HARSH DUBEY

Real-Time Graphics & Game Engine Programmer

Coventry, UK | niffoisme@gmail.com | niffoxic.com | github.com/Niffoxic | LinkedIn

PROFILE

Game engine and graphics programmer specialising in low-level C++23 and DirectX 12. Currently architecting **Trishul**, a reusable real-time rendering engine, and working on **Curse of the Sea**, the Windows action title game. Comfortable across the full graphics stack - descriptor heaps and PSO management up through gameplay systems on EnTT, Jolt and Ozz - with a deep interest in render graph design and GPU profiling and speed & memory optimization. Reading MSc Games Engineering at WMG, University of Warwick.

TECHNICAL SKILLS

Languages: C++ (17 / 20 / 23), HLSL, GLSL, Python

Graphics APIs: DirectX 12 (Agility SDK), DirectX 11, Vulkan(basics)

Engines & Frameworks: Unreal Engine, custom C++ engine development, EnTT ECS, Jolt Physics, Ozz Animation, FMOD, Effekseer

Supporting Libraries: D3D12MA, DirectXTex, DXC, fastgltf, meshoptimizer, ufbx, ImGui

Profiling & Debugging: PIX, RenderDoc, Tracy, spdlog, Intel VTune Profiler.

Build & Tooling: CMake (multi-target), Git & Github, Docker

Concepts: Real-time rendering, GPU optimisation, multi-threading, SIMD intrinsics, cache-coherent data layouts (SoA), ECS architecture, memory management

EDUCATION

MSc Games Engineering | University of Warwick, WMG - Coventry, UK

September 2025 - September 2026

- Modules covering real-time rendering, GPU architecture, engine architecture, and applied games engineering practice.

BCA (Hons.) Data Science | Bennett University - Greater Noida, India

2020 – 2023 | CGPA: 7.9 / 10

PROFESSIONAL EXPERIENCE

Freelance Software Engineer - NLP / LLM Systems | Remote

2023 – 2025

- Delivered production chatbots, text-analysis pipelines and fine-tuned LLM systems against contracted client briefs, taking each project from scoping through to live deployment as sole engineer.
- Shipped Python real-time data servers to live client environments, balancing latency, throughput and accuracy under fixed deadlines - sharpening the self-direction now applied to engine and graphics work.

SELECTED PROJECTS

Trishul Engine - Reusable C++23 / DirectX 12 game engine

Stack: C++23, DirectX 12 (Agility SDK), D3D12MA, DXC, DirectXTex, EnTT, Jolt, Ozz, Effekseer, FMOD, Tracy, CMake

- Designed and built a modular real-time rendering engine from scratch around DirectX 12 and the Agility SDK, integrating EnTT for ECS, Jolt for physics, Ozz for animation, Effekseer for VFX and FMOD for audio.

- Architected the codebase as three independent CMake targets (engine / gameplay shared library / executable), removing engine rebuilds from the gameplay iteration loop and cutting designer-side feedback time.
- Configured four discrete build flavours (Debug, Release, RelWithDebInfo, Production) and integrated Tracy for per-frame CPU/GPU profiling, giving a measurable baseline for ongoing GPU optimisation work.
- Repository: github.com/Niffoxic/CurseOfTheSea

Curse of the Sea - Windows action game built on Trishul

- Shipping a playable Windows action title on a self-authored DirectX 12 engine, exercising the full pipeline end-to-end: rendering, physics, animation, VFX, audio, and gameplay simulation.
- Wrote gameplay code as a hot-swappable shared library against the EnTT ECS, so design changes compile and iterate without rebuilding the engine - closing the inner loop for rapid prototyping.
- Repository: github.com/Niffoxic/CurseOfTheSea

KnightFox - DirectX 12 graphics R&D sandbox

- Maintain an isolated DirectX 12 testbed for descriptor heap strategies, PSO caching, render graph experiments and shader feature work - features graduate into Trishul only after validation here, keeping the production engine stable.
- Repository: github.com/Niffoxic/KnightFox

CPU Software Rasterizer - From-scratch rendering pipeline in C++

- Implemented the full CPU rendering pipeline - vertex transform, clipping, rasterisation, depth testing and shading - without GPU assistance, to internalise what each fixed-function GPU stage actually does.
- Restructured component storage to Structure-of-Arrays (SoA) layout for cache coherence, then applied SIMD intrinsics on the per-pixel hot path to lift throughput on the inner loop.
- Repository: github.com/Niffoxic/CPU-Rasterizer

CPU Path Tracer with ML-Assisted Shading

- Built a software path tracer in C++ experimenting with ML-driven shading components - combining physically based light transport with learned approximations.
- Repository: github.com/Niffoxic/raytracer-with-ai

Additional work: FoxRoom (networked C++ chatroom), PixelFox (CPU-rendered 2D game), 1,000+ DSA solutions. Full catalogue at github.com/Niffoxic*.*

PORTFOLIO

niffoxic.com - full project breakdowns, shader work, engine architecture notes, and contact links.