Tumor-Immune System Cell-DEVS Model Report

SYSC 5104 - Methodologies for Discrete Event Modelling and Simulation

Assignment #2

Rhys Goldstein
Carleton University
Student #100747303

2007 November 19

Table of Contents

Conceptual Model

This section provides an informal description of the tumor-immune system model, which was specified using Cell-DEVS and implemented with CD++. This model will be referred to as the "Cell-DEVS model". The Cell-DEVS model was based on previous work, which will be referred to as the "cellular automaton".

Cellular Automaton

The authors of [1] studied previous models of tumor growth, and expanded those models to include the response of the immune system. Their cellular automaton consists of a 2-dimensional cell-space, in which each cell of the model represents a biological cell. The various types of cells are listed below.

Normal Cell	a cell that is neither part of a tumor, nor a part of the immune system
Immunity Cell	a cell that is part of the immune system. No distinction is made between B cells, T cells, or any other types of cells that respond to tumors.
Proliferative Cell	a tumor cell that divides, facilitating the growth of a tumor
Dormant Cell	a tumor cell that was once proliferative, but no longer divides
Necro Cell	a tumor cell that once once dormant, but is now considered dead

The cellular automaton also included another type of cell, referred to as a "middle" cell. As explained later, this type was not included in the Cell-DEVS version.

The rules of the cellular automaton accounted for growth of tumors, the movement of immunity cells, and the interaction between immunity cells and tumor cells. A tumor is a cluster of tumor cells that grow in a pattern resembling three concentric circles. The proliferative cells on the outside tend to form a ring around dormant cells, which in turn encircle a cluster of necro cells. Immunity cells wander randomly through the space of normal cells that surround the tumor. When an immunity cell encounters a tumor cell, it attacks. If the attack is successful, the tumor cell is destroyed. Otherwise, the immunity cell disappears.

Cell-DEVS Model

Although the Cell-DEVS model was intended to follow the same rules as the cellular automaton described in [1], contributing a formal specification of the model. The formal specification is restricted to mathematical formulas, avoiding variable references and technological instructions. It presents the rules of the algorithm without ambiguity.

As some of the rules in [1] were somewhat ambiguous, several assumptions had to made to define the Cell-DEVS version. One of the ambiguities was in the definition of the neighborhood of a cell, which was described as an "extended Moore neighborhood" but specified in a formula as a discretized circle. The Cell-DEVS version uses the extended Moore neighborhood shown below. Certain rules refer to the set of adjacent cells, which are shaded in the diagram.

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

The rules concerning the probabilistic growth of tumor cells were also somewhat ambiguous, and an earlier model described in [2] was therefore consulted. In the Cell-DEVS version, a proliferative cell is given some probability to divide in a random direction. If that direction points to a normal cell, that cell becomes proliferative. If there are no adjacent normal cells, then there is a chance that the proliferative cell will become dormant. If a dormant cell has no adjacent proliferative cells, there is a chance it will become necro. In [1] and [2], the radius of a tumor is restricted by a constant parameter. This parameter was omitted from the Cell-DEVS version in part for simplicity, but also to allow the tumor-immune system interaction to determine the extent to which a tumor grows.

Before encountering a tumor, an immunity cell either stays still, or moves in a random direction. If at least one of its adjacent cells is proliferative, it attacks one of those cells at random. There is a chance that the proliferative cell will survive the attack, in which case the immunity cell may be destroyed. In [1], a "middle" cell was described as a possible outcome of a tumor-immunity cell battle. This possibility was neglected in the Cell-DEVS version in part for simplicity, but also because the given formula implied that the situation would never arise. Immunity cells never attack dormant cells or necro cells. The rules of the Cell-DEVS model are complicated by cases in which collisions between immunity cells must be avoided.

Model Specification

This section provides a formal specification of the Cell-DEVS model.

