SYSC 5104- Methodologies for Discrete-Event Modeling and Simulation - Fall 2004

Modeling Battlefield Using Cell-DEVS

Rami T. M. Madhoun (100667489)

Introduction

The objective of this work is to model the behavior and interaction between fighters in a land battlefield. The basic ideas of the model are based on the work of **Lim Yew Kia, Jeffrey (**http://staff.science.nus.edu/~parwani/project1/jeff.html) where they described different factors that affect the fighter behavior in a battlefield. However, some assumptions and simplifications were made to limit this model within the required scope. The following points elaborate on the assumptions made about the battlefield:

- The fighter can be in any of the following states: *Alive, Injured, Dead*.
- The battlefield is two dimensional, i.e. we are only concerned with the land part of the fight excluding any missile or airplane engagement.
- The situation awareness of the fighter is limited to his immediate neighborhood excluding any advanced telecommunication equipment used.
- The battlefield contains two armies; each is trying to attack the other's flag.

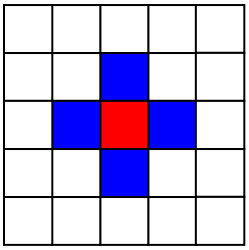

Figure 1: A red fighter surrounded by blue fighters (Van Neumann neighborhood)

State-Change Rules:

- If a fighter is in state *Alive*, and is attacked by one enemy, its state changes to *Injured*.
- If the fighter is in state *Alive*, and is attacked by two or more enemies, its state changes to *Dead*.
- If a fighter in state *Injured*, and is attacked by one or more enemies, its state changes to *Dead*.
- If a fighter in a state *Injured* is not attacked for a time T, its state changes to *Alive*.
- When the fighter becomes *Dead*, its cell becomes free and can be occupied by any other fighter.

Rules of Engagement:

- In general, the fighter tends to move towards the enemy's flag. His movement is directed by the coordinates of the enemy's flag position which is assumed to be provided to him.
- If the fighter is neighbored by one enemy, it engages in a fight. The result of this fight has a probabilistic nature since either one can be injured/ killed.
- If an enemy fighter became the neighbor of the flag and there is no friend fighter close by, the flag is taken.

Formal Specification of the Coupled Model

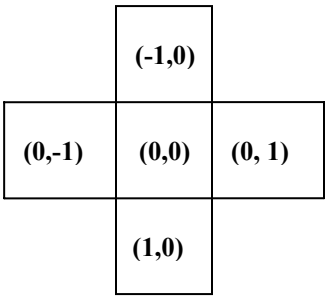

Figure 2: Neighborhood definition

M = < Xlist, Ylist, I, *X*, *Y*, η, N, {r, c}, C, B, Z, select >

Xlist = {Φ}
Ylist = {Φ}
I = <P^x, P^y> P^x={Φ} , P^y ={Φ}
X = Φ
Y = Φ
η = 5
N = {(0,-1), (0,0), (0,1),(-1,0),(1,0)}
r= 10
c = 10

S= {-2.-1.0,1, 2}
B = Φ (wrapped)
C = {C_{ij} / i ε [0,9], j ε [0,9] }

$P_{ij}^{Y1} \rightarrow P_{i,j+1}^{X1}$ $P_{i,j}^{X1} \leftarrow P_{i,j-1}^{Y1}$
$P_{ij}^{Y2} \rightarrow P_{i+1,j}^{X2}$ $P_{i-1,j}^{X2} \leftarrow P_{i-1,j}^{Y2}$
$P_{ij}^{Y3} \rightarrow P_{i,j-1}^{X3}$ $P_{i,j+1}^{X3} \leftarrow P_{i,j+1}^{Y3}$
$Pij^{Y4} \rightarrow P_{i-1,j}^{X4}$ $P_{i+1,j}^{X4} \leftarrow P_{i+1,j}^{Y4}$
$Pij^{Y5} \rightarrow P_{i,j}^{X5}$ $P_{i,j}^{X5} \leftarrow P_{ij}^{Y5}$

SELECT = {(0,-1),(-1,0),(0,0), (0,1), (1,0)}

Rule Implementation Details

In order to implement the previous model, different information need to be stored. This information can be described as follows:

- ## Fighter State

The state of the fighter whether he is dead, injured, or alive is stored in layer 0 as follows:

State	Description
2	Fighter of army A alive
1	Fighter of army A injured
0	Fighter is dead and cell is empty
-1	Fighter of army B injured
-2	Fighter of army B alive

Table 1: Layer 0 Information

- ## Fighting ability

When two or more fighters engage in a fight, the factor determining the outcome of this fight is the fighting ability factor for each one which is represented as a number ranging from 0.0 to 1.0 with 0.0 means that the soldier is dead and 1.0 full fighting ability. In addition, the fighter will be have some effect on the enemy only if his fighting ability is greater than 0.5.

