**Spencer T. Parkin**

2113 S. Claremont Drive

Bountiful, UT 84010

(801) 970-4578

[spencertparkin@gmail.com](mailto:spencertparkin@gmail.com)

<https://github.com/spencerparkin>

**Aim**

This prospective employee aims to advance his career in the profession of software engineering where it can be applied to the solving of interesting problems in the areas of computer graphics and computer entertainment.

**Employment**

**Lead Software Engineer** 2019 – Present

*L3 Driver Training Solutions, Salt Lake City, Utah*

Developed a new graphics solution for the simulation software using Unreal Engine 5. This was a replacement for the previous solution which used Havok’s Vision engine, now deprecated. The UE module synchronized with the brain of the simulation over multicast-IP/UDP and could scale horizontally to take advantage of multiple GPUs.

**Senior Software Engineer** 2016 – 2019

*3M Health Information Systems, Murray, Utah*

Helped to develop and maintain the test automation component of the continuous integration cycle. Wrote a Python/Selenium-based testing framework designed to exercise the coding & reimbursement software developed by 3M HIS. Worked with test automation engineers to make use of the framework.

**Programmer** 2012 – 2016

*Avalanche Software, Salt Lake City, Utah*

Developed new C++/MFC/OpenGL-based tools for use in the asset-to-game pipeline. Worked with tool users (artists and designers) to optimize workflow and educate them on how to use the tools. Modified existing tools to fix bugs and add new features and functionality. Worked closely with those implementing the content-build pipeline and engine consumption of game assets.

**Associate Programmer** 2007 – 2012

*Avalanche Software, Salt Lake City, Utah*

Developed file archiving software for use in packaging up files to be efficiently consumed by the game engine (named Octane) at load time for the title “Bolt.” Worked on the CPU- and GPU-side implementation of the particle system, as well as developed a particle-system authoring tool with live-authoring capabilities that was used in many subsequent titles. Wrote low-level math routines and optimized them with assembly and SIMD intrinsics.

**Lab Aide** 2003 – 2007

*Weber State University, Ogden, Utah*

Helped fellow students with computer-related tasks while earning a 4-year degree (a B.S. in math.)

**Level 1 Programmer** 2001 – 2002

*Acclaim Entertainment, Sugar House, Utah*

Developed the front-end menu system (in C++) used to choose characters, levels, user preferences and other options before proceeding into game-play. Worked closely with artists and designers to fulfil all software requirements. Our title was “Legends of Wrestling II.”

**Programmer Intern** 2000 – 2001

*Acclaim Entertainment, Sugar House, Utah*

Worked on the particle system implementation and character customization feature of the game “Legends of Wrestling I.” For example, splatting blood spots (red texels) on the mat during a fight.

**Education**

**Bachelor of Science in Mathematics** 2003 – 2007

*Weber State University, Ogden, Utah*

While earning a 4-year degree, participated in math club, and submitted solutions to problems published in math journals. Earned a minor in computer science. Occasionally gave talks to the math club on interesting math puzzle solutions.

**High School Diploma** 1998 – 2001

*Viewmont High School, Bountiful, Utah*

While in the first year of high school, played clarinet in the marching band.

**College Credit** 2000 – 2001

*Davis Applied Technology Collage, Kaysville, Utah*

Spent half of the last year of high-school at a community college to earn college credit towards a computer science degree.

**Technologies**

**Programming Languages:**

* 20+ years writing ***C/C++*** code. Taught myself C++ during high school. My high school did not offer any kind of programming classes at the time. Been using it personally and professionally ever since. Well versed in object-oriented principles and design.
* 6+ years writing ***Python*** code, embedding it in C/C++ applications, and extending it with C/C++ modules. Have used it to develop many scripts and tools to aid in other development. Have used it as a backend for developing simple websites. Used it at 3M to help develop their continuous integration systems. Used it at L3 to automate tasks in 3Ds Max.
* 5+ years writing ***Lua*** code as well as embedding and extending it with C/C++. Lua was used extensively at Avalanche Software, both as data and as script in their build system and tools pipeline.
* 4+ years writing ***C#/.NET*** applications. Have used C# to write desktop applications using WinForms, as well as a mobile app using .NET/MAUI. Wrote a Blazer-based website using .NET Core for Azure until my free trial expired.
* 3+ years with a variety of other languages including JavaScript, HTML, CSS, Perl, HLSL, x86 assembly, RISC assembly, and SIMD intrinsics.