General Model Variables

The **TIS** function below results in a tumor-immune system cell-space.

$$TIS(t_1, t_2, d, p_{resisting}, p_{moving}, p_{curing}, p_{dividing}, p_{dying}) = <X_{list}, Y_{list}, X, Y, n, \{t_1, t_2\}, N, C, B, Z, select>$$

The parameters $\mathbf{t_1}$ and $\mathbf{t_2}$ are the dimensions of the cell-space, **d** is the delay for each cell, and the five **p** parameters are probability thresholds that influence the behavior of the model. The cell-space, of course, is 2-dimensional.

$$n=2$$

The model has neither inputs nor outputs.

$$X_{list} = Y_{list} = X = Y = \emptyset$$

The 5-by-5 cell square neighborhood is centered on each cell.

$$N = \{<i,j> \mid (i \in \{-2,-1,0,1,2\}) \wedge (j \in \{-2,-1,0,1,2\})\}$$

The borders of the model are wrapped. This is convenient in that it ensures that immunity cells incident on the border are not removed from the model completely. The disadvantage is that the model losses validity as the growing tumor nears the boundary.

$$B = \emptyset$$

The translation function **Z** is defined as in the formal specification of Cell-DEVS. In this particular model, the order in which states are received from neighboring cells is irrelevant. The **select** function is therefore left unrestricted.

The entire $\mathbf{t_1}$-by-$\mathbf{t_2}$ cell space is occupied by timed DEVS cells, each of which has the same specification.

$$C = \{C_{ij} \mid (i \in \mathbb{Z}) \wedge (0 \leq i < t_1) \wedge (j \in \mathbb{Z}) \wedge (0 \leq j < t_2)\}$$

(where...)

$$C_{ij} = < X_C, Y_C, S, N, delay, d, \delta_{int}, \delta_{ext}, \tau, \lambda, D >$$

As was the case for the cell-space, the DEVS cells have neither input nor output values.

$$X_C = Y_C = \emptyset$$

The state associated with each individual cell consists of both a type and a direction. The type indicates whether the represented biological cell is a normal, immunity, proliferative, dormant, or necro cell. The direction, which is either <0, 0> or the coordinates of an adjacent cell, indicates either the way immunity cells are moving or the way proliferative cells are dividing.

$$S = \{ < type, direction > \mid (type \in types) \wedge (direction \in directions) \}$$

(where...)

$$types = \{ normal, immunity, proliferative, dormant, necro \}$$

(and...)

$$directions = \{ < i, j > \mid (i \in \{-1, 0, 1\}) \wedge (j \in \{-1, 0, 1\}) \}$$

The model uses transport delays.

$$delay = transport$$

The variables $\boldsymbol{\delta_{int}}$, $\boldsymbol{\delta_{ext}}$, $\boldsymbol{\lambda}$, and **D** are defined as in the formal specification of Cell-DEVS. The following subsections define the local computing function $\boldsymbol{\tau}$.

Local Computing Function

The term "state" refers to the value associated with one cell in the cell-space. A state must be a value in the set **S**. A "state function" is a function that takes a vector of relative coordinates as an input, and results in the state of the cell associated with those coordinates. The state function **s** therefore maps values in the set **N** to values in the set **S**.

$$s : N \rightarrow S$$

The example equation below indicates that there is an dormant cell at a relative position of <-2, 1>.

$$s(<-2,1>) = <\text{dormant}, <0,0>>$$

In any scope that contains the state function **s**, the functions **type** and **direction** will also be available. These functions provide the types and directions associated with neighboring cells. They are defined implicitly below.

$$\text{type}: N \rightarrow \text{types}$$

(and...)

$$\text{direction}: N \rightarrow \text{directions}$$

(and...)

$$s(<i,j>) = <\text{type}(<i,j>), \text{direction}(<i,j>)>$$

The local computing function $\boldsymbol{\tau}$ maps the state function **s**, which carries the present states of neighboring cells, onto the new state of the cell. It is defined below.

$$\tau(s) = \begin{bmatrix} \text{type}(<0,0>) = \text{normal} & \rightarrow & \tau_{\text{normal}}(s) \\ \text{type}(<0,0>) = \text{immunity} & \rightarrow & \tau_{\text{immunity}}(s) \\ \text{type}(<0,0>) = \text{proliferative} & \rightarrow & \tau_{\text{proliferative}}(s) \\ \text{type}(<0,0>) = \text{dormant} & \rightarrow & \tau_{\text{dormant}}(s) \\ \text{type}(<0,0>) = \text{necro} & \rightarrow & \tau_{\text{necro}}(s) \end{bmatrix}$$

Note that $\boldsymbol{\tau}$ depends on the functions $\boldsymbol{\tau}_{\text{normal}}$, $\boldsymbol{\tau}_{\text{immunity}}$, $\boldsymbol{\tau}_{\text{proliferative}}$, $\boldsymbol{\tau}_{\text{dormant}}$, and $\boldsymbol{\tau}_{\text{necro}}$. These are defined in subsequent subsections. Several other sets and functions are described here in order to facilitate the specification of the transition rules. The two random value functions below are used extensively.

