Sean Parkinson

portfolio@seanparkinson.net | seanparkinson.net | Sydney, Australia

Education

Bachelor of Computer Science (Honours) Feb 2024 – Present (Expected 2027)

University of Technology, Sydney

Relevant Coursework: Data Structures & Algorithms (HD), Theory of Computing Science (HD)

Experience

Industrial Sciences Group (ISG) Jul 2025 – Present

Software Engineer

- Shipped a production RF link budget tool now used by defence clients to replace manual field testing, implementing ITU loss models over a Longley-Rice propagation core with geospatial elevation data
- Developed a full-stack application predicting 140+ building service asset requirements per facility against Australian Standards datasets, eliminating spreadsheet-based workflows across client sites
- Replacing a rule-based asset prediction approach with unsupervised clustering over 30,000+ records to automate cross-facility classification

Projects

Order Book Engine | Price-Time Priority Matching | C++26

https://github.com/SSeanPP/BookingEngine-LiveDashboard

- Built a price-time priority matching engine benchmarking four order book layouts (flat array vs std::map, AoS vs SoA) to isolate how data structure decisions impact latency during realistic order flow
- Implemented each variant with PMR-backed allocation and fixed-capacity ring buffers to keep the hot path free of heap allocation, with a single OrderBookLike-constrained template ensuring all four shared the same interface and measurement harness across 1M RDTSC samples per configuration.
- VectorSoA achieved 30ns p50 / 210ns p99, a ~20× p99 improvement over the naive baseline, isolating contiguous memory access as the dominant performance factor

GPU-Driven Rendering Engine | Java, OpenGL

https://github.com/SSeanPP/VoxelMVP

- Built a multi-threaded GPU-driven renderer using Multi-Draw-Indirect with compute shader frustum culling, moving per-frame draw call submission off the CPU so total CPU draw work stays O(1) regardless of scene size
- Engineered a persistent mapped buffer with a free-list allocator shared between CPU meshing threads and the GPU, eliminating allocation-swap races by zeroing the draw count before handoff so the GPU never reads memory during replacement
- Built a lock-free async job queue feeding worker threads that performed binary greedy meshing with bit-packing, keeping mesh generation off the render thread to minimise GC pressure and frame-time variance
- Sustains ~72k streamed chunks at ~6ms GPU frame time (~166 FPS budget), with frame-time variance low enough to stay smooth under continuous chunk streaming

Technical Skills & Interests

Languages: C/C++, Java, Python, C#, TypeScript, SQL

Frameworks/Tools: React, Node.js, SvelteKit, .NET, REST APIs, Docker, Git, Linux

Data & Systems: SQL Server, NumPy/SciPy, Azure, CI/CD, Cloudflare Workers

Interests: market microstructure, performance engineering, distributed systems