AI Navigation in Unity with NavMesh - Assignment 1: Simple and Dynamic Movement

Deadline (Soft): Friday July 21st, 7 pm complete tasks 1 - 6.

- Research documentation and YouTube tutorials about the NavMesh library
- Note: Agents can simply be game objects holding the 3D capsule shape.
- [Important] Install GitBash and be able to commit to the shared repository.

Tasks to accomplish:

1. Create a generic plane in which the agent may navigate on, and have the agent travel to an appointed destination. The destination will be assigned by clicking an area on the generic plane with the mouse.
2. Have at least two agents on the same plane to navigate to an appointed destination to verify multi-threading. Research documentation is also an applicable approach.
3. Implement collision avoidance to the agents, both from obstacles and each other.
4. Create a generic plane with baked mesh areas of varying cost to pathfinding. Have the agents navigate through the disorderly plane to reach an appointed destination via mouse-click.
5. Implement dynamic baking to the generic plane of varying cost in which during run time, the plane dynamically rebakes its meshes to have different cost at static intervals. This will test the agents to path find their way to the appointed destination while processing a dynamic environment. This will also showcase the processing power of NavMesh via multithreading.
6. Implement a playable character in which the agents are to follow, hence the player's location is the final destination.
7. Combine the skills learned with NavMesh into the ReGoap architecture, so that agents are navigating through a finite state machine (FSM) that depends on automated planning, in order to chase down the playable character.
8. Implement sensors to the agents. The first sensor to implement is vision, so that agents will not chase the player, unless caught within line of sight. The sensor data will be updated to agent memory, since sensors are just helper methods for memory to help the agent understand their current world state. Agents will remain in idle state in the FSM and only trigger chase if the sensors update the memory that player is detected, thus certain goals will have higher priorities and, or become valid, also the preconditions of certain actions may be fulfilled.

Objective (Shared through Github):

Design a generic level that implements tasks 7 and 8 and commit via GitHub via Git Bash.