Simulating the Effects of Bi-Direction Pedestrian Movement Between a Doorway Using Cell-DEVS **Term Project**

By : Ronnie Farrell

Student # : 100717103

To : Prof. Wainer

Course : SYSC 5104

Date : December 20, 2011

# Introduction

This term project is an extension of the second assignment of SYSC 5104. In the second assignment, a paper entitled “Simulation of bi-direction pedestrian movement using a cellular automata model” was studied and implemented with CellDEVS based on the suggested model and rules.

The model was described in the two-dimensional system. The underlying structure was a W × W cell grid, where W was the system size. Each cell was either empty or occupied by exactly one pedestrian. The size of a cell corresponded to approximately 0.4 × 0.4 m2 since this is the typical space occupied by a pedestrian in a dense crowd. The original work tended to study the behaviours of pedestrians walking in an area with obstacles. The model included two types of walkers; pedestrians that walk north, and pedestrians that walk south, and used a wrapped boundary.

The following figure describes the cellular automata rules that the paper used to represent the behaviour of the pedestrians walking north. The same logic can be applied to pedestrians walking south.



Figure : Pedestrian Behaviour

The second assignment implemented all rules except for case (h). Please see the next section for a background of DEVS and CellDEVS.

# Background

The implementation of this project and the second assignment are both based on the knowledge of CellDEVS which is an extension of DEVS. DEVS represents Discrete Event System Specification. It includes a high level language, as well as a modeling and simulation framework. It features time delays between discrete state changes of the modeled system, as well as a mechanism of differentiating the internal and external events which cause the transition between states. When there is an external input which can occur at any time, the simulation activates the external event transition function which transforms and an input event into another event. Internal transitions occur all of the time. After a certain time has elapsed, and output may or may not be generated, and then the internal event transition function executes which transforms the current state into another state based on its rules.

CellDEVS was introduced for spacial events simulation, which DEVS was not good at. CellDEVS is an extension of DEVS that can be used to model cellular automata where the system can be represented as a cell space. In CellDEVS each cell represents an atomic DEVS model that considers its specified neighbourhood in order to determine its transition according to a set of defined rules. The nice thing about CellDEVS is that for all execution cycles, not all of the cells in the cell space are activated. The simulator only activates the current active cells as well as their inverse neighbourhood. The other cells remain in a passive state waiting to be woken up. This saves a massive amount of execution time for a simulation.

For further information regarding DEVS or CellDEVS, please refer to “Discrete-Event Modeling and Simulation, A Practitioner’s Approach”. [2]

# Problem Statement and Project Definition

Having the modeled space and basic rules understood, an implementation was carried out in the second assignment. The implementation worked very well, and throughout vigorous testing no collisions were observed. Due to the complex nature of the project, the pre-conditions of CellDEVS rules were extremely complex and lengthy. For any pedestrian attempting to walk into another cell, the simulator was required to check that not only was the target cell empty, but that no other pedestrians were potentially going to move into that same cell in the next cycle. The rules needed to be thorough and cover all of the cases, so that no collision would occur. This appeared to be accomplished throughout the completion of the second assignment.

A new simulation model using the pedestrian rules from assignment two was proposed. This model involves taking two room components and allowing pedestrians to walk between them through a doorway. The purpose of this project is to simulate the effect that pedestrians have while walking in a doorway. In order to cover all cases of pedestrian behaviour, new rules will need to be implemented. Refer to section 8 for the definition of all of the rules.

This new model involves creating two different CellDEVS components; one to represent the room, and the other to represent the hall. These two components both use the pedestrian behaviour rules. In order to get pedestrians to walk from one component to the other, a DEVS model is created which connects the two components together. Also, in order to obtain proper behaviour of the pedestrians when they reach the wall with the door, a new zone was implemented. Please refer to the next section for a definition of the model.

# Basic Model Definition

Figure : CellDEVS Coupled Model

Figure 2 shows the coupled model definition of the project. The “Pedestrian Gen” blocks are random sequence generator atomic models built in CD++. The “Hall” is a CellDEVS model with a cell space of 10x6. The “Room” is also a CellDEVS model with a cell space of 10x10. The “Door Controller” block is a DEVS atomic model which has two input ports, and two output ports (one each for the hall and for the room).

