Connor W. Fitzgerald

cwfitz.com ■ connorwadefitzgerald@gmail.com ■ GitHub: cwfitzgerald

EDUCATION

Hunter College, City University of New York May 2021
Bachelor of Arts in Computer Science and Bachelor of Arts in German GPA: 3.87

- Daedalus Computer Science Honors Program
- NYS STEM Incentive Program: Scholarship Recipient

TECHNICAL SKILLS

- Programming Languages: C++ (5yrs), C (5yrs), C# (4yrs), Rust (3yrs), Python (6yrs), Scala (2yrs), WebAssembly
- Graphics: OpenGL, DirectX11, Vulkan, DirectX12, GLSL, RenderDoc, Nsight, VTune, Tracy, WebGPU, RenderGraphs
- Programming Paradigms: Concurrency, Coroutines, SIMD, Threading, Fibers, Async-Await, Data Driven Development, Entity Component Systems
- Misc. skills: Markdown, Kotlin, LaTeX, CLI

RELATED COURSEWORK

- VR, AR, and Mixed Reality
- Computer Architecture 1-3
- Microprocessing & Embedded Sys.
- Computer Theory 1 & 2
- Operating Systems
- Software Analysis and Design 1-3
- Matrix Algebra

WORK EXPERIENCE

SoWork January 2021 – Current
Engine Software Engineer

- Designed and architected a Rust/WebAssembly game engine to replace off-the-shelf game engine in 14 weeks.
  - ECS-based, targeting both single-threaded WebAssembly and multithreaded native builds.
  - Optimized for power usage with aggressive change detection that updates the screen reactively.
  - Supported 250 concurrent users in a single room.
- Optimized user experience through asynchronous, seamless asset loading.
- Prioritized immediate and long-term customer needs against code maintenance and tech debt for quick but sustainable growth.
- Actively reviewed code and created Rust code style guide.
- Built custom Javascript-Rust glue for efficient two-way communication with React-based web app.
- Designed data flows for low-overhead communication between game engine, game server, and database.
- Asynchronously communicated complex technical requirements to both technical and non-technical audiences.

Geopipe July – August 2018
Programming Intern

- Prototyped and developed a Unity game engine plugin in C# to render photo-realistic cities using internal Laser Imaging Detection and Ranging (LIDAR) derived architectural data.
- Streamed user facing 3D mesh objects and texture data from a custom cloud REST API endpoint.
- Developed custom caching and asynchronous processing systems using efficient data structures to hide network latency in a soft real-time environment and to ensure a smooth user experience.

RELEVANT EXPERIENCE

Open Source Project: BVE-Reborn/rend3 Sep 2020 – Present
Owner

- Easy to use, customizable, and efficient 3D rendering library based on WGPU.
- Uses compute shaders, descriptor indexing, and clustered forward light culling for efficient GPU-powered rendering.
- Implemented a render graph to make the library easily extensible and composable.
- Balanced ease of use, performance, and customizability to cater to both new and power users.
- Listened to user's specific needs and prioritized features based on their feedback.

Open Source Project: gfx-rs/wgpu April 2020 – Present
Co-Maintainer

- WGPU is Mozilla's Rust-Lang implementation of the upcoming WebGPU graphics API created as a standard for web browsers.
- Revitalized WGPU's DirectX11 backend fixing many critical bugs and implementing required features
- Improved API input validation, constructed additional entry points, and implemented platform specific features.
- Researched, documented, and implemented optimizations that utilized OS and graphics card specific features.
- Validated, reviewed, and provided detailed feedback on pull requests to help contributors optimize their code.
- Attended W3 Consortium GPU working group meetings and participated in the standardization process.

Open Source Project: BVE-Reborn/bve-reborn Fall 2017 – Present
Owner

- BVE Reborn is a complete ground up rewrite of the OpenBVE Train Simulator written in Rust and WGPU.
- Developed a custom graphics renderer and engine with key features such as: clustered forward light culling, GPU light and object culling, bindless materials, indirect rendering, and physically based rendering.