Mitchell Bizzigotti

San Diego, CA mbizzigotti@ucsd.edu linkedin.com/in/mitchell-bizzigotti github.com/lazergenixdev lazergenix.dev

EDUCATION

University of California San Diego

- M.S. in Computer Science — GPA: 3.91, Expected, Jun 2027
- B.S. in Computer Science — GPA: 3.97, Graduated, Jun 2025

EXPERIENCE

Graduate Researcher, University of California San Diego — Jan 2026 – Present

- Developed a new LLVM backend to generate code for a modified RISC-V architecture, enabling automatic code generation previously not possible.
- Created testing infrastructure to test and verify compiler correctness, helping catch several internal compiler bugs.

Embedded Systems Engineer, GelTech Labs, San Diego, CA — Oct 2026 – Mar 2026

- Developed firmware in Embedded C and ARM Assembly to control a variaty of motors and sensors.
- Used Python to write scripts that tested device functionality, enabling automated device testing.
- Used STM32 Toolchain and Debugger to catch firmware issues that resulted in improved operational reliability.
- Collaborated with team of engineers on RCA to help diagnose and solve multiple device problems.
- Rewrote GUI application from C# to C++ to enable more reliable firmware integration, better user experience, and faster development.

Teaching Assistant, University of California San Diego — Jan 2026 – Mar 2026

- Developed new starter code with Vulkan and C++ to be more up to date with modern C++ software practices.
- Organized and conducted discussion sections to support student's learning.

PROJECTS

FPGA Guitar Pedal

- Worked in a team of 4 using an AMD PYNQ FPGA to create a working guitar pedal.
- Modified device drivers to support realtime data transfer between FPGA and ARM subsystem using an AXI-lite interface

Pipelined Out-of-Order Processor

- Worked in a team of 2 to design, test, and implement a modern MIPS processor using SystemVerilog in a Linux environment with Verilator.
- Implemented out-of-order execution with register renaming to increase processor performance by up to 50%.
- Designed a custom non-blocking data cache to increase memory throughput by 1%

Voxel Rendering Engine

- Developed a dynamic multi-threaded chunk remeshing system with Vulkan to optimize terrain updates without blocking the render thread
- Implemented level-of-detail voxel streaming to allow more than 100× the draw distances.
- Integrated greedy meshing algorithm to decrease cost of CPU to GPU data transfer.

Kai Scripting Language

- Designed an open-source performance-oriented scripting language in C with zero dependencies to target real-time systems.
- Built a custom bytecode interpreter to support compile-time code execution and enable advanced metaprogramming features.
- Utilized arena allocation strategy to decrease total compilation speed, resulting in a 2x speed up in parsing.

SPH Fluid Simulation

- Implemented particle-based physics models with neighbor searching and Eulerian integration techniques to simulate real fluid behavior.
- Leveraged modern GPU compute capabilities through WebGPU compute shaders to simulate and render thousands of fluid particles in real-time.
- Optimized simulation performance by minimizing unnecessary memory transfers to achieve over 60 frames per second with tens of thousands of particles.