Proceedings of the 2023 Winter Simulation Conference
C. G. Corlu, S. R. Hunter, H. Lam, B. S. Onggo, J. Shortle, and B. Biller, eds.

MODELING REACTIVE GAME AGENTS USING THE CELL-DEVS MODELING FORMALISM

Alvi Jawad
Cristina Ruiz Martin
Gabriel Wainer

Department of Systems and Computer Engineering
Carleton University
1125 Colonel By Drive.
Ottawa, ON, K1S 5B6, CANADA

ABSTRACT

Intelligent game agents are vital to modern games as they add life, story, and immersion to the game environment. The requests in the gaming industry for more realism have made intelligent agents more important than ever. Modeling and simulation of game agents and their surrounding environment provide an alternate setting to study dynamic agent behavior before integration into the game engine. The Cell-DEVS formalism, an extension of Cellular Automata, allows modeling such behaviors using the rigorously formalized Discrete Event Systems Specification (DEVS) formalism. This paper explains how to model and test reactive game agents using the Cell-DEVS formalism and the CD++ toolkit. To analyze the dynamic behavior of such agents, we perform several experiments in varying system configurations. Our experimental results confirm the versatility of Cell-DEVS and the functionalities in the CD++ toolkit to model comfort-driven, exploratory, and desire-driven game agents.

1 INTRODUCTION

Game agents are indispensable entities that allow games to add lifelike immersion inside a virtual environment. Intelligent agents are the fundamental building blocks of any game, as the level of intelligent behavior determines the amount of realism the game can provide to the player. Many games, particularly those that make extensive use of autonomous agents, have grown increasingly complex over time. Consequently, the need to improve the agent behavior to match the entities they mimic from the real world has become necessary. While many games have aged well over the years, others have lost their popularity and player base due to not keeping up with this increased demand for realism (Sweetser, Johnson, Sweetser, and Wiles, 2003).

The virtual nature of game environments turns actual experiments to study agent behavior into an impossible task. This, in turn, makes modeling and simulation a prime candidate to experiment on and improve such behavior. Game agents have been modeled using many different mathematical formalisms, including game theory and finite automata (Carmel and Markovitch 1996), cellular automata (CA) and influence maps (Sweetser and Wiles, 2005), reactive planners (Weber et al. 2011), and reinforcement learning (Vinyals et al. 2019), among others. Mathematical models allow the use of formal methods to verify the model correctness and easy alternation of parameters to see variations in agent behaviors. Additionally, this allows formally defined models to be easily translated to a different mathematical formalism and allows continuous improvements to the model.

A system-theoretic modeling formalism, named Discrete Event Systems Specification (DEVS) (Gon, Zeigler, and Praehofer, 2000), has been widely applied for modeling and simulating artificial systems.

DEVS allows building single system components as modular atomic models and connecting multiple atomic and/or coupled models hierarchically to define complex models. An extension of DEVS, called Cell-DEVS, allows asynchronous modeling of timed cellular automata models (Wainer and Giambiasi, 2001). The content-sharing and interoperability advantages of Cell-DEVS have proven that serious games can be enhanced from the rigorous formalization and standardization provided by developing the models using DEVS or its extensions (Wainer, Liu, Dalle, and Zeigler, 2010).

Our objective is to demonstrate the use of Cell-DEVS and the associated CD++ toolkit (Wainer, 2002; Lopez and Wainer, 2004) to model and design intelligent game agents that react to changing game environments under varying system configurations to gain insight into developing highly intelligent game agents. More specifically, our contributions are as follows: (1) Definition of a cellular automata model of reactive game agents in the Cell-DEVS modeling formalism; (2) Simulation of a multi-terrain game system and observation of the comfort-driven, exploratory, and desire-driven behavior of reactive game agents under varying system configurations; (3) Analysis of the conflicting interplay between desire and comfort for intelligent decision-making.

The results are useful to understand complex, and often contradictory, behavior of game agents when multiple incentives in a game terrain are present as potential objectives. In turn, this knowledge can inform analyzing flexible decision-making for intelligent game agents in intricate game environments.

The rest of the paper is organized as follows. Section 2 presents the background and the modeling formalism used for this research. Section 3 provides a formal description of the reactive agents' model. Section 4 details simulations performed using the developed model with various agent behavior. Section 5 presents the conclusions and directions for future research.

2 BACKGROUND

Agents are a crucial part of game worlds. They represent individual non-player characters (e.g., villagers, monsters, allied soldiers, enemies) with distinct personalities and behavior that give the game life, story, and atmosphere. In many cases, agents are hard-coded based on the previous knowledge of the designer but not in the current state of the environment (Sweetser and Wiles, 2005). Although some agents may seem intelligent, as in *Half-Life and Thief: The Dark Project*, they are still hardcoded and look into a database of rules. Other techniques, such as "Smart Terrain" are used in *The Sims*, where intelligence is embedded in the environment and agent behavior is emergent based on the needs and the environment. Each agent has various motivations, and objects in the environment provide information on how they satisfy those needs.

