Immersive NPC Behavior System

By Dima Gordon and Shaked Cohen

Research question

How can advanced NPC design and intelligent behaviors enhance player immersion in dynamic virtual worlds?

Enhancing immersion through intelligent NPCs

Creating dynamic, interactive game worlds with advanced NPC behaviors

Integrating technical frameworks for seamless player experiences

Competitors, Literature and Insights

Introduction

Non-player characters (NPCs) are essential in building immersive game worlds. Modern games utilize advanced AI systems to create lifelike behavior, contributing to the player's experience.

Competition is not hard to come by throughout the gaming industry with many games coming out each year with varied levels of NPC integration. Thus competitors are varied and abundant.

A prime example

Lets take a look at a highly acclaimed game, Red Dead Redemption 2, as a prime example of immersive NPCs exhibiting lifelike behavior.

Rockstar Games emphasized a world where NPCs appear independent and responsive. They created a framework where NPCs interact with their surroundings and player behavior to portray the Wild West convincingly.

Game articles

In a marketing endeavor, half a year before a world-wide release on gaming consoles, Rockstar Games offered a sneak peak into the inner working of their upcoming game to several publications.

In a short article by **SegmentNext**, it was reported that NPCs in *RDR2* dynamically react to subtle player actions (e.g., weapon holstering influencing friendliness or fear). Moreover they exhibited personalities that further deepened realism, making interactions less predictable and more engaging.

Code reverse structure examination

Upon further inspection of the code of the game, several features are made apparent in the manner the NPCs are programed.

- **Schedules:** NPCs follow unique routines based on their societal roles, influenced by time and context.
- **Context-Aware Behavior:** NPCs assess the player's appearance, actions, and surroundings.
- **Reputation Systems:** NPCs remember past player actions, integrating them into future interactions.
- **Emergent Reactions:** Unscripted events spark dynamic responses.

Technical Insights

- **Behavioral states/ routines**: NPCs must follow basic routines and should dynamically switch between various behavioral states based on contextual stimuli.
- **Personality profiles**: Each NPC has its personality profile that dictates its responses to stimuli, such as bravery, fearfulness, or neutrality.
- **Hierarchical Behavior Trees:** NPCs should follow a hierarchical structure where high-priority behaviors override less urgent ones.
- **Emotive Animations:** Employing advanced procedural animation blending for realistic NPC reactions.

Functional requirements

- **Dynamic NPC Behavior**- The system must support creating adaptive behaviors for various NPC types (e.g., civilians, and police officers) with role-specific characteristics.
- **Real-Time Reactions to Player Actions**- NPCs should dynamically respond to player actions (i.e. drawing out a weapon, hitting someone, initiating conversation, and so on), including avoidance (e.g., moving out of the player's way), assistance (e.g., offering guidance or support), or aggression (e.g., confronting the player when provoked).
- **Integration with Modular World Generation**- The system should align with environments created by the Modular World Generation tool, incorporating scene elements such as terrain, objects, and pathways into NPC behavior logic.
- **Environment Constraints Compliance**- NPCs must follow scene-specific rules, navigating only walkable areas and avoiding obstacles or non-walkable zones like cliffs or walls.
- **Contextual Decision-Making**- NPCs should base decisions on contextual factors, such as modifying actions depending on day or night and reacting to nearby events or player actions.

Non-functional requirements

- **Performance**- The system should ensure behavior updates in standard gameplay scenarios.
- **Scalability**- The system must handle at least 10 active NPCs in a single scene without significant performance degradation.
- **Reliability and Availability**- NPC behaviors must remain consistent and error-free in gameplay sessions.
- **Security**- The system must secure NPC behavior logic against unauthorized modifications or tampering.
- **Integration**- Seamless integration with Unity and compatibility with Modular World Generation for synchronized updates.

Architecture

Unity Built-in Tools

Utilizes Unity's native tools for efficient NPC design and behavior integration.

