# Final Project Summary: "HEMA House"

## Final Changes Made

1. Guard Transitions Implemented: Added sprite sheet transitions between guard positions (Mid to High, Mid to Low) with smooth 6-frame animations. Reverse transitions reset the guard back to Mid after each point.

2. Enemy AI Guard Control: AI now randomly changes guard states and mimics player behavior with optional cutting after guard change. Adjusted guard alignment logic was implemented to mirror player actions.

3. Pause Functionality: Pressing P pauses and unpauses the game, halting updates without breaking state.

4. Game Reset Behavior: After each point, both fencers reset to their starting positions and guard to Mid state, avoiding lingering animations or improper states.

5. Collision Feedback: Added tap sounds when fencers are near each other but not cutting, providing more responsive feedback for footwork interactions.

6. Input Debouncing on Menu Selection: Fixed issue where spacebar input would leak into the game state, causing immediate cuts when starting a match.

7. Visual & Audio Polish: Updated idle frames, ensured proper sound triggers on cuts, blocks, and tap proximity. Enhanced visual consistency with proper outlines on menu text.

## Challenges Faced & Solutions

- Animation Timing Conflicts: Ensuring guard transitions didn't conflict with movement or attack states was complex. This was solved by introducing an is_transitioning flag and managing animation priorities explicitly in update_animation().

- AI Guard Adaptation: The AI's ability to react to player guard changes required careful timing to avoid predictable behavior. Random guard changes were balanced with targeted adjustments toward the player's current guard.

- Game Loop Input Persistence: The spacebar event inadvertently carried over from menu selection into gameplay. This was resolved by explicitly clearing the KEYDOWN events when transitioning from menu to game.

- State Reset after Points: Guaranteeing that the fencers reset cleanly after a point was tricky due to overlapping states. A dedicated reset_positions() function in GameManager ensured all fencer attributes were restored reliably.
- Pause Mechanics Without Breaking Flow: Introducing a pause state without interfering with ongoing animations required conditional checks within the main loop and update calls, ensuring animations halt while retaining current state.

## Reflections on the Development Process

This project required careful coordination of multiple systems-animation states, AI logic, collision detection, and UI flow. Managing state machines for the fencers and ensuring responsive, smooth gameplay was a rewarding challenge. Integrating feedback (visual, auditory, and tactile through AI reactions) greatly enhanced player immersion.

The iterative debugging of animation blending and AI behavior highlighted the importance of modular, well-encapsulated code. Each class (Fencer, AIController, GameManager, MenuSystem) evolved into specialized components with clear responsibilities, making it easier to add features and fix bugs.

Overall, the project provided valuable experience in handling game loops, sprite animation sequencing, and AI behavior in a 2D fighting game context.