This sets the grounds for the concept of reactive game agents, which are flexible living entities that can react to changes in their surrounding environment. In (Sweetser and Wiles, 2005), the authors elaborate on this concept by designing the agents by combining influence maps with cellular automata to represent a 3D game world named EmerGEnT. The game models natural phenomena such as fluid flow, heat, fire, pressure, and explosions. The agents in EmerGEnT are able to react and move based on their comfort and desire levels with various reaction times and exhibit different levels of intelligent behavior. Other works on intelligent game agent behavior explore game-theoretic approaches to build adaptive agents that can infer opponent strategies (Carmel and Markovitch 1996), real-time execution and concurrent goal pursuit using reactive planners (Weber, Mateas, and Jhala, 2011), and multi-agent reinforcement learning algorithms that can allow game agents to rival human player at a Grandmaster level in complex real-time strategy games such as StarCraft II (Vinyals et al. 2019).

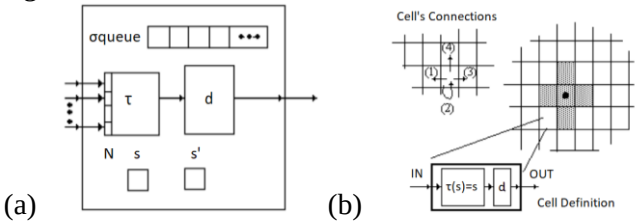
In this work, we elaborate on (Sweetser and Wiles, 2005) by introducing Cell-DEVS to formally model, design and analyze the comfort-driven, exploratory, and desire-driven behavior of reactive game agents.

2.1 The Cell-DEVS Formalism

The Cell-DEVS formalism (Wainer and Giambiasi, 2001) is an extension of DEVS that defines systems as cellular models using the same semantics used in the DEVS formalism (Gon, Zeigler, and Praehofer, 2000). Each cell in the cell space is specified as a DEVS atomic model, as shown in Figure 1(a). A cell takes N

inputs from all cells in a specified cell neighborhood through its interface. Inputs activate the local computing function (τ) that, after a predefined delay (d), produces a state change (s′). This changed state may then be transmitted to other models through the cell interface. Delays belong to one of two types: transport and inertial delays. Transport delays model variable time delays and may schedule outputs in a queue. In the case of inertial delays, the output is preempted, meaning a newly computed value can overwrite all previously scheduled events in case of a state change.

Figure 1(b) shows a coupled Cell-DEVS model built after defining the connections between an atomic model and its neighbors. More than one atomic model can be defined to compose a coupled Cell-DEVS model, where each atomic model is connected to its neighborhood through input and output ports, similar to that of DEVS coupled models. The entire cell space is an R x C grid, where R denotes the number of rows, and C denotes the number of columns. The cell space can also have more than two dimensions. The cell space can be wrapped or non-wrapped (i.e., different behavior defined for border cells), resulting in a uniform and non-uniform neighborhood.



(a) (b)

Figure 1: Cell-DEVS model semantics. (a) Atomic model; (b) Coupled model

2.2 The CD++ Toolkit

CD++ (Wainer, 2002; Lopez and Wainer, 2004) is a modeling and simulation environment based on the C++ programming language. In CD++, Cell-DEVS models are implemented using a custom specification language. The CD++ specification includes (1) the size and the dimensions of the cell space, (2) the shape of the neighborhood, (3) the number and initial values of state variables, (4) the definition of neighbor ports, (optional), and (5) any rules for cells and border cells' behavior.

A cell's local computing function is defined using a set of rules in the following format:

```
{postconditions} {assignations (optional)} {delay} {preconditions}
```

If the `preconditions` defined for a particular rule are satisfied, after the specified `delay` for the rule, `assignations` to state variables will be made (optional), and then the `postconditions` will be evaluated, usually resulting in one or more port assignations so the current state of the cell, as well as any output messages, can be sent via the defined neighbor ports.

New rules are evaluated sequentially after the failure of the previous one. If no rules apply for a cell in the cell space, or more than one rule applies, an error will be produced indicating the conflicting behavior of cells. This simple syntax allows one to define both simple and complex behavior of cells, giving them the flexibility to model increasingly complicated hierarchical cellular models.

3 THE REACTIVE AGENT MODEL

To demonstrate the use of Cell-DEVS and the CD++ toolkit to model, design, and analyze reactive agents, we define a game environment as 10 x10 cell space. Each cell in the game environment represents a terrain of a particular type with different levels of fire, wind, and water attributes (Table 1), resulting in a corresponding value of heat, pressure, and wetness for an agent if they stand on that cell.

Based on the values of heat, pressure, and wind, we divide the terrains into three different categories: (1) fire-attribute terrains (volcano, lava, heated), (2) wind-attribute terrains (tornado, gust, breeze), and (3) water-attribute terrains (river, storm, drizzle). The nine different terrain types and associated heat, pressure, and wind values are represented in Table 1. The values used for each terrain are fictional and are assigned to see different agent behavior in different terrains. In addition to these terrain types, each cell has specific properties that lead to that cell having certain desirability for an agent. It is important to note that the primary

contribution of this work is not the simplified game design but the method to design the agents' behavior and how they are implemented in CD++.

Table 1: Cell values for different terrain types.

