Towards Large-Scale Multi-Agent Based Rodent Simulation: The “Mice In A Box” Scenario

E. Agiriga, F. Coenen, J. Hurst, R. Beynon, D. Kowalski

Abstract Some initial research concerning the provision of a Multi-Agent Based Simulation (MABS) frameworks, to support rodent simulation, is presented. The issues discussed include the representation of: (i) the environment and the characters that interact with the environment, (ii) the nature of the “intelligence” that these characters might posses and (iii) the mechanisms whereby characters interact with environments and each other. Two categories of character are identified: “dumb characters” and “smart characters”, the obvious distinction being that the first posses no intelligence while the second have at least some sort of reasoning capability. The focus of the discussion is the provision of a simple “mice in a box” scenario simulation.

1 Introduction

Multi-Agent Based Simulation (MABS) is concerned with the harnessing of Multi-Agent System (MAS) technology to enable large scale simulations. The challenge is the mechanisms and representations required to build frameworks to support the desired simulation. Using MABS the *characters* that play a part in the simulation, and the environment(s) in which they exist, are conceptualised as agents. MABS has been applied in many domains such as: the monitoring and control of intelligent

E. Agiriga, F. Coenen, D. Kowalski
Dept. of Computer Science, University of Liverpool, Liverpool L69 3BX, UK. e-mail: {grigs,coenen,darek}@liverpool.ac.uk

R. Beynon
Institute for Biocomplexity, University of Liverpool, Liverpool L69 3BX, UK. e-mail: R.Beynon@liverpool.ac.uk

J. Hurst
Mammalian Behaviour and Evolution Group, University of Liverpool, Liverpool L69 3BX, UK. e-mail: G.E.Hurst@liverpool.ac.uk

buildings [2], transport chains [3], malaria re-emergence in the south of France [6], and urban population growth [7], to give just a few examples. To the best knowledge of the authors there is no work on MABS frameworks to study rodent behaviour. This paper describes some early research regarding issues concerned with the provision of MABS frameworks for rodent control. The focus of this report is a simple "mice in a box" scenario. However, the intention is to develop the framework so that it can be used to support large scale mouse simulations comprising some thousand agents.

2 The Mouse in a Box Scenario

The scenario at which the discussion presented in this paper is directed is that of a number of mice contained in a box. The scenario is founded on the sort of experiments conducted by rodent behaviourists who wish to observe the way that mice interact when placed in a closed environment, namely a $1.22 \times 1.22m$ box[1]. The fundamental idea is that one, two or more mice are placed in a box in which they can "run around". Mice have an affinity to walls [1] (they are *thigmotaxic*) and thus tend to moves along walls (although not exclusively so), thus in the absence of any obstructions a mouse's movements tend to be limited to the edges of the box. The mouse can move round the box in either a clockwise or ant-clockwise direction. It can also stop or turn around, occasionally it may venture into the space in the middle of the box. Mice are also interested in exploring their surroundings, the ultimate goal is the find and maintain an optimum nest location. The stronger Male mice have the best territory (nest locations). Females look for males with the best territory. Males mark their territory with scent, the stronger the male the stronger the scent. In the scenarios considered in this paper only male mice are considered. They are driven by the following desires:

1. A preference for wall locations as opposed to open space locations (in open space they are liable to attack by predators).
2. A desire to explore their environment.
3. A desire to avoid locations which feature the scent of other mice (unless that scent is significantly weaker than the mouse's own scent).
4. A requirement to avoid other mice that come into close proximity.

The above provides for some motivation for a mouse agent to move (to explore its locality), although there is no specific goal (reward). Whether the mouse moves or does not move, how long it moves for (or does not move) and which direction it should take, is a decision influenced partly by the above desires and partly by a random element.

[1] The value 1.22 is a result of the fact that the board from which the boxes are typically fashioned comes in $2.44 \times 1.22m$ sheets

3 The MABS Framework

The MABS framework is conceptualised in terms of a “cloud” in which a number of agents exist (Figure 1). From the figure we have three types (classes) of agent: (i) environment agents, (ii) obstruction agents and (iii) mouse (character) agents. The first two are characterised as “dumb” agents in that they do not display any intelligence, while the last has some “thinking” capability. From the figure it can be observed that we have only one environment agent and any number of obstruction and mouse agents (in fact we can have zero obstruction agents, but it would not make any sense to have zero mouse agents). In Figure 1 the arcs indicate communication lines; so the vision is that mouse agents can communicate with one another and the environment agent, while obstruction agents only communicate with the environment. Inspection of Figure 1 indicates that we also have some: (i) house keeping agents to facilitate the operation of the framework, (ii) a simulation interface with which an end user can interact so as to set up individual simulations and (iii) a visualisation unit that allows the end user to observe simulations. Each of the individual classes of agent are described in detail in the following three sections.

