**Prepared By:**  
Kyle Bjornson (100936493)

**Prepared For:**  
Dr. Gabriel Wainer

Modelling Rust Propagation using Cell-DEVS

Assignment 2 – SYSC 5104 – Fall 2018

# Model Proposal

I propose modelling rust propagation using Cell-DEVS, based on the model discussed in [1]. The model defines a three-dimensional cell space. Each cell has a degree of rust progression state with a value from 0 meaning no rust, to ~90 meaning the rust has penetrated the material. The range in between those values represents the progression of rust in that cell.

[1] defines the incidence rate of rust being 0.001%. In other words, at any time step a cell with no rust, and no adjacent rust will have a 0.001% chance of generating one degree of rust. After that, any cell with at least 1 degree of rust will each will generate a propagation of rust to an adjacent cell following the rules defined in [1].

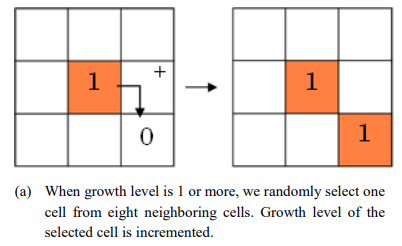
Generally, the rules define vectors based on neighbors and a direction of gravity. These rules decide the pseudo random propagation of rust to the next cell. The authors of [1] explain more in depth the propagation scheme used in a previous paper [2].

I will begin by developing a Cell-DEVS model in two dimensions and if time permits scale up to three dimensions.

# Proposal Modifications

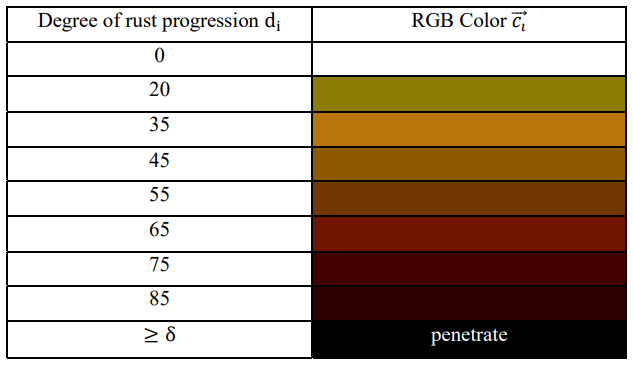
After the initial proposal, I decided to focus on a two-dimensional model using the simplest model discussed in [2]. A cell can have a rust state represented by an integer value from 0 to 90, as mentioned above. Every cell begins with a state of 0 and a subset of cells have a 0.001% chance of changing state to a 1. This subset will be defined as a ‘rust prone’ area/zone, possibly due to moisture, etc.

If a cell has a state of 1 or higher, it will randomly decide at some regular time interval to add 1 to one of its neighbor cells, defined as a Moore neighborhood. This can be seen in Figure 1 below.



**Figure 1: Conceptual model of proposal, taken from [2]**

At specified values of rust state, the colour of the cell will change to indicate a progression of rust. The colour to rust progression table is shown below in Figure 2.



