Spike: Task 18

Title: Navigation with Graphs

Author: Sam Huffer, 101633177

# Goals / Deliverables

* Create a navigation graph simulation demonstrating:
  + A game world divided into tiles, and that supports a navigation graph structure.
  + Agent path planning based on current environment, using cost-based heuristics.
  + Four or more agents of at least two types that can follow their own independent paths.

# 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 17: Graphs, Paths and Search and the agent, weapon and projectile classes and the spike report from Task 16: Goal-Oriented Action Planning into the Task 18: Navigation with Graphs folder, stripping the spike report down to what was needed for this task.
* Box world rendering a stationary agent.
* Start no longer gets a path, agent calculates path to target. R key randomises agent position. On resize, agent stays in its current box.
* Agent moves to next node in path, checks which box it is in and updates its box field. Rate of movement is scaled along each axis by the ratio between the original width and height of the window and the current width and height of the window.
* Multiple agents moving to separate targets, R key now randomises agent positions and targets. Removed start. BUG: sometimes path nodes seem to get strapped to agents and the path goes all over the place.
* All agents wandering. AStar is the default search method. Fixed bug where couldn’t switch path lines off.
* Fixed the bug where sometimes path nodes seem to get strapped to agents. Didn’t figure out what was causing it (apparently a couple of methods in agent that were just accessing the boxes’ positions to read them, that can’t have been it since everything is taking copies and reading those), but did implement an effective countermeasure. Box now has \_vc and position recording its position. If something needs to access the position, it accesses \_vc. If \_vc != position, \_vc is overridden with a copy of position. Also tweaked the code that handles repositioning everything on resize so that it handles the agent’s current node position that it’s going to as well.
* Changed wall blocks to render as circles, make it look like agents aren’t intersecting with them when moving diagonally. Scaling radius of weapons’ effective ranges and agents’ radii and avoidance radii according to screen size. Reworking fov to render properly in the new context.
* Abandoned fov as it was, since it was no longer working adequately, now using a circle centred 65% of its radius in front of the soldier. Anything within that circle is within the soldier’s awareness. Soldier now pursues the closest fugitive it is aware of, wandering if it can’t see any. If pursuing, follows path to the fugitive. If the fugitive moves into a new box, recalculates the path. If the fugitive is in the same box as the soldier, the soldier moves directly towards it.
* Modified aiming to use the target enemy’s path rather than their current velocity, velocity now no longer existing.

# Instructions for Operating the Code

* Left mouse click: place or clear a block, or move the start or target positions.
* 1: mouse clicks now clear blocks.
* 2: mouse clicks now place mud.
* 3: mouse clicks now place water.
* 4: mouse clicks now place walls.
* 5: mouse clicks now move the start position.
* 6: mouse clicks now move the target position.
* M: scroll forward through the list of search algorithms.
* N: scroll backwards through the list of search algorithms.
* Space: plan a path using the current search mode.
* 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.
* P: (un)pause the simulation
* R: randomise the position of the target(s).
* 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.

# Code Snippets

# In-Simulation Screenshots

# What I Found Out

* When using boxes, either got to scale boxes to be 1:1 upon resizing the window and everything needs to fit in the window, or got to scale everything separately for x and y axes where possible, and fudge it as best you can when it’s something like a circle and you can’t scale x and y axes separately. The first is the ideal scenario, but things went weird when I tried scaling the boxes, so instead scaled everything else on the x and y axes, fudging it where necessary.
* Number of different considerations to account for when moving using a path rather than straight to a target, such as:
  + What to do when you’re in the same box as the target, but the target you’re pursuing is not at the same coordinates as the centre of the box.
  + When not using force-based, heading and side cannot derive from velocity, as velocity isn’t being used; gotta derive from the vector to the current target instead. Looks jerkier, could probably get it smoother, wasn’t strictly necessary for this task, so left it.