GEORGIA INSTITUTE OF TECHNOLOGY

CSE 6730 Project #1: Cellular Automata (CA) Simulation of Pedestrian Traffic Leaving Stadium

Dylan Andrew Crocker
dcrocker3@gatech.edu

March 4, 2016

1. INTRODUCTION

In this project a model and simulation is developed to study the egress of pedestrians from the area around Georgia Tech's Bobby Dodd Stadium after an event. More specifically, the project will model the egress from one exit of the stadium and simulate dispersion of people from that exit. The simulation models pedestrians only (does not include vehicles) and is based on the concept of Cellular Automata (CA). The report is structured as follows: section 1 gives background on CA and what is found in the literature for CA models of pedestrians, section 2 will detail the development of a conceptual model for this project, section 3 and 4 describe simulation software development, and finally section 5 will present simulation results.

1.1. CELLULAR AUTOMATA (CA)

Cellular Automata (CA) are set of machines which are capable of changing state based on inputs (automata) arranged in a grid (each to a cell)[1]. The states of the cells are updated in discrete time steps using a state transition function that defines the rules of interaction between a cell and its neighbors. This method can be used to model large complex systems and effectively model group relationships [1].

A basic time stepping CA simulation is described in [1]. The framework of the time stepping simulation is shown in Algorithm 1 below. First the system states are initialized and then observed (e.g., visual graph of pedestrians on a map); then the system states are updated for one discrete time step and again visualized. The update processes is then stepped for discrete time steps. this time stepping model is implemented in the simulation described in this report.
Due to the fact that time, space, and model states are all discrete, only a finite number of possible states and transitions exist. However, an exhaustive search to evaluate all possible states requires exponential running time, which is impractical in most cases. Nevertheless, the discrete nature of the simulations allow CA to model complex and nonlinear systems such as pedestrian traffic flow.

Algorithm 1: Basic CA time stepping code from [1]

```
initialize();
observe();
foreach step in TimeSteps do
    update();
    observe();
end
```

1.2. Pedestrian CA Models in the Literature

The paper *Cellular automata microsimulation for modeling bidirectional pedestrian traffic flow* [2] presents a model for the general case of bidirectional pedestrian flow. This paper built off of its authors' previous work on unidirectional flow which would have been more applicable to this particular project (I was unable to obtain a copy of that report for review). However, the authors summarized much of the unidirectional model in order to describe their additions for bidirectional flow. The new rule set defines rules for avoiding collisions with other pedestrians moving in the the opposite direction. They also define rules that cause the pedestrians to create flow lanes (like humans do naturally). A time step of 1 second and a cell size of $0.21m^2$ ($0.457m$ sides) was used and the speed of pedestrians was varied. The speed of pedestrians was distributed as 5% fast (5 cells/sec), 90% normal (3 cells/sec), and 5% slow (2 cells/sec).

In *Simulation of Evacuation Characteristics Using a 2-Dimensional Cellular Automata Model for Pedestrian Dynamics* [3] the authors use CA to model high density pedestrian evacuations. Their model uses a 1/3 m x 1/3 m cell with one pedestrian allowed per cell. This is somewhat smaller than that in other literature since it is modeling higher density situations. A "floor field" is implemented in order to "attract" pedestrians to the exits. Additionally, a "friction coefficient" is utilized when pedestrians are compacted closely with other pedestrians in order to model the decrease of walking speed with the increase of pedestrian density. Further, an attractive force (desire for travel in the direction) and a repulsive force (to avoid collision with another pedestrian) are both calculated in order to determine a pedestrian's next movement. When multiple pedestrians want to move to the same cell, one is selected randomly and the others remain unchanged. This model is used to simulate several evacuation scenarios including a small stadium.