Scripted Behavior

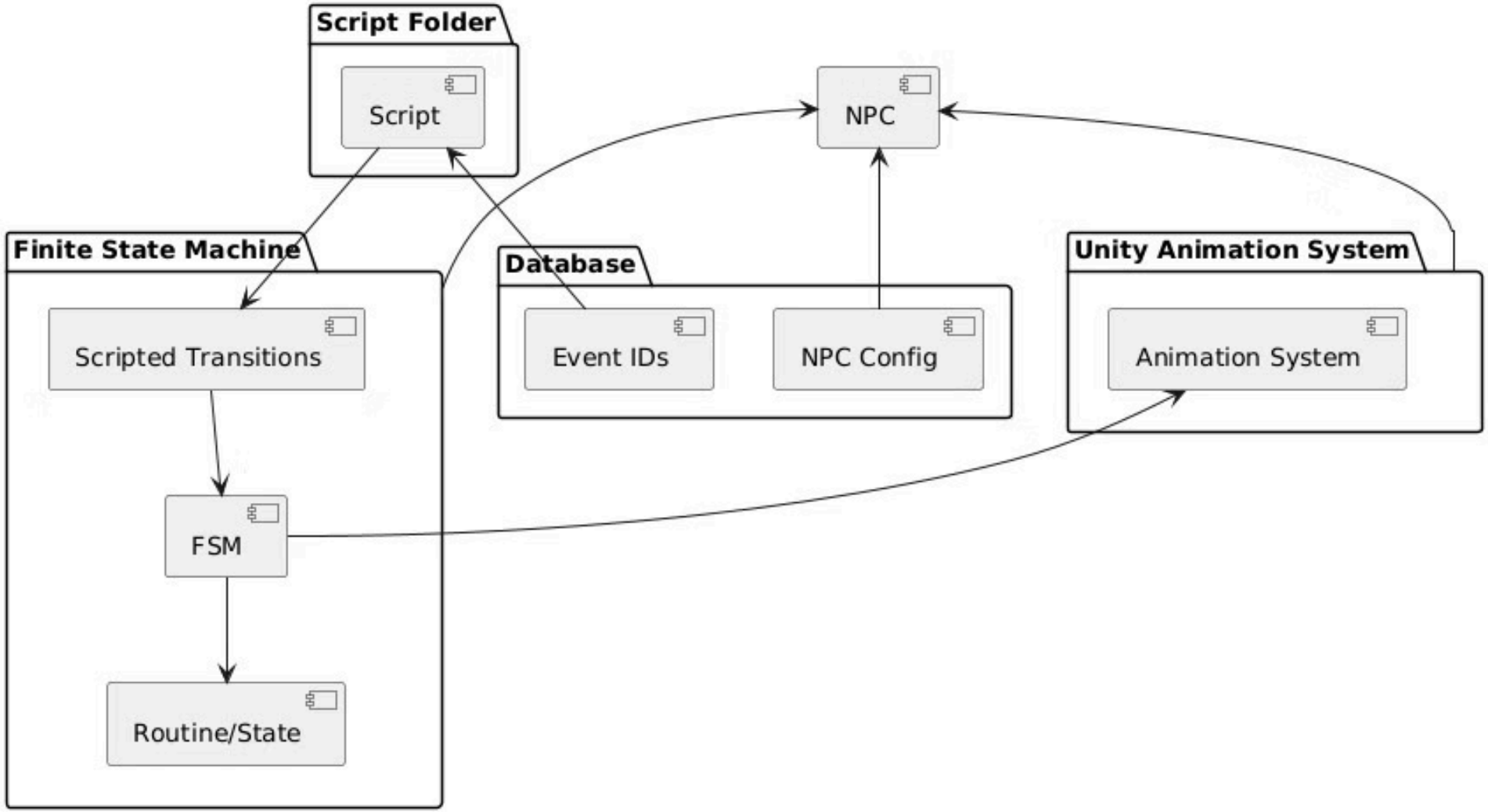
- **Environment-Influenced Script**: NPCs react to environmental changes.
- **Personality-Driven Script**: NPCs' reactions are based on their personality.
- **Player-Driven Script**: NPCs respond to player actions.

Modular Design

The system will isolate NPC behavior logic from scene generation, ensuring flexibility and easier updates.

Decision Trees

- **Decision Trees**-NPC behavior will be driven by decision trees, allowing structured decision-making based on environmental variables, state, and player interactions.
- **Storytelling Package**-Manages branching narratives and NPC memory, affecting future interactions and storylines.



Algorithm

1

Finite State Machine (FSM)

Each NPC will have an FSM that allows seamless transitions between states or tasks, enabling dynamic behavior management.

2

Animation Integration

Unity’s animation system will be used to apply animations to objects, enhancing the illusion of complex movements where necessary.

3

NavMesh for Spatial Movement

Unity’s built-in NavMesh algorithm will be used for basic NPC movement and will serve as the foundation for more complex movement options.

Database Design

NPC Behavior Schema:

NPC_ID (int): Unique identifier for each NPC.

Personality_Type (enum): Defines NPC traits (e.g., Friendly, Hostile).

Situation_State (enum): Current context for decision-making (e.g., Normal, Panic).

Behavior_Tree_ID (int): Links to a pre-defined decision tree for complex logic.

Position (Vector3): NPC's current location.

Story Triggers Schema:

Trigger_ID (int): Unique identifier for story events.

Associated_NPCs (list): NPCs affected by the trigger.

Action (string): Behavioral changes tied to the trigger (e.g., flee, assist).

Development Stages

1 Set Up Initial Scene

We start with an empty Unity space and place 3 NPC characters, each with a different type.

2 Define NPC Routines & Personalities

We will create basic routines for each NPC and assign unique personalities.

3 Create Behavior Scripts for Events

Develop scripts for both global and player-triggered events, linked to NPC personalities

4 Event Injection Criteria

Define how events influence NPC behavior based on global and player-driven inputs

5 Place Main Player & Handle Event Injection

Add the main player to the scene and manage event injections based on player actions

6 Begin Scalability Processes

Prepare the system for scalability to handle additional NPCs and events

Thank You For Listening