

**SYSC 5104**

Assignment 2

November 13, 2020

**Reactive Agents**

**CD++ Simulator**

**Alvi Jawad**

St. ID: 101148341

Email: alvi.jawad@carleton.ca

**Table of Contents**

[**Part I: System Identification** 2](#_Toc56137950)

[**Reactive Agents** 2](#_Toc56137951)

[**Neighborhood Selection** 3](#_Toc56137952)

[**Rules: [AgentBehavior]** 4](#_Toc56137953)

[**Rule 1: Move to the EAST(E) comfortable cell** 4](#_Toc56137954)

[**Rule 2: Move to the NORTHEAST(NE) comfortable cell** 4](#_Toc56137955)

[**Rule 3: Move to the SOUTHEAST(SE) comfortable cell** 5](#_Toc56137956)

[**Rule 4: Move to the NORTH(N) comfortable cell** 5](#_Toc56137957)

[**Rule 5: Move to the SOUTH(S) comfortable cell** 5](#_Toc56137958)

[**Rule 6: Move to the NORTHWEST(NW) comfortable cell** 5](#_Toc56137959)

[**Rule 7: Move to the SOUTHWEST(SW) comfortable cell** 6](#_Toc56137960)

[**Rule 8: Move to the WEST(W) comfortable cell** 6](#_Toc56137961)

[**Rule 9: the default rule** 6](#_Toc56137962)

[**Part II: Formal Specification, Modeling & Simulation** 7](#_Toc56137963)

[**Formal Specification:** 7](#_Toc56137964)

[**Testing** 7](#_Toc56137965)

[**Test Case I** 8](#_Toc56137966)

[**Test Case II** 10](#_Toc56137967)

[**Test Case III – Extended Model** 12](#_Toc56137968)

[**Video Recordings** 14](#_Toc56137969)

[**Possible Extensions** 14](#_Toc56137970)

[**Conclusion** 14](#_Toc56137971)

[**References** 15](#_Toc56137972)

**Table of Figures**

[**Figure 1: Model Neighborhood** 3](#_Toc56137666)

[**Figure 2: Neighborhood Definition** 7](#_Toc56137667)

[**Figure 3: Simulation Steps - Test I** 9](#_Toc56137668)

[**Figure 4: Simulation Steps - Test II** 12](#_Toc56137669)

[**Figure 5: Simulation Steps - Test III** 13](#_Toc56137670)

**Table of Tables**

[**Table 1: Cell Values** 2](#_Toc56137825)

[**Table 2: Cell Colors** 9](#_Toc56137826)

# **Part I: System Identification**

## **Reactive Agents**

Agents are a crucial part of game worlds where they represent the individual non-player characters (NPCs) with their distinct personality and behavior that give the game life, story, and atmosphere. Agents can also be monsters or enemies that the player needs to fight or allies or soldiers in the player’s army. While many modern titles show awareness of other agents in the game environment, very few have made the in-game agents aware of important game events or environmental conditions such as a burning house or severe rainfall. The basic idea of this assignment is to model a game environment with cellular automata where flexible agents are able to react to their surrounding environment while pursuing a goal.

The core idea of this concept is derived from the “*Combining Influence Maps and Cellular Automata for Reactive Game Agents*” article by *Penelope Sweetser* and *Janet Wiles.* In their implementation, they combine influence maps with cellular automata to represent a 3D game world named EmerGEnT that model natural phenomena such as fluid flow, heat, fire, pressure, and explosions. The agents in their game world are able to react and move to other cells based on their comfort level in the current and neighboring cells with varied reaction times and exhibit different levels of intelligent behavior.

Our model derives from the concept of this article and presents a simplified model of reactive agents using CD++. In our model, the game terrain is represented by a W x W cell grid, where W is the system size. Each cell in the system has a predefined value that represents what kind of terrain it is. For the purpose of the assignment, we define 4 different cell types in *Table 1*:

**Table 1: Cell Values**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Cell Type** | **Value** | **Color** | **Description** | **Comfort Level** |
| Lava Cells | 300 | **Red** | Extremely hot, causes death | Unbearable |
| Heated Cells | 200 | **Orange** | Less hot than lava cells | Tolerable |
| Warm Cells | 100 | **Light Orange** | Less hot than heated cells | Mildly uncomfortable |
| Normal Cells | 0 | **Light Blue** | No special characteristics | Comfortable |

The existence of an agent on a specific terrain is represented by the original cell value + 1. In other words, a cell value 301 represents an agent standing on a Lava terrain (300). Likewise, 1 represents an agent occupying a Normal cell (0). In our model, this additional 1 represents a *human agent* that reacts to its *surrounding heat level* and tries to cross the terrain from *left to right* whenever possible.