$\text{random}(M)$	results in a value randomly selected from the set **M**
$\text{random}_{\text{uniform}}()$	results in a value randomly selected from the set of real numbers greater than 0 and less than 1.

The variable **adjacents** represents the set of relative coordinates of all adjacent cells.

$$\text{adjacents} = \{<i,j> \mid (i \in \{-1,0,1\}) \wedge (j \in \{-1,0,1\}) \wedge \neg(i = j = 0)\}$$

The function $\boldsymbol{\text{count}}_{\text{near}}$ results in the number of cells, with coordinates contained in **M**, that are of type $\boldsymbol{\text{type}}_{\text{near}}$. Among other things, it is used to determine whether there is a proliferative cell adjacent to an immunity cell.

$$\text{count}_{\text{near}}(s, \text{type}_{\text{near}}, M) = \sum_{<i,j> \in M} \begin{bmatrix} \text{type}(<i,j>) = \text{type}_{\text{near}} & \rightarrow & 1 \\ \text{type}(<i,j>) \neq \text{type}_{\text{near}} & \rightarrow & 0 \end{bmatrix}$$

The function **$\text{count}_{\text{incoming}}$** results in the number of cells, with coordinates adjacent to <k, l>, that are of type **$\text{type}_{\text{incoming}}$**. In order to be counted, those adjacent cells must also be have a direction that points to <k, l>. Among other things, this function is used to prevent collisions between immunity cells.

$$\text{count}_{\text{incoming}}(s, \text{type}_{\text{incoming}}, <k,l>) = \sum_{<i,j> \in L} \begin{bmatrix} \text{incoming}(<i,j>) & \rightarrow & 1 \\ \neg\text{incoming}(<i,j>) & \rightarrow & 0 \end{bmatrix}$$

(where...)

$$\text{incoming}(<i,j>) = (s(<i,j>) = <\text{type}_{\text{incoming}}, <k-i, l-j>>)$$

(and...)

$$L = \{<k-i, l-j> \mid <i,j> \in \text{adjacents}\}$$

Below is an example that demonstrates the **$\text{count}_{\text{incoming}}$** function.

(-2, 2)		**immunity <1, -1>**		**(2, 2)**
	(-1, 1)		**(1, 1)**	**immunity <-1, 0>**
		(0, 0)	**immunity <0, -1>**	
	(-1, -1)		**(1, -1)**	
(-2, -2)				**(2, -2)**

There are three immunity cells adjacent to the cell labeled (1, 1). The one on the upper left and the one to the right are both approaching that cell. The one below, however, is pointing elsewhere. The cell (1, 1) therefore has 2 incoming immunity cells.

$$\text{count}_{\text{incoming}}(s, \text{immunity}, <1,1>) = 2$$

Normal Cells

A normal cell is considered to be "receiving" an immunity cell if there is exactly one incoming immunity cell. In this event, the cell becomes an immunity cell that is "preparing" its next action. If a normal cell is not "receiving", then it is "praying" that is will not be overtaken by the tumor.

$$\tau_{\text{normal}}(s) = \begin{bmatrix} \text{receiving} & \rightarrow & \tau_{\text{preparing}}(s) \\ \neg\text{receiving} & \rightarrow & \tau_{\text{praying}}(s) \end{bmatrix}$$

(where...)

$$\text{receiving} = (\text{count}_{\text{incoming}}(s, \text{immunity}, <0,0>) = 1)$$

If a "praying" cell has any incoming proliferative cells, the previously normal cell is considered to be "mutating". It then becomes a proliferative cell itself, "plotting" its next division. A "praying" cell that is not "mutating" is "waiting".

