Ryan Gattis

Seattle, WA 98109 ryangattis@outlook.com +1 979-216-8132 linkedin.com/matthew-gattis

Technical Skills

- **Languages:** C++, C#, C, Python, GLSL/HLSL, Bash/Shell, PowerShell
- **Performance & Systems:** Multithreading and concurrency, real-time systems, latency instrumentation, GPU compute (OpenCL)
- **Graphics & XR:** Vulkan, OpenGL, OpenXR, Unity, MRTK, XR Interaction Toolkit, HoloLens 2, Shaders
- **Math:** Linear algebra, quaternions, GLM, Eigen3
- **Libraries, Tools & Practices:** SDL3, VMA, Dear ImGui, shaderc, OpenMesh, GoogleTest, GStreamer, ZeroMQ, CMake, vcpkg, Git, Jenkins, Visual Studio, AI-Assisted Development
- **Platforms:** Windows, Linux, macOS, WSL

Work Experience

Microsoft Corporation *contract through TEKsystems* *July 2024 - April 2025*
XR Software Design Engineer II *Redmond, Washington*
Skills: C#, C++, Unity, MRTK, PowerShell, HoloLens 2

- Built and extended Unity-based test tools using MRTK and XR Interaction Toolkit to validate HoloLens 2 and IVAS (Integrated Visual Augmentation System) functionality.
- Extended C++ head tracking API to surface internal metrics for application-level diagnostics and debugging.
- Collaborated with the head tracking team to identify and resolve real-time tracking inconsistencies.
- Implemented runtime anchor loading and rendering to verify spatial consistency between actual and expected positions.
- Diagnosed Bluetooth connectivity and latency issues; built instrumentation tools for recording and visualizing latency and dead reckoning performance.
- Authored and updated internal documentation to improve onboarding and knowledge transfer for future developers.

E4D Technologies, LLC *November 2015 - May 2024*
3D Software Engineer, Tech Lead, Scrum Master *Richardson, Texas*
Skills: C#, C++, Unity, HLSL, OpenCL, Python, TypeScript, Linear Algebra

- Developed CAD/CAM software for restorative dentistry, processing 3D intraoral scan data to design and manufacture dental restorations.
- Applied OpenCL for GPU-accelerated offset surface computation, improving tool path generation performance, and enabling real-time cutback surface manipulation in the Abutment Implant module.
- Implemented slice plane rendering in HLSL, enabling intricate cross-sectional views of 3D models for detailed design inspection.
- Built camera controls using quaternions and 3D math for precise scene navigation, including double-click focus, interpolated tracking during point cloud acquisition, and automatic framing based on scene AABB.
- Developed controls for positioning and manipulating 3D objects in the scene, including mill block placement with space constraint visualization using linear algebra.
- Computed sprue angle positioning for implant restorations, constraining orientation to discrete angles matching the implant's interface geometry to ensure Ti-base locking cam alignment.
- Solo developed meshfilterapp, a C++ command line tool for loading, transforming, and exporting PLY, STL, and OBJ meshes as part of the manufacturing export pipeline.

- Solo built a TypeScript web demo with Webpack, integrating real-time 3D scene graph construction via WebSockets and ZeroMQ into a Three.js scene, demonstrating a new product architecture.
- Created onboarding infrastructure including multi-repo setup scripts and comprehensive documentation, directly accelerating new hire productivity and reducing ramp-up time.
- Mentored new and existing engineers on MVVM architecture, code patterns, and C# async/await practices.
- Oversaw sprint planning, retrospectives, and code reviews as Tech Lead and Scrum Master to drive team cohesion and deliver on project objectives.
- Collaborated in design reviews, contributing insights and working with cross-functional teams to refine software designs.

Personal Projects

Vulkan Voxel Engine — AI-developed *GitHub: matthewgattis/vulkan-voxel-project*
C++, Vulkan, OpenXR, GLSL, SDL3, GLM, Dear ImGui, CMake

- Vulkan 1.3 voxel terrain renderer with OpenXR HMD support. Implementation AI-generated; designed the layer architecture, chose the rendering approach, and directed the system design throughout.
- Structured the engine as three reusable layers: a Vulkan RAII abstraction layer, an engine layer with custom sparse-set ECS and RAII event dispatch, and an application layer for terrain and controls.
- Implemented multithreaded procedural terrain generation with frustum-based chunk prioritization, per-vertex ambient occlusion, and FXAA post-processing.
- Integrated OpenXR stereo rendering with coordinate space transforms, head tracking, and graceful desktop fallback.

simulife-rs — AI-developed *GitHub: matthewgattis/simulife-rs*
Rust, QUIC, wgpu, tokio, msgpack, zstd

- Client-server evolutionary cellular automaton built as a testbed for exploring distributed simulation, real-time networking, and GPU rendering. Implementation AI-generated; directed the architecture, set the performance methodology, and did the trace analysis.
- Built Chrome-trace instrumentation into both server and viewer plus a CLI trace summarizer, then drove optimization from captured timelines rather than guesswork.
- Chrome traces caught the viewer rendering thousands of frames while idle, when it should have rendered a handful: redraw events were being fed back into egui as input, which kept requesting further repaints. Switching to event-driven repaint took an 8-second idle capture from 9,637 frames to 73.
- Cut a world-pruning phase from 59.3 ms to 1.4 ms by replacing a repeated full-world rescan with a strict cell-locality rule that made updates order-independent, and raised server throughput from 35.5 to 68.6 ticks/sec after the trace showed two async tasks had coalesced onto one worker thread.
- QUIC transport with msgpack serialization and multithreaded zstd compression. Snapshots are handed to the encoder latest-wins so the sim never blocks; faster encoding cut the share of ticks dropped before encode from 22% to near zero.

OpenCL Path Tracer *GitHub: matthewgattis/cl-renderer*
C++, OpenCL, libpng, CMake
Monte Carlo path tracer using OpenCL for GPU-accelerated rendering. Implements importance sampling and solves the rendering equation to produce photorealistic 3D fractal imagery using physically based techniques.

Education

Bachelor of Science, Computer Science; Minor, Electrical Engineering
Texas Tech University *Fall 2010 - Spring 2015*