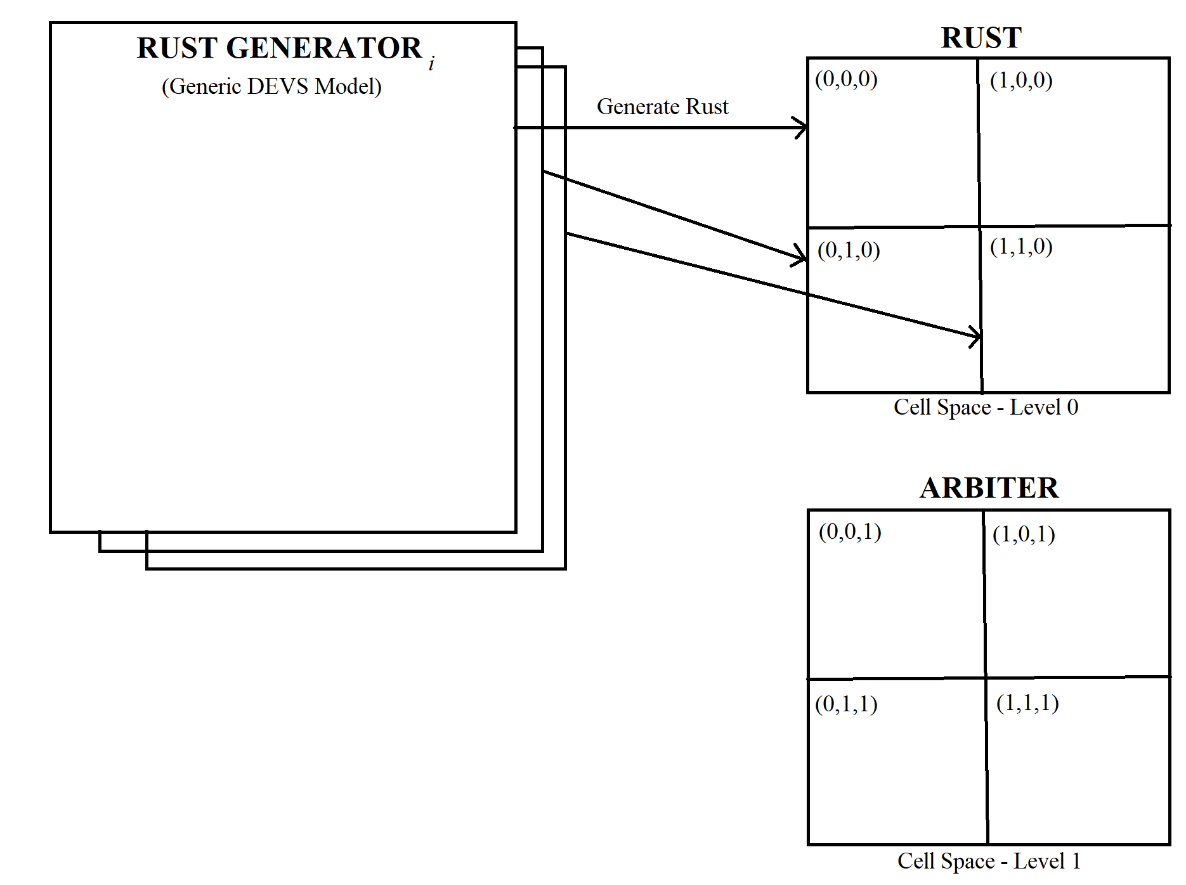
**Figure 2: Rust progression colour palette [1]**

# DEVS Model

The rules for the Cell-DEVS model of the model presented above is slightly different. As each cell is an atomic model, it cannot directly alter the values of neighboring cells. It can only influence other cells based on the state it last broadcasted. Another difference, is that the incident rust only occurs with a 0.001% probability. Utilizing Cell-DEVS, if the value of a cell attempts to calculate that probability then doesn’t change state, it will sleep. These problems will be addressed as follows.

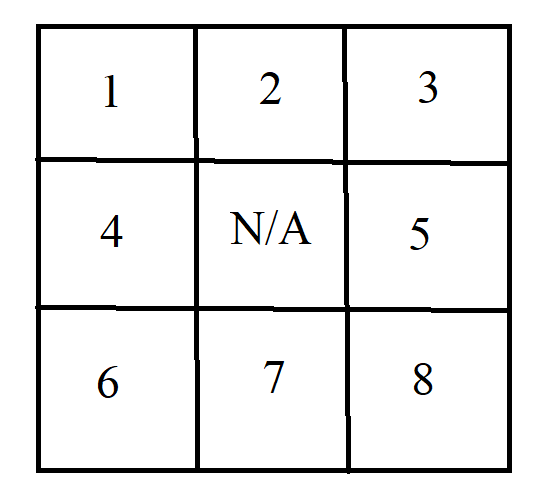
The Cell-DEVS model will utilize two layers making a three-dimensional cell space. One layer will have the state of the rust progression, and the other layer will act as an arbiter to decide which cell rust will propagate to.

