Team Name: D-Crawler

AJAY PAL | ROHAN GUJARATHI | KUSHAL SAMIR MEHTA | RISHABH SACHDEVA

Exploration Strategy:

We are going to consider multiple things while exploring the path which are explained below:

2. **Keeping maximum options open while moving to a tile**:
   Our agent will choose a path that will give him more options to move further. Even if it may cost higher to enter that tile. This will enable agent to avoid possible dead-end and eventually will allow it to avoid backtracking cost.
   Already Explored: If the path is already explored the agent will avoid moving to that tile. However, in some situation agent may move to that tile when necessary.
   Object type: If the object in one of the observed tiles is powerup then agent will move to that tile considering the final power level is positive. If the object in one of the observed tiles is monster, then agent will only move to that tile if he has strength of at least 70% of the monster's strength and it has explored significant portion of map. Eventually the agent must fight the boss, so agent will try to increase its power.
   Agent will avoid fighting monsters with significantly more power than him.
   If agent finds the boss early in the game and learns that its power is significantly less than the boss and still has lot to explore then it will keep track of the boss's location and will not fight at this point. Instead it will explore the map to gain more power using powerups and defeating weaker monsters.

3. We started formulizing our strategy by reading Tremaux's algorithm. We have implemented a variation this algorithm in the assignment and it felt intuitive to build up on this idea for the project too. We played several roguelike games. I.e, Dreamquest and Maze Dungeon. After playing them individually and we discussed our strategies with the team. This brought out different heuristic measures as each one of us prioritized differently. For example, one teammate chose to explore randomly and experienced death several times, whereas the other teammate chose to figure out what the edges of the map contains and backtracked a lot. The agent should be able to anticipate the possible moves of a dynamic monster and choose its next move accordingly.

4. As for our agent the path it takes to reach the goal doesn't matter. The agent can only perceive the environment partially, hence it would be ideal to use a local search strategy. This problem appears to be a good candidate for beam search than hill climbing. The agent maintains a database and continuously updates it as it learns something about the environment. Also, it queries the database to decide in future hence it's a knowledge-based agent.
   A potential technique is Q learning. This is a popular technique that is used to solve

simple games. It is a reward-based reinforcement learning method which finds the best action-selection policy. The function that must be maximized is called the value function. For each state, the next action can be obtained from the Q table. This table is continuously updated until the number of specified episodes are completed. After this, the agent can exploit the environment using this Q table.

5. We will use the following strategies for evaluation
    a. A trivial case that the agent needs to handle is to reach the goal state. This would mean that the agent is able to solve the problem and the algorithm is complete.

    b. Testing the agent with large Map to see if agent's performance is "satisficing". It will satisfice if the agent is able to achieve a defined level of optimality.

    c. Visualizing the explored portion of map. Lesser the explored portion, better the agent's performance

    d. Compare the path the agent takes with the expected one to measure the "error" or the "inaccuracy" of the agent. This could be the number of tiles it is off by from the ideal path it should be taking or its closeness to monsters / powerups.

    e. The agent will compete against humans and other agents.

    f. The agent is expected to behave according to its surroundings. For example, in the vicinity of a monster, it must avoid the monster and never take that path or traverse in the monster's direction. The opposite is true in case of a powerup.

    g. On playing a large number of games, we can evaluate the success rate by dividing the number of successful games over the total number of games played.

6. The functions that we plan to use are as follows (Note: These functions are subject to change (Additions / Deletions) through the implementation.)
    a. **Function Name:** step(location<tuple>, strength<integer>, map<matrix>, objects<dictionary>)
    Returns a direction (N, S, E, W) to move

    b. **Function Name:** find_observed(observed<dictionary>, objects<dictionary>, map<matrix>)
    Behavior: This will find the location of all the nearby tiles that can be observed from the current position
    returns a list of all the tiles that can be observed from the current

c. **Function Name:** choose_move(observed<list>, objects<dictionary>, map<matrix>)
   Behavior: This method will choose the best possible moves among all the moves currently available
   Returns next move

d. **Function Name:** calculate_heuristic(explored, map<matrix>, object)
   Behavior: This method will calculate the heuristic cost of the observed tiles
   Returns heuristic of all the observed tiles

e. **Function Name:** goal_test(location<tuple>)
   Behavior: Checks whether the goal is found or not
   Returns true or false

References

1. How to teach an AI to play Games: Deep Reinforcement Learning
   https://towardsdatascience.com/how-to-teach-an-ai-to-play-games-deep-reinforcement-learning-28f9b920440a
2. Deep Q-Learning with Keras and Gym · Keon's Blog
   https://keon.io/deep-q-learning/
3. Deep Reinforcement Learning & Rogue
   http://www.cs.unibo.it/~asperti/rogue.html
4. Artificial Intelligence: A modern approach 3rd Edition
   Stuart Russel, Peter Norvig
   Sections 4.1.1, 4.1.3