When the Door Controller receives a message on an input port, it sends the value of the message to the associated output port after a specific time. For example, when port inHall receives a value of 2, port outHall will generate a value of 2 after 50msec. In this model there are two Door Controller components representing a doorway with two doors.

The doorway which connects the room and hall components is represented by dotted lines in the coupled model. It consists of cells (0,2),(0,3) in the Hall, and cells (9,4),(9,5) in the Room. The doorway cells have both input and output ports associated with them as pedestrians can both enter and exit the areas through the door. The output ports of the hall and room components are connected to the input ports of the Door Controller component respectively. The output ports of the Door Controller component is connected to the input ports of the hall and room components respectively.

Pedestrians are generated at the bottom of the hall, and at the top of the room. Once the pedestrians reach the doorway, they leave their current area through the door, and enter the other area through the door. The goal of this project is to program the pedestrians to walk from one room to the other via the door, and see the effects that result from the doorway. The speed of all the pedestrians are modeled as a constant speed and everyone travels at this rate. To do this, some rules are used to regulate the behaviours of the pedestrians. These rules are explained in detail in section 8.

# Project Approach

Having the goal of the project understood, the focus of the project was then to determine an appropriate way to apply the proposed priority scheme.

It is clear that in order to apply the priority scheme, a method was needed to indicate which cells have been modified by a level of priority rule. A new version of CD++ which features capabilities of assigning multiple state variables per cell was introduced. In the version of CD++ used in the second assignment, each cell was represented by only one state variable. This new version allows for an implementation that only requires one layer. In the previous version additional layers were added to incorporate other states. By removing these layers and creating state variables, this is why the new version reduces to one layer.

This initial goal of this project was to use this new version of CD++ to perform the simulation of the model using state variables. The implementation was started using the original version of CD++ as there was difficulty using the new version at first. I wanted to get a functioning model using the original version, and then alter the rules to work with the new version. However, due to the complexity of the model, there were many unforeseen problems which took much longer to get the model to behave appropriately. The time constraint of the project, along with the addition of time it would take to change an implementation involving numerous complex rules to an implementation using state variables was infeasible. Because of this, the implementation was done using the original version.

# DEVS Formal Specification

DoorController = < S, X, Y, δint, δext, λ, ta >

* X = {inHall, inRoom}
* Y = {outHall, outRoom}
* S = {Idle, Perform}
* δint (Perform) = Idle
* δext (Idle, port = inHall) = Perform, fromRoom = 0
* δext (Idle, port = inRoom) = Perform, fromRoom = 1
* ta (Idle) = INFINITY
* ta (Perform) = 50 msec
* λ (Perform, fromRoom = 0) = outHall
* λ (Perform, fromRoom = 1) = outRoom

Figure : DEVS Graph of Door Controller

# Cell-DEVS Formal Specification

**Neighbourhood**

Figure 4 is a graphical representation of the neighbourhood. The neighbourhood has been increased from the proposed Von Neumann neighbourhood in the paper [1] to the neighbourhood in Figure 4 to avoid collisions. Cell (0,0) represents the core cell where the pedestrian can move to the adjacent cell either north, south, east or west based on the rules specified above.

Figure : Model Neighbourhood

**Room = <Xlist, Ylist, I, X, Y, ƞ, N, {r,c}, C, B, Z, SELECT>**

* Xlist = Ylist = { (9,4); (9,5); (0,0)...(0,9) };
* I = <Px,Py>, with Px={<X(9,4),1>,<X(9,5),1>,<X(0,0),binary>...<X(0,9),binary>}; }; Py={<Y(9,4),2>,<Y(9,5),2>};
* X = Y = {0,1,2,3};
  + 0 means unoccupied
  + 1 means up walker
  + 2 means down walker
  + 3 means obstacle
* Ƞ = 17;
* N = {(0,0), (0,1), (0,-1), (1,-1), (1,0), (1,-1), (2,-1), (2,0), (2,-1), (-1,-1), (-1,0), (-1,1), (-2,-1), (-2,0), (-2,1),(0,-2),(0,2)};
* r = 10; c = 10;
* C = {Cij | i ϵ [0,9], j ϵ [0,9]};
* B = { Cij | Cij ϵ C ˄ (i=1 ˅ i=r) ˄ (j=1 ˅ j=c)};
* Z = Inverse neighbourhood of N
* SELECT = {(0,0), (1,0), (0,1), (0,-1), (-1,0)};

