SAI SUSHANT KORANNE

sushantsaige@gmail.com • +13126878202 • LinkedIn • Portfolio • GitHub

EDUCATION

DePaul University Chicago, IL
Master of Science in Game Programming (STEM), CDM *2025-2027*

Vellore Institute of Technology Vellore, India
Bachelor of Technology in Computer Science Engineering *2017-2021*

WORK EXPERIENCE

SuperHuge Studio-Xarpie Solutions Bengaluru, India
Game Developer *November 2022 – June 2024*

- Collaborated cross-functionally with designers, artists, and animators to define gameplay mechanics, iterate on prototypes, and deliver cohesive player experiences.
- Designed and implemented game systems in **Unity** using **C#**, including gameplay logic, UI flow, and asset integration for both 2D and 3D projects.
- Optimized performance and memory usage through profiling, batching, and asset/scene optimizations to meet target framerates on target platforms.
- Led QA handoffs and bug resolution by reproducing issues, writing clear bug reports, prioritizing fixes, and verifying regression fixes with the QA team.
- Maintained technical documentation and version control workflows to standardize project setup, coding conventions, and onboarding for new team members.

Cognizant Technology Solutions Pvt. Ltd Hyderabad, India
Programmer Analyst *August 2021 - November 2022*

- Managed Git-based source control and branching strategies to support parallel development, code reviews, and release stabilization.
- Built and maintained CI/CD pipelines to automate builds, tests, and deployments across environments, improving delivery reliability.
- Created and published Docker images and managed the Docker repository to ensure consistent runtime environments across dev, QA, and production.
- Administered application servers and deployments including monitoring, patching, and rollback procedures to maintain application uptime.
- Automated local and environment setup with PowerShell scripts to reduce onboarding time and eliminate manual deployment steps.
- Owned backlog and release artifacts by coordinating with stakeholders, grooming Product Backlog Items, and ensuring timely delivery of prioritized features.

PROJECTS

- Academic

Game Engine (Using Dx11) – InProgress

- Engineered a custom 3D game engine in C++ by abstracting DirectX 11, enabling low-level control over rendering pipelines and GPU resource management.
- Developed a robust asset pipeline using **tinygltf** and **Google Protocol Buffers** to parse GLB models and animations, serializing data into optimized, engine-readable binary files.
- Implemented core graphics and animation systems, including 3D model loading, texture mapping with UV coordinates, and playback of skeletal animations.
- Built a bespoke **Math Library** (Vectors, Matrices, Quaternions) and **PCS (Parent-Child-Sibling) Tree** to manage hierarchical scene graphs and efficient transformation propagation.

Space Invaders (Custom Game Engine - C#)

- Developed a sprite-based Space Invaders game using a custom-built C# engine with real-time collision detection.
- Incorporated key design patterns including Factory, Observer, and State to structure gameplay systems.
- Built a modular architecture supporting scalable enemy types, weapons, and game states.
- Ensured smooth performance through optimized update loops and efficient entity management.

Audio Engine (Multithreading - Academic):

- **Built a high-performance audio engine** using C++ and the XAudio2 API, supporting asynchronous real-time audio streaming with minimal latency and high throughput.
- **Implemented a five-thread architecture** — Main, Audio, File, Auxiliary, and XAudio2 — to handle playback control, file I/O, priority scheduling, and low-level audio processing in parallel.
- **Designed dynamic priority and callback systems** enabling seamless audio transitions, user-defined event hooks, and responsive playback control across gameplay states.
- **Leveraged data-driven design** to support flexible audio configurations, runtime asset management, and scalable integration into custom game engines.

- Personal

Obstacles Path (Unreal)

- **Dynamic obstacle traversal —** Player navigates a path filled with obstacles that move vertically and horizontally.
- **Rotating and sweeping hazards —** Obstacles rotate and sweep across the path, requiring timing and precision.
- **Goal-oriented progression —** Reach a final destination while avoiding and reacting to moving hazards.

Warehouse Wreckage (Unreal)

- **Projectile spawning and launching —** Player's spawn spherical projectiles and launch them at targets.
- **Physics-based destruction** — Realistic collisions knock down stacked cans using physics simulation.
- **Score-driven objective** — Aim and knock down cans to achieve a high score or clear levels.

Multiplayer Top Spin Game (Unity)

- **Multiplayer top combat** — Players control spinning tops and compete in real-time arenas to knock opponents out.
- **Photon PUN2 networking** — Uses PUN2 with RPCs for matchmaking, player sync, and authoritative action handling.
- **Momentum-based elimination** — Collisions reduce tops' speed; a player is eliminated when their speed drops to zero.

Racing Game Using Reinforcement Learning (Unity)

- **Reinforcement learning–driven opponents** — Enemy agents are trained with RL to navigate racecourses and adapt tactics against the player.
- **Checkpoint-based reward system** — Agents receive positive rewards for passing checkpoints and negative rewards for missing them, shaping efficient flight behavior.
- **High-speed racing dynamics** — Player and AI pilots compete in fast aerial courses where learned policies prioritize precision and checkpoint timing.

Maniacs (Unity)

- **Survival-horror setup** — Player spawns in a procedurally darkened environment with limited visibility and atmospheric audio cues.
- **Dynamic enemy AI** — Zombies spawn at randomized locations, navigate the level, pursue the player, and coordinate basic attack behaviors.

- **Diverse combat systems** — Player can find and switch between melee and ranged weapons, each with distinct handling, damage, and cooldowns.
- **Resource and tension management** — Limited ammo and health pickups force tactical decisions and encourage exploration under constant threat.

SKILLS

- **Game Engines:** Unity, Unreal Engine
- **Programming Languages:** C#, C++, Java, C
- **VCS:** Git, Helix-P4V
- **OS:** Windows, Linux, Mac
- **IDE/Code Editors:** Visual Studio, VS Code, JetBrains Rider