Game Design Document (GDD)

Project Title: Dynamic Online Story (Prototype)

By

Akshay raj singh

akshayrajsingh330@gmail.com

Abstract

Dynamic Online Story is a multiplayer, mission-based game prototype developed using Unreal Engine 5.3, built upon Epic Games' Lyra Starter Game framework. It blends strategic decision-making, branching narratives, and real-time multiplayer gameplay to create a dynamic, replayable experience. Set in a detailed urban environment using City Sample assets, the game features unique player abilities powered by Unreal's Gameplay Ability System (GAS), including custom mechanics like an area-wide enemy elimination skill. Missions adapt based on player logic and choices, supporting non-linear progression and varied outcomes. The prototype includes core systems such as multiplayer networking, custom characters from the Unreal Fab library, and a responsive UI/UX. This project serves as a proof of concept for a scalable, player-driven storytelling game. Development was supported by assets and systems from Epic Games, along with contributions from the Unreal Engine community, Unreal Fab creators, and open-source tutorial developers.

Acknowledgement

I would like to express my sincere gratitude to Epic Games for providing powerful development resources such as the Lyra Starter Game framework, which formed the backbone of this prototype. I extend special thanks to the Unreal Engine community for their continuous support, guidance, and open knowledge sharing. I also appreciate the contributions of the Unreal Fab asset creators, whose high-quality models enriched the visual and gameplay experience. The custom map used in this project was built from scratch, and I'm grateful for the tools and documentation that made its creation possible. Lastly, I thank the developers and educators behind various open-source tutorials, whose efforts enabled the integration of advanced features like gameplay abilities, custom characters, and mission scripting in this project.

Table of content

1. Game Overview

- **Title:** Dynamic Online Story (Prototype)
- **Genre:** Multiplayer Action / Strategy.
- **Platform:** PC (Windows)
- **Target Audience:** Players aged 15–35 interested in tactical, mission-driven multiplayer gameplay.
- **Game Summary:** Players engage in dynamic missions that react to team choices and in-game events. Each session plays differently, offering branching paths and objectives. This prototype demonstrates a basic mission, multiplayer mechanics, and an example of dynamic player power.
- **USP:**
  - Player-triggered global ability (within range)
  - Story elements and collectibles
  - Multiplayer ready via Lyra framework

2. Gameplay Mechanics

- **Core Loop:** Enter match → Mission starts → Collect objectives → Engage or bypass enemies → Mission outcome
- **Controls:**
  - Movement: `W`, `A`, `S`, `D` keys
  - Sprint Forward / Backward: `Shift + W` and `Shift + S` (with animation)
  - Fire: Left Mouse Click
  - Aim: Right Mouse Click
  - Special Ability: Press `U` to trigger custom ability (area elimination within 2500 meters)
- **Game Modes:** Multiplayer mission instance (Co-op or Competitive)
- **Progression System:** Control point mission system (prototype); future potential for strategic team objectives and progression systems like experience or skill trees

3. Story and Characters

- **Setting:** The game unfolds in a custom-built futuristic environment—an original high-tech metropolis designed from the ground up. The setting blends urban warfare elements with sci-fi aesthetics, offering a dynamic and immersive tactical battleground.
- **Narrative:** The current prototype features a control point-based mission structure focused on strategic team play and testing core multiplayer mechanics. Future versions will explore branching missions, evolving objectives, and faction-driven narratives influenced by player decisions and interactions.
- **Player Role:** Players take on the role of a custom-designed operative, engaging in high-stakes control missions. The framework allows for future class-based abilities, loadout customization, and narrative flexibility tailored to player progression and team roles

4. Level & World Design

- **World Source:** The game world is built using a custom-designed map, created from scratch to portray a high-tech, futuristic urban landscape. The environment features diverse architectural elements including skyscrapers, alleyways, rooftops, and interior spaces to support a variety of gameplay scenarios.
- **Design Structure:** The level is structured as a tactical mission zone that emphasizes team coordination, positional strategy, and dynamic engagement. Key areas are designed to support multiple routes and vertical movement, enabling both flanking strategies and defensive positioning.
- **Mission Type:** The current mission revolves around capturing and holding designated control points, requiring players to work together to secure and defend strategic zones. This system lays the groundwork for future mission types such as objective-based assaults, territory control, faction conflicts, and branching team missions.