**Hall = <Xlist, Ylist, I, X, Y, ƞ, N, {r,c}, C, B, Z, SELECT>**

* Xlist = Ylist = { (0,2); (0,3); (9,0)...(9,5) };
* I = <Px,Py>, with Px={<X(9,4),2>,<X(9,5),2>, <X(9,0),binary>...<X(9,5),binary>}; Py={<Y(9,4),1>,<Y(9,5),1>};
* X = Y = {0,1,2,3};
  + 0 means unoccupied
  + 1 means up walker
  + 2 means down walker
  + 3 means obstacle
* Ƞ = 17;
* N = {(0,0), (0,1), (0,-1), (1,-1), (1,0), (1,-1), (2,-1), (2,0), (2,-1), (-1,-1), (-1,0), (-1,1), (-2,-1), (-2,0), (-2,1),(0,-2),(0,2)};
* r = 10; c = 6;
* C = {Cij | i ϵ [0,9], j ϵ [0,5]};
* B = { Cij | Cij ϵ C ˄ (i=1 ˅ i=r) ˄ (j=1 ˅ j=c)};
* Z = Inverse neighbourhood of N
* SELECT = {(0,0), (1,0), (0,1), (0,-1), (-1,0)};

# Project Implementation

This section will present the pseudo code for the implementation of the rules of the cellular automata.

## Entrance Rule

Assign a value of 2 to the pedestrians who are entering the hall from the south and walking north, if the cells receive external inputs, and their current values are either 0 or 1.

Assign a value of 1 to the pedestrians who are entering the room from the north and walking south, if the cells receive external inputs, and their current values are either 0 or 2.

## Exit Rule

When an up walker (cell with the value of 1) reaches the most north part of the room, exit the simulation (cell becomes 0).

When a down walker (cell with the value of 2) reaches the south most part of the hall, exit the simulation (cell becomes 0).

## Pedestrian Behaviour Rules

This section will define the behaviour of a pedestrian in the room and in the hall, except for the rows where the doors are located. The rows with the doors are different zones, with different rules. These rules will be defined in section 8.4. All of the rules below are pseudo code specified for the up walker. The same logic applies to the down walker. To make the simulation more interesting and realistic, obstacles were added as a third component to the simulation.

**Rule 1 – No walker/obstacles ahead, move forward**

The upper adjacent cell is unoccupied, the up walker will select it to move into whether his left and/or right adjacent cells are occupied or not. To avoid two pedestrians colliding, this rule will be extended to: move to the upper adjacent cell if the upper adjacent cell is unoccupied, and the upper adjacent cell to that one is not a down walker. This allows the up walker to move forward one cell if an obstacle lays in the cell two spaces ahead. This rule has the highest priority.

**Rule 2 – Walker/obstacle ahead and right side available, move to the right side if possible**

The upper adjacent cell is occupied by another walker or obstacle, move to the right adjacent cell if that cell is unoccupied, and another walker isn’t going to take that cell in the next time-step. This rule has the second highest priority.

**Rule 3 – Walker/obstacle ahead and to the right, move to the left side if possible**

The upper and right adjacent cells are occupied by other walkers or obstacles, move to the left adjacent cell if that cell is unoccupied, and another walker isn’t going to take that cell in the next time-step. This rule has the third highest priority.

**Rule 4 – Avoid collision, move right if down walker is vying for the same cell**

When an up walker and a down walker are trying to walk to the same cell in the next time-step, both walkers move to the right adjacent cell if that cell is unoccupied, and another walker isn’t going to take that cell in the next time step. The purpose of this rule is to avoid collision. This rule has the fourth highest priority.

