Tony Yiding Tian

Philadelphia, PA | (267)249-1202 | tonytg@seas.upenn.edu | tonyxtian.com

Education

University of Pennsylvania - School of Engineering and Applied Sciences Philadelphia, PA

B.S.E. in Computer Engineering & Accelerated M.S.E. in Computer Graphics *May 2027*

- GPA: 3.78 — Relevant Courses: GPU Programming, Advanced Rendering, Interactive Computer Graphics, Computer Animation, Machine Learning, Operating Systems Implementation, Data Structures & Algorithms

Projects

MatForge - Advanced Material Rendering System | *Vulkan, Slang, C++* Nov 2025 – Dec 2025

- Architected production-quality GPU path tracer implementing four SIGGRAPH papers (2023-2024) in unified Vulkan ray tracing pipeline, contributing 2,400+ lines of C++ and Slang code for sampling and geometry systems
- Implemented Quad-Optimized Low-Discrepancy Sequences (QOLDS) from SIGGRAPH 2024, engineering base-3 Sobol' generator with Owen scrambling across 47 dimensions, achieving +2.57 dB PSNR and 44.7% MSE reduction vs traditional PCG sampling at 512 SPP with $< 1\%$ performance overhead
- Developed RMIP (Rectangular MinMax Image Pyramid) intersection shader , enabling tessellation-free displacement mapping through hierarchical texture-space ray traversal with custom Vulkan intersection shaders
- Integrated techniques into complete Monte Carlo path tracing pipeline supporting glTF 2.0 scenes, HDR environment maps, and KHR_materials_displacement extension for physically-based rendering
- GitHub Repo: github.com/MatForge/MatForge

Vulkan Grass Renderer - Physically Based Grass Simulation | *Vulkan, GLSL, C++* Oct 2025 – Nov 2025

- Implemented real-time grass rendering system in Vulkan, simulating up to 1 million grass blades with physics-based animation at interactive frame rates using compute and tessellation pipelines
- Developed GPU compute shader for physics simulation applying gravity, Hooke's law recovery forces, and wind with spatial turbulence to quadratic Bezier curve grass blades, processing 1M+ blade state updates per frame
- Engineered three-tiered GPU culling system (orientation, view-frustum, distance-based probabilistic) achieving 4.31× performance improvement at 1M blades through efficient atomic operations and indirect draw calls
- Built hardware tessellation pipeline with dynamic LOD based on camera distance, generating smooth blade geometry with exponential falloff from 10 to 2 subdivisions for optimal geometry density
- Created comprehensive ImGUI control panel with real-time FPS metrics (average, 1% lows), frame time graphs, and automated performance testing system generating CSV benchmarks across 40 configurations
- GitHub Repo: github.com/tonytgrt/Vulkan-Grass-Renderer

CUDA Path Tracer - 3D PBR Renderer | *CUDA, GLSL, C++* Sep 2025 - Oct 2025

- Monte Carlo path tracer capable of rendering complex 3D scenes with custom 3D models and environment maps
- Implemented shading BSDF kernel supporting global illumination, multiple importance sampling, anti-aliasing, sub-surface scattering, capable of rendering various PBR material types with albedo and texture maps
- Integrated third-party libraries of tinyGLTF to support glTF 2.0 mesh loading and Nvidia OptiX for denoising
- Utilized various techniques to boost performance: material sorting (+5%), Russian Roulette (+6% - 24%), stream compaction (+24% - 67%), and Bounding Volume Hierachy (3× - 160× framerate in complex scenes)
- Project Repo and Demo: github.com/tonytgrt/CUDA-Path-Tracer. A previous standalone performance focused stream compaction project with detailed analysis in Nsight: github.com/tonytgrt/Project2-Stream-Compaction

WebGPU Renderer - Real-time Renderer on Web | *WebGPU, TypeScript, WGSL* Sep 2025 – Oct 2025

- Implemented three advanced rendering techniques for real-time lighting of 5000+ dynamic point lights: Naive Forward, Forward+, and Clustered Deferred rendering using WebGPU compute and graphics pipelines
- Engineered screen-space light clustering system using compute shaders, subdividing view frustum into 16×9×24 grid with exponential depth slicing and sphere-AABB intersection testing for efficient light culling
- Developed G-buffer architecture with 3 render targets (position, normal, albedo) enabling two-pass deferred rendering that decouples geometry complexity from lighting calculations
- Achieved 53x performance improvement over naive rendering (497ms to 9.3ms at 5000 lights) and 3.5x speedup vs Forward+ through overdraw elimination and optimized memory access patterns
- Built automated performance testing system collecting statistical frame time data across 30 configurations (3 renderers × 10 light counts), generating CSV analysis for rigorous benchmarking
- Live demo deployed at webgpu.tonyxtian.com, rendering Sponza scene with real-time performance metrics and interactive controls. GitHub repo: github.com/tonytgrt/Project4-WebGPU-Forward-Plus-and-Clustered-Deferred

Technical Skills

Programming: C++, CUDA, Python, WGSL, GLSL, Parallel algorithms, Memory management, Rendering pipeline
Graphics/Rendering: NSight Profiling, Path Tracing, Deferred Rendering, Rasterization, Animation systems, PBR
Tools/APIs: Nvidia NSight, WebGPU, Vulkan, Visual Studio, Qt, OpenGL, Git, CMake, MakeFile, Clang, GDB