

03

Render Shader Functions

Transforms vertices and calculates pixel colors. Works within the graphics pipeline to create visual output directly to screen.

Performance considerations

FPS Comparison

Compute shaders maintain higher framerates with increasing particles due to parallel processing capabilities and optimized memory access patterns.

02

01

Particle Count Impact

Higher particle counts significantly affect performance in both shader types, with compute shaders handling larger quantities more efficiently.

03

Memory Bandwidth Considerations

Compute shaders reduce memory bandwidth requirements through shared memory utilization, improving performance for particle-heavy simulations.

Compute vs. Render Shaders

Key differences

Memory Access Patterns

Compute shaders allow flexible memory access, whereas render shaders have restricted access following the graphics pipeline flow.

02

01

03

Compute vs. Render Shaders

Compute shaders process data in parallel without fixed pipeline, while render shaders follow graphics pipeline for visual output.

Performance Scalability

Compute shaders scale better with particle count, maintaining higher FPS when processing thousands or millions of particles.

Compute shader advantages

Parallel Processing Capabilities

Compute shaders excel at parallel processing, handling thousands of particles simultaneously without taxing the main CPU thread.

1

Memory Management Efficiency

Direct GPU memory access eliminates redundant data transfers, allowing compute shaders to process particle data where it resides.

2

Higher Performance Ceiling

Compute shaders scale better with increasing particle counts, maintaining higher framerates compared to traditional render-based approaches.

3

Particle count impact

01

Performance Scaling with Particles

As particle count increases, performance impact varies between shader types, with compute shaders maintaining better efficiency at higher particle counts.

02

Rendering Pipeline Bottlenecks

Traditional render shaders experience significant slowdown with increased particles due to pipeline overhead, while compute shaders maintain more consistent performance.

03

Memory Management Differences

Higher particle counts require optimized memory access patterns, where compute shaders excel through shared memory and workgroup synchronization capabilities.

Thanks!