**Rule 5 - Walker/obstacle ahead, to the right, and to the left, move backwards if possible**

The upper, right, and left adjacent cells are occupied by other walkers or obstacles, move to the adjacent cell backwards if that cell is unoccupied, and another walker isn’t going to take that cell in the next time-step. This rule has the fifth highest priority.

**Rule 6 – Default**

If none of the other rules execute, the walker does nothing and waits where they are. This rule also contains the behaviour for the obstacles, as they never move.

## Pedestrian Behaviour Rules at the Boundary

This section will define the rules for all of the pedestrians that reach boundary of the room or the hall components. This boundary represents another zone in the model consisting of all the cells in the row where the doors are. Typically when pedestrians of the western culture are walking, they step right when they are faced with a collision. This behaviour is modeled in section 8.3. However, in order to have the pedestrians walk towards the door (not just stepping right) a new zone was implemented at the boundary to accomplish this. The following are the rules for pedestrians at the boundary of the Hall. The same logic applies to the rules for the Room.

**Rule 1 – Up walker at the door, exit hall**

The up walker enters one of the door cells (ie. value of 1 on cell (0,2) or (0,3)). To represent the walker exiting the hall, in the next time step the cell will become empty. This rule also sends a value of 1 to the doors output port which is connected to the input port of the Door Controller DEVS model. This rule has the highest priority.

**Rule 2 – Enter boundary**

There is an up walker at the boundary, have them walk forward one cell into the boundary in the next time-step, provided that that cell is empty. This rule has the second highest priority.

**Rule 3 – Up walker in boundary to the right of the door, walk left if possible**

There is a walker in the boundary, and they are on the right side of the door, move to the left adjacent cell if that cell is unoccupied, and another walker isn’t going to take that cell in the next time-step. This rule has the third highest priority.

**Rule 4 – Up walker in boundary to the left of the door, walk right if possible**

There is a walker in the boundary, and they are on the left side of the door, move to the right adjacent cell if that cell is unoccupied, and another walker isn’t going to take that cell in the next time-step. This rule has the fourth highest priority.

**Rule 5 – Down walker that entered hall from door, leave boundary**

A down walker that has entered the hall boundary from the room through the door, they leave the boundary in the next time step if the adjacent cell below them is empty.

**Rule 6 – Default**

If none of the other rules execute, the walker does nothing and waits where they are. This rule also contains the behaviour for the obstacles, as they never move.

## Door Entrance Rule

When the pedestrians leaves a room they go through the Door Controller DEVS model which takes 50ms to generate an output. In order to keep the walkers walking on the same time step, the value generated at the output of the Door Controller becomes the value of the cell 50ms later (50ms + 50ms = 100ms). Since the update of the cells is done every 100ms, the walkers travelling from room to room will also be synchronized with the other cells.

# Testing

This section will display the testing results. The Cell-DEVS animation tool will be used to demonstrate the results. A video demonstrating the simulation can be found at the following link:

<http://www.youtube.com/watch?v=1TNFfMQd4SQ>

## Entrance Rule Testing:

These tests are carried out to verify that pedestrians enter at the most northern cells of the room, and at the most southern cells of the hall.

Figure : Room Entrance Test

Figure 5 demonstrates that pedestrians walking south enter at the most northern cells of the room.

Figure : Hall Entrance Test

Figure 6 demonstrates that pedestrians walking north enter at the most southern cells of the hall.

Figure : Entrance Rule Advanced Test

Figure 7 demonstrates a more complex scenario. Figure 5 shows what happens when pedestrians enter freely, while Figure 7 shows the scenario where there will be an up walker (orange) that will enter the space that would otherwise produce a down walker (green) in the next time-step. In this case the up walker has priority and enters that cell. The same applies to the pedestrians entering the hall, but to save space this was omitted.

As shown in the course of simulation, entrance behaves as expected, no collisions occur, nor any sight of losing of a cell.