An in depth discussion of modeling pedestrian flow with CA is presented in *Simulation of pedestrian dynamics using a two-dimensional cellular automaton* [4]. In this model, pedestrians have a strong repulsive force at close distances in order to prevent multiple pedestrians from occupying the same cell. At larger distances there is an attractive force (lanes, groups, curiosity, etc.). Additionally, pedestrian speed is limited to one cell per time step in order to more easily prevent collisions. Cells are 40 cm x 40 cm which is the average space occupied by a pedestrian. The average pedestrian speed is 1.3 m/s, which at a rate of one cell move per time step (0.4 m) correlates to time steps of 0.3 sec (a reference is provided in the paper for the given empirical data). Each pedestrian is given a 3x3 movement matrix that calculates the probabilities of moving to the next cell given the pedestrian's preferred direction of travel. These will overlap with other pedestrians and the resulting conflicts are resolved using the probabilities. The pedestrian model presented is intentionally kept simple, the complex interactions are implemented through a "floor field". This field is analogous to an electric force field which effects the flow of electrons. This field governs pedestrian interactions from a distance by storing historical information (i.e., lanes are created based on previous traffic).

In *Simulation of competitive egress behavior: comparison with aircraft evacuation data* [5] the authors model evacuations using a very similar setup to that in [4]. They use a "floor field" model to determine

pedestrian movement probability. The difference here is the floor field is not dynamic but rather static and used to draw the simple particles (pedestrians) to the evacuation exits. They use a binary factor in their movement calculations to ensure a probability of zero for movement to forbidden cells. They use a friction parameter to describe conflicts. This is very important in their model since evacuation involves tight cramming of pedestrians.

A simulation of pedestrians evacuating a room is presented in *Cellular automaton model for evacuation process with obstacles* [6]. Again a static floor field is used to draw the pedestrians to the exits. The movement decisions are determined based on the static floor field and interactions with other pedestrians. The situation modeled in this paper is not unlike modeling pedestrian egress from a stadium (basically a more complicated version of the same thing). The concepts in this paper helped me to understand the "floor field" concept and greatly influenced the development of the model presented in this report.

The authors of *Pedestrain cellular automata and industrial process simulation* [7] utilize pedestrian cellular automata simulation techniques to model the dynamical system of a manufacturing floor. Their model uses a combination of both static and dynamic floor fields. An interesting aspect of this model is that each individual pedestrian stores their own version of the dynamic field. This allows individuals to adapt to the simulation environment on the fly and change course (take a different route, go where help is most needed, etc.). Each pedestrian does not necessarily follow their predecessor like other "lane forming" CA models using dynamic floor fields.

2. Conceptual Model Development

The conceptual model is a representation of the System Under Investigation (SUI) using some type of formalism. It is an abstraction of a real world system and it is essential to the successful development of a simulation program representing the SUI [8]. The role of the conceptual model is shown below in Figure 2.1.

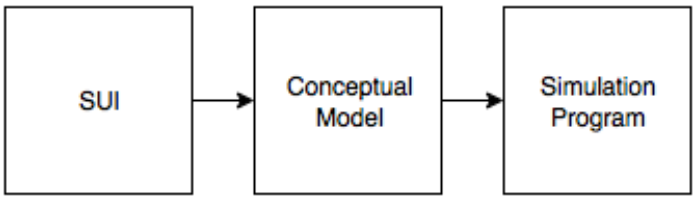

Figure 2.1: The role of the conceptual model

2.1. Objective

For this project, the objective is to model the egress of pedestrians from a stadium (Georgia Tech's Bobby Dodd Stadium). The conceptual model must be generated to represent the SUI which is the pedestrian movements.

2.2. Input

Model inputs includes pedestrian and map information. The number of pedestrians and their characteristics: travel speed and destination. The simulation environment (walkways, destinations, etc.)

must be input as map information. The map information should be input as a grid (for the CA implementation). It must contain definitions of walkways, destinations, stadium exits, and street crossings (crossings can be turned on and off).

2.3. Output

The model output is the simulated egress of the pedestrians (specified during input) through the modeled environment (map information given as an input).

2.4. Content

The model does not include activities inside the stadium; however, the stadium exits do provided inputs to the simulation. Therefore, the stadium and activities within are regarded as exogenous entities for this model. The stadium (Bobby Dodd Stadium) can hold 55,000 spectators [9] which provides an upper limit on the number of spectators to include in the simulation.

