oleg.pustogachev@gmail.com | github.com/rearming | calendly.com/oleg-pustogachev | Bangkok, UTC+7

Oleg Pustogachev

Founding Engineer | Senior Full-stack Engineer

summary

Full-stack engineer, 7+ years in real-time AI rendering, AI-agent platforms and game development. At diffuza I wrote the real-time diffusion renderer on TensorRT and an MCP server that lets AI agents control it; it has been used at clubs and events in Bangkok since spring 2026. Before that I built cnvrg, a multi-agent platform, and spent two years at Banzai Games on SPINE and Shadow Fight: Shades (10M+ downloads). Main languages: C#, TypeScript, Python, C/C++.

skills

Languages	C#, C#/Unity, TypeScript/JavaScript, Python, C/C++
AI / GPU	TensorRT, PyTorch, SDXL-Turbo, CUDA graphs, LLM agents, MCP servers (OAuth 2.1)
Backend	ASP.NET Core, Node.js / Express, WebSocket, SSE, SignalR, REST
Frontend	React, React Native, MobX, Effector, Three.js / R3F, WebGL, Electron
Databases	MongoDB, PostgreSQL, Supabase, Redis, vector DBs
DevOps	Docker, GitLab CI/CD, Nginx, Prometheus / Loki / Grafana, Runpod
English	C1/C2, advanced / fluent

diffuza.com | Founding Full-stack Engineer

Mar 2026 – Present

Real-time AI visuals for clubs, festivals and installations

- **Renderer:** SDXL-Turbo stream diffusion on TensorRT, written to replace the third-party engine the project started with. 30 fps at 512×896 on one RTX 5080, with latent feedback, IP-Adapter, ControlNet and depth
- **Optimisation:** full-VAE frame from 129 to 55 ms. Static-shape TensorRT engines to get GroupNorm fusion (3.2× faster decode), CUDA-graph replay, and a small refiner I trained so the fast decoder gets close to full-VAE quality in 6 ms. Production mode runs at 33.7 fps
- **MCP server:** 16 tools for pipelines, inputs, clock, renderer, DMX lights and screenshots of the output; OAuth 2.1 with dynamic client registration and PKCE; revisioned, idempotent operations. Works with Claude, ChatGPT and Codex
- **Studio and infrastructure:** multi-user browser studio (React + MobX, Express, MongoDB), cloud GPU rigs on Runpod with credits and billing, a desktop app for the venue PC, a supervisor for the render machine
- 30 input modules (beat and onset detection, MediaPipe hand tracking, depth, timelines, DMX), synced to BPM and sending only changed values; C++ Spout2 / DirectX 11 addon for zero-copy output

cnvrg.xyz | Founding Full-stack Engineer

Jan 2025 – Jul 2026

Multi-agent AI platform; sole engineer

- **Usage:** 860+ sign-ups; 225 AI agents wrote 14,000+ of the shared graph's 24,000 nodes and completed 8,000+ scheduled tasks
- **Stack:** React / Three.js frontend, ASP.NET Core API, MongoDB replica set, Redis, 13 Docker services across 3 environments, CI that rebuilds only changed services, Prometheus / Loki / Grafana
- **Models:** 9 backends (Anthropic, OpenAI, Gemini, xAI, OpenRouter, image models, a self-hosted model on my own GPU) behind one streaming interface; 15+ tool suites with group access control; ~9,500 logged calls with per-call cost
- **Agent coding:** agents took coding tasks on the platform's own code: Claude Code in an isolated git worktree, a GitLab MR, a staging deploy and a check in Chrome by a computer-use agent (Xvfb, X11 input). 53 agent-written commits on main, including a rewrite of this runner
- **Graph renderer:** three versions to keep 24,000 nodes interactive: React Flow, then Canvas 2D, then GPU-instanced WebGL with LOD, quadtree hit-testing and layout in a Web Worker; also a 3D UMAP view
- MCP server with OAuth for ChatGPT and Claude connectors, API keys and webhooks; SSE + SignalR real-time layer; Solana token layer; multiplayer Unity world (Mirror)

zavod.ai | Founding Full-stack Engineer

Jun 2023 – Oct 2024

AI code generation platform

- Built a visual pipeline engine for LLM-driven code generation with Babel-based runtime UI compilation and hot reload
- Designed the execution protocol for nested no-code / low-code graphs with real-time multi-user collaborative editing

Banzai Games | Unity Developer

Sep 2021 – Apr 2023

- Shipped features for Shadow Fight: Shades (10M+ downloads): combat systems, shop mechanics, skill progression
- Built the replay system (action recording, playback, full state sync) that the team used to debug combat; debug visualization tools for animation and move state machines
- Moved parts of a 7-year-old codebase to a new architecture; rewrote the combat and animation systems
- Built the early Unity combat prototype for SPINE, an action game, before the project moved to Unreal

Earlier: Vizex (virtual labs for Moscow schools, Unity/WebGL) | Artifice (mobile MOBA, Unity/Photon Quantum ECS) | NTProgress (FOREX trading terminal, C/C++)