Challenge 1: Memory of Agents

Goal

When an agent visits a shop, they will remember the position. However, sometimes they would like to discover new places as well. Implement a small memory in agents so they will remember places they have been to (make his actions randomized).
Result: Implementing this logic and compare distance traveled. Comparing the distance between having and not having memory.

Implementation

1. Attributes of memory

```
list<stores> memory <- []; # Store the visited stores
float totalDistance <- 0.0; # Total distance
point last_location <- nil; # Each step of displacement change
float explorationProbability <- 0.3; # 30% exploring new locations
```

2. Distance tracking mechanism
   Check for changes in position, gradually accumulate the total displacement, and provide quantifiable indicators for subsequent model performance comparisons.

```
reflex trackDistance {
    if last_location != nil {
        totalDistance <- totalDistance + (location distance_to last_location);
    }
    last_location <- location;
}
```

3. Target selection strategy
   When the demand is triggered, the following priority order is adopted:

- First, determine whether it is a need for food or beverages;

- Then, based on explorationProbability:
  - Utilize memory: Select the nearest matching store from memory
  - Explore new stores: Obtain the nearest resource points through infoCenter
    If memory utilization is successful, avoid repetitive long-distance searches.

4. Memory update mechanism
   Automatically record the location of the store when visitors arrive at the resource point.

```
reflex arrivedStore when: target != nil and location distance_to(target.location) <
    ask target {
        if(not (self in myself.memory)) {
            myself.memory <- myself.memory + self;
        }
    }
    target <- nil;
}
```

5. Performance index

```
monitor "Average distance traveled" value: mean(guest collect (each.totalDistance));
monitor "Average memory size" value: mean(guest collect (length(each.memory)));
```

For comparison: there are memory models and non-memory models
To test the effectiveness of the memory mechanism in path optimization.

Evaluation

Simulation cycle	Average distance traveled in Basic Module	Average distance traveled in Memory Module	Difference	Average memory size
20	56.9700	56.983	+ 0.013	0.800
100	296.607	296.704	+ 0.097	2.400
3302	9888.261	9882.621	- 5.64	4.600

5777	17305.271	17289.947	- 15.324	5.133
7613	22805.314	22793.428	- 11.886	4.400
11251	33707.480	33686.464	- 21,016	4.533

Conclusion

In the early simulation cycles (20–100 steps), there is almost no difference in average distance traveled between the Basic Module and the Memory Module. This indicates that memory has little effect before agents have accumulated enough visited locations. As the simulation progresses (after ~3000 cycles), the Memory Module begins to show a consistent reduction in total distance traveled. Agents gradually build a useful memory set (average memory size stabilizing around 4–5), which allows them to reuse known store locations instead of re-discovering them through wandering or querying the information center.

Overall, the memory mechanism provides a long-term efficiency gain, but the improvement is incremental rather than dramatic. The primary benefits only become visible after sufficient experience is accumulated, indicating that:

- Memory reduces redundant exploration and store-searching over extended time.
- The effectiveness of memory depends on the number of visited stores in memory.
- The system exhibits experience-based optimization, converging toward more efficient movement patterns.

In summary, the Memory Module demonstrates progressive efficiency improvement in long-run behavior, confirming that memory enhances agent performance by reducing travel cost once enough store locations have been learned.