Each cell value represents the *level of comfort* an agent feels when standing atop that cell. The level of comfort decreases as cell values increase. For example, Warm cells are much more comfortable for our human agent than both lava and heated cells. This makes cooler cells comparatively much more desirable to the agent, and the agent makes an effort to move to the nearest comfortable cell.

We use finite concrete values to represent the comfort level of the agent in each type of terrain. Lava cells (represented in red in the simulator) are unbearable and immediately causes death for any agent who moves on them. Heated cells are comparatively tolerable, and the agent prefers moving to them instead of lava cells. Warm cells are only mildly uncomfortable and the agent only moves away from these cells if he can move to a normal cell which provides the perfect level of comfort.

The concrete values used here can be replaced to represent other comfort parameters as well. Different agents might also react differently to certain terrain types. Aliens, for example, might be perfectly fine sitting on top of lava cells which causes instant death for humans. On the other hand, humans are might be able to swim over water tiles which evaporates a flame-based monster over time.

Agents start anywhere in the terrain and try to cross the terrain from left to right without colliding with any other agent in the system. While having this right-moving behavior, they always consider their current level of comfort and may move in opposite directions or take turnarounds if necessary to be as comfortable as possible.

## **Neighborhood Selection**

We select the extended Moore neighborhood to use in our model. This choice allows us to model the movement of the agent in all 8 cardinal (N, S, E, W) and ordinal directions (NE, NW, SE, SW) after selecting the cell that has the highest comfort level (lowest cell value) for the agent. Additionally, the second level allows us to avoid conflict with any other agent that is trying to compete for the same cell from all straight and diagonal directions.

|  |  |
| --- | --- |
| The extended Moore neighborhood | Download Scientific Diagram | This compass rose shows ordinal and cardinal directions. |

**Figure 1: Model Neighborhood**

An agent can only occupy one terrain space (one square cell space) at a time. When the simulation starts, all the positions of agents on the terrain are updated synchronously. We use a wrapped boundary and inertial delays in our model.

## **Rules: [AgentBehavior]**

In this section, we describe the rules that agents use to update their position in the cell space. All of these rules define the agents’ behavior in descending priority. We use pseudo code to represent the rules here, the implementation can be found in the **[AgentBehavior]** section of the **reactiveagent.ma** file.

Macros are used in the **reactiveagent.ma** file to make things more readable. The file **macro\_directions.inc** contains the macros for all the cardinal and ordinal directions (N, S, E, W, NE, NW, SE, SW). Another file **macro\_min.inc** contains macros that allow each cell to determine the minimum value of the agent’s immediate neighbors (Moore neighborhood level 1).

#macro(X) --> X represents the four cardinal directions

e.g., #macro(N) = The north cell

#macro(XY) --> XY represents the four ordinal directions

e.g., #macro(NE) = The northeast cell

#macro(MIN\_NBR) = The minimum of all the immediate neighbors

#macro (MIN\_RCVR\_X) --> X represents the receiving cell that has to check whether it is

the most comfortable cell around the incoming agent

e.g., #macro(MIN\_RCVR\_E) = the minimum of all the immediate neighbors of the

incoming agent from the perspective of the East cell

### **Rule 1: Move to the EAST(E) comfortable cell**

If the east cell is empty and

it is also the most comfortable cell out of all the immediate neighbors

-> move to that cell.

This rule has the highest priority and defines the right-moving tendency of agents whenever possible.

***Following Rule 1, Rules 2 and 3 represent the agents’ tendency to move diagonally to the right whenever a straight movement is not comfortable.***

### **Rule 2: Move to the NORTHEAST(NE) comfortable cell**

If the northeast cell is empty and

it is also the most comfortable cell out of all the immediate neighbors and

no other agent is trying to move to that cell using rule 1

-> move to that cell.