**Development Tools:**

* 15+ years using ***Visual Studio***. Am very familiar with its interface, debugging capabilities and features, project organization, package management, etc.
* 4+ years using ***PyCharm*** to develop front and backend applications in Python.
* 9+ years using ***Git*** and ***Git Extensions***. Very much love the feature branching workflow, understand its distributed nature, and am not afraid to merge.
* 6+ years using ***SVN*** and/or ***Perforce***.

**Development Frameworks/APIs/Platforms:**

* 3+ years using ***Unreal Engine***. Have become familiar with its workflow and how to extend it using C++ and Blueprint. Have written a custom asset type serializing with the rest of the build. Wrote an extension of the editor using Slate UI. Wrote code to take advantage of the animation system, dynamic mesh rendering system, and the off-screen rendering system.
* 5+ years using ***OpenGL/WebGL***. Wrote many personal projects for desktop and web using OpenGL.
* 2+ year using ***DirectX***. Used this API to perform occlusion queries for lens-flare effects, and used its instancing features to implement efficient particle systems.
* 0.5+ years using ***React***, ***Cesium***, ***Knockout*** and ***jQuery***.
* 3+ years of using ***PyQt*** to develop Python-based GUI apps.
* 10+ years of using ***wxWidgets*** to develop C++-based GUI apps.
* 2+ years of using ***WinForms*** to develop C#-based GUI apps.
* 20+ years developing for ***Windows***.
* 2+ years developing for ***Linux***.
* 1+ years developing for ***Web***. Have used ***Heroku*** and ***Azure***.

**Persistence & Concurrency:**

* 3+ years using ***MongoDB*** with Python bindings. Learned to do some sophisticated queries as well as use the database to synchronize instances of a process scaled horizontally across the network.
* 3+ years using ***Redis*** with C# and C++ bindings. Took advantage of Redis’ message brokering capabilities as well as a shared key/value store. Wrote my own client using C++. It can be found here: <https://github.com/spencerparkin/Yarc>.
* 1+ years using ***SQL*** with C# bindings. Familiar with relational database principles.

**Mathematics:**

* Well versed in 3D math ranging from linear algebra, vector calculus, quaternions, matrices, etc.
* Familiar with conformal geometric algebra.

**Programming Concepts:**

* Well versed in the object-oriented programming paradigm, including inheritance, encapsulation, polymorphism, abstraction, etc.
* Familiar with MVVM – Model View, View-Model practices in implementing UIs.
* Familiar with multithreaded programming and synchronization using mutexes and semaphores to protect shared resources and improve efficiency. See, for example: <https://github.com/spencerparkin/Frumpy>
* Familiar with multithreaded socket programming over TCP or UDP.

**Programming Principles:**

* ***Single source of truth.*** This applies, not just to data, but also to code. Do not cut and paste code indiscriminately. Subroutines were invented for a reason. Generalize wherever possible. Prefer sensing state rather than duplicating it or caching it.
* ***Separation of concerns.*** This promotes a modular design and discourages the obfuscation of code as well as subroutines with unintended side-effects.
* ***Care about your code.*** A good programmer might spend a whole day writing something only to throw it all away the next day. Do not compromise on the quality or design of your code. You will often become stuck with what you write. Try to get it right the first time, even if that’s impossible.
* ***Learn from others.*** A good programmer is as good at reading code as they are at writing it. A good programmer can learn someone else’s code and then continue with it as if they had written it themselves. You can also learn new and innovative programming techniques by studying other people’s code.
* ***Keep it simple, stupid.*** This is a tried-and-true principle, and whole heartedly agree with it.