$$\tau_{\text{praying}}(s) = \begin{bmatrix} \text{mutating} & \rightarrow & \tau_{\text{plotting}}(s) \\ \neg\text{mutating} & \rightarrow & \tau_{\text{waiting}}(s) \end{bmatrix}$$

(where...)

$$\text{mutating} = (\text{count}_{\text{incoming}}(s, \text{proliferative}, <0,0>) > 0)$$

A "waiting" cell may spontaneously become an immunity cell. This happens with a probability of $\mathbf{p_{resisting}}$, but only if the cell is surrounded by normal cells. Otherwise, the "waiting" cells remains a normal cell.

$$\tau_{\text{waiting}}(s) = \begin{bmatrix} \text{resisting} & \rightarrow & \tau_{\text{preparing}}(s) \\ \neg\text{resisting} & \rightarrow & <\text{normal}, <0,0>> \end{bmatrix}$$

(where...)

$$\text{resisting} = ((\text{count}_{\text{near}}(s, \text{normal}, \text{adjacents}) = 8) \wedge (\text{random}_{\text{uniform}}() < p_{\text{resisting}}))$$

Note that if $\mathbf{p_{resisting}}$ is zero, immunity cells will never spontaneously appear. Instead they will die off until there are none left, or until there are no proliferative cells left that can kill them.

Immunity Cells

An immunity cell is either "leaving" its present position, or staying and "preparing" its next move. It is only "leaving" if its direction is not <0, 0>, the cell it is heading towards is either normal or proliferative, and there are no other immunity cells approaching its own destination.

$$\tau_{\text{immunity}}(s) = \begin{bmatrix} \text{leaving} & \rightarrow & <\text{normal}, <0,0>> \\ \neg\text{leaving} & \rightarrow & \tau_{\text{preparing}}(s) \end{bmatrix}$$

(where...)

$$\text{leaving} = \begin{array}{l} (\text{direction}(<0,0>) \neq <0,0>) \wedge \\ (\text{type}(\text{direction}(<0,0>)) \in \{\text{normal}, \text{proliferative}\}) \wedge \\ \text{count}_{\text{incoming}}(s, \text{immunity}, <0,0>) = 1) \end{array}$$

A "preparing" immunity cell is "attacking" if there are any adjacent proliferative cells. The relative set of coordinates of the target is chosen at random from among these cells. If there are no adjacent proliferative cells, the immunity cell is "wandering". Note that an immunity cell may not attack an adjacent proliferative cell if the two cells had advanced upon one another simultaneously. In this case, the immunity cell may have already chosen to "wander" before the proliferative cell appeared.

$$\tau_{\text{preparing}}(s) = \begin{bmatrix} \text{attacking} & \rightarrow & <\text{immunity}, \text{random}(\text{opponents})> \\ \neg\text{attacking} & \rightarrow & \tau_{\text{wandering}}(s) \end{bmatrix}$$

(where...)

$$\text{attacking} = (\text{count}_{\text{near}}(s, \text{proliferative}, \text{adjacents}) > 0)$$

(and...)

$$\text{opponents} = \{<i, j> \mid (<i, j> \in \text{adjacents}) \wedge (\text{type}(<i, k>) = \text{proliferative})\}$$

A "wandering" cell moves in a random direction with probability $\mathbf{p_{moving}}$. Otherwise it stays.

$$\tau_{\text{wandering}}(s) = \begin{bmatrix} \text{moving} & \rightarrow & <\text{immunity}, \text{random}(\text{adjacents})> \\ \neg\text{moving} & \rightarrow & <\text{immunity}, <0,0>> \end{bmatrix}$$

(where...)

$$\text{moving} = (\text{random}_{\text{uniform}}() < p_{\text{moving}})$$

Proliferative Cells

If a proliferative cell has at least one incoming immunity cell, then there is a probability of $\mathbf{p_{curing}}$ that it will undergo the process of "curing". Otherwise, it is described as "plotting".

$$\tau_{\text{proliferative}}(s) = \begin{bmatrix} \text{curing} & \rightarrow & \tau_{\text{curing}}(s) \\ \neg\text{curing} & \rightarrow & \tau_{\text{plotting}}(s) \end{bmatrix}$$

(where...)