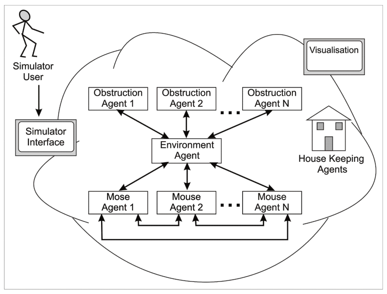

Fig. 1 Proposed MABS Framework

4 The Environment Agent

In the context of the proposed MABS framework an environment agent describes the *playing area*. In the case of the mouse in a box scenario this will be the box. A significant research issue with respect to the desired MABS is how best to represent this playing area. The simplest approach is to represent the playing area as a 2-D grid. However, this may not scale up for large simulations and features the irritation that the centroids of the neighbouring squares of a current square are not equidistant (neighbouring squares on the diagonal are further away than the immediately adjacent squares). Alternative representations include hexagonal grids, vector maps and graphs. However, because of its simplicity, the 2-D grid representation was adopted with respect to the framework described here.

The environment agent thus represents a playing area comprised of a 2-D grid. The dimensions of the environment were defined in terms 1cm units. A mouse was assumed to measures 7cm in all directions (not true, but the assumption can be upheld for the purpose of the simple mouse in a box scenario). A mouse was deemed to move at the rate of one 1cm per 50 mili-seconds. Each grid square (location) was given a numeric code, a Ground Type Identifier (GTI), indicating the nature of the square. The currently available codes were in the range $|0 \ldots 4|$ where: 0 indicated a "no-go" square, 1 a "wall" square, 2 a "space" square (non-wall square), 3 a "choice point" and 4 an obstruction (serving to hide the location of other mouse agents). The mouse cannot move into no-go or obstruction locations.

A mouse agent's location is described by its centroid; thus a mouse cannot get closer to a wall or obstruction than 3cm. Therefore all squares within three units of a wall or obstruction were encoded as no-go squares (0), squares exactly four units away from a wall or obstruction were labelled as wall squares (1), and squares more than four units away from walls as space squares (2). Choice points, at their simplest, are then wall squares that coincide with *obtuse* corners; where the mouse might wish to change direction (or stop); or squares where current movement may proceed in more than one wall direction. The corners of the boxes could also have been marked as choice points; however the movement of a mouse agent entering into these locations will be blocked thus, in effect, the location acts as a choice point without actually being marked as such

The current implementation features six types of environment agent: (i) Box, (ii) H-box, (iii) O-box, (iv) Four Box, (v) Four Nest and (vi) Maze. The first represents the simplest scenario. The H-box introduces the concept of obstructions (agents in their own right) into the box scenario, obstructions can be thought of as "bricks" placed into the box environment so as to impede a mouse agent's progress. The four box scenario comprises four occurrences of the box scenario running simultaneously, but described as a single environment with obstructions placed so as to achieve four boxes. The four nest box was used to simulate the interaction of four mouse agents. The maze scenario comprises a box scenario with a set of obstructions arranged to form a "maze", the objective here was to test whether a mouse object could find its way through this environment. Every environment agent has the following fields:

1. *widthX*, the width of the environment, in terms of grid squares, in the X (East-West) direction.
2. *widthY*, the width of the environment, in terms of grid squares, in the Y (North-South) direction.
3. *groundArea*, the two dimensional grid describing the *locations* that make up the ground area (as described above).
4. *gateCoords*, one or more gates where characters can enter the environment (start points).
5. *obstructionList*, a list of zero, one or more obstruction agents that the environment needs to know about.

Each location within the environment has a GTI and a record of any scent at the location, together with the ID for the mouse agent to which the scent sample belongs. Scent is defined in terms of an integer. Scent typically lasts for 8 to 24 hours depending on the dominance of the mouse. We degrade the mouse scent on each iteration of the simulation. To speed up the simulation we can enhance the degradation factor. Currently the maximum scent strength is 255 and it is degraded by 0.25 on each iteration (a more realistic simulation would require a much lower degradation factor).