The model will also utilize a standard DEVS model, creating a coupled model to determine when incident rust should be produced in a cell. An instance of this DEVS model will be used to output to the input of a subset of cells to tell them when incidence rust should occur. A definition of the coupled model is shown in Figure 3 below. Note that the cell space dimensions are small for illustration purposes.



**Figure 3: Coupled Model**

For example, if the state of (0,0,0) was last broadcast as a 1, the arbiter cell (0,0,1) would then decide to broadcast its own state as a random value representing one of the neighbor cells in the rust plane. That is, (1,0,0), (0,1,0), or (1,1,0). In the model, these will be represented as integers from 1 to 8, as shown in Figure 4 below.



**Figure 4: Arbiter States to decide neighbor cell rust propagation**

# Formal Specification

**Rust – Inertial Delay Cell**

**For Some Rust Layer Cells…**

**Arbiter – Inertial Delay Cell**

(Possible states depend on whether it’s a border cell or not. Only chooses neighbors that exist.)

**Cell-DEVS Coupled Model**

**Incident Rust Generator – Generic DEVS Model**

**External Coupling Definition**

# CD++ Model

Some changes were made to the model during implementation. Firstly, the arbiter layer also now checks whether the neighbor below it, and all its Moore’s neighbors are fully rusted (Have a state of 90). If so, the arbiter will no longer calculate a value for that cell. This improves the performance of the simulation.

A third layer was also added, with just initial conditions and no state changes to indicate which cells are taking input from the generic DEVS model and which are not. They are shown as “rust prone zones” coloured blue.

Also, a note on timing. As this is modelling rust, time must be considered relative. In this model a time of milliseconds is used, but with real rust this should be more like days and years. Importantly, there is often a large time gap between no rust and the first instance of rust, then propagation takes over.

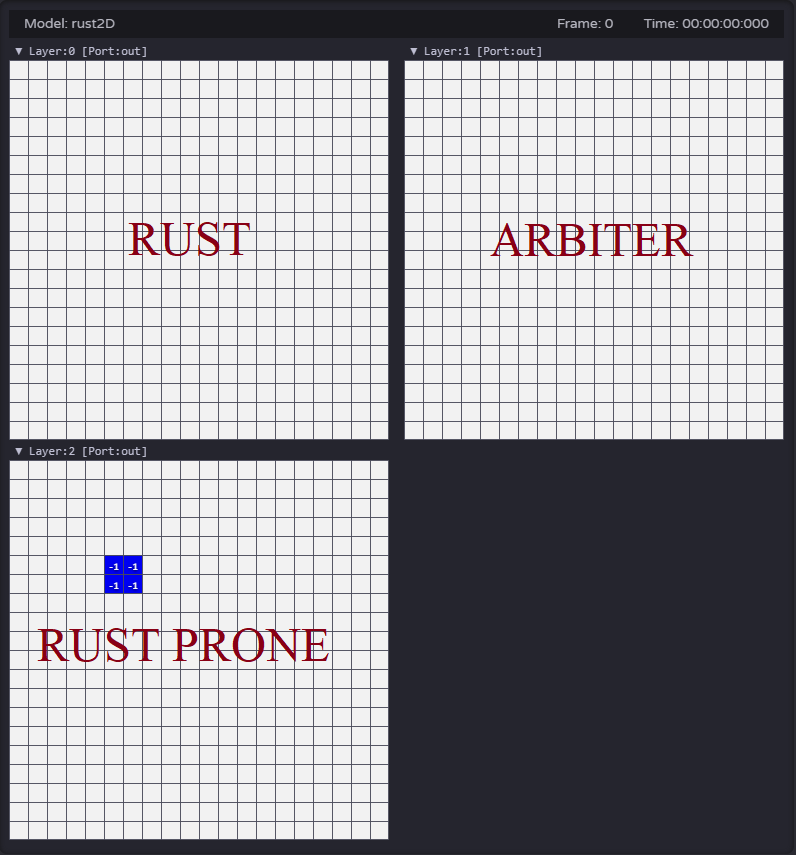
Following the rules from the source material [2] for rust propagation, the borders of the cell space have fewer neighbors, and therefore less of an opportunity to rust. The simulation may behave in an unrealistic way because of the rule.