The endogenous entities in this model are pedestrians and map objects (walkways, streets, etc.). A queuing model is implemented for the pedestrian entities. They are then placed into the model as space opens up at exit locations (previous pedestrians walk away). Pedestrian entities contain properties for waking speed and selected destination. The walking speed distribution is obtained from research in [10] and is similar to that used in [4]. The walking speeds for each pedestrian are drawn from the normal distribution shown in Table 2.1 and Figure 2.2.[1]

Table 2.1: Pedestrian Walking Speed Distribution [10]

	ft/sec	m/sec
Mean	4.40	1.340
Std. Dev.	0.87	0.265

Map information is represented as a grid of cells that measure $0.25m^2$ ($0.5m$ sides). These dimensions were chosen as a simplification of that used in [2]. Each cell is given a type that represents information relevant to this simulation. More specifically, the cells are identified as walkways, crosswalks, streets, or prohibited (no walking allowed).

2.5. Assumptions and Simplifications

Pedestrian entities are assumed to always travel toward their predetermined destination (they do not change destinations during the simulation) and to stay on the allowed paths. The pedestrian model does not include personal attributes, although some attributes may correlate with pedestrian walking speed (e.g., age), this is taken into account through the walking speed distribution. Additionally, diagonal travel is not modeled. Only movements directly to cells that share a side with the current cell are allowed (North, South, East, and West). Further, pedestrians are assumed to travel at a constant speed if possible (the pedestrians will travel less cells than possible in a time step if something is blocking their route).

The map information is simplified to a 2-D grid (for use with CA). Terrain attributes (incline, material, etc.) are not included. Further, due to the grid architecture, diagonal walkways may only be modeled as a stair step of square cells.

These simplifications/assumptions were made in order to focus on the main part of the simulation and get the simulation running within the alloted time. It is fully understood that this type of model

[1] The speeds are capped at a minimum of 0.153 m/s which is the threshold used in [10]to determine walking.

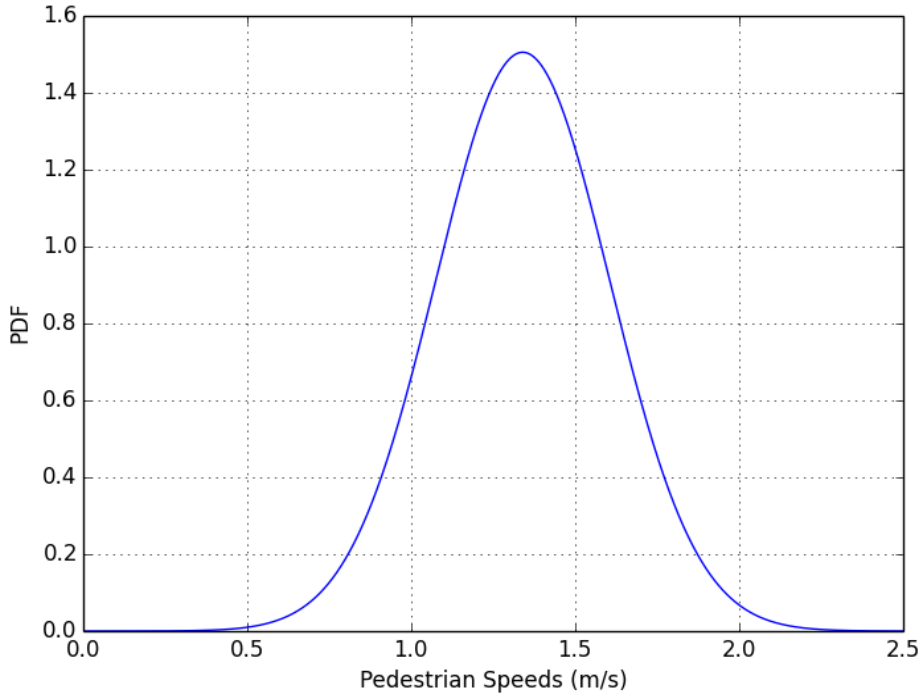

Figure 2.2: PDF of Pedestrian Speeds

is very complex and could have much more detail (especially in the area of pedestrian intelligence). However, the author is confident that much of the essentials to the model were implemented.