This rule has the second highest priority. If the east cell was not the most comfortable one, this will be the target cell to move to unless any other right-moving agent trying to move to it with a higher priority (rule 1).

### **Rule 3: Move to the SOUTHEAST(SE) comfortable cell**

If the southeast cell is empty and

it is also the most comfortable cell out of all the immediate neighbors and

no other agent is trying to move to that cell using rules 1-2

-> move to that cell.

This rule has the third highest priority. If the east and northeast cells were not the most comfortable ones, this will be the target cell to move to unless any other right-moving agent trying to move to it with a higher priority (rule 1, rule 2).

***Following Rules 1-3, Rules 4 and 5 represent the agents’ tendency to move upward or downward whenever a straight or diagonal movement to the right is not comfortable.***

### **Rule 4: Move to the NORTH(N) comfortable cell**

If the north cell is empty and

it is also the most comfortable cell out of all the immediate neighbors and

no other agent is trying to move to that cell using rules 1-3

-> move to that cell.

This rule has the fourth highest priority. If the east, northeast, and southeast cells were not the most comfortable ones, this will be the target cell to move to unless any other right-moving agent trying to move to it with a higher priority (rule 1, rule 2, rule 3).

### **Rule 5: Move to the SOUTH(S) comfortable cell**

If the south cell is empty and

it is also the most comfortable cell out of all the immediate neighbors and

no other agent is trying to move to that cell using rules 1-4

-> move to that cell.

This rule has the fifth highest priority. If the east, northeast, southeast, and north cells were not the most comfortable ones, this will be the target cell to move to unless any other right-moving or up-moving agent trying to move to it with a higher priority (rule 1, rule 2, rule 3, rule 4).

***Following Rules 1-5, Rules 6 and 7 represent the agents’ tendency to move diagonally left whenever a straight or diagonal movement to the right or an upward or downward movement is not comfortable.***

### **Rule 6: Move to the NORTHWEST(NW) comfortable cell**

If the northwest cell is empty and

it is also the most comfortable cell out of all the immediate neighbors and

no other agent is trying to move to that cell using rules 1-5

-> move to that cell.

This rule has the sixth highest priority. If the east, northeast, southeast, north, and south cells were not the most comfortable ones, this will be the target cell to move to unless any other right-moving, up-moving, or down-moving agent trying to move to it with a higher priority (rule 1, rule 2, rule 3, rule 4, rule 5).

### **Rule 7: Move to the SOUTHWEST(SW) comfortable cell**

If the southwest cell is empty and

it is also the most comfortable cell out of all the immediate neighbors and

no other agent is trying to move to that cell using rules 1-6

-> move to that cell.

This rule has the seventh highest priority. If the east, northeast, southeast, north, south, and southeast cells were not the most comfortable ones, this will be the target cell to move to unless any other right-moving, left-moving, up-moving, or down-moving agent trying to move to it with a higher priority (rule 1, rule 2, rule 3, rule 4, rule 5, rule 6).

***Following Rules 1-7, Rule 8 represent the agents’ tendency to move to the left when no other directional movement is comfortable.***

### **Rule 8: Move to the WEST(W) comfortable cell**

If the west cell is empty and

it is also the most comfortable cell out of all the immediate neighbors and

no other agent is trying to move to that cell using rules 1-7

-> move to that cell.

This rule has the eighth highest priority. If the east, northeast, southeast, north, south, southeast, and southwest cells were not the most comfortable ones, this will be the target cell to move to unless any other right-moving, left-moving, up-moving, or down-moving agent trying to move to it with a higher priority (rule 1, rule 2, rule 3, rule 4, rule 5, rule 6, rule 7).

***Following Rules 1-8, Rule 9 represent the agents’ tendency to stay in the same cell when no movement is comfortable and the current cell is the most comfortable. This rule also extends to all terrains maintaining their current value if no agent is trying to move to or leave from them.***

### **Rule 9: the default rule**

If rules 1-8 cannot be followed

-> maintain the current cell state.

This rule has the lowest priority. If the east, northeast, southeast, north, south, southeast, southwest, and west cells were not the most comfortable ones, each cell (agent or no-agent) will maintain their current state.