The following table shows how the fighting ability is assigned to each fighter:

State	Fighting Ability
2	Uniformly distributed number between 0.45 → 1.0
1	Uniformly distributed number between 0.0 → 0.55
0	Fighter is dead and cell is empty 0.0
-1	Uniformly distributed number between 0.0 → 0.55
-2	Uniformly distributed number between 0.45 → 1.0

Table 2: Layer 1 Information

- ## Flag Position

In order to direct each fighter towards the enemy's flag, he needs to know the coordinates of the enemy's flag position. This information is stored in two layers (**layer 2** storing B's flag position, and **layer 3** storing A's flag position).

The following format is used to store both the row and column values in one number:

Row + Column/100 (ex. (row=2, column=4) → 2.04)

- **Moving Direction**

The moving direction layer contains the direction for each fighter in order to reach the enemy's flag. A unique number is assigned for each direction as follows:

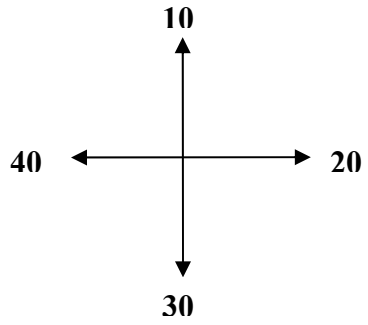

Figure 3: Direction codes

Direction	Value
North	10
East	20
South	30
West	40

Table 3: Layer 4 Information

- **Moving Factor**

This factor represents the fighter ability for the soldier in order to successfully move to a free cell. The values use the same format as the fighting ability factor in layer 1.

- **Fighting Rule Implementation**

The fighter will engage in a fight if he is surrounded by one or more enemy fighter. The following diagram explains this behavior:

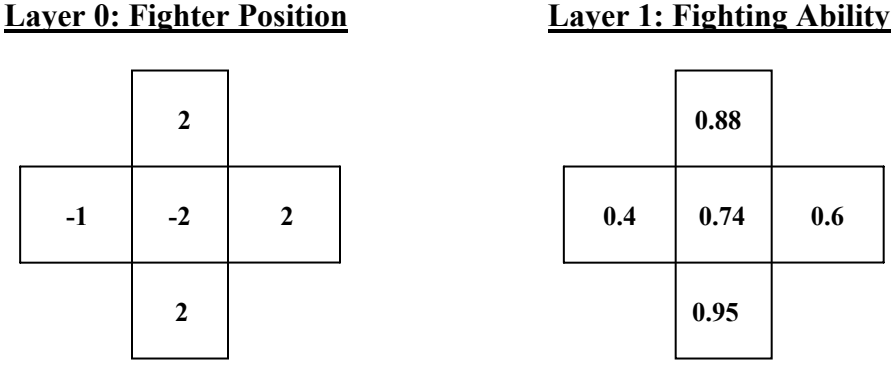

Figure 4: Fighting rule implementation

In the previous diagram, the following points are noted:

- Fighter -2 is surrounded by three enemy fighters and one friend fighter
- Two of the enemy fighters have more fighting ability than (-2), these are the ones which will be successful in attacking -2.
- The fighter -1 was injured and not attacked by anyone for a while, so he goes to state alive again -2.
- One of the enemy fighters has a lower fighting ability than -2, this one will be attacked by -2.

The outcome of the previous alignment is as follows

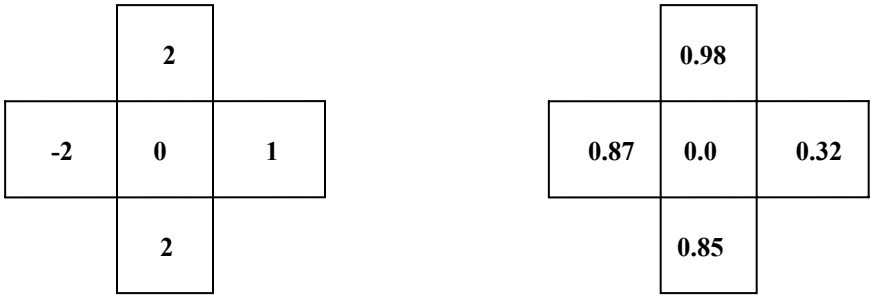

Figure 5: Fighting rule result

Movement Rule Implementation

This rule is implemented through layers 4 and 5. Layer 4 will have the directions for each fighter to get closer to the enemy's flag and layer 5 will have the moving factor for each empty cell.

