Game Development
Summer Semester 2025

Submitted by: Joé Welsch (1526418)

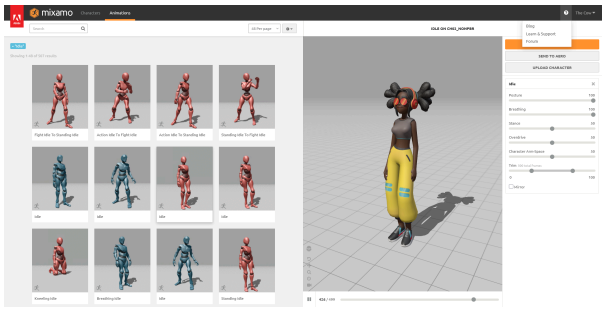
Overview

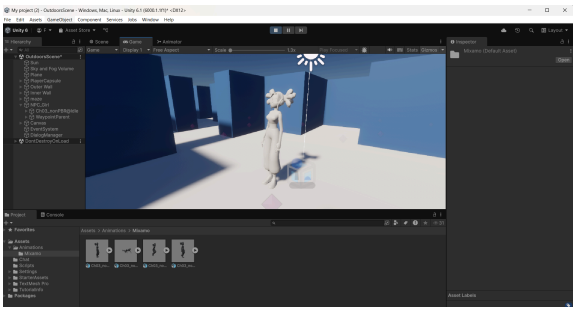
This project demonstrates how an NPC in Unity can be controlled using natural language commands processed by ChatGPT. The NPC can stand idle, walk, run, or crawl between waypoints, depending on the instructions provided by the user. Gameplay video: https://youtu.be/27rN1tsXll8

Character and Animations

I began by exploring how to create NPCs and manage animations in Unity. To learn the basics, I followed this YouTube playlist: https://www.youtube.com/watch?v=-FhvQDqmgmU&list=PLwyUzJb_FNeTQwyGujWRLqnfKpV-cj-eO

After that, I downloaded a character model and several animations (Idle, Walk, Run, Crawl) from Mixamo. These assets were imported into Unity and organized in an Animator Controller containing all four animation states. I set Idle as the default so the NPC would start in place.



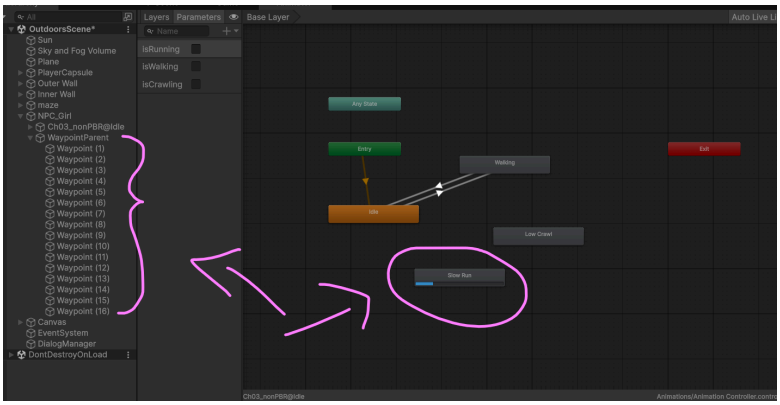


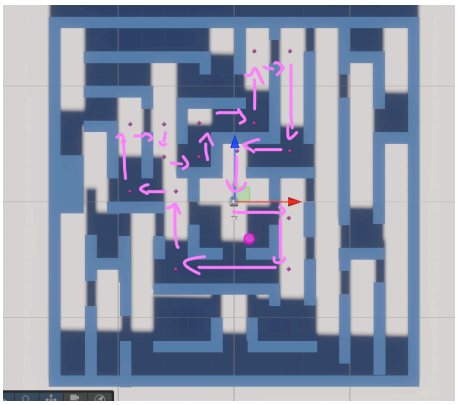
Waypoint Movement

I learned how to move characters along waypoints by following this tutorial:
https://www.youtube.com/watch?v=40oTxM6cSYw

I placed waypoint GameObjects in the scene to define a path the NPC should follow. Then I created a script that moves the character smoothly between these points, rotates to face the next target, and waits briefly before continuing to the next waypoint.

To test how the animations looked in motion, I first wrote a script that automatically cycled through Idle, Walk, Run, and Crawl while the NPC moved along the waypoints (AnimationScript.cs). This helped me confirm that the imported Mixamo animations blended correctly. Later, I updated the movement logic so it could be started, stopped, and have its speed adjusted dynamically based on ChatGPT response using the (NPCAnimationController.cs) Script.



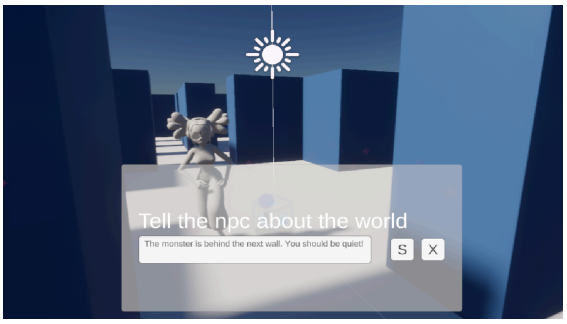


Dialog System

For user interaction, I implemented a pop-up window with an input field where text commands can be entered. This was based on a tutorial explaining the fundamentals of creating in-game UI: https://www.youtube.com/watch?v=JivuXdrIHK0

The popup is made of a canvas containing a dialog panel, buttons, and a title and description. It is hidden when the game starts. Pressing E makes the window appear so the player can enter a story about the world. The Logic is handled in the (`DialogManager.cs`) Script.

I adapted it so that the dialog:



- Automatically focuses when opened
- Player can type whatever a story
- Submits text by pressing Enter
- The content is later send to chatGPT
- Can be closed with Escape
- Pauses the game while active

ChatGPT Integration

To enable natural language control, I integrated the OpenAI Chat Completion API by following this video guide: https://www.youtube.com/watch?v=qgT-quk3JEo

The prompt was configured to force GPT to reply only in a strict format such as “ACTION: WALK” so the responses could be reliably parsed. I also implemented error handling to default to a safe behavior (Idle) if any unexpected output occurred. The logic was also implemented in the (`DialogManager.cs`) Script.

Unfortunately, this does not always work as expected. ChatGPT still sometimes returns unexpected responses. For example, a prompt like “Be quiet, the monster is behind you” might produce the action “BE QUIET” instead of the desired response “IDLE.” More prompt engineering would be necessary to improve the consistency of the output.

Connecting Actions to Behavior

Finally, I connected the parsed ChatGPT responses to the NPC's behavior. Based on the received action, the system:

- Switches the animation
- Starts or stops waypoint movement
- Adjusts movement speed dynamically

Example mapping:

- **ACTION: IDLE** → NPC stands still
- **ACTION: WALK** → NPC plays walk animation and moves slowly
- **ACTION: RUN** → NPC plays run animation and moves faster
- **ACTION: CRAWL** → NPC plays crawl animation and stops moving forward