Michael Scott Fritz

Software Engineer

(360) 355-4365 | michael.s.fritz@gmail.com | www.michaelscottfritz.com

Technical Skills

Languages & Engines: C/C++ (6 years), C#, Javascript, Python, ARM assembly, custom engines, Unity
Math & Graphics: Linear algebra, geometry, OpenGL 3, DirectX 11
Networking: Berkeley socket API/Winsock2, UDP/TCP/IP, Win32
Programming: Game engine development, physics programming, graphics programming, network programming
Concepts: Operating systems, engine design, algorithms, data structures, content pipelines, Agile methodology

Work Experience

AuthorDigital (team of 8, using Agile) **Game Engineer Intern** **5/20 - 8/20**

- Built 4 Unity prototypes; also became the shader/graphics programmer until a senior graphics tech was hired
- Wrote shaders, see-through shaders, and visual fx: color, shape, swirling, glow, high-dynamic range, bloom
- Used procedural terrain-editing node-graph tool, MapMagic, to generate textures, create terrain, build levels
- Wrote error-logging system to show at runtime; wrote auto-tagging system to track version, time, code location
- HIRING: Was highly rated and recommended for full-time hire upon graduation, but they had no openings

DigiPen Game Projects Class **Engine, Graphics, Physics TA** **2/20 - 4/21**

- Mentored graphics, physics, engine programmers on teams developing year-long games in custom engines

DigiPen Projects

***Overdraw* - 3D action cover shooter, team of 15** **Graphics, Physics** **9/19 - 6/20**

- Researched Delaunay Triangulation and the Voronoi Diagram through scholarly articles & whitepapers
- Implemented a mesh-breaking pipeline in Unity, using the Bowyer-Watson algorithm and Voronoi Diagram clipping in order to dynamically break meshes
- Tested the mesh breaking using a debug pipeline to ensure reliability
- Programmed the “force” for the game so the player can manipulate large environmental objects in combat

Personal project: 3D software renderer **Graphics** **6/19 - 9/19**

- Created a 3D software renderer in C/C++ using only Windows libraries (bitmaps and bitblt)
- Optimized triangle rasterization with incremental point-triangle collision to real-time render w/parallelization
- Solved accurate clipping to the NDC cube in clip space to robustly render clipped triangles
- Allowed for diffuse lighting on each fragment in the model.

***Rush Park* - 2D sports action game, team of 12** **Engine, Graphics, Physics** **9/18 - 6/19**

- Developed the architecture and framework for the custom 2D engine in C/C++ to support 6 programmers
- Implemented entity-component system framework to allow for growing functionality with CPU cache efficiency
- Built robust collision detection/resolution system for the ball to prevent tunneling at extreme speeds
- Engineered rendering layer using OpenGL; organized game data to allow for efficient batched rendering calls
- Created a serialization/reflection system and custom game-editor to allow designers to create levels directly
- Incorporated a flexible message system so designers can spawn any game event

Awards

A* Algorithm Speed Contest **AI** **11/20**

Won 1st place in a 60-programmer speed contest for most efficient A* AI pathfinding implementation

Education

BS in CS in Real-Time Interactive Simulation **DigiPen Institute of Technology** **3.88 / 4.0 GPA** **2021**
AA in Computer Science **Centralia Community College (done in high school)** **3.98 / 4.0 GPA** **2017**