- **Direction calculation**

The direction layer (layer 4) is generated as follows:

- The current cell coordinates are compared with the enemy's flag position (from layer 2 or 3).
- If the cell is in the same row as the enemy's flag, the direction will be to the right or left depending on the flag position
- If the cell in the same column as the enemy's flag, the direction will be up or downs depending on the flag position.
- If the cell is not in the same row or column as the enemy's flag, a random selection between up/down or right/left will be performed.

- **Moving factor calculation**

When a fighter is not surrounded by any enemy fighter, he tends to move towards the enemy's flag. However, to avoid any conflicts when more than one fighter wants to move to the same free cell, each free cell will have a moving factor associated with it. The moving factor is the largest fighting ability of the four surrounding fighters. The following figure explains this idea:

Layer 0: Fighter positions **Layer 1: Fighting ability** **Layer 5: Moving factor**

Figure 6: Calculating the moving factor of a free cell

So, if the fighting ability of the fighter wishing to move to the free cell is larger than or equal to the movement factor of that cell, he will be allowed to move and the value of his original cell becomes zero.

Testing strategy

The testing strategy followed, is testing each of the previous rules alone by injecting an input situation that will only trigger that particular rule. After that, testing the model with a scenario that triggers all the rules to check the correctness of the model.

• Testing the Fighting Rule

To test the fighting rule, the following arrangement was used.

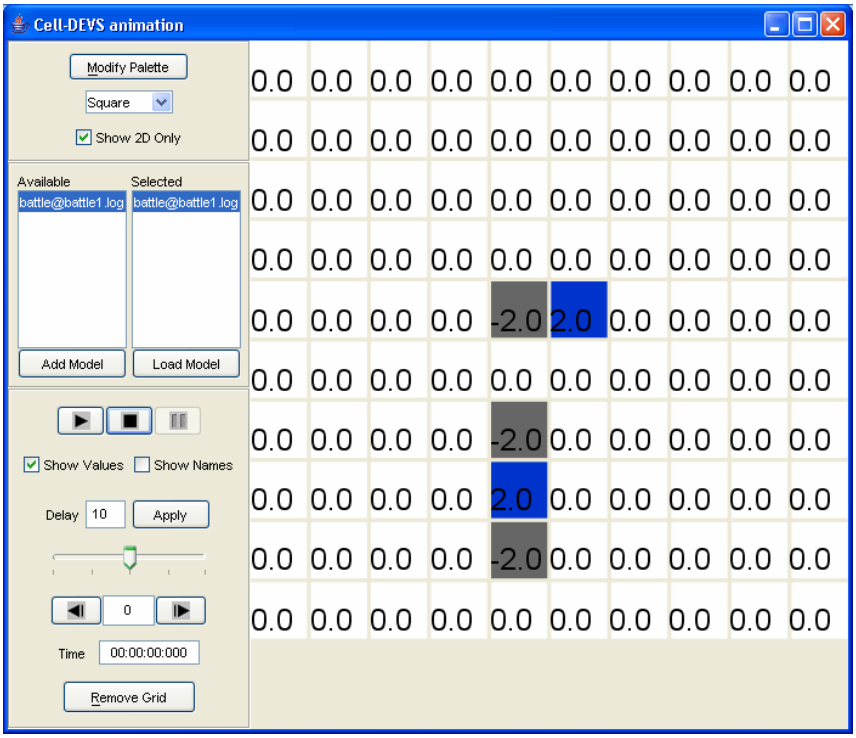

Figure 7: Fighting rule scenario

After executing the file **battle1.bat** and observing the engagement of the fighters, the correct behavior of the rule was verified.

