Aman Singh

amansinghdev.com || matharu689@gmail.com

SUMMARY OF QUALIFICATION

- Gameplay / Software Engineer based in Vancouver, BC with experience building gameplay systems, AI behavior, movement systems, ability systems, custom editor tools, and technical gameplay workflows across Unreal Engine 5 C++ and Unity C#. Strong focus on clean modular code, debugging, documentation, and maintainable systems that support iteration.
- **Languages:** C++, C#, Java, Python | **Game Engines:** Unreal Engine (C++ & Blueprints), Unity | **Tools:** Visual Studio, Git, Trello, Command Line / Bash, Maya
- **Frameworks:** Unreal GAS (AttributeSet, GameplayAbilities, GameplayEffects, Gameplay Tags), Behavior Trees, Blackboard, AI Perception, EQS, CharacterMovementComponent, Enhanced Input, Unity XR Interaction Toolkit, A* Pathfinding (Seeker API), NavMesh

EDUCATION

Bachelor of Arts, Double Minor, Computing Science & Interactive Arts and Technology Sept 2019 – Dec 2024
Simon Fraser University, Burnaby, BC

TECHNICAL PROJECT EXPERIENCE

GAS Melee Combat System (Unreal Engine 5, C++ / Blueprints) — Solo Project Apr 2026 – Jul 2026
UE5 · GAS · C++ · Blueprints · Behavior Trees · EQS · AI Perception · Motion Warping · GASPALS
Gameplay / AI Programmer

- Built GAS combat architecture in C++: ASonicCharacterBase (ASC ownership, ability granting), UBasicAttributeSet (Health, MaxHealth, Stamina, MaxStamina, Damage meta-attribute), and USonicGameplayAbility base class. All damage funnels through a centralized PostGameplayEffectExecute handling health delta, GameplayCue dispatch, HitReaction events via HandleGameplayEvent with populated FGameplayEventData, and UAISense_Damage reporting for AI perception.
- Implemented a multi-hit light combo and hold-to-charge heavy with stamina gating, sprint variant snapshotting cached once at ActivateAbility, and a timed perfect parry window using timestamp comparison. Counter-damage on successful parry flows through the standard PostGameplayEffectExecute pipeline via GE_Damage_Instant with no special-casing, triggering enemy hit reactions automatically.
- Built modular enemy AI using Behavior Trees, Blackboards, EQS, and AI Perception across multiple combat states; designed a combat token system to limit simultaneous attackers and improve encounter pacing, and a struct-driven multi-phase boss with chance-based attack selection, EQS teleport targeting, and arena-control AOE patterns.
- Implemented directional hit reactions via Arrow Components on the enemy serving dual purpose: closest arrow to impact point selects the stagger montage, and arrow world position drives Motion Warping combat magnetism for player attack tracking. Added a Stunned AI state to halt in-flight BT MoveTo tasks during root-motion reaction montages.
- Diagnosed a MetaSound PIE freeze via log forensics on a frozen editor process, tracing it to a GUID collision from file-copying assets rather than editor-duplicating; retracted an earlier misdiagnosis after the log's final lines before silence pointed to the audio registry. Built event-driven UI with zero Tick polling using Wait for Attribute Changed bindings and a Timeline-driven ghost health bar.

Advanced FPS Jetpacking Movement (Unreal Engine 5, C++) — Solo Project Nov 2025 – Feb 2026
Unreal Engine 5 · C++ · Blueprints · CharacterMovementComponent · Enhanced Input · AudioComponent · GameplayAbilitySystem
Gameplay Programmer

- Debugged and migrated Epic's ShooterGame from UE4 to UE5 via C++ across an unfamiliar large codebase, identifying and resolving deprecated engine API calls, modernizing the rendering pipeline (Lumen, VSM, TSR, DX12).
- Built a custom C++ async action system exposing cancellable, curve-driven movement forces to Blueprints via OnComplete/OnFail delegates with structured lifecycle management, replacing LaunchCharacter impulses with predictable, tunable forces and eliminating a class of unpredictable physics edge cases.
- Designed and implemented an 8-state locomotion architecture (Walk, Sprint, Crouch, Slide, WallRun, Jetpack, Prone, Airborne) with centralized state transitions, deterministic conflict resolution between overlapping movement systems, and an analog curve-driven jetpack with frame-rate-independent fuel logic.
- Integrated Unreal's Gameplay Ability System end-to-end: AttributeSet, GameplayAbilities, GameplayEffects, and Gameplay Tags for state conflict resolution. Wired AnimNotify and AnimNotifyState to trigger GameplayEvents and GameplayCues directly from animation frames, keeping gameplay logic driven by animation state rather than code timers.

- Applied 3D vector math for movement direction, projection, and force calculations. Identified and resolved a timer synchronization bug affecting short-thrust jetpack fuel drain, and a double-trigger audio bug by switching Enhanced Input bindings from Triggered to Started/Completed.
- Integrated FPS animation pack via interface-driven component access, enabling layered character behavior (ADS, recoil, FOV transitions) across all locomotion states using Layered Blend Per Bone for weapon animation compositing.

The Storge — Horror 2D Game (Unity, C#) — 4-Person Team Sept 2023 – Dec 2023

Unity · C# · A Pathfinding (Seeker API) · NavMesh · Rigidbody2D · Unity Editor API (C# Reflection) · AudioSource*
Gameplay Programmer

- Built a custom Unity Editor automation tool using C# reflection (FieldInfo/MethodInfo) to iterate CompositeCollider2D geometry and auto-generate configured ShadowCaster2D components across all tilemaps, eliminating repeated manual setup and reducing per-level configuration time for the team.
- Implemented a multi-state AI FSM (Patrolling, Chasing, Attacking, RandomMoving) with centralized SetState() managing transition side-effects including A* / NavMesh component swaps and coroutine-delayed alert phases; validated behavior across multiple structured playtesting sessions.
- Built a sound-as-gameplay mechanic where SFX events dynamically retargeted enemy patrol routes, turning audio into a tactical gameplay input. Implemented 3D positional AudioSource emitters in a 2D game using temporary world-space GameObjects with spatial blend.
- Developed noise-driven and line-of-sight detection systems dynamically rerouting pathfinding targets and modifying AI aggression states based on environmental triggers.
- Synchronized animation-event-driven damage registration via animation callbacks to ensure hit windows matched character state exactly, removing frame-timing inconsistencies identified in playtesting.

Slow & Steady — Rogue-like 2D Shooter (Unity, C#) — 4-Person Team Sept 2023 – Jan 2024

Unity · C# · A Pathfinding (Seeker API) · Physics2D · Singleton / State / Observer / Factory patterns · AudioSource*
Gameplay Programmer

- Engineered a modular hitscan combat system separating input (Update) from physics (FixedUpdate) for frame-rate-independent gameplay; iterated on combat feel through structured playtesting, adding hit marker SFX, kill confirmation audio, and an ADS damage multiplier based on direct player feedback.
- Implemented a time-scaling slow-motion mechanic with extended duration tied to consecutive hit accuracy, a recharge system triggered by kills after a combo threshold, and compensated physics forces and camera interpolation to preserve responsive player control under altered Time.timeScale.
- Developed a multi-archetype enemy AI suite (5 types + boss) using State pattern enums and A* pathfinding; CybertruckBossAI uses health-threshold BossState transitions with randomized spawn-point selection.
- Built a centralized AudioManager pipeline routing gameplay events to context-specific SFX with per-callsite volume control; implemented slow-motion pitch scaling synchronized to Time.timeScale.