$$\text{curing} = (\text{count}_{\text{incoming}}(s, \text{immunity}, <0,0>) \geq 1) \wedge (\text{random}_{\text{uniform}}() < p_{\text{curing}})$$

There are two situation in which a "curing" proliferative cell is transformed. If there are multiple incoming immunity cells, then none of the immunity cells may enter and the cell becomes normal. In this case, the proliferative cell can be described as "suffocating". If there is exactly one incoming immunity cell, then that immunity cell it moves in and becomes "preparing".

$$\tau_{\text{curing}}(s) = \begin{bmatrix} \text{suffocating} & \rightarrow & <\text{normal}, <0,0>> \\ \neg\text{suffocating} & \rightarrow & \tau_{\text{preparing}}(s) \end{bmatrix}$$

(where...)

$$\text{suffocating} = (\text{count}_{\text{incoming}}(s, \text{immunity}, <0,0>) > 1)$$

A "plotting" proliferative cell divides with a probability of $\mathbf{p_{dividing}}$.

$$\tau_{\text{plotting}}(s) = \begin{bmatrix} \text{dividing} & \rightarrow & \tau_{\text{dividing}}(s) \\ \neg\text{dividing} & \rightarrow & <\text{proliferative}, <0,0>> \end{bmatrix}$$

(where...)

$$\text{dividing} = (\text{random}_{\text{uniform}}() < p_{\text{dividing}})$$

A "dividing" proliferative cell becomes "sleeping" if there are no adjacent normal cells. Otherwise, it attempts to create a new proliferative cell in a random direction.

$$\tau_{\text{dividing}}(s) = \begin{bmatrix} \text{sleeping} & \rightarrow & <\text{dormant}, <0,0>> \\ \neg\text{sleeping} & \rightarrow & <\text{proliferative}, \text{random}(\text{adjacents})> \end{bmatrix}$$

(where...)

$$\text{sleeping} = (\text{count}_{\text{near}}(s, \text{normal}, \text{adjacents}) = 0)$$

Other rules ensure that a proliferative cell will only convert the randomly-selected cell if that cell happens to be normal. Also, there must not be any immunity cells incoming on the target.

Dormant Cells

A dormant cell has a probability of $\mathbf{p_{dying}}$ of "dying", but only if there are no adjacent proliferative cells.

$$\tau_{\text{dormant}}(s) = \begin{bmatrix} \text{dying} & \rightarrow & <\text{necro}, <0,0>> \\ \neg\text{dying} & \rightarrow & <\text{dormant}, <0,0>> \end{bmatrix}$$

(where...)

$$\text{dying} = (\text{count}_{\text{near}}(s, \text{proliferative}, \text{adjacents}) = 0) \wedge (\text{random}_{\text{uniform}}() < p_{\text{dying}})$$

Necro Cells

Necro cells do nothing but remain as they are.

$$\tau_{\text{necro}} = <\text{necro}, <0,0>>$$

Model Implementation and Testing

This section describes the implementation of the Cell-DEVS model using CD++, and discusses the results of four different tests.

CD++ Implementation

CD++ refers to both a typed specification language for DEVS and Cell-DEVS models, and a tool that facilitates computer simulation with DEVS models. Although the tool provides many convenient functions for operating on cell neighborhood values, the version used supported neither functions nor variables with limited scope. Because the specification in the previous section relies on these language features, much of the CD++ code was prepared in a spreadsheet. The table below illustrates the technique.

Pseudo Code	Spreadsheet Formula	Generated CD++ Code
tau_wandering(s) = if(moving, <immunity, rand(adjacents)>, <immunity, <0, 0>)	=CONCATENATE("if(";B11;", ";DEVS_Cell_Values.B3;" + ";Primitive_Functions.B9;", ";DEVS_Cell_Values.B3;")")	if((uniform(0, 1)) < #Macro(p_moving)), 1 + ((trunc(uniform(0, 8)) + 1*0.0625), 1)
moving = (random_uniform() < p_moving)	=CONCATENATE("(";Primitive_Functions.B7;" < "; Parameters.B5;")")	((uniform(0, 1)) < #Macro(p_moving))

The spreadsheet cells on the left contained pseudo code to describe the corresponding cells on the right. Those cells contained spreadsheet formulas that concatenated strings with the contents of other cells. The spreadsheet cells acted as variables, and in some cases the functionality of the spreadsheet was used to evaluate functions. The result was generated CD++ code. All of the CD++ code in the spreadsheet was concatenated into a single cell. The CD++ rule contained in that cell expressed the entire specification of the previous section.