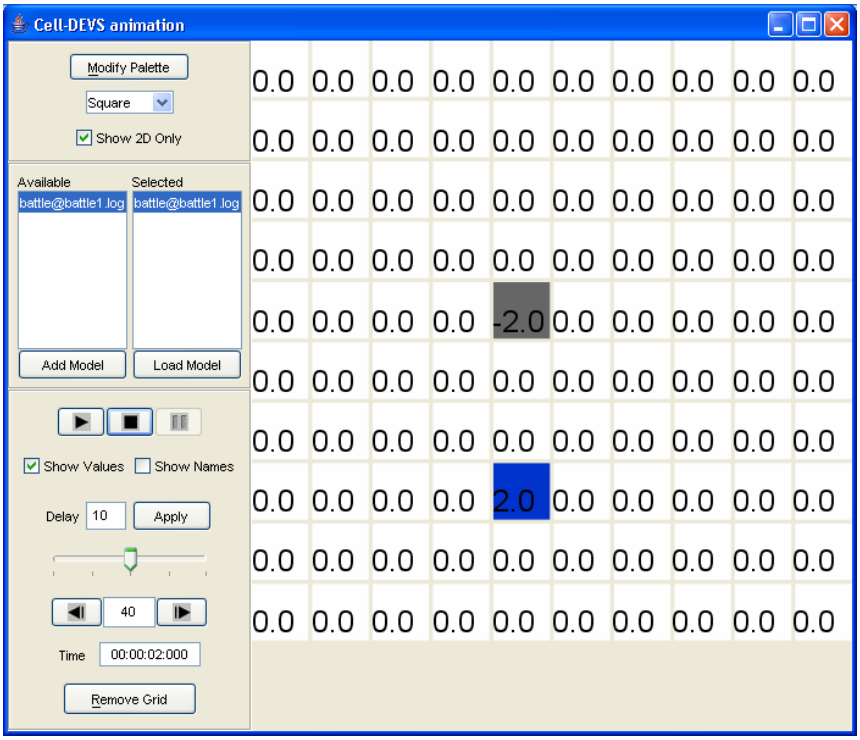

Figure 8: Fighting rule outcome

- **Testing the Movement Rules**

The following arrangement was used to test the movement of fighters towards the enemy's flag.

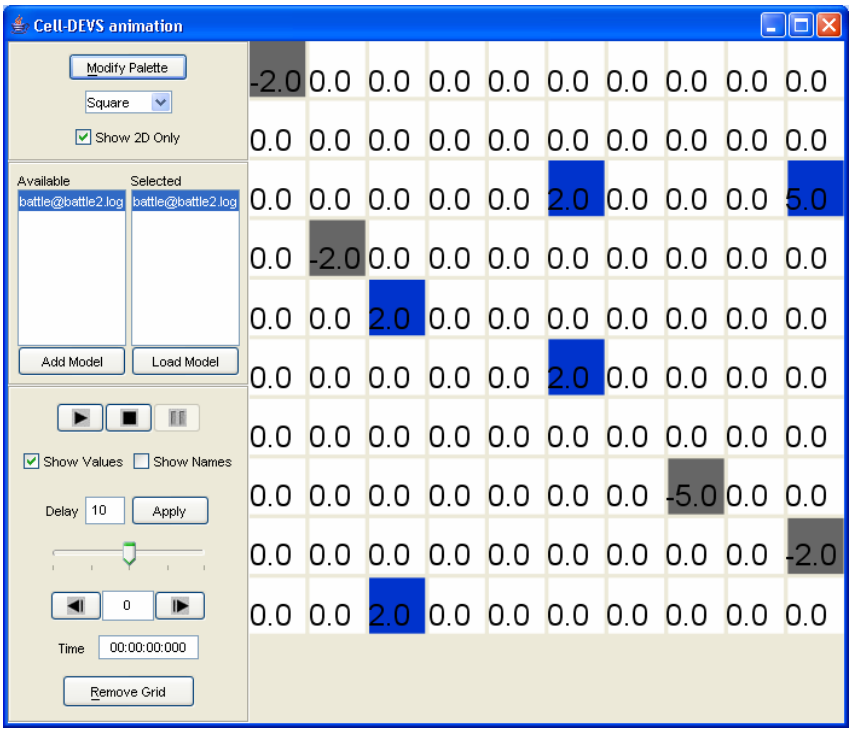

Figure 9: Movement rule scenario

After executing the file **battle2.bat**, and observing the movement of different fighters, the correct behavior of the rule was proved as shown in the following figure.

