ID2209 – Distributed Artificial Intelligence and Intelligent Agents

Assignment 3 – Coordination & Utility

Group 36
Matay Mayrany

27.11.2019

N Queens and Stage Matching problems

In this assignment, we were given 2 tasks. One to simulate the N queens problem using the GAMA platform and queen agents. The other is to simulate guest agent choosing stage agents to go to, based on their preferences and how they match the attributes of the given stage agent.

How to run

Run GAMA 1.7 and import Task1.gaml and task2.gaml as a new project. Press main to run the simulation. Note that changing parameters the N parameter in the first task will increase the size of the chessboard and the queens placed on it. Changing the numbers of festival guests and stages in task 2 will do just that in the simulation.

Species

Queen Agent

The queen agent is responsible for finding a space in the column matching its index in the list of queens, with the condition of having a spot that is not under attack by other previous queens. If it cannot do so it is then responsible to inform the previous queen so that it moves and respond to such responses when they are given from the successor queen.

Festival Guest Agent

This agent was responsible with requesting the values of the attributes of the existing stages. Based on the values it receives and the values in its preferences, it chooses a stage to head to.

Stage Agent

This agent has a set of values associated with the stages attributes at a given time. When the guests request it sends those values so that they can choose whether they fit their preference better than the other stages.

Implementation

Task 1

Started by implementing the chessboard and having N queen agents place themselves in each column. The checkDiagonal and checkRow functions were then implemented so that the queens can decide whether a spot is suitable or not. Finally, the back-propagation function of the agents informing the previous queen to move its position when no spots are available was implemented.

Task 2

The stage agent was first implemented as it is simpler and as the guest agent relies on its implementation for its own. It initiates itself by randomly assigning values out of 100% for each of its attributes. It then awaits messages from the guests and sends those values. Finally, it regenerates the values every given time interval. The guest agent was then implemented, and it simply requests the values of all the stages. Calculates its interest in each stage and sets its target to one of the stages, that gives the highest interest. It requests for the values again after the same time interval it takes for the stage to regenerate them.

Results

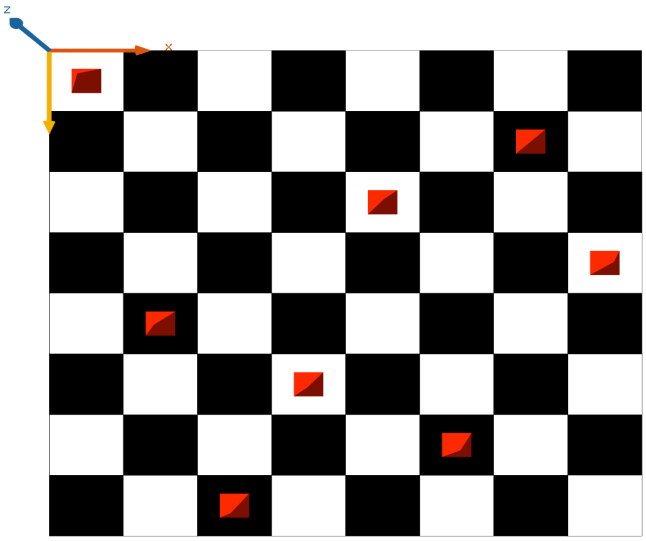
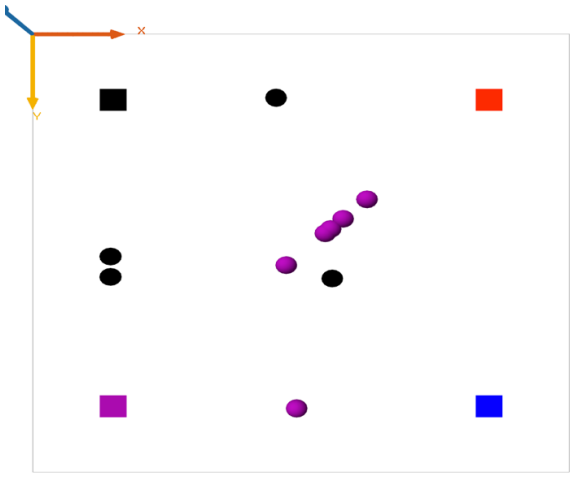


Figure 1: A screenshot of the final solution for task 1.

```
Guest0: I want to know the stage attribute values!
Guest1: I want to know the stage attribute values!
Guest2: I want to know the stage attribute values!
Guest3: I want to know the stage attribute values!
Guest4: I want to know the stage attribute values!
Guest5: I want to know the stage attribute values!
Guest6: I want to know the stage attribute values!
Guest7: I want to know the stage attribute values!
Guest8: I want to know the stage attribute values!
Guest9: I want to know the stage attribute values!
Stage0: Guests are asking for my values so I'm sending them!
Stage1: Guests are asking for my values so I'm sending them!
Stage2: Guests are asking for my values so I'm sending them!
Stage3: Guests are asking for my values so I'm sending them!
Guest0: found my match! going there now.
Guest1: found my match! going there now.
Guest2: found my match! going there now.
Guest3: found my match! going there now.
Guest4: found my match! going there now.
Guest5: found my match! going there now.
Guest6: found my match! going there now.
Guest7: found my match! going there now.
Guest8: found my match! going there now.
Guest9: found my match! going there now.
```



Figures 2 & 3: A screenshot of the logs and simulation task 2 showing guests choosing appropriate stages.

Discussion / Conclusion

The N queens problem took a bit longer than task 2 in this assignment the main reason being that I was evaluating the diagonal occupancy incorrectly from the beginner. This was solved when I split my thinking to checking the diagonal occupancy upwards and downwards, which made the task quite simple. Overall, good assignment and cool simulations as a result.