	Terrain	Symbol	Cell type	Heat	Pressure	Wetness
Fire Attribute Terrains	Volcano	v	0.001	100	50	0
	Lava	l	0.002	80	30	0
	Heated	h	0.003	40	20	20
Wind Attribute Terrains	Tornado	t	0.004	0	100	40
	Gust	g	0.005	0	60	20
	Breeze	b	0.006	0	30	10
Water Attribute Terrains	River	r	0.007	0	20	100
	Storm	s	0.008	20	60	70
	Drizzle	d	0.009	0	10	40

Figure 2 shows the game's terrain that we use to analyze the reactive agents' emergent behavior. Each cell contains the cell coordinates and the type of cell as specified in Table 1. Fire-Attribute terrains are represented in various shades of red. Similarly, Wind-Attribute and Water-Attribute terrains are in shades of yellow and blue, respectively. To be consistent, the same terrain will be used in all the experiments.

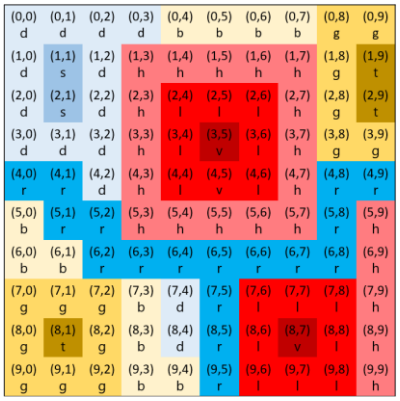


Figure 2: Game terrain.

We design two types of reactive agents: (1) Comfort-Driven Agents and (2) Desire-Driven Agents.

Comfort-Driven Agents are driven by their level of comfort based on the attributes of each cell. For example, human agents may feel the most uncomfortable if they stand on a fire-attribute cell but deal well with wind and water-attribute cells. On the other hand, a fiery monster may feel perfectly fine on a fire-attribute cell but might die instantly if forced to stand on a water-attribute cell. We assign weights to heat, pressure, and wetness, allowing us to easily alter the agents' comfort preferences and model many different agent types. Each agent, while standing atop a particular cell, determines their current level of comfort and the prospective comfort if they were to move to a particular cell. If they find a nearby, more comfortable cell than their current cell, they move to that cell. Agents also have a right-moving tendency; if all cells are equally comfortable, agents prefer moving to the right compared to staying in the current cell, etc.

Desire-Driven Agents prefer moving to cells with the highest desirability for them as long as the movement is not too uncomfortable. The desirability of cells may change after a certain time, and the agents should react accordingly to those changes.

All reactive agents considered in this work (see Table 2) can react to their environment based on comfort only (comfort-driven agents), desire only (desire-driven agents), or a combination of both. With this method, we can observe the interplay between desire and comfort and see what happens when both behaviors are equally important or when one dominates the other.

3.1 Reactive Agents Formal Specification

This section provides the formal specification for the Reactive Agents model.

3.1.1 Neighborhood Selection

We selected the extended Moore neighborhood to use in our model. This choice allows us to model the agent's movement in all eight cardinal (North, South, East, West) and ordinal directions (Northeast, Northwest, Southeast, Southwest), as shown in Figure 3. The cell marked in green and yellow represents the agent and its immediate neighborhood, respectively, whereas the grey cells represent the extended neighborhood. This second level allows us to implement more complex behaviors and avoid conflict with any other agent competing for the same cell from all straight and diagonal directions.

(-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 3: Cell neighborhood.

3.1.2 Formal Specification

The ReactiveAgents atomic model is formally specified as follows:

RA_Atomic = < *X; Y; S; N; type; d;* τ*;* δ_{int}*;* δ_{ext}*;* λ*; ta* >

X = Y = {Ø}
S = {agent, cell, heat, pressure, wetness, desirability} as detailed in section 3.1.3
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 = transport;
d = 100; // in milliseconds
τ: N → S; as defined by the rules in section 3.1.5
δ_{int}; δ_{ext}; λ; and ta are defined using the Cell-DEVS specification, and the definition is not needed by the user.

The ReactiveAgents coupled model is formally specified as follows:

RA_Coupled = < *Xlist; Y list; X; Y; I;* η*; N; {r,c}; C; B; Z; select* >

Xlist = Ylist = {Ø}
X = Y = {occ, celltype, comfort, desire} as detailed in section 3.1.4
I = < P_x; P_y >
η = 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)}
C = {Cij / i ∈ [0; 9]; j ∈ [0; 9]}
B = Ø; // wrapped border
Z = the translation function as defined in the formalism

3.1.3 State Variables

Each cell is defined based on six state variables:

`cell`: defines the terrain type specified in Table 1: {0.001, 0.002, 0.003, ... 0.009}

`agent`: the type of agent in each cell. The agent state can take four values: 0 (i.e., no agent), 1 (i.e., human agent - afraid of fire), 2 (i.e., Fiery Monster agent - afraid of water), and 3 (i.e., Vapor Alien agent - afraid of pressure).