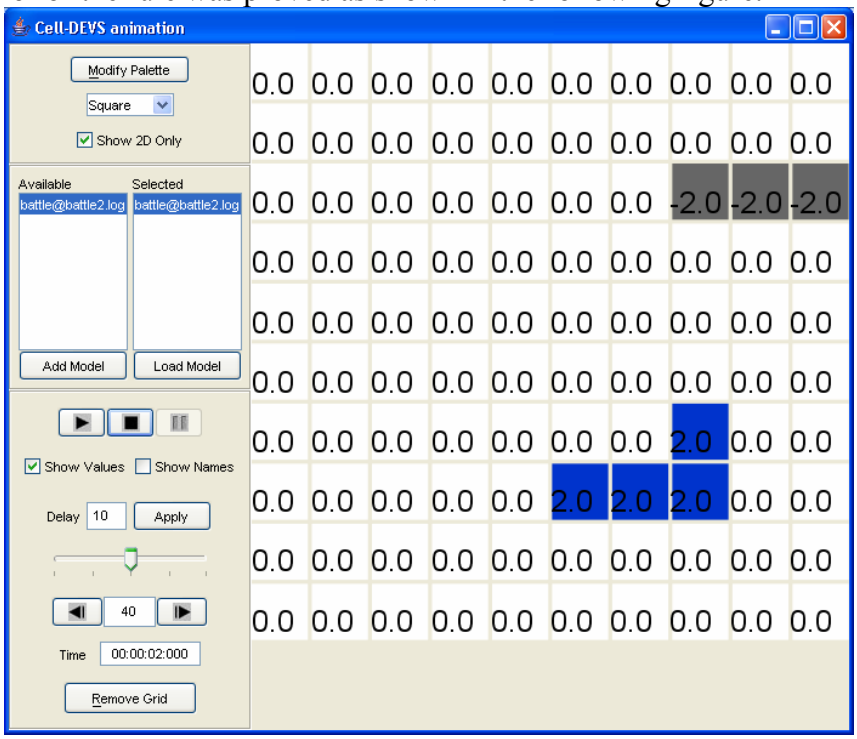

Figure 10: Movement scenario outcome

- **Testing the behavior of injured soldiers**

The initial allocation of injured soldiers was as follows:

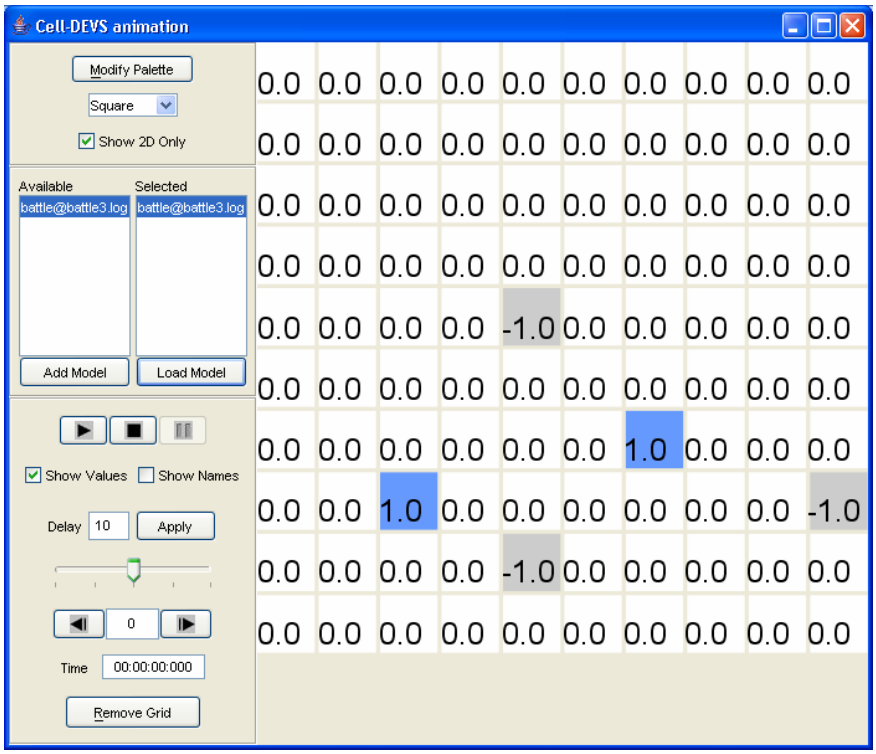

Figure 11: Injured soldier scenario

After executing the **battle3.bat** file, the following allocation was generated, which proves the correct behavior of the injured soldiers rule.