3. Simulation Development

Pedestrian movement is a dynamical system and for this project, pedestrian egress from a stadium, it will be modeled as a discrete time dynamical system using CA. The simulation grid is made up of cells that are either "allowed" or "prohibited" for pedestrian traffic. A "floor field" [6] is calculated for each destination and used to influence the movement decisions of the pedestrians. The pedestrians enter the simulation at the cells designated for the stadium exit(s). A simple time stepping loop (see Algorithm 1) is then used to move the pedestrians to their respective destinations. A high level representation of the simulation algorithm is presented below (Algorithm 2).

Algorithm 2: High Level Simulation Algorithm

```
   Data: Map information, number of pedestrians
 1 Define distribution of pedestrian speeds
 2 Define distribution of pedestrian destinations
 3 Create a queue of pedestrian objects
 4 Create map object
 5 Create separate static fields for each destination
 6 for number of time steps do
 7 |   pop pedestrinas off queue and place at stadium exit
 8 |   update pedestrian positions
 9 |   visualize map
10 end
```

3.1. Simulation Grid

A simulation grid is created to represent the desired map divided into cells. Each cell has an attribute that indicates its type. A cell can have any of the types listed below.

1. Prohibited
2. Walkway
3. Street
4. Crossing
5. Simulation Entry point (also allowed walkway)
6. Simulation Exit point (also allowed walkway)

Using these attributes the simulation will be able to make decisions. If a cell is marked as prohibited, pedestrians will not be allowed to move to those locations. Walkways and crossings will be allowed. Crossing cells are identified separately so that the simulation can periodically open and close the crossing to simulate real life traffic scenarios. The simulation will insert and remove pedestrians at entry and exit point cells respectively.

A "floor field" [6] is calculated for each destination by assigning a value of 1 to the destination cells and then incrementing the value of neighboring cells for each step they are away from the destination. This "field" is used to "force" pedestrians through the grid to their desired destination. The calculation of this field is similar to a breadth-first search in graph theory and runs in $O(m+n)$ time (where n is the number of gird rows and m is the number of grid columns). Example floor fields for the map simulated are presented in Appendix C.

Each "cell" object also has attributes for the pedestrian currently occupying the cell. This provides information to the movement calculations (state transitions).

3.2. Pedestrian Objects

The simulation defines a pedestrian object with the following attributes:

1. Coordinates (location on the grid)
2. Destination ID
3. Speed (m/s)

The pedestrian speeds are drawn from the distribution defined in Section 2 which describes the conceptual model. The destination is selected at random from a list of possible destinations defined by the map input. All destinations have equal probability of being chosen.

3.3. Pedestrian Movements (State Transitions)

Each time step of the simulation represents 1 second of time. At each time step pedestrians are placed at the stadium exits. In order to make the simulation more realistic, the stadium exit cells are randomly filled by new pedestrians at the beginning of each time step. This prevents the unrealistic situation of a solid wall of pedestrians exiting the stadium at exactly the same time walking shoulder to shoulder. Once the pedestrian is placed in the simulation grid, the moves are calculated by first calculating the distance the pedestrian may move. The distance d (number of cells) a pedestrian can travel for a given time step is calculated by Equation 3.1. Where s is the pedestrian speed, t is the loop time step (1 second), and r_{prev} is the remainder of the previous distance calculation for the particular pedestrian.

$$d = \lfloor s * t + r_{prev} \rfloor \tag{3.1}$$

A list of possible cells to which the pedestrian may move is then assembled using an algorithm similar to a breadth-first-search in graph theory. As the algorithm looks for cells that are within the pedestrians movable distance, prohibited cells (including those occupied by another pedestrian) are excluded. A move probability p_m is then assigned to each reachable cell using Equation 3.2. Where $U_{current}$ is the floor field potential at the pedestrian's current cell and U_{new} is the floor field potential at the reachable cell.

$$p_m = exp^{U_{current} - U_{new}} \tag{3.2}$$

The equation is designed such that lower potentials result in higher probabilities. The cell with the highest probability is chosen as the pedestrian's next move. Conflicts (when two or more pedestrians pick the same cell) are mitigated by taking the pedestrian that wants it the most (highest probability) while the rest select the next cell on their list and the process repeats (ties are given to the first pedestrian to pick the cell). The cell currently occupied by the pedestrian is left in the list of possibilities so that if no lower potential cells are available the pedestrian will stay put (useful for implementing waiting due to traffic signals). In the case where a pedestrian is confronted with multiple cells of equal probability, a random choice is made.