`heat`: defines the amount of fire heat in each cell: {0, 1, 2, ... 100}

`pressure`: defines the amount of wind pressure in each cell: {0, 1, 2, ... 100}

`wetness`: defines the amount of water wetness in each cell: {0, 1, 2, ... 100}

`desirability`: defines the desirability of a cell to an agent from 0 (no desirability) to 0.99 (the maximum desirability). Specific levels of desirability is defined as a real number $\in$ [0, 0.99]

All cells are initialized with certain heat, pressure, and wetness levels. The values assigned are based on the type of terrain based on the values specified in Table 1. The desirability of the cell is also initialized.

The agent cell is also initialized based on the number of agents at the start of the simulation and their locations.

3.1.4 Neighbor Ports

We define four ports:

`occ`: depending on the type of simulation being performed, outputs the value of either the agent variable or a sum of the agent variable and another additional value. This is the primary port that will be used to visualize the simulations.

`celltype`: outputs the value of the cell state variable.

`desire`: outputs the value of the desire state variable.

`comfort`: outputs the value after calculating the comfort of a certain agent using the heat, pressure, and wetness state variables of the level 1 Moore neighborhood, and the assigned weights to each of them. The comfort level (CL) of an agent is defined as:

```
CL=heat*W_H + pressure*W_P + wetness*W_W
```

W_H, W_P, and W_W are the weight of the discomfort felt by the agent because of heat, pressure, and wetness, respectively. Table 2 represents the agents and weights assigned to W_H, W_P, and W_W. Note that these values can be modified to simulate other types of agents.

Table 2: Types of Agents and associated element weights.

Agent ID	Agent Type	W_H	W_P	W_W
Primary Agents				
Agent01	Human	0.008	0.002	0.001
Agent02	Fiery Monster	0.0	0.001	0.009
Agent03	Vapor Alien	0.0001	0.0098	0.0001
Secondary Agents				
Agent04	N/A	0.005	0.005	0
Agent05	N/A	0	0.005	0.005
Agent06	N/A	0.005	0	0.005
Agent07	N/A	0.0033	0.0033	0.0033

3.1.5 Rules

We define the behavior of reactive agents based on the following set of rules:

The Insta-death Rule: If an agent has reached or exceeded the extreme discomfort level (in our case, defined as 80% CL) using the equation presented in section 3.1.4, that agent dies, and the value of the `agent` state variable changes to zero. An excellent example of this would be human agents on volcanoes, where the comfort level of volcano cells for human agents is exactly 80%. This rule is the first to be evaluated. If the agent dies, the rest of the rules are ignored.

The Movement Rules: These determine the agents' movement upon confirming the comfort values of neighboring cells or desire based on their type of behavior. These rules also contain priority between agents when two agents of different types try to occupy the same cell. If there is a clear winner among all neighboring cells, the agent will move to that cell, becoming more comfortable than before or being in a more desirable cell based on the agents' behavior. In case of a tie, agents always try to move toward the right if possible.

The Default Rule: This rule is evaluated when all other rules fail to satisfy and therefore has the lowest priority. If none of the other rules apply, the cell state of the cell remains the same.

4 SIMULATION RESULTS AND ANALYSIS

In our experiments, 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. They have a right-moving behavior (i.e., if two cells have the same comfort, desire, etc., they decide to move to the right). Note that more complex behaviors can also be implemented. We performed several experiments. For visualization purposes, dark green represents agents in the uncomfortable state and light green represents agents in the comfortable state.

4.1 Simulating Single Comfort-driven Agents

In the first experiment, we run simulations where all agents are Comfort-driven agents of the same type. The initial configuration is shown in Figure 4, with the ten agents represented in dark green.

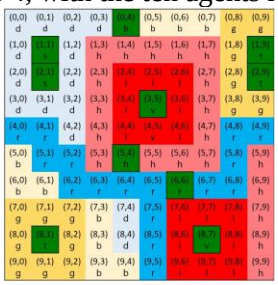


Figure 4: Initial configuration.

Figure 5 shows a complete run of this experiment using ten human agents until the simulation reaches a steady state. We can see the game terrain from a human agent's point of view (POV) without and with agents in the first two snippets. In Step 1, the two agents previously standing on volcano terrains die immediately, and the number of agents goes down to eight. Simultaneously, agents in the middle move to the river, and agents to the top-left move to areas with drizzle to become more comfortable, as water is the most preferred element for human agents. In subsequent steps, we can see both agents in the middle and to the top-left becoming comfortable (less than 10% discomfort). One agent moves through the river until it finds a drizzle terrain on the other side of the wrapped boundary and becomes comfortable by moving to it.

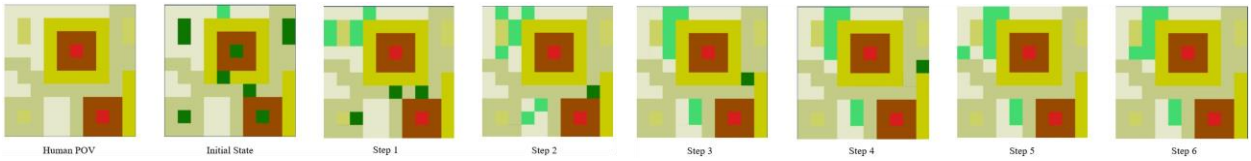


Figure 5: Full simulation run with ten comfort-driven human agents.