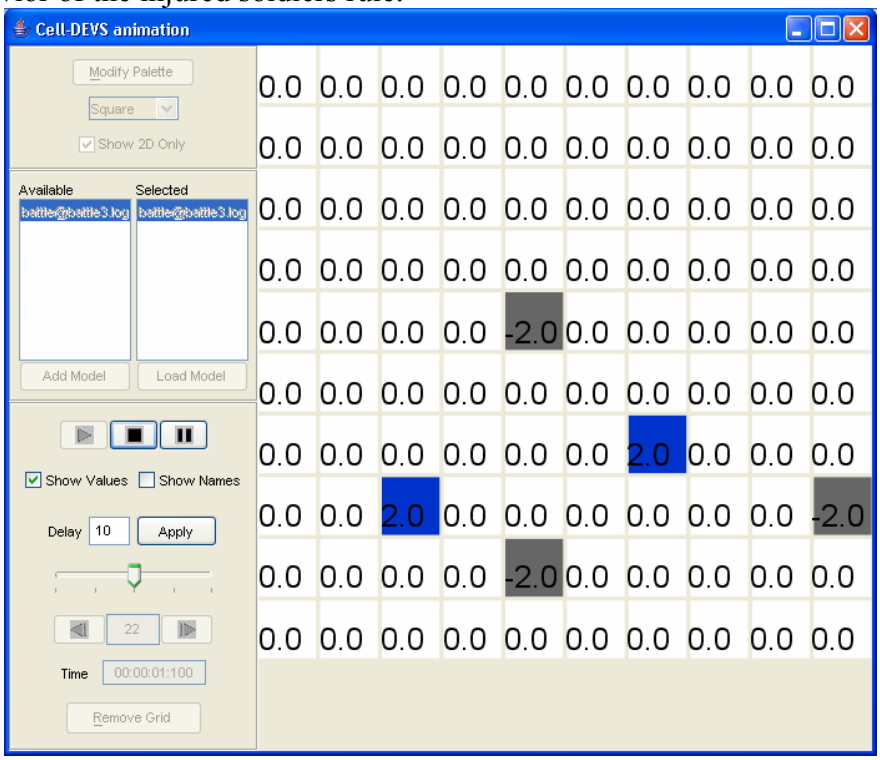

Figure 12: Injured soldiers scenario outcome

- **Testing the Overall Behavior of the Model**

To test the overall behavior of the model, a scenario involving all the rules was used. But the result seemed to be incorrect when the rules are activated together. After testing different scenarios, I came to a conclusion that the reason for this inconsistency is that the fighting ability rule is evaluated and delayed (for the value of transport delay). And because the fighting ability factor is a key in evaluating most of the other rules, this created an inconsistent behavior of the model. So, the following steps were performed:

- Evaluating the fighting ability rule with zero delay but this created some instability in the model and generated "Unknown Exception" error.
- Decreasing the delay for the fighting ability rule to a very small value. But this made the execution of the model extremely slow.
- The last step was to use 50 time units delay for the fighting rule and 100 for the other rules. In addition, the delay type was changed from transport to inertial. More correct behavior was detected after performing this step.

The following scenario was used to test the overall behavior of the model.

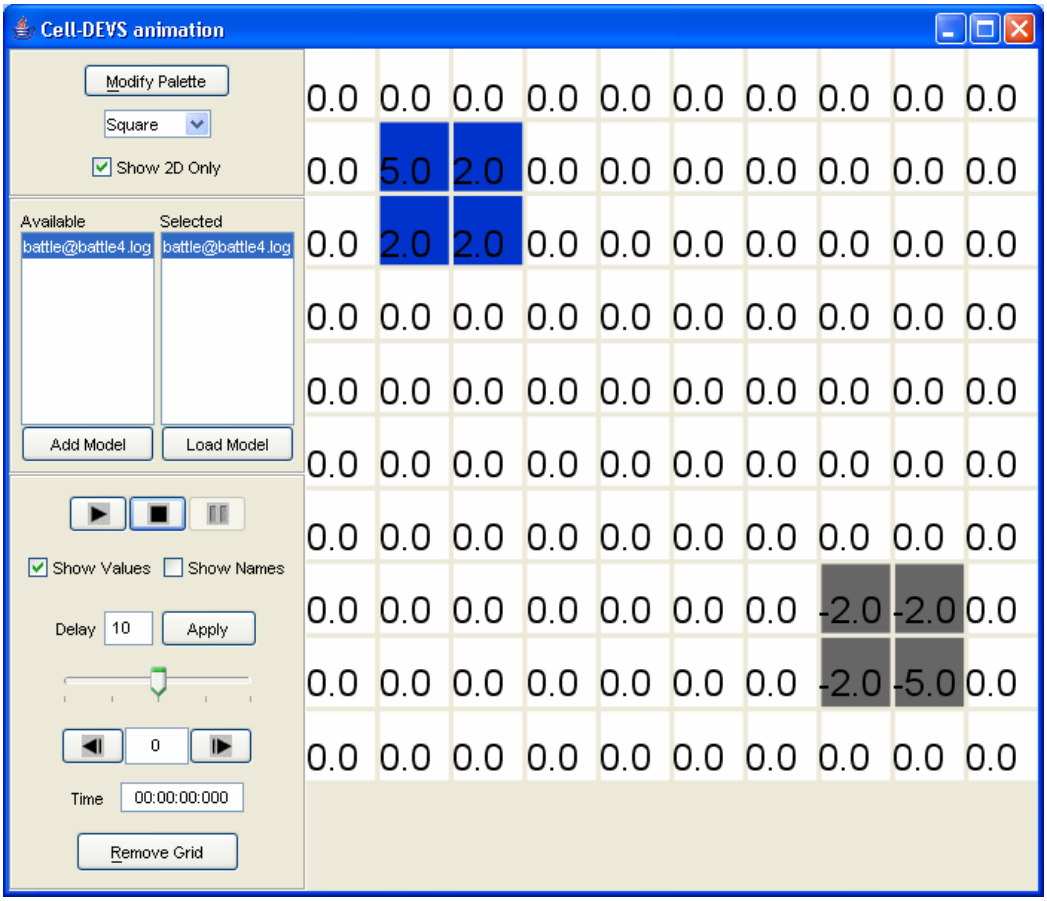

Figure 13: General fighting scenario

After executing the file **battle4.bat**, the result was as follows:

