

Last Sonata

Design and Analysis

What have I done so far as a solo game developer?

By Lejia Mu

October 19, 2025

Weapon System

- **Weapon Framework** – Core architecture supporting modular weapon design and extensibility for multiple weapon types.
- **Firearm Mechanics** – Implemented shooting logic (projectile, hitscan, and spread patterns), recoil, and rate-of-fire balancing.
- **Melee Combat Mechanics** – Designed close-range attack system with swing detection, hit registration, and stamina integration.
- **Reloading System** – Created magazine and chamber-based reloading logic, supporting partial reloads and interrupt handling.
- **Weapon Switching & Slot Management** – Built seamless equip/unequip system with animation synchronization and hotkey mapping.
- **Visual & Audio Feedback** – Designed VFX/SFX for muzzle flash, bullet impact, shell ejection, and weapon-specific audio cues.

Character Input & Control System

- **Buffered Input System** – Implemented input buffering to queue player actions during animation states
- **Action Priority & Interrupt Logic** – Built rules to determine which buffered actions can override or interrupt current animations
- **Combat Input Integration** – Synchronized weapon firing, melee attacks, and ability activations with input buffering for tight combat responsiveness.

Game UI System

- **HUD (Heads-Up Display)** – Designed real-time overlays displaying health, stamina, ammo, and status effects with dynamic updates.

Dynamic Crosshairs System

- **Crosshair Framework** – Core architecture supporting modular crosshair styles and behaviors across weapon types.
- **Weapon-Specific Profiles** – Implemented configurable crosshair layouts (rifle and pistol) with customizable parameters per weapon class.
- **Dynamic Spread & Accuracy Subsystem** – Linked crosshair expansion to weapon recoil, bullet spread, and character movement for real-time accuracy feedback.
- **Hit & Damage Feedback** – Added contextual crosshair responses (color change, hit marker, kill indicator) on enemy impact.

AI Systems

- **AI Framework Architecture** – Designed a modular enemy AI system fully implemented in Blueprints, with a shared parent class and branched child classes for distinct behaviours
- **Perception Subsystem** – Configured vision and hearing detection using Pawn Sensing
- **State-Based AI Logic** - Implemented a lightweight blueprint logic for switching between Idle, Patrol, Chase, Attack, and Die

AI Animation System

- **Snapshot Pose Integration** – Captured current animation poses for smooth blending between active animations and ragdoll states (e.g., transition from knockdown to collapse).
- **Hit Reaction & Force Application** – Applied directional forces based on weapon impact and projectile trajectory to produce realistic responses.
- **Animation Blending Subsystem** – Used Blend Spaces and Anim Montages to seamlessly interpolate between locomotion, combat, and physics-driven poses.
- **Damage Type & Context Awareness** – Triggered context-sensitive reactions based on damage type (rifle, pistol, knife) and hit location (headshot, bodyshot).

AI Pathfinding System

- **Navigation Mesh Integration** – Configured static and dynamic NavMesh to allow AI to traverse complex environments.
- **NavLink Proxy Subsystem** – Implemented jump, climb, and vault links to connect disconnected areas, with configurable priority to control AI traversal order.

Climb & Vaulting System

- **Traversal Framework** – Developed a modular system enabling smooth character interaction with environmental obstacles like vehicles and fences.
- **Obstacle Detection Subsystem** – Implemented collision and raycast-based detection to identify climbable and vaultable surfaces in real-time.
- **Motion Warping Integration** – Dynamically adjusted animation root motion to match target locations, ensuring precise positioning during vaults, climbs, and jumps.

AI Door Interaction & Breach System

- **Door Interaction Framework** – Modular system allowing AI to detect, evaluate, and interact with doors in the environment.
- **Area Detection Subsystem** – Implemented spatial checks and trigger volumes to determine if players or objects are behind doors before initiating an action.
- **Door Breach Logic** – Enabled AI to dynamically decide whether to break or bypass doors based on player presence.
- **Destructible Door Integration** – Linked AI actions to destructible door assets, including physics-driven breakage.
- **Priority & Pathfinding Coordination** – Integrated with AI pathfinding to adjust routes and avoid bottlenecks when multiple AI agents target the same door.

AI Dismemberment System

- **Dismemberment Framework** – Modular system for dynamically handling limb detachment and replacement on enemy characters during combat.
- **Limb Replacement Subsystem** – Enabled procedural replacement of damaged limbs with corresponding “dismembered” mesh assets for visual accuracy.
- **Bone Hiding & Skeleton Adjustment** – Implemented hidden bone logic to prevent clipping and maintain mesh integrity after limb removal.
- **Damage Localization System** – Integrated hit detection to determine limb-specific damage based on attack type, weapon, and impact location.
- **Visual Effects & Feedback** – Added niagara effects, blood decals, and audio cues tied to limb dismemberment for immersive combat feedback.
- **Performance Optimization** – Managed limb pool to minimize performance impact in encounters with multiple enemies.

Object Pooling System

- **Pooling Framework** – Designed a modular object pooling system to manage high-frequency spawning and recycling of dismembered limb assets for performance optimization.
- **Limb Spawn & Recycle Subsystem** – Dynamically spawned dismembered limbs upon damage events and returned them to the pool after reaching the capped amount

AI Pooling System

- **AI Pooling Framework** – Developed a system to efficiently manage spawning and recycling of AI characters for large-scale horde encounters.
- **Spawn & Despawn Subsystem** – Controlled AI lifecycle to dynamically spawn enemies in the world and return them to the pool when reaching the capped amount
- **State Reset & Reinitialization** – Reset AI variables, health, animations, and physics states when reactivating from the pool to ensure consistent behavior.

Boss Fight System

- **Multi-Phase Combat Framework:** Implements distinct combat phases triggered by health thresholds or scripted events, introducing new attack patterns, environmental hazards, and AI behaviors.
- **Adaptive Attack Logic:** Utilizes procedural decision-making to vary attack timing and combos based on player distance, direction, and nearby obstacles.

Level Design

- **Branching Area Design:** Implements interconnected pathways and optional exploration zones that offer tactical choices and replayability.
- **Combat Encounter Zones:** Defines AI spawn points, cover placement, and dynamic engagement regions based on player presence.
- **Environmental Narrative System:** Uses visual storytelling elements—such as lighting cues, environmental decay, and interactive props—to reflect world lore and emotional tone.
- **Resource Distribution System (In Progress):** Dynamically places health packs and ammos to encourage strategic exploration and survival pacing.

For visual level-design breakdowns and annotated images, visit my Behance page:

<https://www.behance.net/gallery/223852639/Abandoned-City>