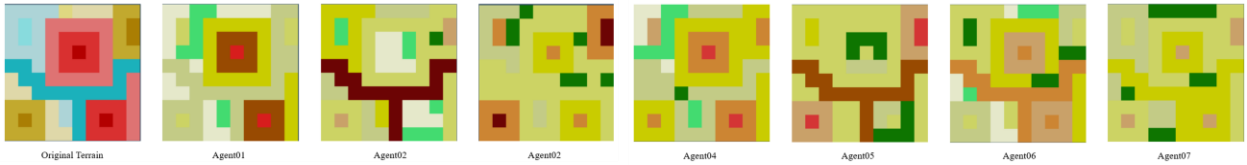


Figure 6: Final state of a complete simulation run with ten agents behaving as comfort-driven agents.

For all the other agents, we show the original terrain and final state of the simulation (Figure 6). In each image, the terrain view corresponds to that agent's comfort level in each cell, and the original terrain is also provided for easier comparison.

For human agents, (Agent01), drizzle terrains seem to be the most comfortable location. Eight agents out of ten survive, and 100% of the surviving agents become comfortable.

Interestingly, fiery monster agents (Agent02) prefer lava cells over heated cells due to their aversion to water present in such cells. Their high fire tolerance makes both volcano and lava cells comfortable, with lava cells being slightly more comfortable. The agent whose initial position is the river dies at the beginning, and out of the remaining nine, eight end up in a comfortable state.

For vapor aliens (Agent03), tornado cells are deadly, causing the fatality of three agents at the beginning. There are no cells where these agents can feel comfortable; hence, all seven remaining aliens stay uncomfortable until the simulation ends.

Agent04 shows very similar behavior to human agents and prefers drizzle cells over others. However, since no strong discomfort is felt due to any element, no agents die at the beginning, and nine out of ten agents become comfortable. Agent05 ends up occupying lava cells exclusively due to their low pressure and wetness value. Although all agents survive, none of them are able to reach a comfortable state.

The preference for Agent06 is all over the place, but they only become comfortable when occupying breeze cells. All ten agents survive, and 40% of them end up being comfortable. Agent07 does not feel particularly uncomfortable in any of the cells, as seen from the high density of yellow cells in the terrain. However, this also means that the reverse is true, and there is not a single cell where these agents are able to become comfortable. As such, all surviving ten agents remain uncomfortable.

Overall, this experiment allowed us to see some interesting behavior from different agent types and the flexibility to model different agents in varying simulation terrains.

4.2 Simulating Multiple Comfort-driven Agents

When simulating multiple comfort-driven agents, we remove the death rule as we do not want agents to die. Instead, we are interested in observing how they interact when different agent types are placed together.

We performed two experiments with several units of multiple agent types placed together. Figure 7 presents 15 agents with five units of all three agent types scattered throughout the game terrain. Human, monster, and alien agents are depicted in green, dark blue, and magenta, respectively. The game terrain retains its original terrain colors, as it is impossible to show the comfort level of all three agent types simultaneously. For conciseness, we only show the initial, final, and first two steps of each experiment.

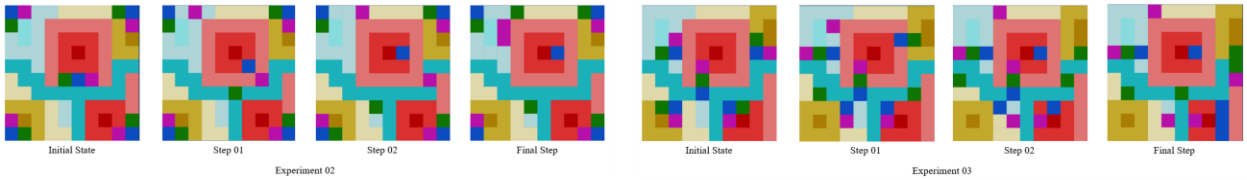


Figure 7: Experiments with multiple types of comfort-driven reactive agents.

In experiment 02, five trios of all three agent types are placed in the four corners and the middle of the map. During different steps, although the agents in the middle start moving based on their comfort in certain cells, the agents in the corners barely shift their position. Eight out of ten corner agents retain their starting position at the end. This is because all corners are connected due to our wrapped border, and the agents are negating the directional movement of each other due to being too close together.

In experiment 03, five trios of all three agent types are placed again on the map. However, the difference this time is in the placement change of the corner agents, allowing them more room to move around. This results in improved behavior in the directional movements, and all but two agents end up moving to different positions than their starting position.

These experiments allowed us to observe the reactive movement of multiple agent types simulated simultaneously. While the model can be easily scaled to include as many agent types as we want, careful

consideration must be placed on the number of rules and the priorities, as the inclusion of a single agent type involves adding, at a minimum, sixteen directional rules, and redefinition of the priorities.

4.3 Simulating Reactive Explorers

In many games, the game terrain is not immediately visible to the agents unless they can access a map. Many real-time strategy games (e.g., Age of Empires franchise, Warcraft) start with unexplored maps at the beginning. Only those portions of the terrain with buildings or agents under the player's control are visible to the player. Unexplored areas can be explored by moving an explorer agent (e.g., a scout) through those areas, and the revealed area depends on the visual capability of the explorer agent.

