# Final Report - Immersive NPC Behavior Systems

**Names**: Dima Gordon (319630182) & Shaked Cohen (312324023)
**Institution**: Afeka Academic College of Engineering
**Date**: 30/06/25

## 1. Abstract

The contemporary gaming industry is heavily dominated by handcrafted open-world titles that prioritize visual fidelity and cinematic presentation. While these games offer rich, curated environments, they often rely on predictable, pre-scripted interactions that limit variability and player agency over time. In particular, non-player characters (NPCs) in such games tend to follow rigid, repetitive patterns, which can detract from immersion and the illusion of a living world. In contrast, games that embrace procedural generation or randomness are relatively rare, and those that do often suffer from incoherent design or a lack of narrative and behavioral depth.

This project proposes a proof-of-concept (POC) for a next-generation game system that blends modular procedural world generation with intelligent, adaptive NPC behaviors to create a more immersive and replayable experience. The game world is constructed from modular tiles–pre-designed yet procedurally assembled scenes–that connect seamlessly to form a coherent, ever-changing landscape. This approach balances the consistency of handcrafted environments with the unpredictability and replayability of generative design.

At the core of this system is the Immersive NPC Behavior System, which enables NPCs to exhibit logical, goal-oriented actions that respond dynamically to environmental changes and player behaviors. NPCs are not merely scripted entities but autonomous agents capable of adapting to evolving in-game contexts. When integrated with the modular world, these adaptive behaviors contribute to a dynamic ecosystem where gameplay is emergent rather than predetermined.

Together, these systems aim to challenge traditional design paradigms by offering a game experience that is not only visually and structurally dynamic but also rich in behavioral complexity. The result is an open world that feels alive, responsive, and capable of offering unique narrative and gameplay possibilities with every session.

## 2. Table of Contents

## 3. Introduction

Non-player characters (NPCs) play a crucial role in shaping the player's experience within modern video games. They contribute not only to world-building and storytelling but also to the sense of immersion and realism that defines many contemporary open-world and role-playing games. As players increasingly expect dynamic and lifelike interactions, the need for intelligent, adaptive NPC behavior has become more critical than ever.

**Motivation -** Traditional NPC systems often rely on rigid scripts and predictable patterns, which can break immersion and reduce engagement. In contrast, modern games like Red Dead Redemption 2, Cyberpunk 2077, and The Witcher 3: Wild Hunt showcase the power of sophisticated AI-driven behavior to create vibrant, believable worlds. These examples demonstrate how advanced NPC behavior systems can greatly enhance the sense of presence and interactivity in a game. However, designing such systems remains a complex challenge, especially when combined with procedurally generated environments or dynamic world states. Addressing this challenge is vital for pushing the boundaries of interactive storytelling and open-world gameplay.

**Problem Statement -** This project addresses the problem of creating an intelligent and immersive NPC behavior system that can operate within dynamic and procedurally generated game worlds. Specifically, the challenge is to design NPCs that can respond to both environmental changes and player actions in a believable and context-sensitive way. Existing systems often fail to scale or adapt well in such settings, leading to either shallow interactivity or brittle behavior patterns.

**Project Goals and Objectives -** The goal of this project is to develop the Immersive NPC Behavior System–a modular AI framework for NPCs that supports adaptive, context-aware decision-making and animation logic. The system aims to:

- Simulate realistic social, reactive, and survival behaviors in NPCs
- Integrate seamlessly with the Modular World Generation tool to support dynamically changing environments
- Handle both scripted and emergent behavior based on player actions and environmental triggers
- Support a variety of NPC roles (e.g., civilians, law enforcement, hostile agents) with distinct behavioral patterns

**Overview of Approach and Contributions -** The approach combines behavior trees, state machines, and environmental triggers with customizable NPC roles and animations. The system includes a Danger Sub-State Machine that switches behaviors based on external events (e.g., tornadoes, drawn weapons), as well as role-specific logic (e.g., lawmen approaching armed players, civilians fleeing danger). Key contributions of this project include:

- A robust framework for danger perception and role-based response
- Smooth transitions between idle, alert, and reactive states using Unity's animation and AI tools

- Demonstrated scenarios of emergent gameplay based on dynamic interactions between NPCs and the environment
- Comparative analysis with behavior systems from major titles, identifying both inspiration and improvements

This work contributes toward the development of richer, more immersive AI agents that not only enhance gameplay but also push forward the design principles of believable digital characters.

## 4. Background and Related Work

As games have evolved, so too have expectations for non-player characters (NPCs). No longer is it enough for NPCs to stand idly or follow repetitive paths–today's players expect them to react intelligently, display social awareness, and contribute meaningfully to immersive worlds. While modern game engines and toolkits offer a variety of solutions for behavior control, they often fall short when applied to dynamic and procedurally generated environments. The Immersive NPC Behavior System aims to close this gap by combining modular behavior logic, role-based AI, and environmental reactivity into a cohesive, developer-friendly framework.

