UNITY DEVELOPER INTERNSHIP TRIAL - TASK 3 & 4.1 REVIEW

-

Task 3 & 4.1: AI State Machine for Enemy Behavior & Implement a Basic Combat System

Overview

This document details the implementation of Task 3 (AI State Machine for Enemy Behavior) and the beginning of Task 4.1 (Basic Combat System) in the Unity Developer Internship Trial. The core AI system for the enemy is fully implemented, while bonus features like pathfinding remain. Additionally, the combat system has been integrated and is around 75% complete.

Timeline and Initial Steps

- 12:30 PM → Began development after working on university assignments.
- Afternoon → Completed core Task 3 logic, including state transitions and AI behavior.
- Evening → Moved into Task 4.1 (Combat System), implementing attack animations, input handling, and hit detection.
- Total Time Spent → Efficiently managed both tasks within estimated timeframes.

Day's Progress & Key Milestones

Enemy AI State Machine & Behavior (Task 3 Core Implementation)

- Created Enemy GameObject Parent & Prefab Structure.
- Set up Rigidbody2D, Collider, and Animator.
- Designed 8-direction movement animations and Blend Tree.
- Implemented EnemyStateMachine & Base State Structure.
- Developed and refined Idle, Patrol, Chase, Attack, and ReturnToPatrol states.
- Fixed patrol logic issue (enemy not cycling through patrol points).

- Fixed attack logic (triggering only on collision).
- Prevented unintended enemy-player pushing interaction.

Initial Combat System Implementation (Task 4.1 Progress)

- Created PlayerCombat.cs as a separate component.
- Integrated attack input handling using GameInputSO.
- Developed PlayerAttackBlendTree with 4-directional attack animations.
- Revised animation approach (moved attack animations to a separate child object CombatVisual).
- Implemented knockback force & collision-based attack detection.
- Created visual hit feedback with sprite flickering.
- Fixed double hit detection by using a HashSet to track hit enemies.
- Fixed input stacking issue (attack inputs queuing during cooldown).

Approach & Execution (Chronological Order)

1. ***Enemy AI Implementation & Prefab Setup***
    - Created Enemy Parent GameObject and added EnemyVisual as a child with SpriteRenderer & Animator.
    - Created Enemy Animator Controller with an Idle default state and an EnemyMovementBlendTree for 8-direction movement.
    - Configured Rigidbody2D (Dynamic, Gravity 0, Interpolation, Collision Detection Discrete, Z-Axis Rotation Frozen).
    - Created CapsuleCollider2D, adjusted to fit the enemy's shape.
    - Converted Enemy into a Prefab inside Prefabs/Characters/Enemy/.

2. *Obstacle System & Level Structuring*

- Added 10 rock obstacles with BoxCollider2D, a new Obstacle tag & collision layer.
- Grouped obstacles inside Obstacles Parent GameObject for hierarchy cleanliness.
- Created an Environment Parent GameObject to hold the Tilemap, Obstacles, and Light for better project organization.

3. *AI State Machine Implementation*

- Created EnemyStateMachine.cs, Patrol Points, Detection Range, and Movement Speed.
- Developed EnemyBaseState.cs as the foundation for AI behaviors.
- Implemented and refined the following enemy states:
  - Idle State → Enemy remains still, transitioning to Patrol or Chase based on player proximity.
  - Patrol State → Enemy moves between randomized patrol points and waits at each point before continuing.
  - Chase State → Enemy follows the player when detected.
  - Attack State → Enemy logs an attack message when close to the player and transitions back to chase.
  - Return to Patrol State → Enemy moves back to the closest patrol point when the player escapes detection range.

4. *Refining Enemy Behavior & Addressing Issues*

- Fixed patrol logic issue where the enemy stopped at the first patrol point.
- Refined attack logic so the enemy only attacks upon collision, not from a distance.
- Adjusted enemy physics to prevent being pushed by the player by increasing mass and tweaking Rigidbody settings.

5. Combat System Implementation for Task 4.1

- Created PlayerCombat.cs to handle combat behavior independent of state machine logic.
- Implemented attack input handling via GameInputSO.
- Developed PlayerAttackBlendTree using 4-directional attack animations.
- Moved attack animations to a separate child object (CombatVisual) to fix sprite visibility issues.
- Implemented knockback force upon enemy hit and sprite flickering effect for hit feedback.
- Used a HashSet to track hit enemies, fixing double-hit detection.
- Prevented attack input stacking during cooldown by introducing an isProcessingAttack flag.

Challenges & Reflections

1**. Patrol Logic Not Functioning as Expected**

- Issue: Enemy wasn't continuing its patrol after reaching the first patrol point.
- Solution: Fixed patrol transition logic and ensured proper cycling through points.

2. **Enemy Attacking Too Early**

- Issue: Enemy was attacking before reaching the player.
- Solution: Adjusted attack trigger to activate only upon collision.

3. **Player Could Push the Enemy (Unwanted Behavior)**

- Issue: Player could push the enemy due to Rigidbody interaction.
- Solution: Increased enemy mass and adjusted collision behavior.

4**. Enemy Could Push the Player (Unintended Behavior)**

- Issue: When increasing enemy mass, the enemy began pushing the player.
- Solution: Dynamically adjusted mass and drag in the attack state to prevent movement issues.

5. **Attack Animation Visibility Issue**

- Issue: Using Animator Layers caused player sprite to disappear when attacking.
- Solution: Created a separate CombatVisual GameObject to handle attack animations independently.

6. **Double Hit Detection Bug**

- Issue: Multiple hit detections per attack due to overlapping colliders.
- Solution: Implemented HashSet tracking system to ensure each enemy registers only one hit per attack.

7. **Input Stacking Issue**

- Issue: Attack inputs were queued/stacked during cooldown.
- Solution: Added isProcessingAttack flag to prevent unintended queued attacks.

Time Spent & Estimations

- Task 3: Took the estimated 3.5 - 5 hours as planned.
- Task 4.1 (so far): Reached 75% completion in ~4 hours, maintaining efficiency despite debugging challenges.
- Managed to stay within estimated timeframes while handling unexpected obstacles (bugs, animation issues, AI refinements, etc.).

Next Steps

- Finish Task 4.1: Implement Health System & Enemy Abilities.
- Complete Bonus Features for Task 3: Implement Pathfinding System for enemy AI.
- Extra Task: Since 3 days remain, take initiative to implement Task 4.2 (Inventory System).

Final Thoughts

This phase of the trial has been incredibly rewarding. Throughout Task 3 and Task 4.1, I have significantly improved my ability to tackle problems efficiently while maintaining structured documentation and debugging workflows. The integration of AI behaviors and a robust combat system has been a fascinating challenge that allowed me to deepen my knowledge of Unity's animation, physics, and AI systems. I am proud of my efficiency, attention to detail, and ability to stay within estimated timeframes despite various challenges. Looking forward, I am excited to complete the health system, enemy abilities, and pathfinding while taking the initiative to implement an extra feature (Task 4.2: Inventory System). The process so far has been engaging and enjoyable, and I am very satisfied with my progress!

❖ **Written by:** Juan F. Gutierrez
February 21st, 2025.