To add the area exploration capability to our agents, we replace our Insta-death rule with the *Area Exploration* rule. This rule states that every time an agent occupies or moves to a cell, it reveals all unexplored cells in its immediate neighbor. To deal with the unexplored state of a cell, we add a new state variable `visibility` that keeps track of explored and unexplored cells. All cells in the map start as unexplored (visibility set to 0) and become visible when explored by a neighboring agent (visibility set to 1). A new neighbor port `explored` outputs the visibility status of a cell to its neighbors.

Figure 8 shows the reactive exploratory behavior of one single human agent. Gray cells represent unexplored areas and turn into cells matching those of our original game terrain as they are explored. The agent is placed at the cell with coordinates (5,5) at the beginning, and the explored neighborhood around the cell can be seen clearly. The agent explored all areas that could be comfortably explored and ended up in a comfortable position.

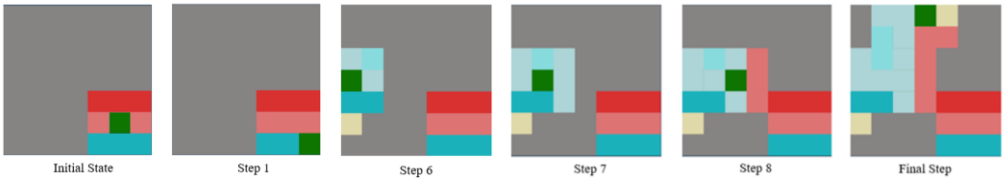


Figure 8: Exploratory behavior of a single human agent

Figure 9 shows the initial state and the simulation results of the nine human agents placed in the game terrain; four agents occupy the four corners, four others occupy the cell in the middle of the borders, and one additional agent occupies a cell in the middle.

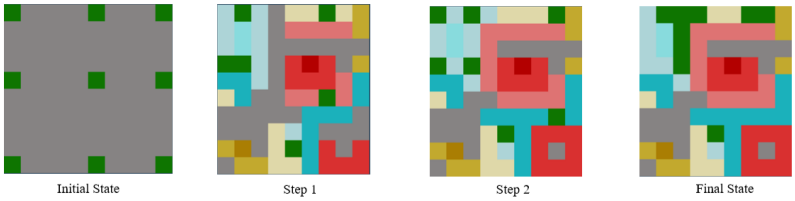


Figure 9: Muti-agent exploratory behavior with nine human agents.

In this experiment, 81% of the cells become visible after all agents finish their exploration and end up flocking to the top-left drizzle cells of the map.

These experiments allowed us to see the interesting exploratory behavior of both single and multiple agents placed in the game terrain and observe the relationship between agents facing changes during movement and the area explored. Note that other exploratory or viewing behaviors can be implemented.

4.4 The Desire-Driven Agent Model

The main objective of games is to achieve a mission. Based on this mission, agents might display entirely different behavior. The desire to achieve the mission has a clear contention with how comfortable the agent feels while moving towards their goal. In this section, we investigate the interplay between desire and comfort and how agents with different levels of these parameters react.

We update our model to add the capability of agents to make decisions based on both desire and comfort. An additional state variable called `inclination` now considers both the comfort level and desirability (calculated as explained in section 3).

The `inclination` is calculated as follows:

```
I = CL * W_C + D * W_D
```

W_C and W_D represent the weight of comfort and desire for the specific type of agent being modeled.

Based on this idea of weights, table 3 presents the five different types of agents that we experiment on. All experiments will be performed using a human agent with the comfort level as defined in Table 2. All experiments done until now have been done with purely reactive agents (R-01) that do not respond to the desirability of cells, and as such, we do not show results for these agents any further.

Table 3: preference percentage of comfort and desire for each agent.

Agent ID	Agent Type	Comfort Preference	Desirability Preference
R-01	Purely Reactive	100%	0%
R-02	Highly Reactive	75%	25%
R-03	Evenly Weighted	50%	50%
R-04	Highly Goal-directed	25%	75%
R-05	Purely Goal-directed	0%	100%

We have kept the *Insta-death rule* as we want to analyze whether, and at what level of desire, agents are willing to give their lives to reach their goal. All movement rules have been updated to include the preference port, and agents now react and make decisions based on their preference for a certain cell.

We perform an experiment where we try to see the behavior change in human agents as their preference changes due to the increasing desirability of cells over time. The results are shown in Figure 10. The game terrain shows the preference from a human agent's POV and uses the same color profiles used along the paper to depict the comfort level. This experiment has three *objective cells* with maximum desirability with a position identical to the initial destinations shown at the top of Figure 10 in purple. Over time, desire from objective cells spreads through the cells to the left, which can be seen by those cells taking lower (more desirable) preference values. The two objective cells at the right border require agents to walk over volcano cells (V) to reach them. This will allow us to understand the forces between desirability versus comfort.

Type R-01 Agents react purely based on their comfort (100%), so the spread of desirability does not affect them. The results from these agents are the same as in Figure 5, and thus, not shown in Figure 10.