Finally, to calculate random arbiter propagation, the uniform function is used in coordination with the round function. This is ok for cells not on the border, but some cells on the border do not get a perfect uniform distribution between them. For example, on the left edge, the arbiter cell needs to select a random cell between 2,3,5,7 and 8. Since there is a gap in the sequence, the arbiter has a chance of choosing 4 or 6, meaning no cells will get rust propagation that round. This also contributes to the edge rust propagation as mentioned above.

# Simulation Results

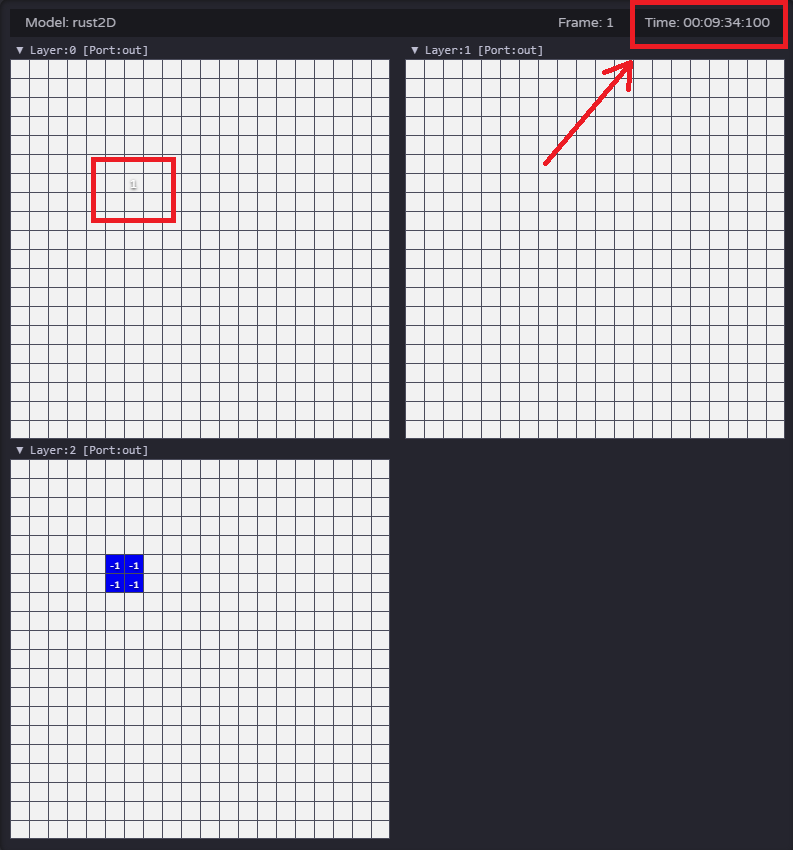
The simulations show a random propagation of rust throughout the cell space. Let’s walk through an example simulation.

Figure 5 below shows the initial conditions, and labels what each grid represents.



