UNITY DEVELOPER INTERNSHIP TRIAL - TASK 3, 4.1 & 4.2 REVIEW

-

Task 3: AI State Machine & Pathfinding | Task 4.1: Combat System & Health | Task 4.2: Inventory & Gameplay Mechanics

Overview

This document presents a comprehensive breakdown of the final major tasks of the Unity Developer Internship Trial at Runic Dices Entertainment.

- Task 3 focused on implementing a modular AI state machine, refining enemy behaviors, and integrating custom pathfinding.
- Task 4.1 introduced combat mechanics, including health systems, damage handling, animations, and visual effects.
- Task 4.2 expanded the game further with an inventory system, item usage mechanics, and additional environmental effects.

With the completion of these tasks, the project now features:

- A fully functional AI system capable of patrolling, chasing, attacking, and dynamically avoiding obstacles.
- A robust combat system with melee and ranged attacks, hit detection, knockback mechanics, and feedback animations.
- A health system with UI elements, health bars, and death state handling.
- A fully interactive inventory system, allowing item collection, usage, and stack management.
- A polished gameplay experience with audio, visual effects, and environmental mechanics like movement-affecting terrain.

This final report details the technical implementation, problem-solving approaches, debugging challenges, and overall development experience.

Timeline and Initial Steps

- Finalized Enemy AI State Machine behaviors.
- Integrated advanced enemy pathfinding system.
- Refactored movement logic to ensure smooth transitions.
- Fixed enemy patrol issues and state transition inconsistencies.
- Completed all AI enhancements and Task 3 bonus objectives.
- Moved into Task 4.1 - Combat System & Health.
- Integrated health bars, UI feedback, and damage logic.
- Implemented combat hit effects and refined animations.
- Implemented Task 4.2 - Inventory System.
- Added collectible items, stackable inventory slots, and health potions.
- Integrated gameplay enhancements like movement-affecting terrain (grass zones).
- Polished audio, visual feedback, and environmental effects.

Day's Progress & Key Milestones

1. *AI State Machine & Pathfinding System (Task 3 - Completed)*

- Enhanced AI Movement & State System:
    - Implemented enemy patrol, chase, attack, idle, and return states.
    - Fixed pathfinding logic to prevent enemies from getting stuck.
    - Improved enemy reaction to obstacles & player movement.
- Custom Pathfinding Implementation:

- Developed multi-ray obstacle detection system.
- Implemented smooth movement interpolation for realistic enemy tracking.
- Added debugging visualization tools to monitor AI behavior.

- Advanced Movement Refinements:
  - Added fallback behaviors to prevent stuck AI.
  - Adjusted AI rotation & directional movement calculations.
  - Implemented enemy self-avoidance to prevent AI from blocking itself.

2. Combat System & Health UI (Task 4.1 - Completed)

- Player Combat Mechanics:
  - Implemented attack animations & blend trees.
  - Created knockback system & hit detection logic.
  - Separated motion and combat animation layers for smooth visual transitions.
  - Added input buffering to prevent attack spam.
- Enemy Combat AI:
  - •Implemented AI attack cooldown system.
  - Refined enemy targeting logic.
  - Fixed player/enemy pushback issue by dynamically adjusting Rigidbody2D mass.
- Health System & UI:
  - Created a modular health system applicable to both player and enemies.
  - Implemented visual health bars with smooth UI updates.
  - Added animated health bar fading effect for polish.

3. Inventory System & Game Mechanics Enhancements (Task 4.2 - Completed)

- Inventory System:
    - Implemented item collection, stack management, and UI display.
    - Integrated health potions with real-time usage effects.
    - Refactored item data management for efficiency.
- Projectile System & Ranged Combat:
    - Introduced player projectiles (arrows, magic bolts, etc.).
    - Implemented object pooling for optimized projectile handling.
    - Added impact effects & hit registration for enemies.
- Environmental Effects & Movement Zones:
    - Created grass zones that slow player movement.
    - Adjusted animation speeds dynamically based on terrain type.
    - Ensured seamless transition between different movement states.
- Audio & Visual Enhancements:
    - Integrated custom combat sound effects.
    - Added hit sound variations for impact feedback.
    - Implemented enemy dodge ability with visual indicators.

Approach & Execution (Chronological Order)

1. AI & Pathfinding Enhancements

- Modular AI System Design:
    - Structured the Enemy AI system using a state machine approach, ensuring that transitions between states were smooth and logical.

    - Implemented five core states, Idle, Patrol, Chase, Attack, and Return-to-Patrol, ensuring a scalable and reusable system.

- Initial Pathfinding System Implementation:
    - Designed a custom pathfinding system that relied on multiple raycasts to detect obstacles and determine alternative movement directions.
    - Implemented a vector-based scoring system to prioritize the best movement paths while avoiding collisions with obstacles.
    - Integrated debugging tools using Gizmos, which allowed visualization of movement vectors, raycasts, and potential pathfinding errors.