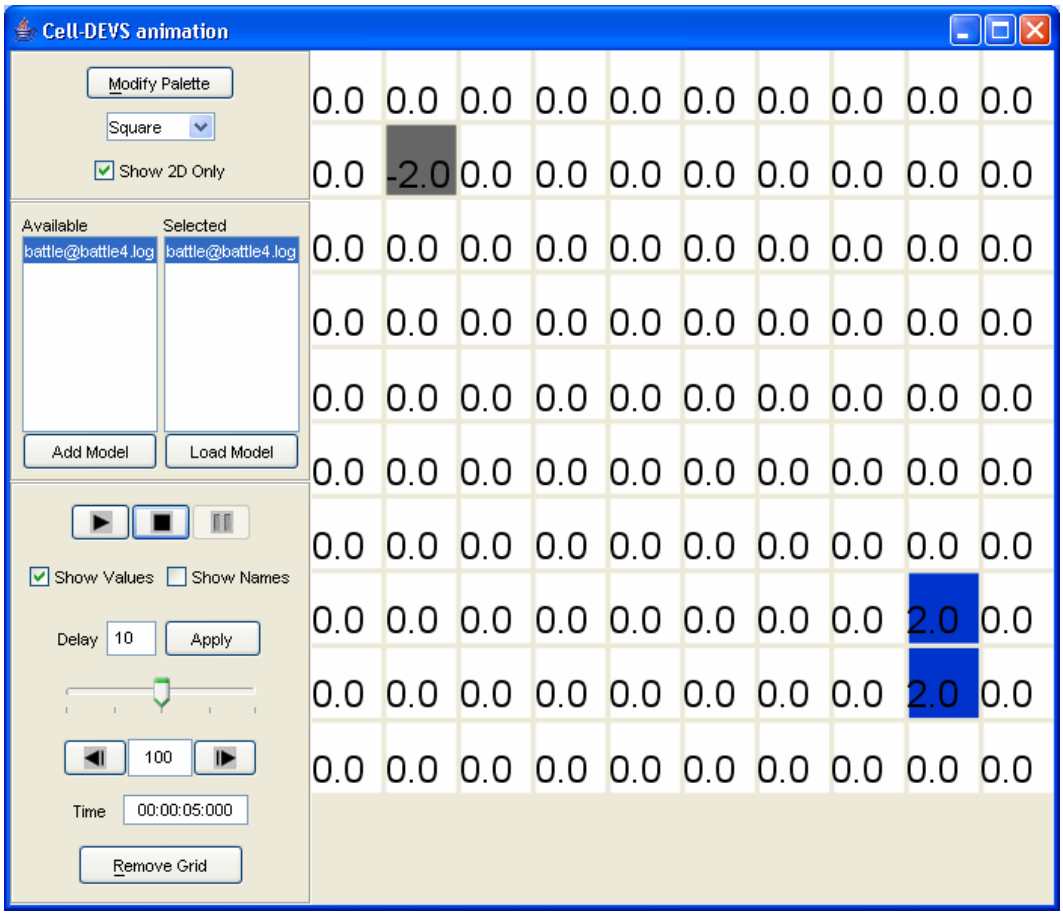

Figure 14: General fighting scenario outcome

3D Visualization of the Model Execution

The results of the model execution were visualized using the VRML 3D visualization tool. The following figures show the result for different scenarios.