5 The Obstruction Agent

Obstruction agents are simple agents that, as noted above, can be conceptualised as "bricks" that may be located within an environment. The bricks may be placed in the box as the scenario progresses, hence obstructions are considered to be agents in their own right. The H-box environment contains two obstruction agents so that the environment, when observed in plan view, formed an "H" shape. The O-box contained a single obstruction in the middle of the box so that the environment, when observed in plan view, resembled an "O" shape. The four box and four nest environment also contained two obstruction agents, but arranged to form an intersecting cross so as to divide the environment into four sub-boxes (Our "bricks" can intersect) and to form four "nest area" respectively. The maze environment had eighteen obstruction agents arranged in a "maze" formation. Similar to an environment agent, obstruction agents are dumb agents. The significance of obstruction agents is that mouse agents cannot "see" behind them; they obstruct a mouse agent's "field of view".

6 The Mouse Agent

A mouse agent is the central character in our mouse simulator. Mouse agents have the following fields:

1. *state*, the current state of the mouse agent, either *moving*, *stopped* or *turning*.
2. *stateTime*, the time spent in the current state.
3. *coordX*, the mouse agent's current X location with respect to the environment agent.
4. *coordY*, the mouse agent's current Y location with respect to the environment agent.
5. *direction*, the direction the mouse agent is facing, a number in the range of $|0 \ldots 7|$ representing N, NE, E, SE, S, SW, W or NW respectively.
6. *goalDirection*, the direction the agent wishes to face (only applicable when turning).
7. *turnDirection*, the "turning direction", either *clockwise* or *anticlockwise* (also only applicable when turning).
8. *scentStrength*, the strength of its scent.

9. *visionMap*, a disc of locations, with radius *v*, representing the part of the environment which a mouse agent can “see”. Thus a mouse agent’s field of vision is equivalent to *v*.

Table 1 Action Table

Current State	Event	Action	Comments	New State
stopped	None	Agent decides to move in direction faced	$sateTime = 0$	*moving*
stopped	None	Agent decides to move another direction	$sateTime = 0$	*turning*
moving	At choice point	Agent decides to move in new direction		*turning*
moving	Obstruction	Agent decides to move in new direction		*turning*
moving	Obstruction	Agent decides to stop	$sateTime = 0$	*stopped*
moving	None	Agent decides to stop	$sateTime = 0$	*stopped*
turning	Completed turn	None		*moving*

Mouse agents are dynamic agents in that they can change their location, direction, goal direction, turn direction and state. At the same time they are “intelligent” agents in that they can make decisions about which way to face and where to go.The operation of our mouse agent is founded on the well established concept of a Finite State Machine (FSM) [5, 8]. FSM are used to model processes in terms of a finite set of *states*. A change from one state to another is called a *transition*. Transitions are caused by *events* or *actions* (something happening to the agent or the agent doing something). The possible transitions to a new state, caused by an event or action, are typically described using a *transition table* (*state diagram* or *state table*). FSMs can be conceptualised as graphs (state models) where the vertexes represent states and the edges transitions caused by events or actions. An alternative approach would be to use the Belief-Desire-Intention (BDI) model [4]. This offers the advantage that it is supported by existing logic models. However, planning is typically outside the scope of the model. Given that in our model we think of mouse agents being in a certain state; and that changes from one state to another with an element of randomness as well as intention (expressed in the form of preferences), a finite state machine mechanism of operation was adopted.

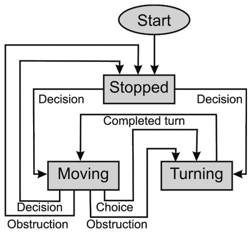

Fig. 2 State Model

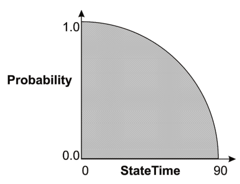

Fig. 3 Cosine Probability

The transition table for the mouse object is given in Table 1, which should be interpreted with respect to the state model presented in Figure 2 . There are seven different possible transitions. At the start of each simulation the default state for a mouse object is *stopped*. Eventually the mouse will decide to move (how this is determined is discussed below). The mouse object can either move in the direction it is currently facing or turn to face another direction and then move (how this is determined is also discussed below).Thus there are two possible state transitions associated with the *stopped* state.

