Brandon Erickson

San Diego, CA | (619) 403-0190 | brandonscotterickson@gmail.com | linkedin.com/in/brandonscotterickson | 1beric.github.io

EDUCATION

FALL 2018 – SPRING 2021

University of Arizona

Bachelor of Science in Computer Science

Academic Honors

EMPLOYMENT

JANUARY 2021 – SEPTEMBER 2022

Full Stack Developer, Orgo Systems, Tucson, AZ - Remote

- Designed and implemented a robust zoomable and pannable mapping system for org charts, an app-spanning analytics suite, a documentation site for future developers, and an extensive permissions system for security purposes.
- Worked with React, Node, Express, and PostgreSQL to create the web application.
- Held weekly stand-ups and demos to showcase updates to the application.
- Applied concepts such as screen culling, viewport virtualization, and matrix transformations.

MARCH 2019 – JANUARY 2021

Emerging Technologist, Tech Core, Tucson, AZ

- Designed and implemented complex algorithms in virtual reality, such as simulating snow properties in a 3D environment, utilizing light rays in a 3D game engine to simulate acoustic properties for an ultrasound in virtual reality simulation, and building various 3D objects from complex dendrochronology scans.
- Managed teams of interns, teaching them how to apply computer science to business endeavors.
- Worked with various companies to bring virtual reality to classes for the University of Arizona.

PROJECTS

MARCH 2020

Beryl Game Engine, Java

Beryl Game Engine is a 3D game engine written in Java with the intent to introduce users to game design. It utilizes a complex rendering system that is easy-to-use through the exposure of high-level classes. An Entity Component System (ECS) is the heart of the project, allowing the user to create custom components with ease and start new projects without hassle. A library of components is also available in the game engine for users to take advantage of without needing to implement everything on their own.

JANUARY 2021

C Compiler, C

I wrote a grammar using a limited feature-set of C to teach myself more about compilers. While designing and implementing the compiler, I learned about LL(1) parsing, syntax analysis, designing an efficient symbol table with “smart” type checking, and many three-address code optimizations—register allocation/reallocation, data-flow analysis, jump to jump elimination, common subexpression elimination, loop optimizations, and inlining. In the future, I plan to create a compiler for an object-oriented programming (OOP) language to understand the nuances between compiling OOP and procedural languages.

LANGUAGE EXPERIENCE

Java (6 years), Python (5), C (5), ReactJS (4), C# (3), GLSL (3), Rust (1.5)