Artem Kirienko

(813) 709-9515 – artem.kirienko1999@mail.ru – 5009 N Sheridan Rd, Chicago, IL 60640

CAREER OBJECTIVE

Aspiring software engineer with a strong foundation in graphics, real-time systems, and multithreading. Mastering real-time architectural design of custom game engines, optimizing performance, and creating scalable frameworks for complex systems. Explore innovative techniques in advanced C++, Design Patterns, Multithreading, and GPU Programming (HLSL, CUDA) to deliver cutting-edge real-time applications.

SUMMARY OF QUALIFICATIONS

- Proven experience in Real-Time Game Systems {Engine, Graphics, Audio, Networking}.
- Real-Time profiling and refactoring through Multithreaded development and low-level optimizations.
- Experience working with version controlled large-scale projects (150+ classes).
- Excel in challenging environments, solving complex, performance-oriented tasks.
- Proven capacity to manage workloads under extreme time pressure and deadlines.

EDUCATION

DePaul University (Chicago, IL).....**Jan/2023 – March/2025**
Master of Science in Software Engineering

- *Concentration on Real-Time Game Systems*

University of South Florida (Tampa, FL).....**Aug/2017 – Aug/2022**
Bachelor of Science in Computer Science

RELEVANT DEVELOPMENT EXPERIENCE

More information, videos and UMLs of my projects can be found at: artushdeveloper.github.io/portfolio/

Real-Time Rendering Engine `[C++/DirectX 11/Google Protocol Buffers]`

- Engine Core:
  - Architect a real-time game engine from scratch using **DirectX**, **HLSL**, and **C++**.
  - Programmed 20+ DirectX shaders {**Vertex, Pixel, Compute**} to manage complex rendering pipelines.
  - Robust C++ framework encapsulating high-level concepts {Meshes, Textures, Cameras, Lights, etc.}.
  - Engine capable of mesh texturing, multiple cameras, skinning and animation, lighting and 2D fonts.
  - Conducted debugging and performance profiling using **Visual Studio Graphics Debugger**.
- Lighting:
  - HLSL dynamic and static lights system {Point, Directional, Spot and Area using Blinn-Phong BRDF}.
  - Implemented a **Volumetric Deferred Lighting Pipeline** capable of handling up to **10,000 lights** (~360 lights with forward lighting) before noticeable drop in framerate.
- Animation and Skinning:
  - Developed skeletal animation and mesh skinning systems, through matrix transformations.
  - Distributed matrix calculations to 1000+ GPU threads using DirectCompute for efficient execution.
  - Compressed animation clip data by a factor of ~3x with no perceivable visual degradation.
- 2D Font System:
  - Developed a system to render 2D font sprites using orthographic camera and Flyweight design pattern.
- Mesh Converter:
  - Programmed an offline tool to convert mesh information {Geometry, Skinning, Animation, Texture, Fonts} from .glb files into engine-compatible format.
  - Serialized the above mesh information into run-time files using **Google Protocol Buffers**.
- Math Library:
  - Designed a library for complex math operations on Vector, Matrix, and Quaternion data types.
  - Optimized library performance by a factor of ~4x using **SIMD intrinsics** {MMX, SSE, SSE2}.
- Object Library:
  - Developed a library with a custom tree-like data structure to handle hierarchies of objects in memory.
  - Used the library to implement scene graphs and bone hierarchies in the engine.

Artem Kirienko

(813) 709-9515 – artem.kirienko1999@mail.ru – 5009 N Sheridan Rd, Chicago, IL 60640

Real-Time Multithreaded Development [C++/WaveOut]

- Real-Time Audio Streaming Player:
  - Developed a real-time player to stream and play audio wave files using C++ and WaveOut framework.
  - Used 25 C++ threads and double buffering technique to play raw audio data from wave files.
  - Played small audio buffers one by one on 20 threads simultaneously downloading the rest from the file.
  - Created a Monitor thread to automatically terminate and clean up other threads at the end of playback.
  - Communicated between threads with C++ **mutexes**, **locks**, **condition variables**, **futures** and **promises**.
- Multithreaded Maze Solver:
  - Optimized a DFS-based maze solver by splitting it into 2 threads {Top-Down and Bottom-Up}.
  - The 2 threads start from the opposite ends of the maze, colliding in the middle to give a full solution.
  - Used C++ **atomics** for the fastest communication between threads.
  - Optimized multithreaded solution solves large mazes **~3x times faster** than single-threaded.