Type R-02 Agents are highly reactive (75% comfort, 25% desire) and produce results similar to Type R-01 agents. The two agents standing on volcano cells die at the beginning, and most agents end up flocking to the top-left drizzle cells. The big difference is seen in agents close to the highly desirable objective cells. One agent in the middle does not move into the river but prefers to stay on the objective cell (a heated cell and therefore less comfortable) due to the high preference resulting from high desire values. Another agent that started in the river, instead of moving through the river, waits for the objective cells to be unoccupied to move to that. This, however, never happens because the objective cell never becomes unoccupied. One other agent at the top-right stays close to the highly desirable target. In contrast, another agent at the bottom travels halfway from its initial position to the bottom-right objective. The agent that stopped halfway was due to the insufficient desire of the next right cell for it to move from the drizzle cells to river cells.

Type R-03 Agents have the same comfort and desire values (50% for both) and end up in a final state very similar to that of Type R-02 Agents. One slight difference is the agent at the bottom being able to move one step to the right due to the increased percentage of desire, making that river cell the preferred destination for itself.

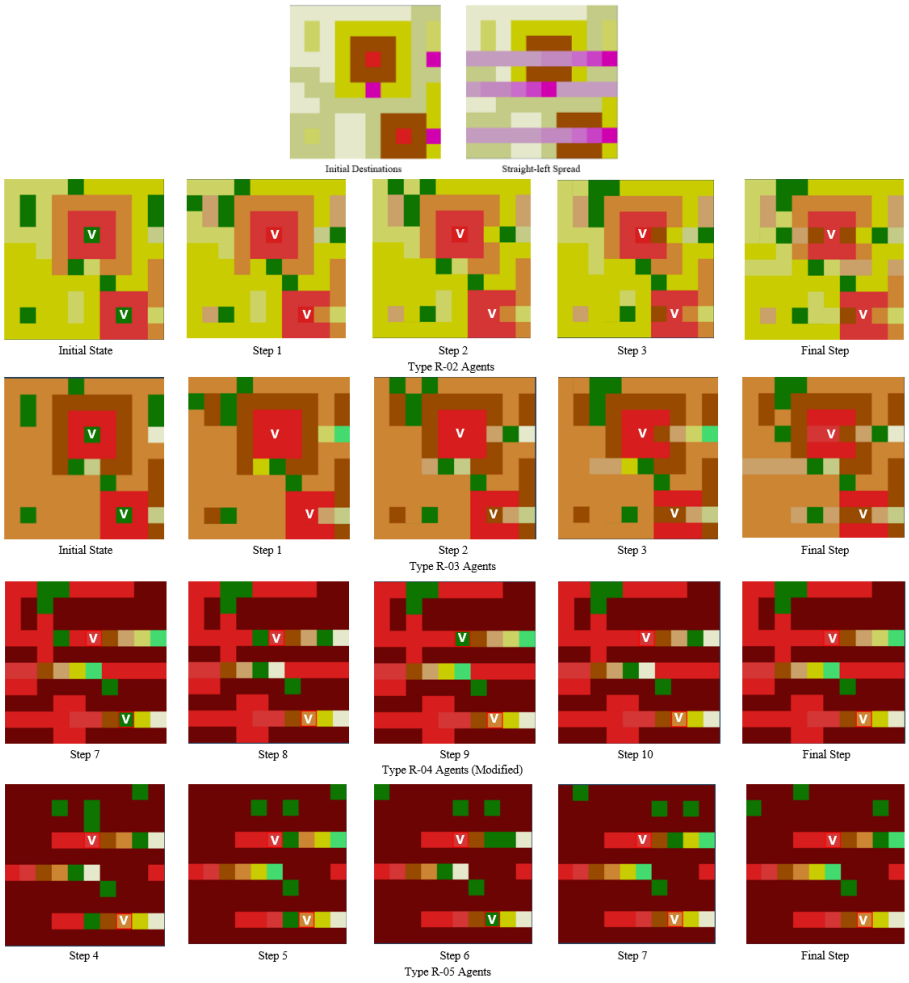


Figure 10: Desire-driven human agent behavior

Type R-04 Agents are highly goal-driven agents (25% comfort, 75% desire) and produce the same final state as the previous experiment. Therefore, we slightly modify these agents to have a higher desire (10% comfort, 90% desire), which shows a drastic change in emergent behavior in agents. The agent at the bottom, due to its very high desire to reach the bottom target, steps over the volcano cell at step 7, while another agent at the top-left, until now stuck right before the fire-attribute cells for all experiments, takes its first step towards the top objective. The bottom agent dies at step 8, whereas the to-left agent follows the same tragic fate by stepping into the volcano cell at step 9. Only six agents are left in the final state.

Type R-05 Agents are driven purely by desire (100%), so their movement does not consider the terrain type at all. This can be seen by the top three agents, previously stuck at drizzle cells, now moving directly right, ignoring everything in their path. The bottom agent dies one step earlier at step 6. One interesting thing happens in the case of the previously dead top-left agent. When it tries to move to the path with higher desirability (step 4 to step 5), it moves past the volcano cell as the lava cell has much higher desirability due to the reduced propagation value, which allows it to survive. This is an interesting emerging behavior as it had to die when it cared about comfort and survived when it did not care about comfort at all. In the end, seven agents out of the starting ten survive.

