Christopher Wilt

Gameplay & Engine Programmer — C++ / Unreal Engine 5

Mifflinburg, PA **— remote, or hybrid in central PA** | 570-541-9807 | ctwilt23@gmail.com
christopherwilt.github.io | github.com/ChristopherWilt | Playable build on itch.io

PROFILE

Gameplay and engine programmer working in C++ across Unreal Engine 5 and from-scratch renderers. Built a server-authoritative multiplayer FPS framework with custom lag compensation, a data-driven weapon and recoil system, and an EOS-backed lobby flow; shipped a networked UE5 title with a team while running the Perforce depot and sprint board. Comfortable below the engine as well: a bare-metal Vulkan 1.3 PBR pipeline, an EnTT data-oriented level loader, and a CPU software rasterizer written from first principles. Also designed and launched a live production platform solo — evidence that I finish and ship, not just prototype.

TECHNICAL SKILLS

Engines: Unreal Engine 5 (C++ & Blueprint), Unity, custom C++ engines

Gameplay: Server-authoritative replication, client-side prediction, lag compensation, RPCs, character movement & state machines, weapon systems, UMG/HUD architecture, data assets

Graphics: Vulkan 1.3, HLSL, PBR & Cook-Torrance shading, image-based lighting, software rasterization, 3D math (vectors, matrices, quaternions)

AI & Systems: Alpha-beta minimax, A* with timesliced search, behavior trees, flocking & steering, ECS / data-oriented design, multithreading, memory management, profiling

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

Pipeline: Perforce (Helix Core) administration, Git, Jira sprint planning, EOS / Epic Online Services, CI/CD, Blender asset integration, animation retargeting

GAME PROJECTS

Multiplayer FPS Framework — Unreal Engine 5, C++ (solo) **Sept 2025 – Present**

- Built a modular, **server-authoritative** competitive shooter framework: the client predicts the hit trace locally for zero-latency feedback while `Server_FireShot` validates ammo and recalculates the shot authoritatively.
- Engineered **custom lag compensation** for hit-scan weapons, rewinding server-side hitboxes to the shooter's client timestamp so high-ping players are neither punished nor advantaged.
- Architected a hybrid, data-driven **recoil solver** decoupled from the weapon class via `UDataAssets` — hot-swappable between deterministic pattern sprays and procedural variance, layered over a spring-mass-damper system for visual kick and camera roll.
- Wrote a **socket-driven procedural ADS system** that removes hard-coded camera offsets entirely, resolving the transform from the active optic's sight socket so any modular attachment centers correctly without per-weapon tuning.
- Extended the Character Movement Component with a replicated **slide state** that alters ground friction and shrinks the capsule, with procedural arm interpolation blending input sway into the first-person mesh.
- Built a **killcam replay system** capturing compressed player snapshots at 20 Hz on a rolling five-second buffer, replayed against a master clock with collision-safe VFX traces.
- Shipped gameplay UI in C++ and UMG: a buy menu with real-time affordability gating, a procedural crosshair scaling to fire spread, a networked minimap fed by a multicast ping RPC, and a killfeed using `TWeakObjectPtr` and timer handles to avoid leaks — with impact decals capped through a 50-instance global pool and animation state replicated as compressed integer flags to hold the rendering and bandwidth budgets.
- Integrated **Epic Online Services** for peer-to-peer matchmaking, persistent lobbies and social-overlay invites, including a custom log writer that scrubs portal credentials before output.

Delivery Detonation — UE5 networked capstone, 4-person team · playable on itch.io **Jan 2026**

Technical Lead & Infrastructure Administrator — Full Sail University capstone

- **Shipped a playable networked build** of a physics-driven third-person delivery game, and served as the team's technical lead through production.
- Administered the remote **Perforce (Helix Core)** depot for the engineering team — defined the branching strategy, and manually resolved deep binary merge conflicts that standard tooling could not.
- Ran sprint planning in **Jira**, decomposing systems like networked physics into atomic tasks; the project hit every milestone.
- Developed a **server-authoritative session state machine** over EOS to prevent client-side state exploitation and synchronize state across clients.
- Optimized bandwidth with a customized **replication graph**, using raw UDP for high-frequency actor translation while reserving reliable RPC nodes for critical game state.

Engine & Graphics Programming — C++, Vulkan 1.3, HLSL (solo) **June 2025 – Aug 2025**

- Wrote a bare-metal **Vulkan 1.3 PBR renderer** from the ground up: Cook-Torrance microfacet shading, image-based lighting from HDR cube maps, and a pre-computed BRDF lookup table enforcing energy conservation — managing pipelines, descriptor pools, uniform buffers and storage views directly, with glTF assets streamed into staging buffers.
- Built a data-driven **ECS level loader** on EnTT, replacing object-oriented game loops with cache-friendly component arrays, plus a flyweight instancing factory that shares pooled geometry and packs per-instance transforms into instanced buffers to avoid pipeline stalls.
- Implemented a **CPU software rasterizer** from first principles beforehand: parametric line drawing, barycentric triangle fill, Z-buffering, perspective-correct UV mapping, backface culling, and a hand-written 4x4 matrix math library with a full MVP pipeline.

Game AI & Low-Level Systems — C++, C# **Feb 2025 – July 2025**

- Built a recursive **alpha-beta minimax** agent with branch cutoffs to contain exponential growth, driven by a phase-adaptive heuristic that shifts scoring bias from early positional weighting to mid-game mobility restriction to late-game material count.
- Engineered **A* pathfinding** over coordinate grids with hash-map visited-set lookups, and a **non-blocking timesliced update system** that splits long searches across frames so deep queries never starve the main loop.
- Implemented **Reynolds steering** for emergent flocking — cohesion, alignment and inverse-distance separation resolved into one delta-time-scaled kinematic acceleration loop for large agent counts.
- Wrote an **N-ary behavior tree** on a hand-rolled template linked-list container with custom iterators, supporting breadth-first, pre-order and post-order traversal with static leak tracking across live node trees.
- Engineered thread-safe C++ pipelines using `std::unique_lock` and condition variables to override scheduling non-determinism, and tracked concurrent hardware contexts inside a single 32-bit integer with atomic bitwise operations.

Archmage (Unity RPG, C#) · **Space Race** (custom C++ arcade) **2025**

- Built interface-driven ability and damage systems in Unity so spells, enemies and destructibles share one contract, plus a C++ arcade title with its own game loop, audio layer and menu flow. Gameplay demos for both are linked from the portfolio.

SOFTWARE ENGINEERING EXPERIENCE

Founder & Lead Systems Architect — The Veil (theveilwedding.com) **April 2026 – Present**

- **Designed, built and launched a live production platform solo** — an iOS, Android and web client from one React Native codebase plus a Next.js business portal, totaling ~300,000 lines across 550 ordered SQL migrations and 75 edge functions, with QuickBooks, Zapier and MCP (Claude and ChatGPT) integrations.
- Enforced authorization in PostgreSQL rather than the client, deriving five user roles from two SQL predicates, and built payment infrastructure across five Stripe flows in which the client submits a plan name and never an amount, with an idempotent webhook as the single writer against at-least-once delivery.

EDUCATION

Bachelor of Science in Game Development — Full Sail University, Winter Park, FL **Graduated April 2026**

Core focus: Systems Architecture & Software Engineering

Coursework: Computer Graphics, Artificial Intelligence for Games, Operating Systems & Systems Programming, Advanced Computer Networks, Advanced Database Management.