Multithreaded Game Audio Engine [C++/XAudio2]

- Engineered a real-time game audio engine with the **Actor Model** multithreaded architecture.
- Game, Audio, File, User, Error and Auxiliary threads handling distinct tasks across 7,000+ lines of code.
- Designed a **thread communication system** using the **Command** design pattern and a circular queue.
- Built a custom **Handle Library** utilizing **C++ mutex** to ensure thread-safe access to shared resources.
- Developed a linked list-based **Manager Library** for efficient asset management and memory integrity.
- Extracted, processed, and played raw audio data using XAudio2 framework on the Audio thread.
- Provided API for user-customizable callbacks and executed them on the User thread.

Omega Race Game Multiplayer Network Layer [C#/.NET/Lidgren Network]

- Added a **Client-Server** network layer to Omega Race game for 2 players and a server in C#.
- Created and serialized packet data types, sending them over the network with Lidgren Network library.
- Programmed data-driven queues for processing incoming and outgoing network packets.
- Synchronized game state with a Lock-Step protocol and Cristian's algorithm for clock synchronization.
- Implemented **Client-Side Prediction** and **Dead Reckoning** to compensate for network latency.
- Created a debug tool to record game sessions in real-time and play them back to reproduce errors.

Space Invaders Game Clone [C#/.NET]

- Developed Space Invaders clone game in C# using modern techniques and **12 Software Design Patterns**.
- Game capable of drawing, moving, animating and colliding sprites and processing keyboard inputs.
- Implemented an **Object Pooling Method** to minimize dynamic memory allocations.
- Created game sprites {aliens, shields, ships and missiles} on demand using **Factory** design pattern.
- Used the **State** design pattern to switch between select, play and game over scenes.
- Programmed sprite collision system using early-out and **Visitor** and **Composite** design patterns.
- Applied the **Flyweight** pattern to efficiently render fonts {player's score, level labels and menu options}.
- Other design patterns include **Singleton**, **Observer**, **Proxy**, **Command**, **Iterator** and **Strategy**.

Game Particle System Optimization [C++]

- Refactored a 200,000-particle game system, achieving a **20x performance boost**.
- Developed vector and matrix math library enhanced with SSE2 SIMD intrinsics.
- Integrated the above library into the project, optimizing vector and matrix calculations.
- Reengineered project data structures for efficient dynamic memory allocations and **CPU caching**.
- Applied **Return Value Optimization** to eliminate unnecessary temporary constructions.
- Modified the project to use float data type instead of double.

Other Research Areas [C++/C#/.NET]

- GPGPU Programming.
- API Development.
- Real-Time Collision Detection.
- Applied Algorithms and Data Structures.

Artem Kirienko

(813) 709-9515 – artem.kirienko1999@mail.ru – 5009 N Sheridan Rd, Chicago, IL 60640

WORK EXPERIENCE

C++ Software Engineer for a NASA-funded project at *DePaul University* (Chicago, IL)....**9/2024 – Present**

- **Drone Traffic Management Ecosystem** `[C++/Unreal Engine 5]`
  - In a team of 7, developed a system for remote drone operation using **C++** and **Unreal Engine 5**.
  - Owned the implementation of a feature to stream video from drone cameras over the network.
  - Video streaming implemented using **GStreamer** library**, RTP** protocol and **H.264** video format.
  - Moved video streaming code to a separate thread using UE5 threading API.
  - Constructed **UE5 blueprints** to support video streaming from drone camera.
  - Performed local C++ optimizations including RVO, loop invariants, memory caching and allocations.

TECHNICAL SUMMARY

Programming languages: C, C++, HLSL, C#, CUDA C/C++.
Libraries and API: DirectX 11, Unreal Engine 5 API, CUDA, GStreamer, Win32, .Net, XAudio2.
Networking Protocols: TCP, UDP/RTP, IP.
Software and IDEs: Unreal Engine 5, Visual Studio Enterprise.
Software Engineering: UML, Test-driven, Agile and Data-driven development.
Version Control: Perforce, Git.
Platforms: Windows.