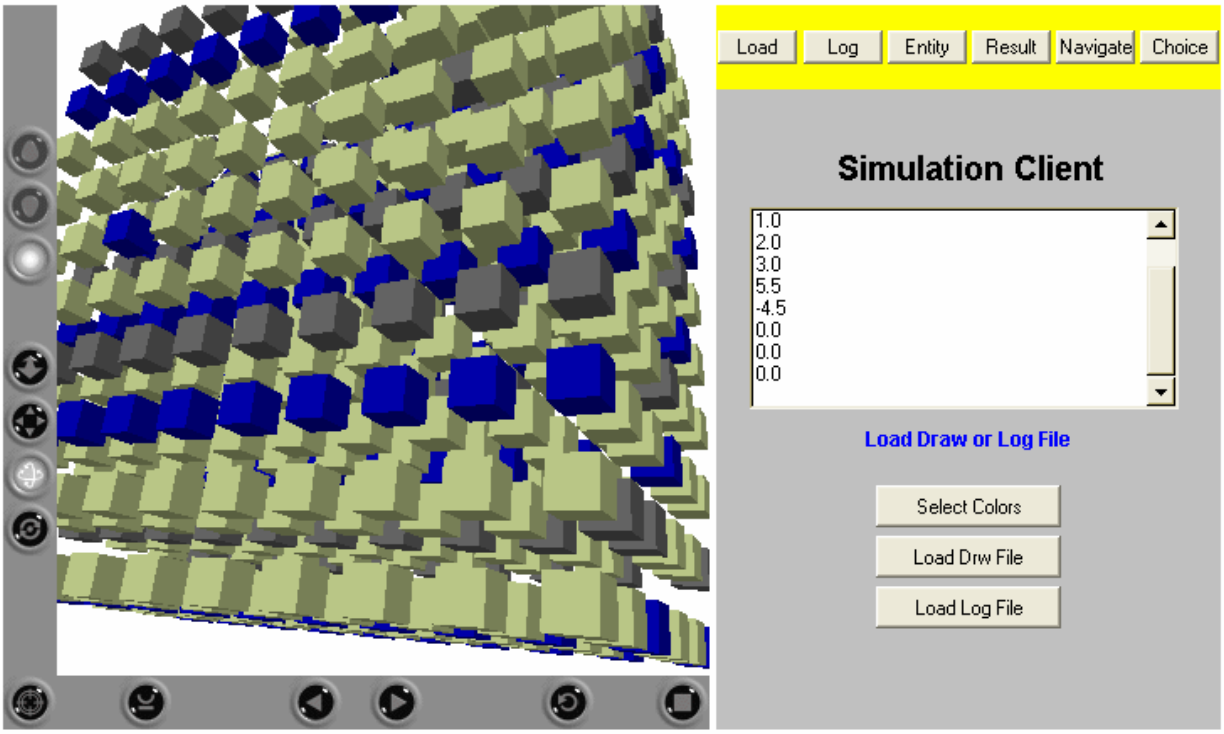

Figure 15: Visualizing the movement rule behavior

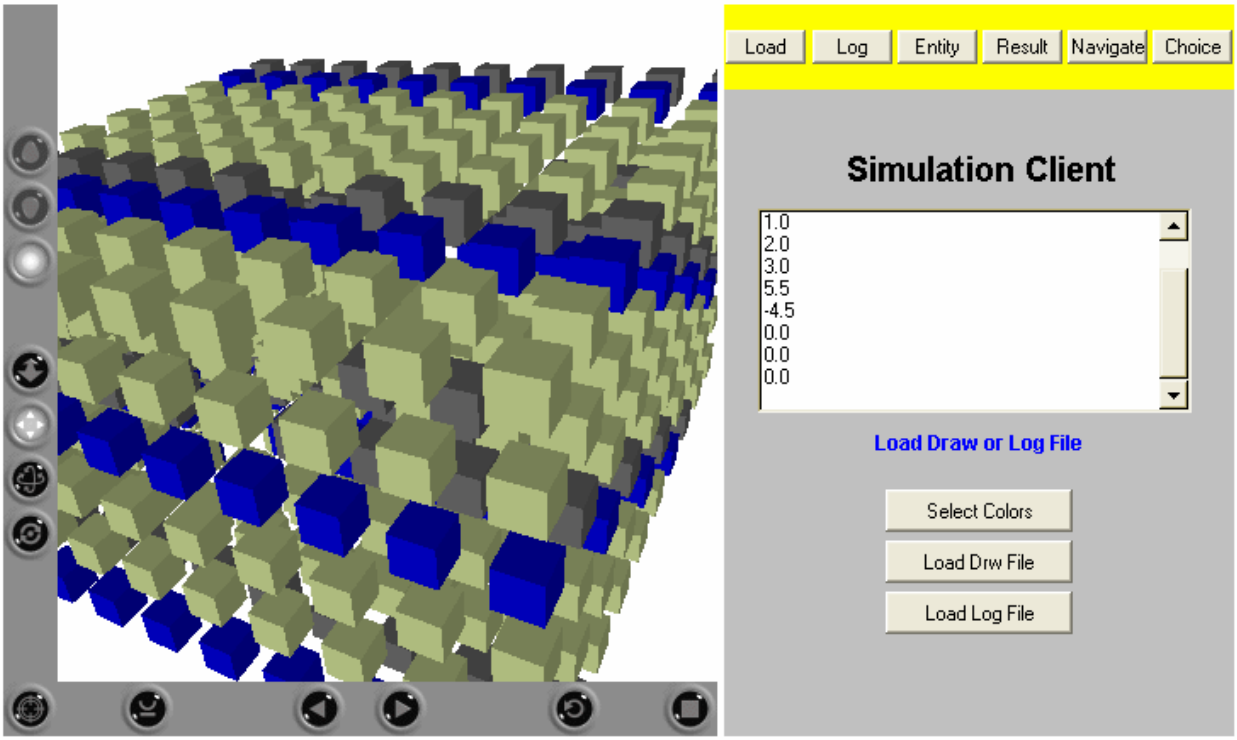

Figure 16: Visualizing the general fighting scenario