There are four possible state transitions associated with the *moving* state. The first is when the mouse reaches a choice point. From the above, mice are "wall huggers". A choice point is a location where there are more than one possible next wall locations (as in the case of the maze environment) or the next possible wall location requires a change in direction. In the first case the mouse may decide to continue to move in the current direction, in which case there will be no state change; alternatively the mouse may decide to turn and move in a new direction, thus adopt a *turning* state. The second and third movement state changes are where the mouse's movements are blocked (for example at the corner of a box environment). In this case the mouse can decide to stop (adopt a *stopped* state) or head off in a new direction (change to a *turning* state). Note that in the case of a choice point, in the current implementation, the mouse does not have an option to stop. Finally a mouse in a *moving* state may simply decide to stop (how this is determined is discussed below).

The final state transition in Table 1 occurs when a mouse agent completes a turn, in which case the mouse will move in the direction it is now facing (i.e. adopt a *moving* state). The assumption here is that the only reason for a mouse to turn is to move in a new direction.

At the start of a simulation the state of the mouse agent is always *stopped*. Conceptually the mouse agent can only stay stopped for a finite period of time T. The probability that the mouse will stay stopped decreases as the the current *stateTime* increases (i.e. as the time the mouse agent has spent in its stopped state increases). When $stateTime \equiv T$ the probability that the mouse will stay stopped is 0.0 (definitely decide to move), when $stateTime \equiv 0$ the probability is 1.0 (definitely stay stopped). This probability distribution was modelled using a cosine probability curve (Figure 3); we could have used a linear probability, or some other alternative, however the cosine probability has the feature that the likelihood of the mouse agent staying stopped remains high at low *stateTime* values, and becomes negligible (reducing to 0.0) as *stateTime* approaches T. On each simulation iteration, when the mouse object is stopped, a random number r is generated. A state transition will then occur when:

$$r < cosin\left(\frac{90 \times stateTime}{T}\right) \tag{1}$$

A similar process was applied where a mouse agent's state is *moving*. The assumption is again that the mouse will continue to move for a finite period of time, but in this case the time period was assumed to be $2T$. Thus, on each iteration, when the mouse object is moving a state transition will occur when:

$$r < cosin\left(\frac{90 \times stateTime/2}{T}\right) \quad (2)$$

7 Selecting a Direction of Travel

When a mouse agent reaches a choice point or discovers an obstruction (i.e. it cannot or may not proceed any further in the current direction) the agent must make a decision. Where an obstruction is reached the mouse has the option to stop or proceed in a new direction (see Figure 2); the decision whether to stop or not is determined using identity 2. Where a change of direction is indicated a mouse agent has between 0 and 8 potential directions it can choose from. A mouse agent cannot enter no-go locations ($GTL = 0$); thus, depending on the mouse agent's current location, some directions will not be permissible. It is possible for a mouse agents movement to be entirely blocked by obstructions and/or the presence of other mouse agents. in which case the mouse will adopt a *stopped* state. Assuming a mouse agent has one or more potential directions it can move in each potential direction has a preference value p of between $|0.0\ldots1.0|$. The complete set of preference values, P, is then defined as:

$$P = \{p_0, p_1, \ldots, p_n\} \quad (3)$$

such that:

$$\sum_{i=0}^{i=n} p_i = 1.0 \quad (4)$$

(where n is the number of available directions/locations).

Preference values are made up of a number of components $C = \{c_1, c_2, \ldots, c_m\}$, where m is the number of components. Each component describes some factor of the decision making process. A specific component j associated with a specific direction i is indicated as c_{ij}. Each component has a value of $|0.0\ldots1.0|$. Such that $\sum_{i=0}^{i=n} c_{ij} = 1.0$ (i.e. the set of values describing a particular component across the set of potential directions is equivalent to 1.0). Some components may be considered to have greater significance than others, thus the components are weighted[2]. The weighting associated with a component c_j is indicated by w_j. The preference (p) for a particular location (i) is then calculated as follows:

$$p_i = \frac{\sum_{j=0}^{j=m} w_j p_{ij}}{\sum_{j=0}^{j=m} w_j} \quad (5)$$

In the current simulation implementation four components are considered ($m = 4$) as follows:

[2] Although not a feature of the current implementation, these weighting mat be dynamic (i.e. they may be changed according to circumstances).

c_1 Preference according to GTL (desire for wall locations over space locations).
c_2 Preference for locations not recently or never visited (desire to explore).
c_3 Preference for avoiding locations where the scent of another mouse is significant compared with a mouse agent's own scent strength (desire to avoid the scent trails of other mice).
c_4 Preference for directions that tend to move way from other mouse agents if within sight (desire to avoid other mice).

