Alex Mollard

Senior Generalist Programmer · Big Ant Studios · Melbourne, Australia

alexmollard@protonmail.com · alexmollard.dev · github.com/AlexMollard · au.linkedin.com/in/alex-mollard

PROFILE

Game programmer with eight shipped console and PC titles since 2022, working across UI, gameplay, networking, platform certification, and internal tooling. Outside work I build AetherCore, a Vulkan engine with an editor-first workflow, hot-reloadable C# gameplay, and owner-authoritative multiplayer with full NAT traversal. I increasingly work as an architect and reviewer of agent-directed development: I set the design, constraints, and verification floor, then drive implementation through multi-model agent harnesses, turning weeks of work into days.

EXPERIENCE

Big Ant Studios - Senior Generalist Programmer 2022 - present

- Implemented the client-side WebSocket stack for Cricket 26, including cross-platform TLS handshake support, and carried it through platform certification.
- Built and maintained the UI backend behind every screen in AFL 23, and owned most of the UI implementation and several complete game modes in TIEBREAK, including the move to NoesisGUI.
- Delivered TRC/platform certification across Cricket 24, Rugby 25, Rugby League 26, and Cricket 26, taking builds through submission on PC, PlayStation and Xbox, and integrated PlayFab and PlayGo online services.
- Worked directly with producers through submission windows, picking up the certification tickets most at risk of missing the date, including other seniors' queues in their absence - a responsibility I was given before I held the title.
- Ported and merged gameplay systems between studio titles and stabilised the resulting integrations across the AFL, Rugby and Rugby League codebases.
- Built internal production tooling with the programmers and QA who use it: I write up their requirements, build the tool, and document it so the team runs it without me. Memory analysis, crash capture and triage on Windows Error Reporting, a distributed QA orchestrator, and a Python to CSX automation migration.

SHIPPED TITLES

- Cricket 26 (2025)
- Rugby League 26 (2025)
- AFL 26 (2025)
- Rugby 25 (2025)
- TIEBREAK (2024)
- Cricket 24 (2023)
- AFL 23 (2023)
- Cricket 22 (2022)

SELECTED PERSONAL WORK

AetherCore - Vulkan game engine (C++ / C#) github.com/AlexMollard/AetherCore

- GPU-driven Vulkan renderer with bindless descriptors, a render graph, tiled lighting, GTAO, screen-space reflections and a full post stack.
- Editor-first 2D and 3D workflow with a standalone launcher, dockable inspectors, prefabs, undo, autosave and crash recovery.
- Hot-reloadable .NET 10 C# gameplay SDK with collectible load contexts, inspector-exposed script fields and a managed immediate-mode editor GUI API.
- Owner-authoritative multiplayer over ENet reached through a cheapest-first ladder: LAN broadcast, UPnP/NAT-PMP/PCP, STUN hole punching, rendezvous signalling and opt-in TURN relay, with no port forwarding at any rung.
- Architecture, design, and review are mine; much of the implementation is agent-directed against a hard verification floor - 1,271 tests, eight in-repo games as the acceptance test, an automated GPU abstraction guard, and CI architecture checks on every push.

Earlier engines and experiments

- SlimeProject - co-developed Vulkan 1.3 engine; Odyssean Engine - C++ OpenGL/Vulkan 2D/3D engine that preceded AetherCore.
- Dreaded Escape - unfinished weekend project with two friends; hands-on Unreal Engine 5 replication and Steamworks session work; it reached playable co-op and stopped there.
- Crash Twinsanity Improved - PS2 reverse-engineering patch set (agent-directed, verified in-emulator); RS3 Analyzer - Rust desktop market analyzer; Fly - C++ audio player with real-time visualisation.

TECHNICAL

Languages	C++, C#, Rust, Python, GLSL/HLSL, PowerShell
Graphics	Vulkan, OpenGL, render graphs, bindless rendering, PBR, post-processing, GPU profiling
Engines	Proprietary studio engine, own C++ engines, Unreal Engine 5, Unity
Systems	Netcode (ENet, WebSocket, TLS, NAT traversal), physics (Jolt, Box2D), ECS, asset pipelines
Platforms	PC, PlayStation, Xbox; TRC/platform certification, PlayFab, PlayGo, Steamworks
Tooling	Editor and debug tooling, Tracy instrumentation, crash reporting (WER), CI, CMake, Git
Workflow	Agent-directed development (omp, Claude Code), multi-model orchestration, spec-and-review discipline, automated verification gates, continuous evaluation of new Vulkan SDK extensions and systems tech before adoption

EDUCATION

Advanced Diploma of Professional Game Development - AIE Melbourne

Fundamentals of engine architecture, mathematics, and game systems programming, extended by self-taught C++ and graphics work.

Case studies and source: alexmollard.dev · github.com/AlexMollard · au.linkedin.com/in/alex-mollard