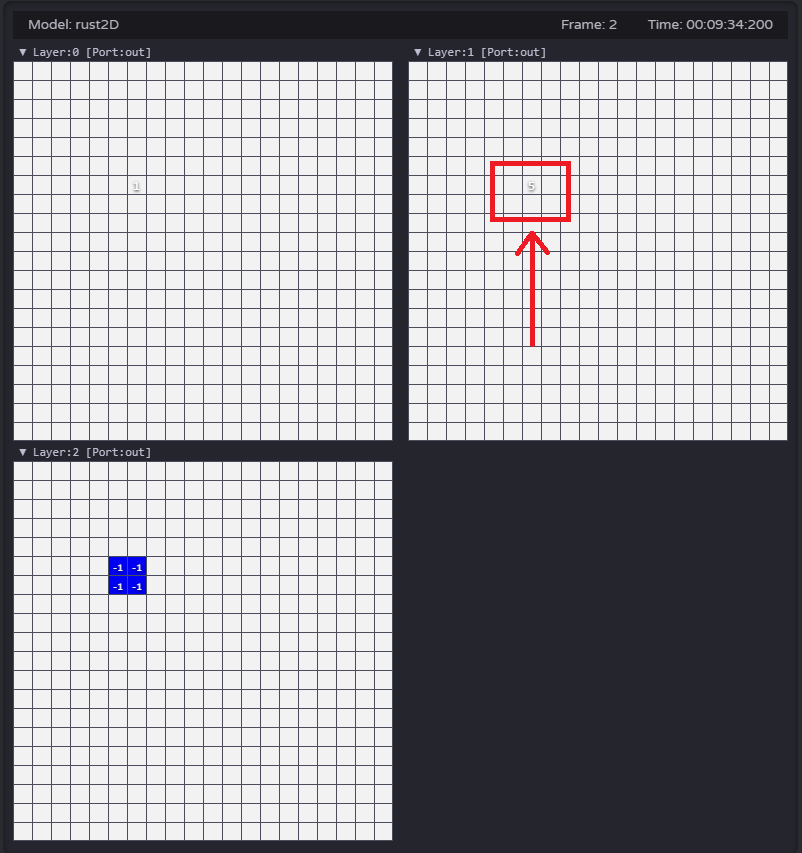
**Figure 5: Initial Conditions**

Next, we can see one step into the simulation. The rust generating DEVS model has randomly added 1 to cell (6,6,0) after 9 minutes, 34 seconds. This is shown in Figure 6.



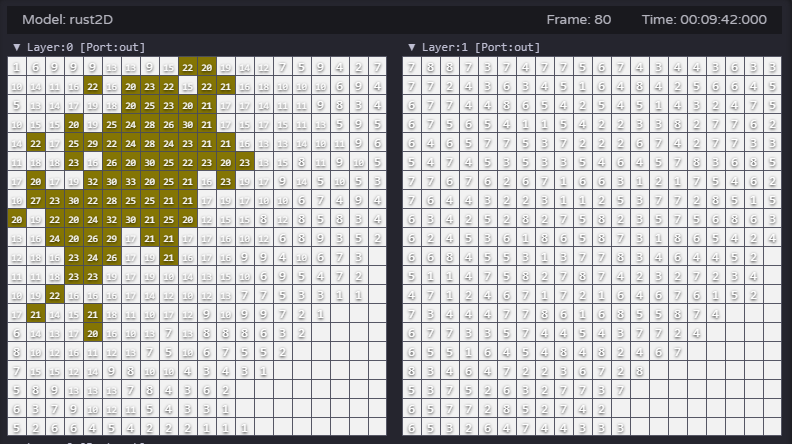
**Figure 6: After one time advance.**

Now the rust will start to propagate from this one cell. The arbiter will detect this change one cell after and update its cell accordingly. Figure 7 is one time-advance after Figure 6.



**Figure 7: After two time-advances.**

Now the rust will begin to propagate and fill the cell space. The cells will start to change colour the cell states reach the specified target values. The remaining Figures 8, 9, 10 and 11 show the propagation of rust until the simulation completes. They omit the third layer as it does not change.



**Figure 8: Cells begin to change colour**



**Figure 9: Rust has filled the cell space, Arbiter is working continuously**

# 

**Figure 10: Arbiter begins to shut down as rust maxes out**

# 

**Figure 10: Rust complete, Arbiter off, simulation complete**

# References

1. Y. Suzuki, Y. Yamabe, T. Moriya and T. Takahashi, "A corrosion and deformation simulation method of 3D iron objects based on voxel automaton," 2018 International Workshop on Advanced Image Technology (IWAIT), Chiang Mai, 2018, pp. 1-5. URL: <http://ieeexplore.ieee.org.proxy.library.carleton.ca/stamp/stamp.jsp?tp=&arnumber=8369703&isnumber=8369620>
2. R. Tanabe, Y. Morimoto, T. Moriya and T. Takahashi, "A Generation Method of Rust Aging Texture Considering Rust Spreading," The Institute of Image Information Television Engineers, 2014, pp. 1-4. URL: <http://nishitalab.org/user/UEI/publication/Tanabe_IWAIT2015.pdf>