Algorithm 1 Determination of preference for wall location component (c_1)

```
$L = Set\ of\ potential\ locations$
$N_n = Number\ of\ nonspace\ locations\ in\ L$
$N_s = Number\ of\ space\ locations\ in\ L$
if $N_s \equiv 0$ then
    $p_n = 1.0/N_n$
else
    $p_n = P_n/N_n$
    $p_s = P_s/N_s$
end if
for $i = 0 \to |L|$ do
    if $L_i.groundType \equiv space\ location$ then
        $L_i.c_1 = p_s$
    else
        $L_i.c_1 = n_s$
    end if
end for
```

7.1 Desire for Wall Locations over Space Locations (c_1)

As noted above mice prefer to move along walls, thus a preference should be given to directions (next locations) adjacent to walls. A mouse agent will have potentially N_n wall locations and N_s space locations to choose from, where N_n and N_s are whole numbers in the range of $|0\ldots 8|$. Except in the special case where a mouse agent is blocked in, $1 \leq (N_n + N_s) \leq 8$. Of these directions zero, one or more will be space locations, and one or more will be non-space (wall or choice point) locations. The overall probability that a non-space location, L_n, is selected is given by P_n; and the overall probability that a space location, L_s, is selected by P_s, where P_n is assumed to be significantly greater than P_s. If $N_s \equiv 0$ then $P_n = 1.0$. Thus the probability of selecting a specific non space location is given by P_n/N_n, and the probability of selecting a specific space location (if such locations exist) is given by P_s/N_s. The process of determining the values for the preference component that reflects a desire for wall locations is given in algorithm 1.

7.2 Desire to Explore (c_2)

The desire to explore is expressed according to where a mouse agent has been recently, which in turn is expressed according to the scent strength of the mouse agent's own scent strength found at neighbouring locations. A mouse agent prefers locations (directions) where its own scent is not present, or at least weak. Thus the preference for new locations is expressed as a fraction of the inverse of the mouse agent's own scent strength (s_{inv_i}) at a given location i. If no scent is present $s_{inv} = 1.0$. The process for calculating the desire to explore preference component is given in algorithm 2. The c_2 component at a particular candidate location q is given by:

$$c_{q2} = \frac{s_{inv_q}}{\sum_{i=0}^{i=n} s_{inv_i}} \tag{6}$$

In algorithm 2 the factor k is used to reduce the influence of the scent strength at recently visited locations. The current maximum scent strength is 255, and thus the k value has been set to 10; if we simply used the inverse of the scent strength the influence of very recent directions will be negligible, 0.004 $(1/255)$ as compared to 0.039 $(10/255)$.

Algorithm 2 Determination of desire to explore component (c_2)

```
$L = Set\ of\ potential\ locations$
$S = Set\ of\ inverses\ scent\ strengths$
$total = 0.0$
for $i = 0 \to |L|$ do
    if $L_i.ownScentStrength \equiv 0$ then
        $S_i = 1$
    else
        $S_i = k/L_i.ownScentStrength$
        $total = total + S_i$
    end if
end for
for $i = 0 \to |L|$ do
    $L_i.c_2 = S_i/total$
end for
```

7.3 Desire to Avoid Scent Trails of other Mice (c_3)

The desire to avoid the scent trails of other mice is encapsulated in a similar manner to the desire to explore new locations. We use the inverse of the strength of the strongest scent belonging to another mouse agent, or 1.0 if there is no such scent. The process is presented in algorithm 3 where $maxScentStrength$ is the scent strength associated with the scent strengths at a location belonging to other mice, 0 if there is no such scent strength.The constant K is again used.