## Exit Rule Testing:

Please refer to Figure 7 to see the illustration of the up walkers (orange) leaving the north of the room. The same applies to the down walkers leaving the south of the hall, but to save space this was omitted. Over the course of the simulation, the behaviour of the rule functions well.

## Pedestrian Behaviour Rules Testing:

**Rule 1 Testing:**

These tests are carried out to verify that pedestrians move forward, provided that there are no obstacles or walkers directly in front of them.

Figure : Rule 1 Test

Figure 8 displays all of the cases for walking forward. It demonstrates that a walker with no one within two cells ahead of them will move to the cell in front of them. It also demonstrates that since there are no two pedestrians going for the same cell in the next time-step, they move forward even though there are obstacles or walkers of the same type two cells ahead. This is as expected and no collisions result.

Over the course of the simulation, the rule functions very well.

**Rule 2 Testing:**

These tests are carried out to verify that when a walker has an obstacle or pedestrian directly in front of them, they move to the right if it is possible. Figures 9 and 10 demonstrate different scenarios for this rule.

Figure : Rule 2 Basic Test

As seen in Figure 9, the pedestrians move right when they are blocked by either another pedestrian or obstacle directly ahead of them.

Figure 10 demonstrates a more complex test for this rule. In the scenario on the bottom left of the figure, the orange cell (up walker) has another pedestrian in front of it. However, since the down walker (green) will be going to the right of the orange cell in the next time-step, the orange cell waits because the down walker has the right of way. Another collision is avoided.

Similarly, in the upper scenario on Figure 10, the green (down walker) cell has a pedestrian in front of it, and an opening to its right. However since the up walker has precedence and will take that cell in the next time-step, green waits.

Finally, even though there are two obstacles NE, and SE of the other orange cell, they pose no threat to a collision in the next time step, so the up walker moves right, in between them.

Over the course of the simulation this rule behaves beautifully.

Figure : Rule 2 Advanced Test

**Rule 3 Testing:**

These tests are carried out to verify that when a walker has an obstacle or pedestrian directly in front and to the right of them, they move to the left if it is possible. Figures 11 and 12 demonstrate different scenarios for this rule.

Figure : Rule 3 Basic Test

As seen in Figure 11, the pedestrians that have obstacles and/or pedestrians in front of them, and to their right, move to the left when the left cell is available and not going to be taken by another pedestrian in the next time-step.

Figure 12 shows a more complex scenario for this rule. In the bottom left scenario of the figure it shows both an up walker (orange) and a down walker (green) with obstacles / pedestrians in front of them and to their right. The orange cell does have the left cell available, but it also has to check if there will be another pedestrian going for that cell as well in the next-time step. In this case a down walker will be going for it, so the orange cell will wait since the down walker has priority. The green cell has two obstacles in the cells NE, and SE of it and the left cell proportional to it is available. Since these obstacles pose no threat of collision in the next time step, the down walker is able to move left. The upper scenario behaves the same way except with the roles reversed.

Over the course of the simulation this rule behaves as expected.

Figure : Rule 3 Advanced Test

**Rule 4 Testing:**

These tests are carried out to verify that pedestrians that are going for the same cell in the next-time step move right if it is possible to avoid collisions. Figures 13 and 14 demonstrate different scenarios for this rule.

Figure : Rule 4 Basic Test

Figure 13 shows two different scenarios. The first scenario involves two sets of pedestrians that have empty cells in front of them, but if they both move forward, there will be a collision. In the bottom left of the figure it shows that the cells to the right of the pedestrians are available, so they move right to avoid collision. In the upper portion of the figure the down walker (green) has an obstacle to the right so it waits and stays where it is.

Figure : Rule 4 Advanced Test

In the lower scenario on Figure 14, the down walker (green) with a potential collision has an available cell to the right with no one fighting for it, so it moves there in the next time step. However, the up walker (orange) has an available cell to its right, but a down walker in its NE cell that will move to its right cell in the next-time step. As observed, to avoid collisions, the orange cell waits, and does not move right.

In the upper scenario on Figure 14, the up walker has obstacles to its NE and SE cell. Since these obstacles pose no threat of collision in the next-time step, the up walker is able to move to the right cell. The down walker has an available right side, but since it knows that its fellow down walker is going to take that cell in the next time step, it waits to avoid collisions.