# **Part II: Formal Specification, Modeling & Simulation**

In this section, we provide the formal specification for the ReactiveAgents model followed by modeling the system using CD++ and testing different simulation scenarios. As mentioned before, we use the extended Moore neighborhood, represented in *Figure 2* below. The cells marked in green and red represent the agent and its immediate neighborhood, respectively. The grey cells represent the extended neighborhood that is necessary to avoid collision with other agents.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **(-2,-2)** | **(-2,-1)** | **(-2,0)** | **(-2,1)** | **(-2,2)** |
| **(-1,-2)** | **(-1,-1)** | **(-1,0)** | **(-1,1)** | **(-1,2)** |
| **(0,-2)** | **(0,-1)** | **(0,0)** | **(0,1)** | **(0,2)** |
| **(1,-2)** | **(1,-1)** | **(1,0)** | **(1,1)** | **(1,2)** |
| **(2,-2)** | **(2,-1)** | **(2,0)** | **(2,1)** | **(2,2)** |

**Figure 2: Neighborhood Definition**

## **Formal Specification:**

|  |
| --- |
| **Coupled Model, ReactiveAgentsAtomic = <** **X, Y, S, N, type, d, τ, δint, δext, λ, ta >**  X = {};  Y = {};  S = {0, 1, 100, 101, 200, 201 , 300, 301};  // 0 is a normal cell  // 1 is an agent occupying a normal cell  // 100 is a warm cell  // 101 is an agent occupying a warm cell  // 200 is a heated cell  // 201 is an agent occupying a heated cell  // 300 is a lava cell  // 301 is an agent occupying a lava cell  N = { (-2,-2), (-2,-1), (-2,0), (-2,1), (-2,2),  (-1,-2), (-1,-1), (-1,0), (-1,1), (-1,2),  (0,-2), (0,-1), (0,0), (0,1), (0,2),  (1,-2), (1,-1), (1,0), (1,1), (1,2),  (2,-2), (2,-1), (2,0), (2,1), (2,2) };  type = inertial;  d = 100; // in milliseconds  τ: N🡪S; // (please refer to the *Rules: [AgentBehavior]* section)  δint, δext, λ, and ta are defined using the Cell-DEVS specification |
|  |
| **Coupled Model, ReactiveAgents = <Xlist, Ylist, X, Y, I, ƞ, N, {r, c}, C, B, Z, select>**  Xlist = {};  Ylist = {};  X = Y = {0, 1, 100, 101, 200, 201 , 300, 301};  I = <Px, Py>  ƞ = 25;  N= { (-2,-2), (-2,-1), (-2,0), (-2,1), (-2,2),  (-1,-2), (-1,-1), (-1,0), (-1,1), (-1,2),  (0,-2), (0,-1), (0,0), (0,1), (0,2),  (1,-2), (1,-1), (1,0), (1,1), (1,2),  (2,-2), (2,-1), (2,0), (2,1), (2,2) };  r = 10; c = 10;  C = {Cij / i ϵ [0,9], j ϵ [0,9]};  B = {}; //wrapped border  Z = the translation function that determines the dynamics of the agent behavior  //(please refer to the *Rules: [AgentBehavior]* section)  select = { (0,0), (0,1), (-1,1), (1,1), (-1,0), (1,0), (-1, -1), (1, -1), (0,-1) } |

## **Testing**

We start by testing each of our rules in sequence. Each test case will involve agents being put in different terrains where they will try to move towards the right using the most comfortable path possible. They will also organize their movement based on the rule priority and either wait or move towards an equally comfortable cell in a different direction if their primary target cell is to be occupied by another agent that is using a higher priority rule.

### **Test Case I**

In this test case, we put two agents in a terrain surrounded by lava, heated, and warm cells. Both agents are slightly far away from being perfectly comfortable. According to our ruleset, they should move towards the right, while trying to use the most comfortable path whenever possible. In this test case, we test the rules 1-4.

*Table 2* represents the cell colors to be used in all of our simulations. New here is the addition of the existence of the agent that now reacts to the terrain around it. The additional two colors, dark green and light green, represent the uncomfortable and comfortable state of the agent, respectively.