- Pathfinding System Refinement & Debugging:
    - After testing, several movement inconsistencies emerged, including:
        - The enemy getting stuck around patrol points.
        - Sudden direction changes causing erratic movement.
        - Pathfinding not dynamically adapting to new obstacles.
    - Iteratively refined the obstacle detection and movement interpolation, implementing a fallback system that recalculated movement paths when an enemy became stuck.
    - Adjusted arrival logic at patrol points, ensuring that enemies wouldn't endlessly loop around them.

- Finalized AI System with Full Pathfinding Implementation:
    - Successfully merged obstacle avoidance logic with pursuit logic, balancing enemy movement between optimal paths and direct player tracking.
    - Ensured state machine integrity, meaning enemies correctly switched between behaviors without breaking animation flow or movement smoothness.
    - Optimized performance by implementing interval-based path recalculations instead of recalculating every frame.

- Enhanced Enemy Interaction System
    - Multi-Enemy Pathfinding:
        - Implemented layer-based detection system with self-collision avoidance
        - Created combined layer mask system for efficient obstacle and enemy detection
        - Modified Physics2D.RaycastAll for comprehensive environment scanning
    - Movement Optimizations:
        - Implemented closest valid hit detection and collision filtering
        - Added dynamic fallback behaviors to prevent AI from getting stuck
        - Enhanced group behavior while maintaining individual pathfinding integrity
    - Performance Optimizations:
        - Optimized movement calculations with interval-based updates.
        - Implemented efficient collision detection hierarchies.
        - Added movement interpolation for smooth transitions.
- Audio System Implementation
    - **Scene-Aware Audio Architecture:**
        - Developed adaptive audio system that responds to scene context.
        - Implemented graceful fallbacks for test scenes.
        - Created scene detection system to enable/disable audio features appropriately.
    - **Sound Design Collaboration:**
        - Integrated custom sound effects created by sound designer.
        - Implemented variety in combat feedback through multiple hurt sounds.
        - Added weapon effects for enhanced combat feel.
    - **Technical Implementation:**
        - Single AudioSource management for clean transitions.

        - Automatic sound replacement system.
        - Scene-specific audio validation.

- Visual Enhancement Systems
    - **Combat Feedback:**
        - Implemented sprite flickering system for damage indication.
        - Created healing visual effect with color transitions.
        - Added dodge ability visual indicators.

    - **UI Polish:**
        - Smooth health bar transitions.
        - Fade effects for non-critical UI elements.
        - Dynamic cooldown indicators.

2. Combat System Implementation

- Melee Combat System Design & Animation Layers:
    - Designed a separate animation layer for attacks to ensure combat animations played independently from movement animations.
    - Experimented with Animator Override Layers, but ultimately created a CombatVisual child object with a separate Animator Controller to handle attack animations smoothly.

- Attack Hit Detection & Input Handling:
    - Implemented precise collision detection for melee attacks, ensuring that enemies only took damage on direct collision with the player's attack hitbox.
    - Added input buffering to prevent attack input stacking, ensuring only one attack plays at a time.
    - Implemented knockback mechanics for enemies when hit, including force scaling based on attack strength.

- Enemy Attack System & AI Combat Behavior:

- Designed the Enemy Attack State to function on a cooldown system to prevent spam attacks.
- Implemented state-specific physics modifications, where enemies dynamically adjust their Rigidbody2D mass during attacks, preventing unwanted knockback effects.
- Ensured attack animation synchronization, meaning that attacks would only deal damage once the attack animation was visually aligned with the hit frame.

- Health System & UI Integration:
  - Implemented a modular Health System component that worked for both player and enemies.
  - Designed a smooth, responsive health bar system, using fade-in/fade-out effects to avoid cluttering the UI unnecessarily.
  - Integrated health bars with the AI system, ensuring enemies visually reflect their damage state via sprite flickering and UI updates.

3. Inventory & Gameplay Mechanics

- Inventory System Core Implementation:
  - Created a fully functional inventory system, allowing players to collect, store, and use items.
  - Designed a UI-based inventory with dynamic slot allocation, ensuring collected items stacked correctly instead of occupying duplicate slots.
  - Implemented an input-driven selection system, allowing the player to navigate the inventory using both keyboard and gamepad controls.

- Item Collection & Usage Mechanics:
  - Designed item collection logic, ensuring only one item is picked up at a time, preventing stacking bugs.
  - Implemented health potions as the first usable item, allowing the player to restore health through the inventory system.

    - Integrated real-time UI updates, ensuring that item consumption immediately updates both the UI and the health system.

- Projectile Combat System:
    - Designed a player projectile system, allowing ranged attacks in addition to melee combat.
    - Implemented object pooling for projectiles, ensuring optimized memory usage and efficient instantiation.
    - Designed projectile behavior logic, including directional movement, collision detection, and impact effects.