Over the course of the simulation this rule behaves as expected.

**Rule 5 Testing:**

These tests are carried out to verify that when a walker has an obstacle or pedestrian directly in front of them, to the right of them, and to the left of them, they move backwards if it is possible. Figures 15 and 16 demonstrate different scenarios for this rule.

Figure : Rule 5 Basic Test

As seen in Figure 15, the pedestrians that have obstacles and/or pedestrians in front of them, to their right, and to their left move backwards when the adjacent cell behind them is available, and not going to be taken by another pedestrian in the next time-step.

Figure : Rule 5 Advanced Test

Figure 16 shows a more complex scenario for this rule. In the bottom left scenario of the figure it shows an up walker (orange) that has an obstacle to the right and left of them, as well as a down walker in front of them. The cell below them is empty, but it also has to check if there will be another pedestrian going for that cell as well in the next-time step. In this case an up walker will be going for the same cell in the next time-step, so the orange cell will wait since the other walker has priority. The upper scenario behaves the same way, except that it demonstrates that this rule also avoids collision when there is a walker 2 cells away going for that same cell in the next time-step.

Over the course of the simulation this rule behaves as expected.

**Border Testing:**

This test is carried out to verify that when pedestrians get to the sides of the rooms, that they behave as if there is a wall there.

Figure : Border Test

As seen Figure 17, the down walker (green) is facing the up walker (orange). Typically the down walker would walk right, but since it is at the wall, it walks left. Also, in the scenario on the right side of the figure, there is a down walker at the wall with an obstacle ahead of them, and to their left. The only option is to walk backwards.

Over the course of the simulation the pedestrians behaved very well and treated the border as if there was a wall there, which is as expected.

**Rule 6 Testing:**

This test is carried out to verify the default case that when none of the other rules apply, the pedestrian does nothing and waits where they are. This also covers the behaviour for obstacles.

Figure : Default Rule Test

The left scenario on Figure 18 shows an up walker (orange) who has obstacles / pedestrians to its right, left, front and behind adjacent cells. Since none of the other rules apply, the up walker remains where it is until to coast is clear. The right scenario demonstrates an obstacle in front and to the left of the up walker, as well as another up walker to its right. The cell below it is empty, however since the up walker to its bottom left is going to take that cell in the next time-step, it remains where it is.

For the purposes of this simulation, the obstacles are meant to be permanently configured to the ground and not moved. This rule provided this.

Over the course of the simulation this rule behaves as expected.

## Pedestrian Behaviour at the Boundary Testing

**Entering the Boundary Testing:**

This test was carried out to verify that the pedestrians are able to enter the other zone, which is the boundary of the room where the doors are located.

Figure : Entering Boundary Test

Figure 19 demonstrates the up walkers are in fact able to enter the boundary of the hall, and that down walkers are able to enter the boundary of the room. The boundary is represented by a blue rectangle. The larger rectangle is the boundary of the room, whereas the smaller rectangle is the boundary of the hall.

This rule performs well over the course of the simulation.

**Leave the Boundary Testing:**

Figure 19 also demonstrates walkers leaving the boundary. This becomes useful when pedestrians travel from one room to the other. Once they have transferred to the other room’s boundary, they must leave. Over the course of the simulation, the rule performs well. However, due to the complexity of this simulation, there was not enough time to guarantee an implementation that collisions will not occur anymore.

**Move Towards Door Testing:**

These tests are carried out to verify that once a pedestrian is in the boundary, they move towards the door.

Figure : Move Towards Door Basic Test

Figure 20 demonstrates that once pedestrians from the hall, who are trying to walk to the room (orange), walk towards the door in the hall. The same is true for the pedestrians in the room trying to walk to the hall.

Figure : Move Towards Door Advanced Test

Figure 21 demonstrates that when a pedestrian in the boundary that is supposed to move towards the door, first checks to see if another walker is going to move to that cell. If there is another walker going to move to that same cell, the walker in the boundary waits to avoid collision.