Two additional rules were added for the convenience of simulation. The first one, shown below, was included to repetitively place a single proliferative cell at the center of the model, provided that the cell was otherwise normal. The second rule created an initial random distribution of immunity cells, a trick that required the entire cell-space to be given some "nil" value, -1 in this case, at time zero. The density of immunity cells was governed by the parameter $p_{initial}$, the probability that each cell would be initialized as an immunity cell.

```
rule : 2 #Macro(d) { ((0,0) = 0) and (cellpos(0) = 20) and (cellpos(1) = 20) }

rule : { if(uniform(0, 1) < #Macro(p_initial), 1, 0) } #Macro(d) { (0,0) = -1 }
```

Because the selected version of CD++ did not allow vectors to be used as states, the direction was encoded as a multiple of 0.0625. This number was added to an integer part representing the cell type.

The use of random numbers in CD++ presented two additional complications. First, it was difficult to accommodate the "attacking" immunity cell, which had to choose at random a proliferative cell to advance upon. Without a temporary variable to store the result of the random value function, it was practically necessary to reformulate the problem and obtain a separate random value for almost every direction. In the end, the specification was satisfied, but at a high cost in terms of execution speed.

A second complication related to random values was a threat to the specification itself. In order to save time, CD++ avoids the unnecessary evaluation of the local computing function for passive cells. A cell is considered passive if it is receiving no external inputs, and the state of none of its neighbors has changed. In many cases this is an extremely valuable feature. However, it is possible for a cell to be considered passive when its state is determined in part by a random number. The problem is largely mathematical in nature. Random number generating functions violate the law below, which states that a function's result may change only if its argument changes.

$$\neg(x = y) \vee (f(x) = f(y))$$

Thus in the presence of random values, the result of the local computing function can change without changes in the cell's neighborhood.

Test A - Tumor Growth

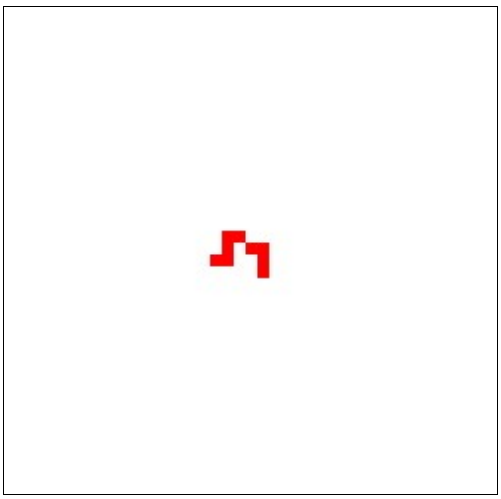
All tests were performed on spaces of 41-by-41 cells, with a duration of 1 time unit between transitions. Test A investigated the growth of a tumor in the absence of immunity cells.

$$p_{initial} = p_{resisting} = 0$$

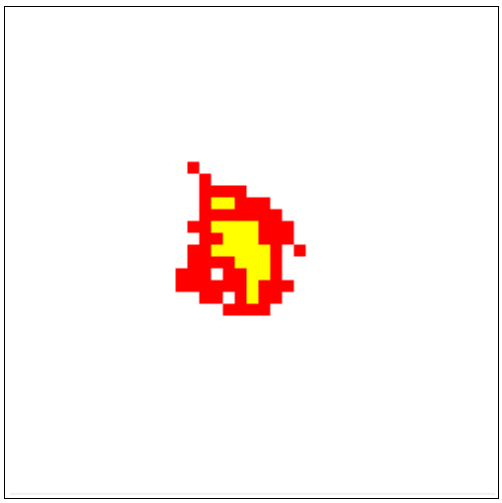
The probability of dividing and dying were both 50%.