- Enemy Dodge Ability & AI Enhancements:
    - Implemented enemy dodge mechanics, allowing AI to detect incoming projectiles and evade accordingly.
    - Designed a cooldown-based dodge system, ensuring that enemies don't spam dodge repeatedly.
    - Implemented dodge animations and UI indicators, providing visual feedback on dodge cooldowns.

- Environmental Effects & Movement Modifiers:
    - Created grass zones that slow down player movement, adding environmental variety to traversal mechanics.
    - Adjusted player animation speeds dynamically to match movement changes in different zones.
    - Integrated seamless transitions between different movement states, ensuring smooth player control.

Challenges & Reflections

1. Enemy Pathfinding Stability Issues

- Challenge:

After implementing the pathfinding system, the enemy movement became unpredictable in patrol and return-to-patrol states. The AI would sometimes get stuck circling patrol points, failing to transition properly.

- Solution:

- Increased arrival threshold logic, ensuring that the AI stopped movement smoothly upon reaching a destination.
- Implemented a state flag system to separate movement transitions more clearly, preventing state confusion.
- Debugged enemy AI navigation using Gizmos visualization, allowing me to identify inconsistencies in movement calculations.

2. Combat Animation Handling & Attack Buffering

- Challenge:

Originally, the attack animations overlapped with movement animations, causing erratic sprite behavior. Additionally, attack inputs could stack, resulting in multiple attack actions triggering at once.

- Solution:

- Separated combat animations into a dedicated CombatVisual object, ensuring smooth attack playback without overriding movement animations.
- Implemented an attack buffering system, restricting attacks until the current animation completes.

3. Item Collection & Inventory System Refactoring

- Challenge:

The inventory system initially created duplicate slots for the same item, instead of stacking them properly. Additionally, item usage did not properly update UI elements in real-time.

- Solution:

- Separated combat animations into a dedicated CombatVisual object, ensuring smooth attack playback without overriding movement animations.
- Implemented an attack buffering system, restricting attacks until the current animation completes.

Time Spent & Estimations

- Task 3 (AI & Pathfinding): Took longer than expected due to pathfinding debugging, requiring multiple refinements before completion.
- Task 4.1 (Combat System & Health): Completed within 5 to 7 hours, aligning with original estimates.
- Task 4.2 (Inventory & Gameplay Enhancements): Took between 4.5 to 6 hours, with most of the time spent on UI navigation & inventory refactoring.

Additional Note:

Over the weekend, I was away without internet access but continued working on the project. This explains the lack of commits and communication with RDE staff over Saturday & Sunday. Upon returning, I pushed the final commit on Monday, completing the trial within the 5-day timeframe.

Technical Analysis & Future Considerations

As this project involved complex systems and multiple interconnected components, a detailed technical analysis is valuable for understanding the challenges faced, solutions implemented, and potential future improvements. This analysis provides insights into the development process, architectural decisions, and opportunities for optimization.

Time Management Insights

- **Task 3 Timeline Analysis:**
    - Initial pathfinding implementation: 2-3 hours
    - Debugging and refinement cycles: 4-5 hours
    - Integration with other systems: 2-3 hours.
    - Major challenges that extended timeline:
        - Complex enemy interaction behaviors.
        - State transition edge cases.
        - Performance optimization requirements.

Architecture Review

- **Current System Integration:**
    - State machine implementation provides good behavior separation.
    - Health and combat systems show some coupling that could be improved.
    - Event-based communication could enhance system independence.
- **Potential Improvements:**
    - Interface-based abstraction for combat interactions.
    - Centralized event system for cross-component communication.
    - Enhanced data management for inventory system.

Edge Case Management

- **Systematic Debug Approach:**
    - Implemented comprehensive logging system.

    - Created visual debugging tools for pathfinding.
    - Established reproducible test scenarios.

- **Key Resolved Issues:**
    - Item stacking synchronization.
    - Collision detection edge cases.
    - Animation state conflicts.
    - Multiple enemy interaction scenarios.

Future Optimization Opportunities

- **Performance Enhancements:**
    - Implement spatial partitioning for enemy pathfinding.
    - Optimize UI updates for inventory system.
    - Add object pooling for visual effects.

- **Feature Expansions:**
    - Enhanced inventory categorization system.
    - More sophisticated enemy behavior patterns.
    - Advanced pathfinding algorithms.

- **Polish Improvements:**
    - Additional visual feedback systems.
    - Enhanced sound effect variety.
    - More dynamic camera behaviors.

Final Thoughts

Completing this trial has been a deeply fulfilling experience. It has challenged my ability to problem-solve complex AI behaviors, refine gameplay mechanics, and implement scalable game systems.

- This process strengthened my skills in:

  ✓ AI Development & Pathfinding

  ✓ Combat System Implementation & Animation Handling

  ✓ UI & Inventory Management

  ✓ Debugging, Testing, and Optimization

I am extremely proud of the structured approach I took, my ability to adapt to unexpected challenges, and my efficiency in delivering a complete, polished prototype.

❖ **Written by:** Juan F. Gutierrez
February 23rd, 2025.