Spike Extension Report
Marco Giacoppo 104071453
COS30002 - AI for Games

Introduction
This report presents the spike extension work completed during the COS30002 - AI for Games unit. The purpose of this document is to demonstrate how each spike aligns with the unit's Intended Learning Outcomes (ILOs), with a brief justification and explanation for each.

ILO's
ILO1: Discuss and implement software development techniques to support the creation of AI behaviour in games.

ILO2: Understand and utilise a variety of graph and path-planning techniques.

ILO3: Design and create realistic movements for agents using steering force models to simulate natural and responsive behaviour.

ILO4: Design and develop agents capable of planning actions that require adaptive problem-solving techniques and innovative solutions.

ILO5: Combine multiple AI techniques to develop sophisticated game AI capable of solving intricate, multi-layered problems that arise in dynamic game environments.

Spike Extension Alignment:
Include a list or table of the extensions, and indicate which ILO's the work align with. If possible, see if you can indicate how much they support the ILO's. Here is a basic example:

Spike Extension	ILO 1	ILO 2	ILO 3	ILO 4	ILO5
Task 3 – AI Battle	X			X	X
Task 4 – Perfect AI Battles				X	X
Task 6 – Death-based Path Cost	X	X		X	
Task 7 – OO Combat + RPG	X			X	X
Task 8 – GOAP Survival Sim	X	X		X	X
Task 9 – PlanetWars Tactical AI	X	X		X	X
Task 16 – Enemy FSM + Resupply Mechanism	X		X	X	X

Justification for Each Extension:

Task 3 – AI Bot Battle (Week 2)

This task involved designing two AI agents that battle based on predefined logic. The simulation did not require graphics but focused on internal decision-making and damage resolution. It demonstrates:

ILO1: Procedural code for AI simulation and output.

ILO4: Agents assess and attack each other dynamically.

ILO5: Potential to run multiple battles and evolve tactics.

The extension codes related will be in *main.py* with comments on the relevant codes.

Task 4 – Perfect AI Battles (Week 3)

This task contains an AI agent that always plays the perfect game. A 4x4 board can also be played on, it just takes a long time to provide the output.

ILO4: A perfect-playing AI agent reflects advanced planning and problem-solving.

ILO5: Handles both standard and 4x4 boards suggests scaling algorithms to increased complexity.

The extension codes related will be in *main.py* with comments on the relevant codes.

Task 6 – Death-Based Pathfinding (Week 4)

Here, agents that collide are marked as dead, and tiles where many agents die are avoided by others. This advanced the system via death-tracking:

ILO1: Update to agent simulation and tracking structure.

ILO2: Integration of modified tile costs into A.*

ILO4: Agents change paths to avoid lethal terrain.

The extension codes related will be in *agent.py, game.py, and searches.py* with comments on the relevant codes.

Task 7 – OO Console Combat + RPG (Week 5)

Created a structured, object-oriented battle simulation between two NPCs. Added both a planning system and combat resolution:

ILO1: Clear OOP design with classes for NPC, Action, and Goal.

ILO4: Agents choose actions based on changing internal state.

ILO5: Combines action planning and damage tracking.

The extension codes related will be in both of the files with comments on the relevant codes.

Task 8 – ASCII GOAP Survival System (Week 5)

A fully interactive text-based agent that plans actions (gather, drink, cook) using GOAP and pathfinding:

ILO1: Complex system integration.

ILO2: Uses BFS to move agent across terrain.

ILO4: Predictive planning (e.g., cook before hunger).

ILO5: Merges GOAP, resources, inventory, and world map.

The extension codes related will be in *main.py with* comments on the relevant codes.

Task 9 – PlanetWars Tactical AI (Week 6)

Developing a more strategic AI agent in the PlanetWars environment by:

Avoiding far-distance attacks, sending scouts, choosing weak targets, using simulation-based planning.

While the current agent assumes full visibility, it has been designed with fog-of-war awareness in mind. For example, the bot incorporates conservative decision logic, avoids overcommitting forces, and could be extended with memory or probabilistic models to handle unknown or outdated data. These features support tactical decision-making even under incomplete information.

Supports:

ILO1 & ILO2: Uses planetary graphs and decision layers.

ILO4: Chooses attack strategies based on game state.

ILO5: Combines tactical heuristics, goal-switching, and adaptive fleet management.

Task 16 – Enemy FSM + Soldier Resupply Mechanism (Week 11)
Extension codes will all be in *agent.py* with comments.
This extension added a layered AI system to both enemies and the soldier agent.

- ***Enemy FSM (Finite State Machine):***
  Implemented two high-level states (PATROL and EVADE), enemies switch to EVADE after being hit and flee at triple speed.
- ***Soldier Resupply Logic:***
  A new RESUPPLY state was added to the soldier's FSM. After reloading twice, the soldier must move to a predefined resupply zone to resupply on ammunition.

ILO1: New software architecture for state management and behavior switching.

ILO3: Realistic responsive behaviors using steering + state transitions.

ILO4: Adaptive planning (e.g., enemies fleeing, soldier needing to resupply).

ILO5: Combining multiple AI models (FSM + steering + cooldowns + predictive targeting).

Conclusion

Across the tasks, I have applied a range of AI techniques that align directly with the unit's learning outcomes. From simple decision-making bots and FSMs to GOAP-driven simulations, and tactical planning in games like PlanetWars, each spike task helped deepen my understanding of how AI systems are designed, integrated, and applied in game environments. These extensions reflect a progressive mastery of both foundational and advanced concepts in game AI..