5. Visual Style & UI

- **Visuals:** The game features a realistic visual style set in a high-fidelity, custom-built futuristic cityscape. The environment leverages photogrammetry-inspired textures, Lumen-powered global illumination, and Nanite-rendered assets to achieve dense geometry and smooth runtime performance. Immersive visual effects such as dynamic lighting, reflective surfaces, atmospheric fog, and volumetric effects further enhance the futuristic tone and realism during gameplay.
- **UI:** Built on the Lyra Starter Game UI framework, the interface includes a clean and responsive player HUD displaying health, ammo, ability indicators, and mission objectives. The UI leverages Slate and UMG (Unreal Motion Graphics) for modularity, responsiveness, and real-time state transitions with support for animated elements.
- **Enhancements:**

❖ A new skeletal mesh character model was imported from Unreal Fab, fully rigged and integrated with UE5's Control Rig and IK Retargeting systems for seamless animation blending.

❖ UI elements were customized using data-driven design principles, allowing rapid iteration on gameplay feedback, alerts, and ability cooldowns.

❖ Support for World Partition streaming ensures that large map sections load efficiently without compromising UI responsiveness.

❖ Planned improvements include context-sensitive prompts, dynamic minimaps, and diegetic UI elements such as in-world holograms and AR overlays to further immerse players in the environment.

6. Audio Design

- **Audio System:** The game builds on the Lyra Starter Game's audio foundation, utilizing Unreal Engine's **MetaSounds** system for modular, real-time audio processing and dynamic sound behavior. It supports **spatialized 3D audio**, allowing immersive sound placement based on player position and the surrounding environment.
- **Sound Effects (SFX):** The prototype features essential sound cues including **gunfire**, **footsteps**, **ability usage**, **UI interactions**, and **control point captures**. Each sound is configured using **cue assets** with adjustable **attenuation settings** to provide accurate directional feedback and distance-based volume scaling. The use of **convolution reverb** and **reverb zones** improves acoustic realism when transitioning between indoor and outdoor spaces.
- **Background Music:** Ambient music tracks are currently used as **placeholders**, delivering a tense, futuristic atmosphere suited to urban combat and mission-driven gameplay. Music is managed using **Audio Components** and **Sound Mixes**, enabling smooth transitions between exploration, combat, and objective-based sequences.
-

7. Technical Details

- **Engine:** Unreal Engine 5.3
- **Core Framework:** Built on the **Lyra Starter Game**, leveraging its modular architecture including **Enhanced Input**, **Gameplay Framework**, and the **Gameplay Ability System (GAS)** for responsive multiplayer gameplay and scalable ability logic.
- **World Assets:** The environment is constructed using a **custom-built map**, designed with high-quality modular assets and enhanced using **Nanite** for dense geometry rendering. The map includes layered verticality, performance-optimized **Level of Detail (LOD)** setups, and dynamic interaction zones to support scalable multiplayer gameplay across a range of hardware.
- **Character:** Fully rigged **Unreal Fab** character asset, integrated with **Control Rig** and **IK Retargeting** for smooth animation blending and precise movement.
- **Special Features:**

❖ **Area-Based Ability Execution:** Pressing the 'U' key triggers a powerful gameplay ability that eliminates all enemies within a 2500-unit radius, implemented through the **Gameplay Ability System** with radial targeting logic.

❖ **Custom Character Integration:** The player character is a custom-imported skeletal mesh from **Unreal Fab**, supporting high-fidelity animation, modular attachments, and seamless movement transitions.

❖ **Modular Mission System:** Missions are built using a **data-driven**, modular structure. The current mission type focuses on **control point capture**, with future scalability for **branching objectives**, **class-specific tasks**, and **environment-triggered events**.

❖ **Open-Ended World Interaction:** The custom environment supports **vertical traversal**, **flanking routes**, and **strategic zone control**, designed for freeform player engagement and team-based objectives.