This series of experiments confirmed that agents are willing to die for their cause if the desire to reach their objective is adequately high. Further experiments can be performed to better understand the emergent behavior seen for different combinations of desire and comfort.

5 CONCLUSIONS

In this paper, we developed a model of reactive game agents in the Cell-DEVS modeling formalism, where agents can react to environmental changes based on their comfort and desire. To verify model correctness, we performed several experiments demonstrating the comfort-driven, exploratory, and desire-driven behavior of multiple agents in a fixed game terrain containing nine different terrain types. Cell-DEVS allows for rapid design, prototyping, and analysis of reactive agents using formal modeling and simulation.

Our experiments proved that agents could react to their surrounding environment by determining the best course of action based on where they felt more comfortable or which direction led them to more desirable goals. The exploratory behavior of agents led to agents revealing unexplored part of the game terrain based their comfort-driven movement. Moreover, we determined that agents with a highly desire-driven nature are willing to ignore their discomfort and walk to their death if that meant that they would reach their desired objective.

The experiments allowed us to dive deep into the dynamic behavior of different game agents and gain insight into modeling intelligent reactive agents. The study presented could help investigate various other phenomena by simulating game agents with additional capabilities and constraints in intricate game environments to discover hidden, unintended, or undesired interactions alongside flexible decision-making.

REFERENCES

Carmel, D., and S. Markovitch. 1996. “Learning models of intelligent agents”. In Proceedings of the thirteenth national conference on Artificial intelligence-Volume 1, 62–67

Gon, K. T., B. P. Zeigler, and H. Praehofer, “Theory of modeling and simulation: integrating discrete event and continuous complex dynamic systems,” 2000.

Lopez, A, and G. Wainer, “Improved cell-DEVS model definition in cd++,” in International Conference on Cellular Automata, pp. 803–812, Springer, 2004.

Sweetser, P and J. Wiles, “Combining influence maps and cellular automata for reactive game agents,” in International Conference on Intelligent Data Engineering and Automated Learning, pp. 524–531, Springer, 2005.

Sweetser, P., Johnson, D., Sweetser, J. and Wiles, J.: Creating Engaging Artificial Characters for Games. Proceedings of the Second International Conference on Entertainment Computing. Carnegie Mellon University, Pittsburgh, PA (2003) 1-8

Vinyals, O., I. Babuschkin, W. M. Czarnecki, M. Mathieu, A. Dudzik, J. Chung, D. H. Choi, R. Powell, T. Ewalds, P. Georgiev et al. 2019. “Grandmaster level in StarCraft II using multi-agent reinforcement learning”. Nature 575(7782): 350–354.

Wainer G., “Cd++: a toolkit to define discrete-event models,” Software, Practice and Experience, vol. 32, pp. 1261–1306, 2002.

Wainer G. and N. Giambiasi, “Application of the cell-DEVS paradigm for cell spaces modelling and simulation,” Simulation, vol. 76, no. 1, pp. 22–39, 2001.

Wainer G, Q. Liu, O. Dalle, and B. P. Zeigler, “Applying cellular automata and DEVS methodologies to digital games: A survey,” Simulation & Gaming, vol. 41, no. 6, pp. 796–823, 2010.

Weber, B. G., M. Mateas, and A. Jhala. 2011. “Building human-level ai for real-time strategy games”. In 2011 AAAI Fall Symposium Series.

AUTHOR BIOGRAPHIES

ALVI JAWAD is a Ph.D. student in the Department of Systems and Computer Engineering at Carleton University (Ottawa, ON, Canada). He received his M.A.Sc. degree in Electrical and Computer Engineering in 2021 from Carleton University. His research interests involve intelligent agents and using formal model-based approaches to assess the cybersecurity of critical infrastructures. His email address is alvi.jawad@carleton.ca, and his website is https://carleton.ca/cybersea/people/alvi-jawad.

CRISTINA RUIZ MARTIN is an Instructor at the Department of Systems and Computer Engineering at Carleton University (Ottawa, ON, Canada). She received the DEVS Modeling and Simulation Ph.D. Dissertation Award from SCS (2019) and the Young Simulation Scientist Award from SCS (2020). She has been a member of the Board of Directors of SCS since July 2021. Her email address is cristinaruizmartin@sce.carleton.ca, and her website is https://carleton.ca/sce/people/ruiz/

GABRIEL WAINER is a Professor at the Department of Systems and Computer Engineering at Carleton University (Ottawa, ON, Canada). He is the author of three books, over 320 research articles, and the editor of four other books. He is the head of the Advanced Real-Time Simulation lab at Carleton University. He has received various awards, including the IBM Eclipse Innovation Award, SCS Leadership Award, and Best Paper awards. He is a Fellow of SCS and a member of the Board of Directors of SCS. His email address is gwainer@sce.carleton.ca, and his website is http://www.sce.carleton.ca/faculty/wainer.