Once a pedestrian has reached its destination it is removed from the simulation. The simulation terminates once there are no longer any more pedestrians left in the simulation grid.

4. Description of Simulation Software

4.1. Architecture

The simulation software was written in Python and is based on the example CA simulation given in [1]. It consists of a time stepping loop that updates the states of the cells (pedestrian locations) similar to that shown in Algorithm 1. Object Oriented (OO) practices are implemented by the software. Objects are used to represent pedestrians, the simulation grid, cells in the grid, and the simulation itself.

4.2. Interfaces

The code is contained in a single file that can be run via the Python interpreter. The simulation requires as inputs the number of pedestrians to create and a path to the map text file (see Appendix B) which is used to create the simulation grid. The simulation was built to be very general in that different pedestrian walking scenarios could be simulated simply by changing the map file and the desired number of pedestrians. The interface is very simple and provides good usability.

5. Simulation Results

This section details the simulation of pedestrian egress from Georgia Tech's Bobby Dodd stadium.

5.1. Simulation Grid

Due to the fact that author has never seen the stadium in person, map data from the Internet[2] was utilized for building the representation in the simulation grid. A screen shot of the map used to generate the simulation grid is shown in Figure 5.1 while the simulation grid created from the map is shown in Figure 5.2.

[2] http://binged.it/1TLWUNr

Figure 5.1: Aerial map of Bobby Dodd stadium and surrounding area.

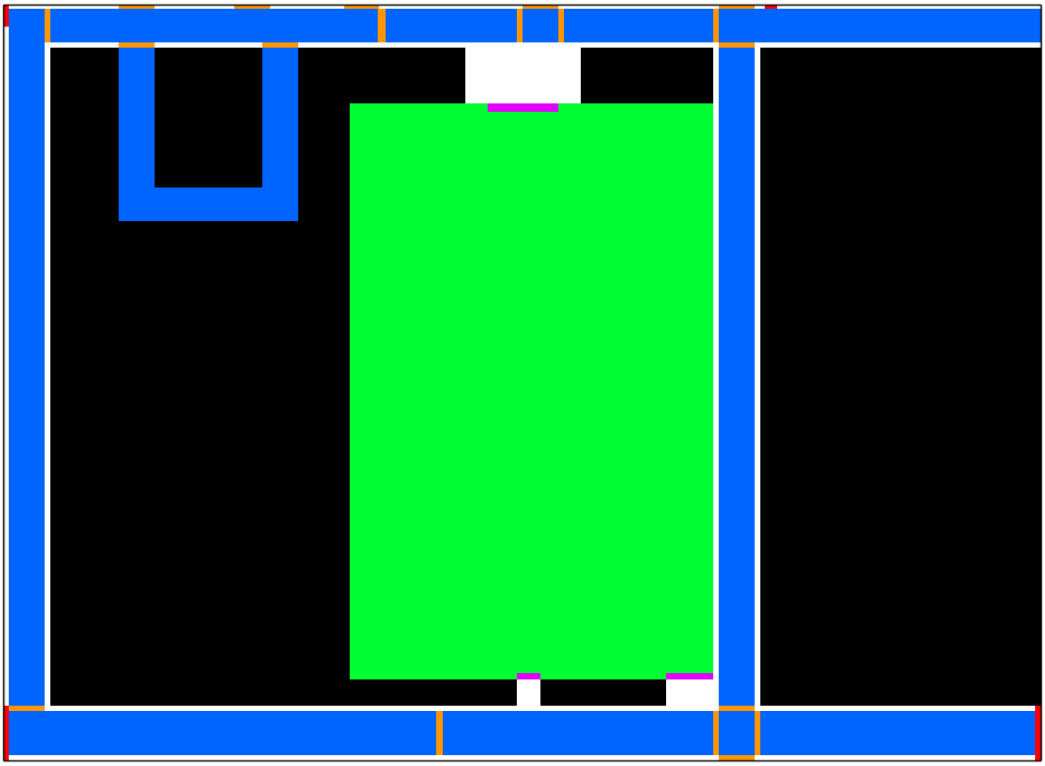

Figure 5.2: Simplified map of Bobby Dodd stadium and surrounding area implemented in the simulation. Legend: red = exit, purple = stadium exit, white = walkway, orange = crossing, black = prohibited, green = stadium

Note the map was substantially simplified in order to get the most relevant data in a reasonable amount of time as generating the map file (see the map file used in Appendix B) was significant undertaking.

