Kanishk Gupta

(312) 900-5609 | kanishkgupta410@gmail.com | Lake View, Chicago, IL 60613

Career Objective

Work as a developer for a game company where I can use and improve my technical and problem solving skills to create and enhance cutting-edge real-time technology.

Visa Sponsorship: STEM OPT compliant (3 years)

Qualification Summary

- Master's Degree in Game Programming with a focus on Real-Time Systems & Software Development.
- Experienced with multiple areas of Game Programming such as Graphics, Audio, Physics, etc.
- Demonstrated analytical and quantitative problem solving skills.
- Proven capacity to manage workloads under extreme time pressure and deadlines.

Technical Summary

- **Programming Languages**: C, C++, C#, HLSL, CUDA.
- **Frameworks & Libraries**: DirectX, XAudio2, Win32, CUDA, Google Protocol Buffers, Lidgren.Network, Box2D.
- **Tools & Technologies**: Perforce, Visual Studio.
- **Skills**: OOP, Multithreading & Concurrency, Version Control, Test Driven Development, Library Development, Caching Strategies, Debugging & Pair Programming, Memory Management.

Education

DePaul University (Chicago, IL, USA) — GPA: 3.977/4.0
Master of Science: Game Programming — 1/2023 - 3/2025

Amity University (Noida, UP, India) — GPA: 7.6/10.0
Bachelor of Technology: Computer Science — 2018 - 2022

Relevant Development Projects

- **Game Engine Development & Optimization** [C++ / DirectX / HLSL / Google ProtoBuf]
  - Engine Core
    - Architect a real-time game engine from scratch using DirectX, HLSL, and C++.
    - Programmed 20+ DirectX shaders (Vertex, Pixel & Compute) for complex rendering pipelines.
    - Robust C++ framework encapsulating high-level concepts (ex. Meshes, Textures, Cameras, Lights).
    - Engine capable of object transformations, mesh texturing, multiple cameras, skinning and animation, lighting, and 2D sprites.
    - Conducted debugging and performance profiling using Visual Studio Graphics Debugger.
  - Animation & Skinning

- Developed skeletal animation and mesh skinning system through matrix transformations.
- Efficient matrix calculations, distributed to 1000+ GPU threads using DirectX Compute Shaders.

- 2D Sprite System:
  - Implemented a system to render 2D sprites using an orthographic camera and a plane mesh.
  - Used the system and the Flyweight design pattern to render dynamic 2D fonts.
- Mesh Converter
  - Programmed an offline tool to convert mesh data {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.
- Object Library
  - Developed a library with custom tree-like data structure to handle object hierarchies in memory.
  - Created an iterator to traverse the tree (DFS forward & reverse) in a linked list fashion.
  - Used the library to implement scene graphs and bone hierarchies in the engine.

- **Real-Time Multi-Threaded Development** [C++ / Win32]
  - Real-Time Audio Streaming Player
    - Developed a real-time application to stream and play audio wav files using C++ and WaveOut.
    - Used 25 C++ threads and the double buffering technique to play raw audio data from wav files.
    - Played small audio buffers one by one on 20 threads, while also loading the rest from the file.
    - Created a Monitor thread to automatically terminate and clean up other threads at the end of the program.
    - Communicated between threads via C++ mutexes, locks, condition variables, futures and promises.
  - Multi-Threaded Maze Solver
    - Optimized a DFS-based maze solver by splitting it into 2 threads (Top-Down & Bottom-Up).
    - The 2 threads start from the opposite ends of the maze, meeting in the middle for a full solution.
    - Used C++ atomics for fast and efficient communication between threads.
    - Achieved ~2x increase in performance compared to the single-threaded solution.
- **Multi-Threaded 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.
  - 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, executed on a separate User thread.

Kanishk Gupta

(312) 900-5609 **|** kanishkgupta410@gmail.com **|** Lake View, Chicago, IL 60613

- **RayMarching** [C++ / DirectX / HLSL]
  - Built a scene in HLSL using Parametric math-based Models and Signed Distance Functions.
  - Implemented Soft Shadows, Ambient Occlusion & Anti-Aliasing.
- **Game Physics & Collision** [C++]
  - Implemented a Rigid-Body physics engine for 3D rigid-body collisions.
  - Implemented low-latency techniques by minimizing garbage collection overhead.
  - Designed and Developed architecture to improve the performance of Collision Detection and Resolution by using techniques like Spatial Partitioning, Bounding Volume Hierarchy, and Primitive Bounding volumes for complex shape objects and multiple design patterns.
- **Omega Race Multiplayer Network Layer** [C# / Lidgren.Network]
  - Added a Client-Server network layer to Omega Race game for 2 players in C#.
  - Created and serialized packet data types, sending them over the network with Lidgren 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 Remake** [C#]
  - Developed Space Invaders in C# using modern techniques and 10+ Software Design Patterns.
  - Game capable of drawing, moving, animating, and colliding sprites and processing keyboard inputs.
  - Implemented Object Pooling to minimize dynamic memory allocations.
  - Created game sprites (aliens, shields, ships & missiles) on demand using the 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.
  - Other design patterns include Singleton, Observer, Proxy, Command, Iterator & Strategy.
- **Angry Birds Remake** [C++, Box2D]
  - Developed Angry Birds in C++ using Box2D.
  - Game capable of drawing, moving, animating, and colliding sprites and processing keyboard inputs.
  - Implemented Object Pooling to minimize dynamic memory allocations.
  - Implemented Game Objects to have different Collision Shapes (Box/Circle/Triangle).
  - Built custom Contact Listener to implement Object Degradation/Destruction.
  - Implemented Sprite Swapping to show Damage in Blocks.
  - Created Birds and Blocks using Factory design pattern.
  - Implemented Smoke Trails for tracing Bird flight path in Back/Front Buffer for accurate pattern.
  - Implemented Bird Abilities (speed up, boomerang).
- **Other Projects**:-
  - CUDA DCCI Image Upscaler [C++, CUDA]