**Table 2: Cell Colors**

|  |  |  |  |
| --- | --- | --- | --- |
| Cell type | Cell color |  | Cell value |
| Lava | Red |  | 300 |
| Heated | Orange |  | 200 |
| Warm | Light Orange |  | 100 |
| Normal | Light Blue |  | 0 |
| Agent (uncomfortable) | Dark Green |  | 101, 201, 301 |
| Agent (comfortable) | Light Green |  | 1 |

*Figure 3* demonstrates an eight-step simulation scenario that makes use of the **DRAWLOG** and **GrafLog** utilities. The input values for this test case can be found in the **Test I – Rules 1-4** folder inside the project directory. To run the same simulation, please copy the **reactiveagent.val** file found there to the project directory, run **reactiveagent.bat**, and follow the instructions in the command-line interface(CLI) to navigate the simulation scenarios to observe the same results and more.

|  |  |  |
| --- | --- | --- |
| 0 | 1 | 2 |
| 3 | 4 | 5 |
| 6 | 7 | 8 |

**Figure 3: Simulation Steps - Test I**

Step 0 represents the initial state of the cell space where both reactive agents are in their uncomfortable state (represented by dark green) being surrounded by high-temperature cells. The agent in the bottom is surrounded by lava and heated cells, whereas the top agent is surrounded by lava, heated, and warm cells.

In the next step (step 1), both agents take a step to the right (east cell) despite several other cells being equally comfortable for them. This indicates the agents’ tendency to reach the right border as fast as possible.

In step 2, the bottom agent moves diagonally up to the right (northeast) as that cell is deemed to be the most comfortable among all the neighboring cells. This represents the negation of rule 1 and rule 2 being followed by the agent. The exact same cell is considered to be most comfortable for the top agent as well as both agents want to move to the right. But since it has to follow rule 3 to do the same, it instead waits to make room for the bottom agent that is using a higher priority rule (rule 2).

In step 3, the bottom agent follows rule 3 to move diagonally down to the right (northwest) whereas the top agent moves one cell in the upward direction following rule 4. Both cells at this point are on the verge of leaving the uncomfortable heated area and move towards more comfortable positions.

Step 4 represents both cells moving out of the heated area and becoming comfortable (represented by light green) in their current situation. The bottom agent moves straight to the right (prioritizing rule 1 over rules 2 and 3), and the top agent moves toward the northeast direction, even though the north and northwest cells being equally comfortable (prioritizing rule 2 over rules 4 and 6).

From step 5 through step 8, both cells keep moving straight to the right following rule 1. The bottom agent reaches the right border at step 7 and appears on the other side of the cell space in the next step due to the wrapped nature of the boundary. From that point onwards, that agent faces a wall of lava which it tries to avoid by moving around it, which can be seen by running more simulation steps.

***This test case simulated the two agents following rules 1 - 4 to satisfy their desire to be as comfortable as possible while also maintaining a right-moving behavior.***

### **Test Case II**

In this test case, we put one agent in a very uncomfortable situation where it has no other way except moving backward to stay comfortable. We also put two other comfortable agents to see interesting interactions between all three agents. In this test case, we test the rules 5-8.

*Figure 4* demonstrates a fourteen-step simulation scenario. The input values for this test case can be found in the **Test II – Rules 5-8** folder inside the project directory. To run the same simulation, please copy the **reactiveagent.val** file found there to the project directory, run **reactiveagent.bat**, and follow the instructions to observe the same results and more. We do not provide separate .bat files for these tests as the .ma file contents stay the same and this allows us to do a simple copy and paste avoiding clutter in the primary project folder.

|  |  |  |
| --- | --- | --- |
| 0 | 1 | 2 |
| 3 | 4 | 5 |
| 6 | 7 | 8 |
| 9 | 10 | 11 |
| 12 | 13 | 14 |

**Figure 4: Simulation Steps - Test II**

In the first step, the uncomfortable agent in the middle of high-temperature cells can only move back following rule 4 to stay comfortable. In step 2, it finds a more comfortable cell to the southwest cell and follows rule 7 to move there. In the third step, it sees two ways to become perfectly comfortable to its northwest and west. However, since rule 6 takes priority over rule 8, it moves to the northwest cell and becomes comfortable. In step 4, we can see the utilization of rule 5, as the cell moves downward.

