**Spike:** 08

**Title:** Goal-Oriented Action Planning (GOAP)

**Author:** Ben Holmes, 103024841

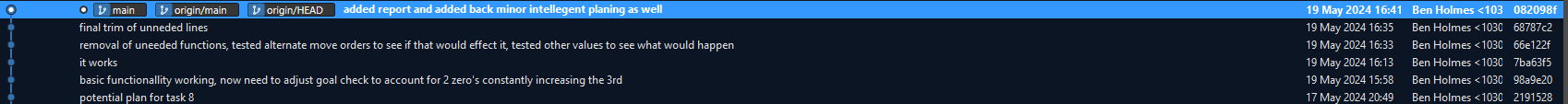
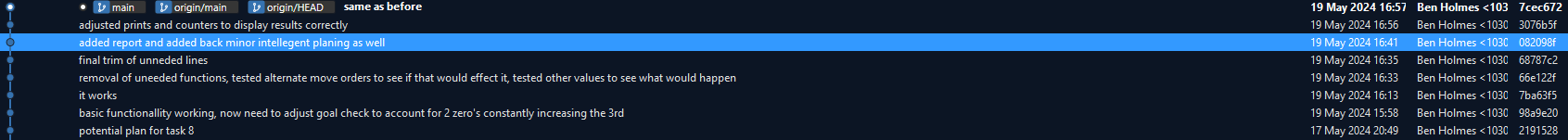
**Goals / deliverables:**

1. Create a GOAP simulation that demonstrates the effectiveness of the technique in considering long-term outcomes of actions (related to side-effects and/or time delays) and can plan and act intelligently.

**Technologies, Tools, and Resources used:**

* Visual Studio Code
* Python 3.12.2

**Tasks undertaken:**

****

* Copied gop from task 7
* Adjusted to use 3 values instead of 2 and renamed them to energy, hunger and fitness (fitness still wants to go to 0, its lack of fitness shortened to fitness for ease of typing)
* Created new recursive function for path planning called action\_paths, it loops through each possible action and applies it to a temp instance of the goals, in the loop this temp instance of the goals is then passed to another action\_paths which loops through the next moves (move number 2). This eventually generates a path (moves done and final goals of moves done) in a Path object that is then added to a list of path objects called paths which is returned. Action paths also has a temp\_paths which is looped through and appended to paths if another action\_paths is called

A computer screen shot of text

Description automatically generated

* Path\_apply\_action, is the same as the old apply action just to a provided set of goals (path\_goals

A screen shot of a computer

Description automatically generated

* Action\_paths is called in the chose\_action\_path function, which if a path has not been generated before, generates a paths list and then loops through it checking against the first paths goals to see if the path produces better results using the path\_check function. (it also compares length of path and choses the shortest path if one is shorter as there is a check in action\_paths that shortens the path if 0 0 0 is hit) The best path is stored in the global variable, the path moves are then iterated through till the move\_counter (number of path moves) hits the move\_amount (max number of path moves), once move\_amount is hit, the counter is reset and next time chose action\_paths is called, then the paths are regenerated.

A black screen with colorful text

Description automatically generated

A computer screen shot of a program

Description automatically generated

**Results:**

Path move number shows how many moves through each path the choice is through.

Moved number shows the overall move count

Path moves is the list of moves the path generates (shows the same for all 3 or more path moves done)

best action is action done on that move

goals is current overall goals

path goals is final goals after path is run

Depth 3 last 2 paths chosen:

**A screenshot of a computer program

Description automatically generated**

Depth 4 last 2 paths chosen:

**A screenshot of a computer program

Description automatically generated**

**A screenshot of a computer program

Description automatically generated**

**What we found out:**

For the deliverable, the simulation was created to have 6 actions with different increases and decreases and stability of the 3 goals so that the planning function had to account for side effects of its actions. The intelligent actions are determined in 2 ways, first is checking to see if the move count is less than the move depth (or current best path) then it choses that path. The next intelligent check is when it checks for the decrease in the largest goal after checking for all 3 values decreasing. This is to make each path more efficient. The second check improves efficiency at depth 3-4 but depth 5 its nearly identical.

If you wish to test the depth side of things, simply change the move depth number, although be warned at depth 6 and above it becomes much slower each increment.

If you wish to test the efficiency of the largest goal check please comment out these lines