$$p_{dividing} = p_{dying} = 0.5$$

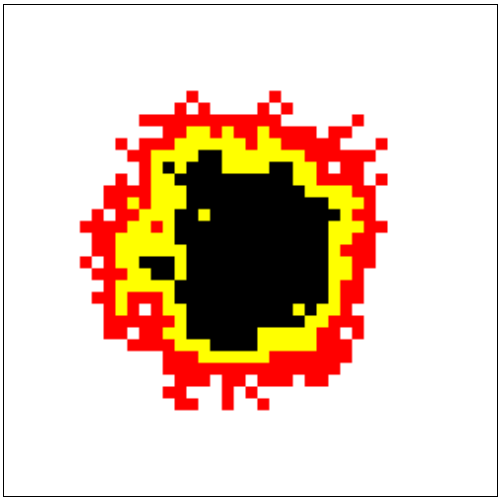
The images below show the tumor at various times. The proliferative cells on the outside are shown in red, surrounding the yellow dormant cells and black necro cells. Although the shape of the tumor is irregular at time 11, it becomes continually more circular as it grows. The tumor reaches the boundary of the model at time 64.



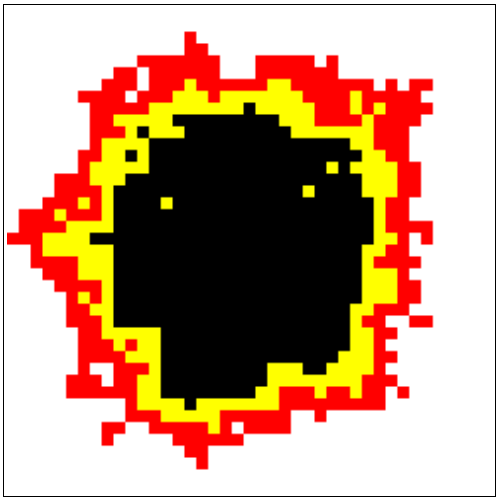
Test A (time = 11)



Test A (time = 23)



Test A (time = 49)



Test A (time = 64)

Test B - Immunity Cell Random Walk

In Test B, a small number of immunity cells were added at the beginning of the simulation.

$$p_{initial} = 0.01$$

$$p_{resisting} = 0$$

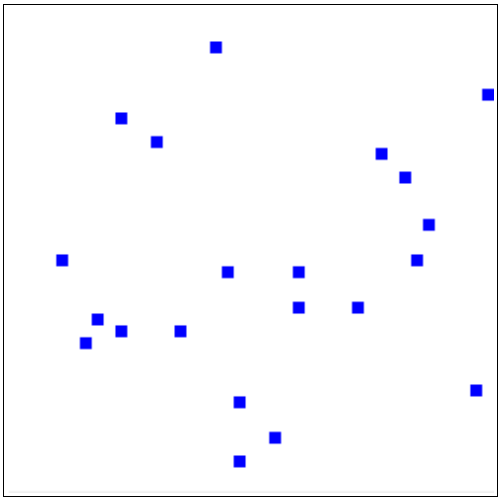
The immunity cells were to always move, and always succeed in destroying proliferative cells.

$$p_{moving} = p_{curing} = 1$$

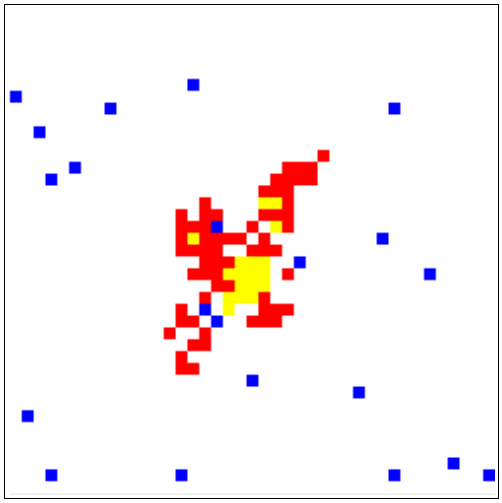
Proliferative cells divided with a 20% probability, but never died.

$$p_{dividing} = 0.2$$

$$p_{dying} = 0$$



Test B (time = 1)



Test B (time = 100)

The only case in which an immunity cell could have died is if it moved onto the center cell as the proliferative cell was placed. This possible situation, brought about by the extra rule added for convenience, did not occur in Test B. There were 21 immunity cells at the beginning and end of the simulation. The test served to partially verify that the issue of collision detection was correctly handled.

