The Design Process behind the Guard AI

When I received the task, I started opening my drawing board and plan out each feature needed to be in the ai system. The system should be modular as required and should not be too dependent on each other too much. I categorize each feature as follows;

Feature Categorization:

Controller: Serves as the central hub for all essential data and components required for both basic AI functionality and decision-making processes.

View Detection: Serves as detection for guard Ai and Player.

View Detection Graphic: Provides a visual representation of the AI's field of view detection.

Patrol Feature: Moves from one point to another in a loop.

Chase Feature: Chase Player upon detecting Player in view.

Investigate Sound Feature: Investigate sound emitted by Player.

Navigation System: Initially, A* pathfinding was considered for the navigation system. However, after carefully reviewing the project requirements, it became evident that Unity's Navmesh system would be better suited to meet my needs.

Controller System: Various controller systems were evaluated such as having one update method in controller, ultimately leading to the decision to incorporate Unity Events for decision-making. This choice allows collaborators such as Artists or Sound Designers to seamlessly integrate their logic during the development process, thereby enhancing system cohesion and adaptability.

Development Journey:

View Detection System: Initially, I considered using a sphere collider for detection. However, this approach proved ineffective when obstacles obstructed the guard's view. Subsequently, I implemented a solution using Physics.OverlapSphere to detect the guard's surroundings, including obstacles and players within its field of view.

Patrol System: A basic patrol system was developed to enable the guard to move between waypoints and loop back to the initial waypoint upon reaching the last one.

Idle System: To enhance the guard's behavior, an idle system was implemented. Basic start and stop idle times were introduced, with a countdown mechanism controlled by the controller.

Integration: Idle logic was integrated into the patrol system, along with start and stop patrol functionality, culminating in the completion of these systems within a two-day timeframe.

Chase System: On the third day, focus shifted to implementing a chase system. The guard's behavior was augmented to pursue players within its field of view, adding a natural dynamic to its actions.

Sound Detection: Towards the end of the third day, a sound detection feature was added. Similar to player detection logic, this feature detects sounds within the guard's radius, providing alerts when players emit noises such as running.

Investigation Decision Making: On the fourth day, the basic framework for investigation decision-making was implemented, alongside the addition of visual detection for guard logic. Challenges arose during this phase, particularly in unfamiliar territory such as visual representation. However, after exploring various approaches, a solution involving mesh-generated triangles was successfully implemented.

Refinement: The fifth day was dedicated to code refinement and optimization, eliminating unnecessary lines, and ensuring smooth functionality.

Conclusion: The Guard AI development process involved careful consideration of modularity, system integration, and iterative refinement. Through the collaborative efforts of the development team, a robust and adaptable AI system was realized, capable of seamlessly integrating additional behavior modules to meet evolving project requirements.