Levi Rankin

Sacramento, CA | (916) 970-0496 | levirankin1@gmail.com | GitHub | LinkedIn | Website

EDUCATION

University of California, Santa Cruz *Santa Cruz, CA*
B.S. Computer Science *Sept 2021 – June 2025*
- Coursework: Distributed Systems, Web Applications, Algorithms, Databases, Networking, ML/AI (DL, NLP, CV, Game)

SKILLS

TypeScript/JavaScript, React, Python, Rust, C/C++, Haskell, Go, SQL, Bash, PyTorch, TensorFlow, Keras, FastAPI, OpenCV, Docker, Kubernetes, AWS, GCP, Terraform, Ansible, Git, PostgreSQL, Redis, Prometheus, Grafana, Datadog

EXPERIENCE

Independent Software Engineer / Researcher *Sacramento, CA*
(Simulation / Visualization, Systems Engineering) *Jun 2025 – Present*
- Designing and implementing simulation and visualization software to study dynamic system behavior, performance, and emergent phenomena.
- Conducting systems engineering and infrastructure work, including concurrency, fault tolerance, and system design; publishing source code and technical artifacts (15+ repos).

Uxly *San Francisco, CA*
Software Engineer *(TypeScript, React, Python, Node.js)* *Sep 2024 – Dec 2024*
- Built a real-time blockchain analytics platform in React/TypeScript integrating data from 20+ chains.
- Implemented graph-based community detection (Louvain, Label Propagation) to analyze transaction networks, reducing investigation time by 50%.
- Designed client-side caching and batching systems that reduced repeated graph-query latency by 99%.

Synthura *Santa Cruz, CA*
Software Engineer *(Python, FastAPI, PyTorch, TypeScript, React)* *Apr 2024 – Jun 2024*
- Led a team of 6 to build an end-to-end security monitoring platform supporting live and historical data from 10+ concurrent camera streams.
- Architected a FastAPI backend for real-time camera ingestion and async frame processing, integrating CPU/GPU inference pipelines and reducing detection latency from 5s to <10ms.
- Built a React/TypeScript frontend for environment configuration, live video feeds, and analytics dashboards, enabling per-camera and per-environment event monitoring.

Baskin Engineering at UCSC *Santa Cruz, CA*
Undergraduate Researcher *(Python, OpenCV, TypeScript, React)* *Jan 2024 – Jun 2024*
- Developed Python pipelines to extract and process drone video frames for rip current detection research.
- Built a lightweight React/TypeScript interface to visualize live and recorded model predictions.
- Automated dataset generation and labeling workflows with structured metadata and annotation tools.

PROJECTS

Particle Physics Simulation *(TypeScript, Three.js, WebGPU, WGSL, Web Workers)* *Feb 2026 – Present*
- Built a real-time 3D particle physics engine with dual CPU/GPU backends switchable at runtime, supporting 50,000 particles at 60fps on GPU and 3,000 on CPU via WebGPU compute shaders in WGSL.
- CPU backend uses a Barnes-Hut octree (O(N log N)) with a adaptive theta, and zero-copy ping pong array buffers via Web Worker; GPU backend implements three pass velocity Verlet with tiled shared-memory force computation for parallelizing particle reads.

ASCII Procedural Rendering Engine *(TypeScript, Canvas, Web Workers)* *Nov 2025 – Present*
- Constructed a browser-based procedural rendering engine generating dynamic ASCII and character-grid visual effects.
- Developed a modular rendering pipeline with noise generation, palette/glyph transforms, and real-time controls, achieving 60–120 FPS across 50+ visual styles.

Radar Signal Simulator Platform *(TypeScript, React, Canvas, Web Workers)* *Dec 2025*
- Built an interactive real-time UAV radar simulation dashboard allowing users to configure chirp parameters, targets, and environmental noise to explore Range–Doppler behavior.
- Implemented real-time FFT-based signal-processing pipelines and optimized compute-heavy visualization workloads using Web Workers for smooth interactive performance.

Distributed Job Scheduler with Redis Cluster *(Go, Redis, Docker, UUID)* *Oct 2025*
- Engineered a distributed job scheduler supporting prioritized, multi-queue task execution across horizontally scalable worker nodes.
- Designed and implemented the system in Go using Redis Cluster with weighted load balancing, atomic job retrieval, and resilient producer–consumer architecture.

Distributed Consensus & Blockchain Systems *(Rust, Tokio, Axum, Serde, SHA-256, HTTP/JSON)* *Jul 2025*
- Developed fault-tolerant distributed systems in Rust, implementing Raft and HotStuff under crash and Byzantine failure models.
- Built a HotStuff blockchain prototype and a Raft-based key–value store with persistent state, REST APIs, integrity checks, and 90-node simulations.