Test C - Tumor Victory

In Test C, a tumor victory arose from the following parameters,

$$p_{initial} = 0$$

$$p_{resisting} = 0.01$$

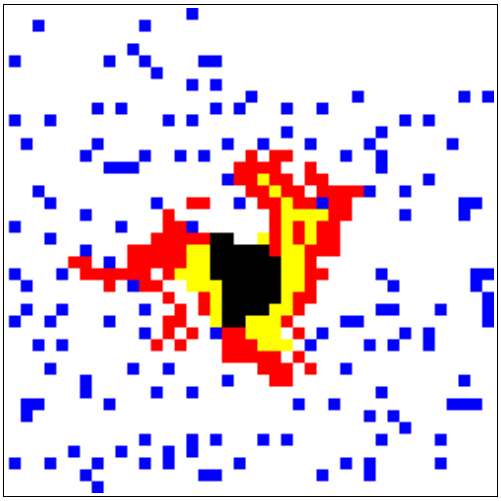
$$p_{moving} = 0.5$$

$$p_{curing} = 0.8$$

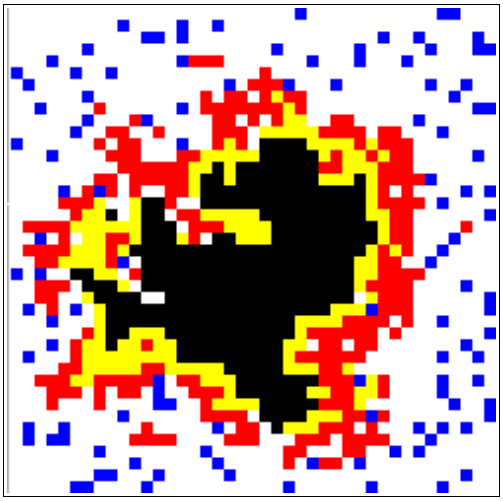
$$p_{dividing} = 0.6$$

$$p_{dividing} = 0.6$$

Near the beginning of the simulation, the immunity cells cleared away a large section of proliferative cells on the north side of the tumor. Later the tumor regained this region, and proceeded to overwhelm the immune system.



Test C (time = 38)



Test C (time = 64)

Test D - Immune System Victory

In Test D, the parameters were changed from Test C to favour the immune system.

$$p_{initial} = 0$$

$$p_{resisting} = 0.03$$

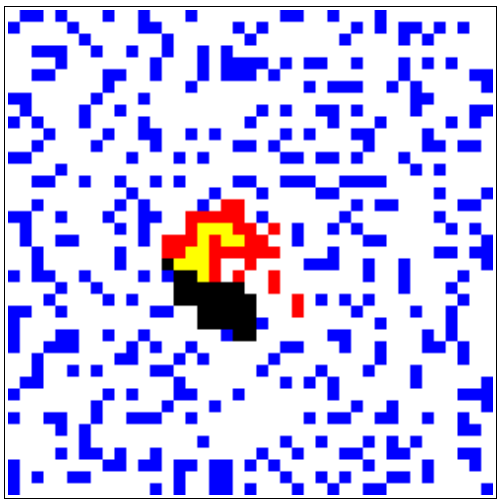
$$p_{moving} = 0.4$$

$$p_{curing} = 0.8$$

$$p_{dividing} = 0.4$$

$$p_{dividing} = 0.6$$

By time 65, the proliferative cells were restricted to a single side of the otherwise dead tumor. The mass of necro cells increased roughly four times thereafter, but the last proliferative cell was destroyed at time 150.



Test D (time = 65)

Test D (time = 150)

References

1 R. Huricha, X. Ruanxiaogang, “A simple cellular automaton model for tumor-immunity system,” Proceedings. 2003 IEEE International Conference on Robotics, Intelligent Systems and Signal Processing, 2003.

2 A. R, Kansal, S. Torquato, G. R. Harsh, E. A. Chiocca, T. S. Deisboeck, “Simulated Brain Tumor Growth Dynamics Using a Three-Dimensional Cellular Automaton,” Journal of Theoretical Biology, 2002, 203(4): 367-382.