As described in Section 3.1, the simulation calculated "floor fields" for each destination. The fields gave lower values to cells closer to the destination. The movement calculations utilized these fields (see Section 3.3). The floor fields for the destinations shown in Figure 5.2 are presented in Appendix C.

5.2. Pedestrian Queue

As discussed in Section 2.4, the stadium can hold 55,000 spectators. The simulation thus generates 55,000 pedestrian objects with random walking speed and destination. The destinations are chosen with equal probabilities and the speeds are drawn from the distribution discussed in Section 2.4. The equal probability of destination selection is verifies by the distribution shown in Figure 5.3. Additionally, the correctness of the distribution of pedestrian speeds is demonstrated by Figure 5.4.

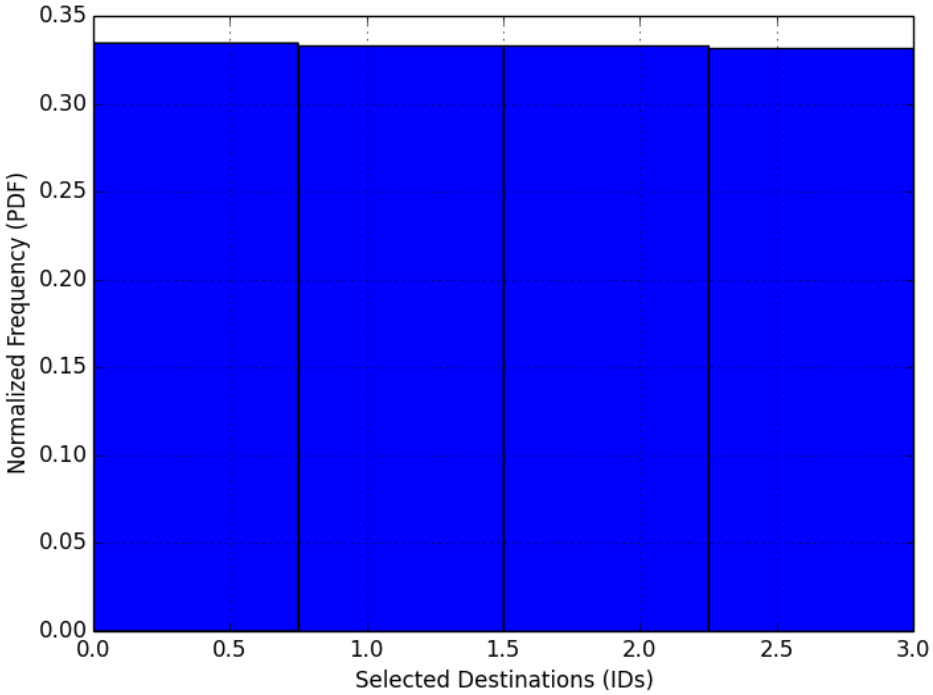

Figure 5.3: Histogram of pedestrian destination distribution.

