Otto Pfefferkorn

410-934-2445 | pfefferkornotto@gmail.com | Linkedin | Github | Portfolio Website | plentyfast.com

Education

University of Maryland — College Park, MD
Bachelor's in Computer Science, Upper Level Concentration in Mathematics — *December 2024*

Anne Arundel Community College — Arnold, MD
Associate's in Computer Science — *May 2022*

Experience

Fullstack Developer | ASP.NET Core, SQL Server, Bootstrap — January 2025 - Present
Annapolis Computing — *MD*

- Rewrote a CRM application from **ColdFusion** to **.NET** using **Bootstrap** templates for frontend, **ASP.NET Core MVC** for backend, and a scaffolded instance of **Entity Framework** to interact with the **SQL Server** DB
- Repaired the underlying database by setting primary and foreign keys to **allow EF Core to properly identify and implement the intended relationships**, as well as converting columns to more convenient datatypes for space efficiency and ease of use in the application
- Tested for common **security vulnerabilities** in the old and new version of the application to ensure robust security
- Discussed problems concerning the older version of the application and worked out solutions to be implemented in the newer with the project manager

Administrator — June 2025 - Present
PlentyFast IT — *MD*

- Designed and Maintain PlentyFast.com website with SiteBuilder
- Technician/Help-desk for solving general hardware/software setups and maintenance, including laptop screen replacement, hard drive transfer, operating system installation and diagnostics, and printer setup

Intern | .NET Core, Svelte, Typescript, PostgreSQL — May 2023 - August 2023
SessionSentry — *MD*

- Developed monitoring scripts that tracks **user input and process information**
- Browser tracking: Scripts also include tracking of the user's browser activity (URLS, tab titles) through the Windows process accessibility features (done for Chrome, Edge, and Firefox)
- Stored program variables on the user's registry using **Win32 Registry Library**
- Contiguous storage: robust data stream transfer mechanism writes user data simultaneously to file and **Azure BLOB storage** to ensure that recorded data will make it to the server if internet connection is lost
- Created the **CRUD API** for user and session information for **Postgres** DB
- User Interface: Designed the **Administrator UI** for viewing live streams of local machines' activity using **Svelte**
- Wrote **Unit Tests (100% Coverage)** for all aspects of the software, mostly concerned with data transfer

Projects

Mining-Game | *C++, SDL2, Valgrind, GDB* — Present

- Desktop game written in **C++ using SDL2** libraries for OS-agnostic system calls
- Custom-built Physics Engine, Renderer, Textures, Input and Sound Managers
- 2-Dimensional interactive tile-based world, procedurally generated to ensure the uniqueness of the game world on startup
- **AABB-based collision detection** between entities and the game world prevent them from tunneling through impassable tiles
- Raycasting used for ending paths at solid/obstructive tiles (discovery of terrain without seeing past stone, determining if a player has line-of-sight to mine a block)
- Input Manager accounts for 8-directional normalized vector movement, screen-to-game coordinate translation, and dynamic assignment of actions to keystrokes and delay through frames

- Debugging done with **Valgrind and GDB**, Compiled in both Linux and Windows
- Ported to the Browser using **Emscripten**, compiled to **WebAssembly** (See on Portfolio Website)

geekOS | *C, x86 Assembly, QEMU* December 2024

- Implemented various features of an operating system in geekOS, an educational operating system that uses QEMU to simulate the processor, ram, and disk storage
- Unix System Calls: **Implemented fork(), exec(), pipe(), exit()**, and signals between processes
- Multiple Core Utilization: Implementation of **Per-CPU** variables that allows each of the 8 simulated CPU cores to have exclusive access to a given process, allowing for parallelization and speedup
- Paging: In **replacement of allocating segments which subjects memory to external fragmentation**, allocated pages for processes using a **two-level table**
- GFS3 Filesystem: **Opening, writing to, closing, deleting** files and directories from the simulated disk, regular syncing in case of unexpected shutdown

"What The Dog Doin?" | *C#, Unity* December 2023

- Tower-defense style game written in the Unity Engine where the player has to defend against waves of "cat" enemies using automatic weaponry
- Fast and responsive player controls allow for player movement, sprinting, and "barking" to scare away enemies
- Turret scripts that target onto enemies and use a diverse range of projectiles with different effects and mechanics to create a large range of play styles
- Large variety of enemy cats, ranging from large and tanklike to small and speedy, including a "Schrödinger's Cat" which has a 50% chance of respawning into a more powerful enemy when eliminated
- Interactive world is affected by enemy's progress, most notably the "corporate tower" that steadily collapses as more cats are able to reach it

Technical Skills

Languages: C#, C/C++, Javascript/Typescript, HTML, CSS, WebAssembly, Java, Python
Frameworks/Libraries: .Net Core, Entity Framework, ASP.NET, SQL Server, NodeJS, React, React Native, NextJS, Express, Svelte, AJAX, Unity, Flask, NumPy, Pandas, SDL2, OpenGL, Win32, Winsock2
Developer Tools: Git, Docker, Visual Studio, VSCode, Cursor, IIS, Azure Devops, Emscripten, GDB, Valgrind, Makefile
Database: PostgreSQL, SQLite, SQL Server, Azure BLOB Storage and KeyVault, MongoDB
Other: Unix/Linux CLI, Command Prompt, Powershell