**Existing Industry Practices and Limitations -** Contemporary open-world titles like Skyrim, Grand Theft Auto V, and The Witcher 3 have set a high bar for environmental immersion, with massive, detailed worlds populated by hundreds of NPCs. However, the underlying behavior of these characters is usually governed by predefined routines and simple state machines. While effective for maintaining narrative consistency, these models often result in static, repetitive behavior that lacks the nuance or spontaneity of real-life interaction.

For example, in Skyrim, NPCs may follow daily routines–eating, sleeping, working–but fail to adapt when unexpected events occur unless explicitly scripted. Similarly, in GTA V, crowd NPCs rarely react meaningfully beyond hardcoded panic or aggression triggers. These behaviors are often built using finite state machines (FSMs), which offer straightforward logic flow but can become brittle and bloated in complex systems.

**Tools and Frameworks -** Middleware tools like Unity's NavMesh system simplify NPC navigation and pathfinding across terrain but provide no built-in mechanisms for behavioral adaptation or decision-making. Additionally, many of these systems are not optimized for procedurally generated environments. Most are designed with static world assumptions in mind, and adapting them to environments where object placement, terrain, or events change in real-time can be tedious or error-prone.

**Academic Approaches and Behavior Models -** In the academic and commercial sectors, procedural content generation (PCG) has garnered substantial attention, particularly for terrain and object generation in games like Minecraft or No Man's Sky. However, these games prioritize world creation over dynamic NPC interaction. Few PCG frameworks deeply explore NPC behavioral variability or emergent AI in interactive spaces.

On the behavior modeling side, behavior trees (BTs) remain a dominant technique in commercial games, offering scalable hierarchical control. Paired with FSMs, they enable robust control structures, though they still require extensive manual design effort and can struggle to support adaptive behavior without added complexity. More advanced techniques, such as utility-based AI and GOAP (Goal-Oriented Action Planning), offer dynamic decision-making but introduce performance and debugging challenges in large-scale NPC systems.

**Gaps Addressed by This Project** - This project addresses several major shortcomings in existing approaches:

- Lack of Adaptability: Most NPC systems assume static environments and cannot easily adjust to dynamic world changes. This system is built with procedural generation in mind, allowing NPCs to interpret and respond to unfamiliar scenes and layouts.
- Limited Role-Based AI: Many tools treat NPCs as generic agents. The Immersive NPC Behavior System defines distinct role behaviors (e.g., law enforcement, civilians, troublemakers) with a shared but adaptable sub-state machine that allows them to respond differently under the same conditions.
- Complexity vs. Flexibility Trade-off: Existing frameworks are either oversimplified or overly intricate. This system emphasizes a lightweight, reusable design that balances structured behavior with opportunities for emergent interactions and unpredictability.
- Integration with Environmental Context: Unlike tools focused solely on AI logic or movement, this system integrates NPC behavior with scene-level elements like time of day, proximity to the player, and social context, yielding more responsive and immersive gameplay.

By enabling NPCs to adaptively interact with both players and procedurally generated environments, this project lays the groundwork for more believable, context-aware, and narratively meaningful AI in future games.

## 5. System Design / Methods / Approach

### Functional Requirements

- NPCs simulate logical, task-driven behavior.
- NPCs adapt to player proximity and presence.
- NPCs can wander, gather, flee, and form group dynamics.
- NPCs change behavior states based on nearby NPCs and objects.
- The tool allows developer-defined prefabs and spawn settings.

### Non-Functional Requirements

- Performance: must operate efficiently with multiple NPCs in a large scene.
- Modularity: easy to plug into different scenes or projects.
- Scalability: new behavior states should be easily added.
- Usability: clear inspector UI for designers to configure NPC types and zones.

### System Architecture

- Core Modules: NPC Generator, NPC Movement, NPC Behavior
- NavMesh: Provides navigable terrain for all movement decisions.
- Animator: Manages transitions based on state flags (e.g., run, dance, idle).

### Technology Stack

- Engine: Unity (2022+)
- Language: C#
- AI Tools: Unity NavMesh, custom logic scripts
- Animation: Unity Animator with parameter triggers

## 6. Methodology

The system was built through iterative testing within Unity's Play Mode.

**The development process involved:**

1. Creating modular NPC prefabs with standard components: NavMeshAgent, Animator, and NPC scripts.
2. Designing an editor-based NPC Generator that enables designers to spawn and manage NPCs.
3. Defining behavior logic in the NPCBehavior script: idle, walking, running, group interaction, fleeing.
4. Implementing condition-based state transitions using triggers and float parameters within the Animator.
5. Testing interactions in various scenarios: solo wandering, crowd reactions, law enforcement presence.
6. Debugging spatial navigation issues and animation syncing.

Key design choices focused on clarity and reuse–scripts are modular and abstract enough for developers to plug in different models and animations without altering core logic.

**Challenges addressed include:**

- Animator Complexity: Simplified with boolean flags and int routeIDs.
- Spatial Navigation: Integrated escape logic and pathfinding validation.
- Group Logic: Designed a group behavior radius system to detect when multiple NPCs trigger events (e.g., group dancing).