7.4 Desire to Avoid other Mice (c_4)

A mouse agent knows nothing about the locations of other mice until they appear on its vision map. In the current simulation the radius of the vision map (v) is set to 20, however if the location of another mouse agent is obscured by an obstruction the current mouse agent will not know anything about this other mouse. To ensure the mouse agents do not actually crash into each other a buffer region of ten units is place round other mouse agents. Our mouse agents are currently programmed to avoid other mouse agents that are on its vision map. The values for this preference component are calculated according to the distance d from each candidate location to the nearest other mouse (if any). The c_4 component at a particular candidate location q is the distance d from the given candidate location q divided by the sum of the distances from all of the locations. Thus:

$$c_{q4} = \frac{d}{\sum_{i=0}^{i=n} d} \tag{7}$$

Algorithm 3 Determination of desire to avoid scent trails of other mice (c_3)

```
$L = Set\ of\ potential\ locations$
$S = Set\ of\ inverses\ scent\ strengths$
$total = 0.0$
**for** $i = 0 \rightarrow |L|$ **do**
    **if** $L_i.maxScentStrength \equiv 0$ **then**
        $S_i = 1$
    **else**
        $S_i = k/L_i.maxScentStrength$
        $total = total + S_i$
    **end if**
**end for**
**for** $i = 0 \rightarrow |L|$ **do**
    $L_i.c_3 = S_i/total$
**end for**
```

Algorithm 4 Next Location Algorithm

```
$L = Set\ of\ potential\ locations$
$Prob = 0.0$
$R = randomNumberGenerator()$
$L_{final} = -1$
**for** $i = 0 \rightarrow |L|$ **do**
    $Prob = Prob + L_i.prob$
    **if** $R < Prob$ **then**
        $L_{final} = L_i$
        $break$
    **end if**
**end for**
$return(L_{final})$
```

7.5 Decision making process

From the above each location has four components which are used to calculate a preference value for the location. Experiments indicated that the weighting that should be associated with c_1 and c_3 should be higher than those associated with the other components, w_1 and w_3 were therefore set to 2, while the remaining weightings were set to 1. The total preference for a particular location q was this given b:

$$p_q = \frac{2c_1 + c_2 + 2c_3 + c_4}{5} \tag{8}$$

The selection of a new direction was then determined using algorithm 4. The weightings can of course be adjusted as desired by the end user.

7.6 Change of Direction

Having selected a new location it may be necessary to change direction, if so a state transition from moving to turning will occur. Where a turn is initiated the mouse agents *goalDirection* and *turnDirection* fields must be reset. The value for the *turnDirection* field is calculated as follows as shown in algorithm 2 (recall that directions are specified as integers within the range $|0\ldots7|$).

Algorithm 5 Direction of Turn Algorithm

```
$diff = absolute(direction - goalDirection)$
**if** $if(goalDirection > direction)$ **then**
    **if** $diff \leq 4$ **then**
        $return(``clockwise'')$
    **else**
        $return(``anticlockwise'')$
    **end if**
**else**
    **if** $diff \leq 4$ **then**
        $return(``anticlockwise'')$
    **else**
        $return(``clockwise'')$
    **end if**
**end if**
```

8 Operation

The operation of the simulator was controlled by a Loop which iterated every 50 milliseconds. Thus, given that the mouse agent (when in a moving state) moves at a rate of one grid square per iteration and a grid square measures 1cm, the mouse agent travels at 1200cm per minute (or 72km per hour). Experiments were conducted using a number of different environments with a turn rate of 45 degrees per iteration, $T = 90$, $P_n = 0.95$, $P_s = 0.05$ and $k = 10$. The Box experiment was intended to establish that the mouse agent behaved in a reasonably realistic manner, as confirmed by domain experts. The H-box was intended to establish that the mouse agent could react to obstructions, the O-box was intended to observe the mouse agent's behaviour should it cross the open space between the outer wall of the box and the obstruction, the Maze experiment was used to evaluate the mouse agent's ability to negotiate choice points and the 4-box to demonstrate that mouse agents did not behave in the same way given four identical spaces. Finally the four nest box simulation was used observe how a group of mouse agents might interact given a hypothetical situation that they each might want to guard their own nest site.

