Task: Task 21

Title: Custom Project

Author: Sam Huffer, 101633177

# Goals / Deliverables

* A customizable box world
  + Corridors formed from wall tiles
  + Patrolled by a squad of soldier agents
    - Using group behaviours adapted for graph-based navigation.
    - Attack fugitives on sight.
    - If fugitives flee beyond soldiers’ awareness, they scout the immediate area, resume attacking if they find them, resume patrolling if they don’t.
    - If overwhelmed by fugitives, one soldier returns to base to lead reinforcements up to max squad size back to the rest of the soldiers.
    - Squad respawns if all soldiers die.
  + Populated with fugitive agents
    - Stay hidden until they see the soldiers, then attack.
    - On attack, fear increases. If it gets too high, fugitives flee.
    - Respawn at random points on map when they die.
  + All agents flee explosives they’re in range of until they finish detonating.

# Technologies, Tools, and Resources Used

* SublimeText (for editing, executing and testing the code)
* Learning materials on Canvas (for instructions and sample code)

# Tasks Undertaken

* I started by copying the project from Task 18: Navigation with Graphs and my custom project plan into the Task 20: Custom Project folder, stripping the spike report down to what was needed for the custom project.
* Next, I devised a larger map for the agents to wander around in, making a blank 30 x 30 map text file, designing it in the simulation itself, and copying it over to the text file. To adapt the mobile classes to it, I reduced the size of the agents, made the projectiles split their movement into several steps to ensure they don’t jump over agents or walls they’re supposed to hit, modified the speed of projectiles so that they don’t move too fast but not so slow that the agents can overtake them and set them off, and reduced the inaccuracy margin of the handgun, hand grenade, and shotgun.
* I modified the soldiers’ code so that while patrolling, the lead soldier would decide where the soldiers were going, and that the rest would calculate their own paths to the same location, recalculating if the lead soldier’s target changed. However, I noticed that occasionally the soldiers would move “backwards” then continue along their path when they recalculated their path to the lead soldier’s target, so I also tweaked agent.plan\_path() so that when a path was calculated, if the distance between the soldier and the second waypoint was less than the distance between the first and second waypoints, the first would be discarded and the soldier would just head to the second.
* I changed the read-from-file code to read the patrol waypoint box numbers from the map file and store the corresponding boxes in a list of waypoints in the box world class. While doing so, I got sick of the low framerate, I changed box.draw() to not fill the circle rendered for walls, but just do a thick outline, saving iterations of circle outline drawing and increasing the framerate.
* I altered the agent setup code so that the soldier leader would have its target set at the start and plan a path to the first waypoint, prompting the rest of the soldiers to follow it. Then I reorganised the various conditions for planning a new path into one method that could manage target selection and path planning for all of its preceding code, and replaced them with a call to that method. This way, the target selection and call of self.plan\_path() is all in one location and can be more easily modified as needed.
* Ran into errors where target\_enemy was None when it shouldn’t have been, put it down to misalignment between how look() calculated if the agent could see an enemy, and how update\_soldier() got the closest enemy. While testing, also removed explosive weapons from the list of weapons, as explosives weren’t working and just kept detonating behind the soldier.
* Noticed after attacking, soldiers would sometimes patrol in the reverse direction, so created a directional map for soldiers to follow. However, realised this would require the customizability of the simulation to be disabled or for functionality to allow the changing of a tile’s preferred direction for soldier agents, so decided against this. Instead decided to use a similar technique to how the agent bypasses redundant nodes on the path, using a path-length calculation method similar to what’s used for the aiming and shooting predictive calculations. However, this ran into the issue of the soldier agent somehow bouncing back and forth between the first two nodes.
* Solution: waypoints aren’t single boxes, but a line of them such that when the lead soldier moves backwards along the patrol path while attacking, if it triggers the boxes, it decrements its waypoint so that when it finishes attacking, it knows which one is the next one along the patrol path. However, to retain customizability of the box world, also requires keyboard controls for customizing waypoints. First changed code so that the box world reads in all boxes for a waypoint, then added a waypoint class that lists all boxes within the waypoint. Then added drawing code for displaying waypoints, inputs for editing waypoints, and a method to handle adding and removing boxes from a waypoint.
* Added methods for switching between waypoints both when the soldiers are patrolling, and when they are not, incrementing and decrementing the index of the current and last waypoints as appropriate. Spotted weird errors with the waypoint editing code in that on every number, clicking a box to add it to a waypoint would attempt to add or remove it from the correct box, but display 9 as its waypoint, and waypoints read from file would be unable to be removed. Fixed this by reworking BoxWorld.edit\_waypoint\_node() to perform checked based on the waypoint number of the node passed to it, and BoxWorld.active\_waypoint rather than searching through waypoints to check if the node was present in their lists of nodes or not.

# Instructions for Operating the Code

* Space: alternate between placing blocks and modifying waypoints
* Placing blocks:
  + Left mouse click: change a box’s kind to the currently selected kind.
  + 1: mouse clicks now clear blocks.
  + 2: mouse clicks now place mud.
  + 3: mouse clicks now place water.
  + 4: mouse clicks now place walls.
* Modifying waypoints:
  + [0-9]: select a waypoint
  + Left mouse click: toggle whether a box is in the currently selected waypoint. Cannot add a box to a waypoint if it belongs to another waypoint.
* Search parameters:
  + D: scroll through the list of ways of calculating diagonals.
  + M: scroll forward through the list of search algorithms.
  + N: scroll backwards through the list of search algorithms.
* Display options:
  + B: alternate thickness of box lines.
  + C: toggles markers of the centre of boxes.
  + E: toggles displaying of movement network edges.
  + L: toggles box labels.
  + O: toggles highlighting of agents’ optimal paths in red.
  + T: toggles display of graph options that were considered but did not end up being the optimal path.
  + U: toggles circle markers of boxes considered during the search.
* P: (un)pause the simulation
* ~~R: randomise the position of the agents and targets.~~

# State Diagrams

# UML Class Diagram

# Code Snippets

# In-Simulation Screenshots

# Retrospective

* What I found out / what I now know
  + Grid-based directional map > wouldn’t be as easy to do as changing box types would mean the directions would need to be customizable as well.
  + Single nodes for a waypoint aren’t enough when agents can move backwards along the patrol path to attack. Need to be enough that it acts like a trigger collider.
* what I’d do different
* what I’d change
* further changes I’d suggest