## 7. Implementation

**The final implementation includes three key scripts:**

- NPCGenerator: Spawns NPCs during Play Mode based on developer parameters (count, type, prefab, radius).
- NPCMovement: Handles all movement logic, including free wandering, purposeful walking, and fleeing from threats using NavMesh.
- NPCBehavior: Governs state transitions such as gathering, dancing, or fleeing based on proximity to NPCs or players.

**Animator parameters used include:**

- Run (bool), Dance (bool), Discrete (bool)
- runRoute (int), danceRoute (int)

All logic ensures that NPCs react fluidly in real-time. Escape behaviors calculate a vector away from the player, validate NavMesh points, and then navigate NPCs to safety.

## 8. Experiments and Results

Although this project is not rooted in formal experimental research, it underwent extensive in-engine prototyping and scenario-based testing to validate its functionality and effectiveness. Key testing scenarios included:

- **Wandering and gathering behaviors** in low-density and high-density NPC populations.
- **Reactive behaviors** are triggered by player proximity or the presence of law enforcement.
- **Fleeing logic** in constrained or obstructed environments.
- **Animation-state transitions** under varying interaction frequencies.

Results demonstrated that:

- NPCs transitioned between states smoothly under real-time conditions.
- Fleeing logic using calculated escape vectors produced believable evasion patterns.
- Group-based behaviors (e.g., dancing when 3+ hooligans gather) are consistently triggered and desynchronized properly.

## 9. Discussion

### Insights

- Even relatively simple behavior patterns–like fleeing, gathering, or responding to player proximity–create a powerful illusion of awareness and autonomy. Players perceive these NPCs as intelligent because their actions reflect environmental and social stimuli.
- A key challenge was balancing complexity with performance and usability. Systems like GOAP or utility-based AI offer greater realism but introduce debugging and scalability issues. This system favors a modular approach using FSM and BT-style components, offering clarity and extensibility.
- Unscripted moments–like NPCs gathering into a group, responding to a lawman, or dispersing during conflict–can feel organic and surprising to players. This unpredictability fosters replayability and engagement.

### Limitations

- The system does not currently support long-term memory or persistent NPC states.
- Inter-NPC communication is limited to broadcast reactions; deeper dialogue or emotional models are future goals.
- The simulation runs only during Play Mode, meaning no persistent data is carried across sessions.

### Potential Improvements

- Incorporate machine learning or behavior trees for longer-term decision making.
- Enable saving/loading of NPC states across sessions.
- Add environmental awareness, like weather or terrain conditions.

## 10. Conclusion and Future Work

This project demonstrates a modular, adaptive NPC behavior system that responds fluidly to environmental triggers and player actions. By integrating procedural scene generation with role-based behavior logic, it offers a foundation for building immersive, believable NPCs in dynamic game worlds.

Key achievements include:

- A flexible behavior framework built in Unity using standard tools and custom scripts.
- Distinct behavior sets for different NPC roles, with logical transitions between passive and reactive states.
- A spawning and scene integration system compatible with modular world layouts.

**Future Work** may involve:

- Adding persistent memory and emotional states to NPCs for deeper interactivity.
- Incorporating dialogue systems to support social interaction and branching narratives.
- Integrating more advanced decision-making systems like GOAP for long-term planning.
- Porting the system into networked environments or multiplayer sandboxes.
- Conducting formal user testing to measure perceived realism and engagement.

## 11. References

- ❖ Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., & Zaremba, W. (2016). *OpenAI Gym*. arXiv preprint arXiv:1606.01540.
- ❖ CD Projekt Red. (2015). *The Witcher 3: Wild Hunt* [Video game]. CD Projekt.
- ❖ IGN. (2020). *Cyberpunk 2077 Review*. https://www.ign.com/articles/cyberpunk-2077-review
- ❖ Isla, D. (2005). Handling complexity in the Halo 2 AI. *Game Developers Conference (GDC)*.
- ❖ Orkin, J. (2006). Three States and a Plan: The AI of F.E.A.R. *Game Developers Conference (GDC)*.
- ❖ Robicquet, A., Sadeghian, A., Alahi, A., & Savarese, S. (2016). Learning social etiquette: Human trajectory understanding in crowded scenes. *European Conference on Computer Vision*, 549–565.
- ❖ Rockstar Games. (2013). *Grand Theft Auto V* [Video game]. Rockstar North.
- ❖ Rockstar Games. (2018). *Red Dead Redemption 2* [Video game]. Rockstar Games.
- ❖ Bethesda Game Studios. (2011). *The Elder Scrolls V: Skyrim* [Video game]. Bethesda Softworks.
- ❖ Togelius, J., & Yannakakis, G. N. (2016). Procedural Content Generation. In *Artificial Intelligence and Games*.
- ❖ Unity Technologies. (2024). Unity Manual: NavMesh Agent. https://docs.unity3d.com/Manual/nav-Agent.html
- ❖ Isla, D. (2005). Handling Complexity in the Halo 2 AI. *Game Developers Conference*.
- ❖ Millington, I., & Funge, J. (2016). *Artificial Intelligence for Games*. CRC Press.