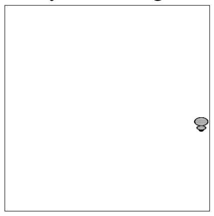
Fig. 4 Box Simulation

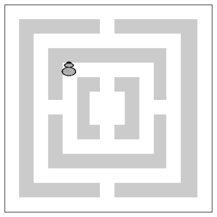
Fig. 5 Maze Simulation

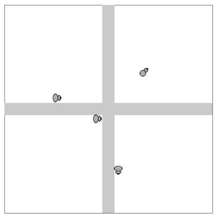
Fig. 6 4 Box Simulation

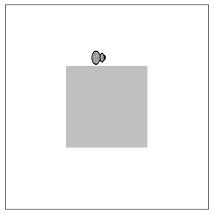
Fig. 7 O Box Simulation

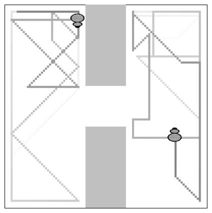
Fig. 8 H Box Simulation (with scent traces)

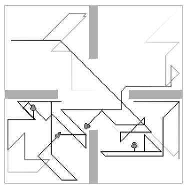
Fig. 9 Four Nest Box Simulation (with scent traces)

Figures 4 to 10 illustrating the simulations. Inspection of the figures indicates how mouse agents, when not influenced by the presence of other mouse agents, tend to follow walls. In the case of the O-box environment (Figure 7) the mouse agent has crossed the open space and is now hugging the wall of the obstruction. Figure 8 shows the H-box environment with two mouse agents and Figure 9 the four nest environment with four mouse agents. Both figures include scent trails. The objective in both cases was to observe how mice agents might define there own space. For the benefit of the simulation, and to allow easy observation, the "lifespan" of the scent deposits was kept deliberately shirt. Better results would be achieved by increasing the longevity of the scent trails however in this case the simulation has to be run over a much longer (and more realistic) time period. The experiment demonstrated in Figure 10 was designed to demonstrate that the simulator could function with a reasonable number of mouse agents.

9 Discussion and Conclusions

In this paper we have described a simple Multi-Agent Based Simulation (MABS) framework to describe the mouse in a box scenario. The intention was to provide a simple start point for the development of large scale rodent simulations. Features of the framework are: (i) that it can be used to create sophisticated environments using the concept of obstruction agents, (ii) several mice can operate in these environments and (iii) the mice operate in a sufficiently realistic manner. Experiments indicated that environments were easy to create and that simulations were easy to run and observe. The authors therefore believe that they have established a sound foundation on which to build. Current work is directed at techniques to support more sophisticated scenarios and to allow mouse agents to learn about their environments.

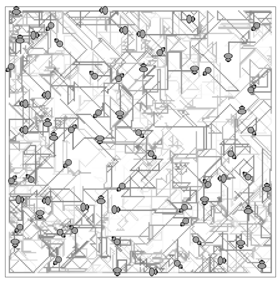

Fig. 10 Large (64 mouse agent) box simulation (with scent trails)

References

1. P. Crowcroft. *Mice All Over.* The Chicago Zoological Society, 1973.
2. P. Davidsson. Multi agent based simulation: Beyond social simulation. In *Proc. Workshop on Multi Agent Based Simulation (MABS) 2000*, pages 141–155. Springer-Verlag, LNCS 1979, 2001.
3. P. Davidsson, J. Holmgren, J.A. Persson, and L. Ramstedt. Multi agent based simulation of transport chains. In *Proc. 7th Int. Joint Conf. on Autonomous Agents and Multi-Agent Systems (AAMAS), Volume 2*, pages 1153–1160. Springer-Verlag, LNAI 3415, 2008.
4. B. Georgeff, M.P. andPell, Pollack M.E., M. Tambe, and M. Wooldridge. The belief-desire-intention model of agency. In *Proc. 5th Int. Workshop on Intelligent Agents: Agent Theories, Architectures and Languages (ATAL'98)*, pages 1–10. Springer-Verlag, London, 1998.
5. A. Gill. *Introduction to the Theory of Finite-state Machines*. McGraw-Hill, 1962.
6. C. Linarda, N. Pononb, D. Fontenilleb, and E.F. Lambin. A multi-agent simulation to assess the risk of malaria re-emergence in southern france. *Ecology Modelling*, 220(2):160–174, 2009.
7. X. Pan, C.S. Han, K. Dauber, and K.H. Law. A multi-agent based framework for the simulation of human and social behaviors during emergency evacuations. *AI and Society*, 22(2):113–132, 2007.
8. F. Wagner. *Modeling Software with Finite State Machines: A Practical Approach*. Auerbach Publicationsl, 2006.