Over the course of the simulation this rule performs well.

## Entering and Exiting the Door Testing

These tests are carried out to verify that when a pedestrian reaches the door, they will exit the room through the door and enter the other room at the door.

Figure : Entering / Exiting Door Test 1

Due to the way that the CellDEVS animation tool displays each of the components, the doors do not line up, so I displayed the cells representing the doors with arrows pointing to them. In the room the cells representing doors are (9,4) and (9,5), and in the hall they are (0,2) and (0,3).

Figure 22 demonstrates that when there is an up walker (orange) at door2 in the hall, in the next time-step they leave the hall and reappear at door2 in the room. The same is true for the down walker (green) in the room at door1, in the next time-step they leave the room and reappear at door1 in the hall. This is as expected.

Figure : Entering / Exiting Door Test 2

Figure 23 demonstrates the pedestrians walking through the other door from in Figure 22, and shows that when pedestrians can walk to the other room without collision, they do so and it works well.

However, due to the complexity of this situation at the door, and the time restraint of the project, collisions will occur at the door.

# Experiment Results

This section presents the experiment results including the results of varying the pedestrian’s parameters.

Over the course of the simulation, the pedestrians behave reasonably well. As seen in the simulation videos, they begin by entering at the proper section of the room or hall while avoiding collisions. They begin walking about their business while still remaining collision free until they get to the boundary of the room or hall which includes the row of cells where the doors are located. Due to the complexity of the model, collisions will occur around the doorway at both the hall and room. When collisions do not occur, the pedestrians are observed to walk from one area to the other. They then exit the boundary and begin walking towards the opposite end of the area. Once the pedestrians walk to the end of the area they are removed from the simulation. Since the pedestrians are generated randomly throughout the simulation, there is always a constant flow of pedestrians walking from one area to the other even though they get removed from the area. Overall the simulation works well, but in order to be complete future work would include making the model function so that it is collision free.

There were several simulations done that included varying the parameters of the pedestrian generators to see the effects of at the doorway. Since collisions occurred in all simulations at the door, the behaviour of the pedestrians at the door were similar except that where there was more or less pedestrians walking around the areas. In the case where there were more pedestrians in the room generated, it appeared that the doorway was constantly used by the pedestrians leaving the room as expected. Since the room was larger than the hall, there were more pedestrians generated in the hall, and made it seem that there were always more people walking from the room to the hall. To offset this, and make the pedestrians in each room more balanced, the pedestrian’s generator values in the room were set higher, and the halls parameters were set lower. This also resulted in fewer collisions and in my opinion the best overall simulation.

The following are links to videos displaying the results of different parameter values:

Similar mean values (0.8 – 1.8) in both hall and room:

<http://www.youtube.com/watch?v=I77-zLCRpS4>

Low mean values in hall (0.4 – 0.9):

<http://www.youtube.com/watch?v=QfbfRvWOP2c>

High mean values in room (1.8 – 2.6) and low mean values in hall:

<http://www.youtube.com/watch?v=1TNFfMQd4SQ>

# Conclusion

To conclude, this rule set behaves reasonably well. Multiple simulations were performed, and collisions were only observed at the doorway of the model. The pedestrians flowed very naturally. Multiple simulations were also performed while varying the pedestrian’s generator values and it was noticed that by increasing the generators values of the pedestrians entering the room and lowering the generators values of the pedestrians entering the hall resulted in fewer collisions at the door, as well as represents the best looking simulation.

Since time was a major factor in this project, the simulation was carried out using the version of CD++ that was used in the second assignment. Future work would be to implement this model using the new version of CD++ where and implementation with state variables or ports to make the simulation more efficient. Other future work would be to add more rules to remove the collisions of the pedestrians at the doorway.

Much knowledge was learned throughout this project, and overall this project was a success.

# References

[1] F. Weifeng, Y. Lizhong, F. Weicheng. “Simulation of bi-direction pedestrian movement using a cellular automata model”, 2002.

[2] Gabriel A. Wainer. “Discrete-Event Modeling and Simulation: A Practitioner’s Approach”, CRC Press, April 30, 2009.