The behavior of the two comfortable cells to the right is rather simple until the fourth step. They simply keep moving toward their right following rule 1 and appear at the first column at step 4. From that point onwards, we continue to see an interesting interplay of various rules between these three agents in their effort to move right while staying as comfortable as possible.

***This test case simulated three agents following rules 5 - 8 to satisfy their desire to be as comfortable as possible while also maintaining a right-moving behavior.***

### **Test Case III – Extended Model**

Next, we check for the scalability of the model. Extending the model to a WxW grid where (W>10) is as simple as changing the *width* and *height* value in the **reactiveagent.ma** file. Due to the way the model is built, extending the range of uncomfortable cells only requires adding new values that are a multiple of 100 (e.g., 400, 500, 1000) to the **reactiveagent.val** file (higher values mean lower comfort). Finally, to visualize the new type of cells, new color palettes must be added to the **reactiveagent.col** file.

*Figure 5* represents an extended version of our previous model with a 20x20 cell space. We have also added two new levels of discomfort for the reactive agent. Magma cells, represented in dark red(), are even hotter, and as such, more uncomfortable than lava cells. Another new addition, plasma cells, are marked in magenta () and represent the highest level of discomfort for the agent. This test case can be simulated by moving to the **TEST III - Extended Model** directory from the project folder and running the **reactiveagent.bat** file there.

|  |  |
| --- | --- |
| 0 | 1 |
| 2 | 3 |

**Figure 5: Simulation Steps - Test III**

What is rather interesting about this test case is that regardless of the number of uncomfortable cells around the agent, it always finds its way to become perfectly comfortable three steps after the initial system state. This is represented by all agents taking the light green color at step 3. The input values were generated using a pseudorandom number generator in C++. In our experience, different sets of random values generate widely varying behavior but always following the rules of the ReactiveAgent model.

***This test case simulated five agents trying to reach the most comfortable cell using various rules from our ruleset.***

***In all test cases, the behavior related to the default rule can be seen once an agent has reached a state where it feels the most comfortable among all of its neighbors, and thus, it maintains that state.***

## **Video Recordings**

Video recordings of each of the test cases can be found in .mp4 format in the project directory. To view each of the test cases, please run the corresponding .mp4 file named after the test.

* Test Case I : run **TEST I Recording.mp4**
* Test Case II : run **TEST II Recording.mp4**
* Test Case III : run **TEST III Recording.mp4**

# **Possible Extensions**

# In the current implementation, an agent reacts based on the concrete and unchanging values of its neighboring cells. But this also means that if the agent finds that the current position is the most comfortable, it will never move away from that cell. One great extension to this model could be the introduction of dynamic inputs that randomly change the temperature of each terrain at each time step. This will force the agent to move to the most comfortable cell based on the dynamic values at each time step and should result in a much more realistic and interesting behavior.

# Another possible extension could be introducing prioritized objectives for the agent. In the current implementation, due to the right-moving tendency of the agent, sometimes the agent moves back and forth between two cells when those are of the same value, and every other neighboring cell provides a higher level of discomfort. If we were to model the system in a way where the agent prioritizes moving to an objective a certain percentage of the time (e.g., 75%), this might lead to some fascinating behavior. Agents may now have to move away from their current comfortable cell and go through uncomfortable cells to exhibit their prioritized objective-focused behavior and vice-versa.

# **Conclusion**

To conclude, we model the behavior of a flexible agent that reacts to environmental changes or important events within a system by moving towards cells where they feel the most comfortable. We use the extended Moore neighborhood and the CD++ simulator to model such behavior of a human agent in four different types of terrain. A set of nine rules represent the behavior of the agent, and the first two test cases are used to verify such behaviors. Finally, the third test case exhibits the scalability of the model, and possible extensions to the model are discussed as future work.

# **References**

1. Sweetser, P., & Wiles, J. (2005, July). Combining influence maps and cellular automata for reactive game agents. In *International Conference on Intelligent Data Engineering and Automated Learning* (pp. 524-531). Springer, Berlin, Heidelberg.
2. Wainer, G. A. (2017). *Discrete-event modeling and simulation: a practitioner's approach*. CRC press.