❖ **Networking:** The game follows a **client-server model**, where the **server authoritatively manages** all core gameplay mechanics including movement, shooting, ability execution, and mission progress. Multiplayer is powered by Unreal's **replication system**, ensuring real-time synchronization of player actions and shared game states. Each player is assigned a **Player Controller**, **Player State**, and **Character**, while match data is shared via **replicated game mode and game state classes**. Additional support includes **matchmaking**, **session management**, and **dynamic content streaming** for flexible multiplayer scalability.

8. Development Roadmap

- **Phase 1:** Set up the core framework using the **Lyra Starter Game**, integrating its multiplayer systems, Enhanced Input, and Gameplay Ability System. Imported a fully rigged **custom character** from Unreal Fab and configured Control Rig + IK Retargeting for seamless animation.
- **Phase 2:** Designed and implemented a **custom-built futuristic map** to serve as the main tactical environment, complete with vertical traversal paths, combat zones, and strategic sightlines.
- **Phase 3:** Developed the first playable **control point-based mission**, where teams must capture and hold zones to win. This tested the core mechanics of team coordination, area control, and replicated mission triggers.
- **Phase 4:** Added a **custom gameplay ability** bound to the 'U' key, eliminating nearby enemies within a 2500-unit radius. Integrated this ability using Unreal's Gameplay Ability System with replicated execution and visual/audio feedback.
- **Phase 5:** Enhanced the **UI** using Lyra's base framework and Unreal Motion Graphics (UMG), introducing HUD indicators for health, ammo, abilities, and control point status. Added data-driven elements to support future scalability.
- **Phase 6:** Planned and began implementation of a **modular mission scripting system** designed to support **branching mission paths**, team-specific objectives, and environmental triggers.
- **Phase 7:** Conducted a **comprehensive testing phase** focused on gameplay balancing, ability replication across clients, mission objective syncing, and stability in multiplayer sessions.

9. Future Steps & Expansion Plans

- **Expand Mission Variety:** Add new mission types such as **hostage rescue**, **stealth infiltration**, **sabotage**, and **timed objectives** to increase gameplay diversity and challenge.
- **Dynamic Branching Missions:** Implement a **branching mission system** where player decisions lead to alternate win/loss outcomes, unlocking varied paths and strategic replayability.
- **Role-Based Characters:** Introduce distinct **character roles** (e.g., assault, support, recon) with **unique abilities** and **team-based mechanics**, encouraging cooperative play and tactical depth.
- **Community Playtesting:** Conduct **regular multiplayer playtests** with players to gather feedback on game balance, UI clarity, and mission pacing, allowing for iterative improvements.
- **Narrative Campaign Mode:** Develop a structured **campaign experience** with progressive missions, **character growth**, unlockable content, and a functional **save/load system**.
- **VR Exploration:** Investigate **VR support** for delivering an immersive **first-person multiplayer** experience, focusing on intuitive interaction, motion safety, and environmental immersion.

10. Testing & Feedback

- **Manual Playtesting:**
  Extensively tested core gameplay systems—including player movement, control point capturing, shooting, ability execution, and UI behavior—using **Play In Editor (PIE)**, **Standalone mode**, and **Dedicated Server setups** for both single-player and multiplayer environments.
- **Multiplayer Network Testing:**
  Simulated multiplayer scenarios by launching multiple clients to test **replication**, **latency**, and **prediction accuracy**. Used network commands such as `net PktLag` and `net PktLoss` to emulate real-world conditions and assess client-server synchronization.
- **Gameplay Ability System (GAS) Testing:**
  Validated **ability activation**, **effect application**, **cooldowns**, and **replication consistency** using the **Gameplay Debugger** (`ShowDebug AbilitySystem`) along with live in-game testing under multiplayer conditions.
- **Automated Functional Testing:**
  Integrated **FunctionalTest actors** to run scripted gameplay checks, and used Unreal's **Gauntlet framework** for automated **build validation** and **regression testing** across development milestones.
- **Blueprint & Runtime Debugging:**
  Debugged gameplay flow using **breakpoints**, **variable watches**, `Print String`, and `UE_LOG` for real-time analysis. Focused on detecting issues in mission triggers, UI state updates, and replicated gameplay elements.
- **Unit & Automation Testing (C++):**
  Implemented targeted **unit tests** using Unreal's **Automation Framework** to verify the reliability of C++-based systems and custom logic extensions. Tests were executed and tracked through the **Session Frontend > Automation tab**.