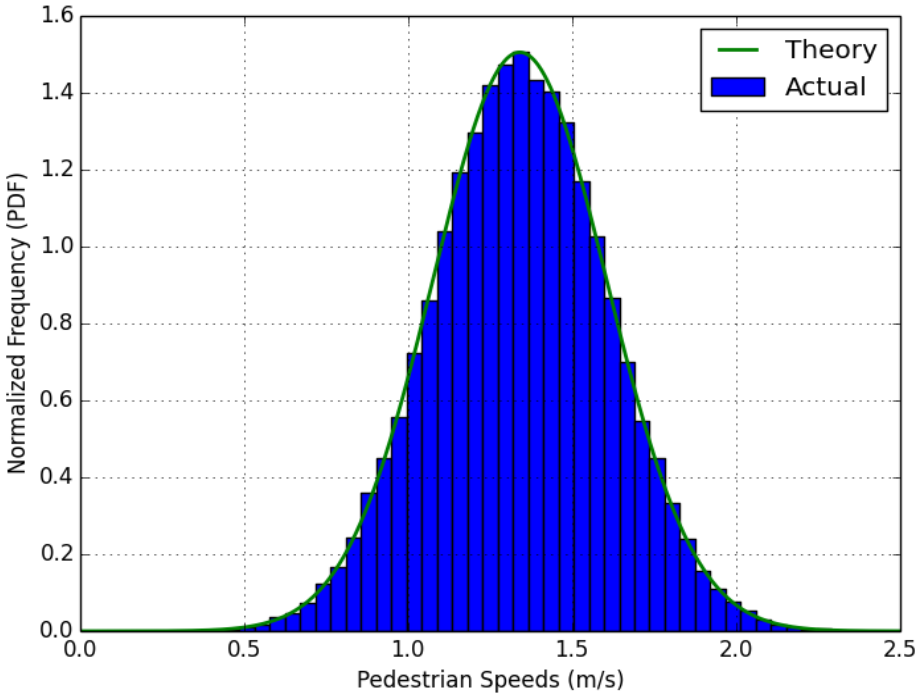

Figure 5.4: Histogram of pedestrian speeds generated by the simulation.

5.3. Simulated Pedestrian Egress

The figures below show the movement of pedestrians along the map toward their respective destinations. Per the project requirements for DL students, the egress from only one exit was modeled (the north exit was selected). The animation feature was utilized to verify the performance of the simulation was as expected.

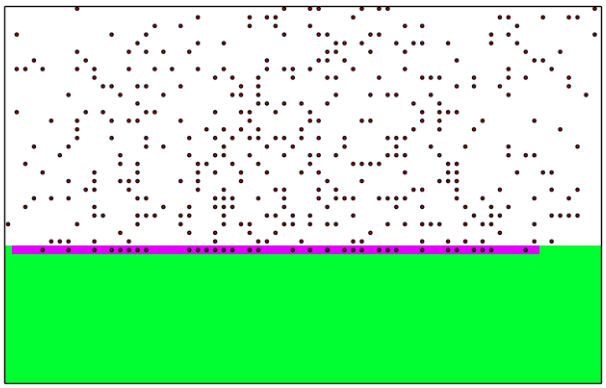

Figure 5.5: Random distribution of pedestrians at North exit.

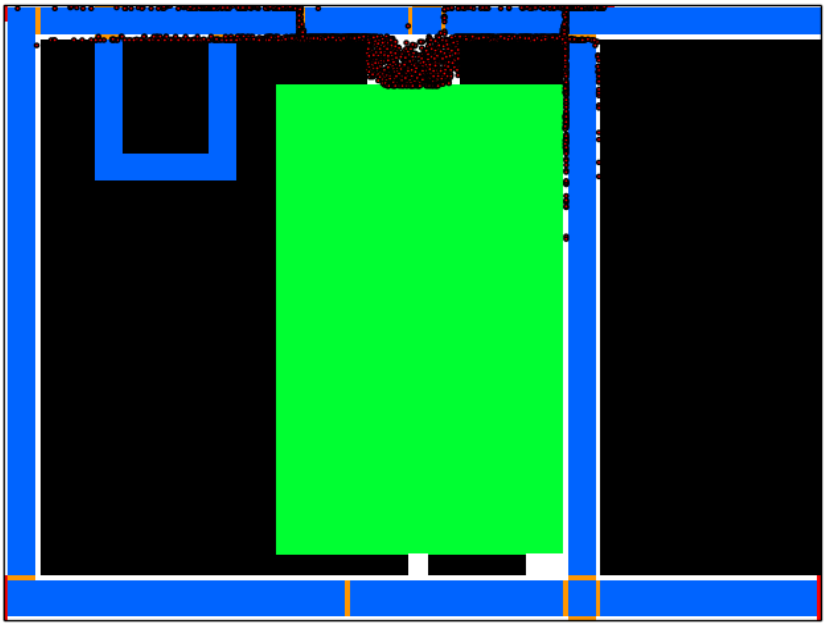

Figure 5.6: Simulation state at 100th time step. Note: The dots representing pedestrians are larger than the grid cells.

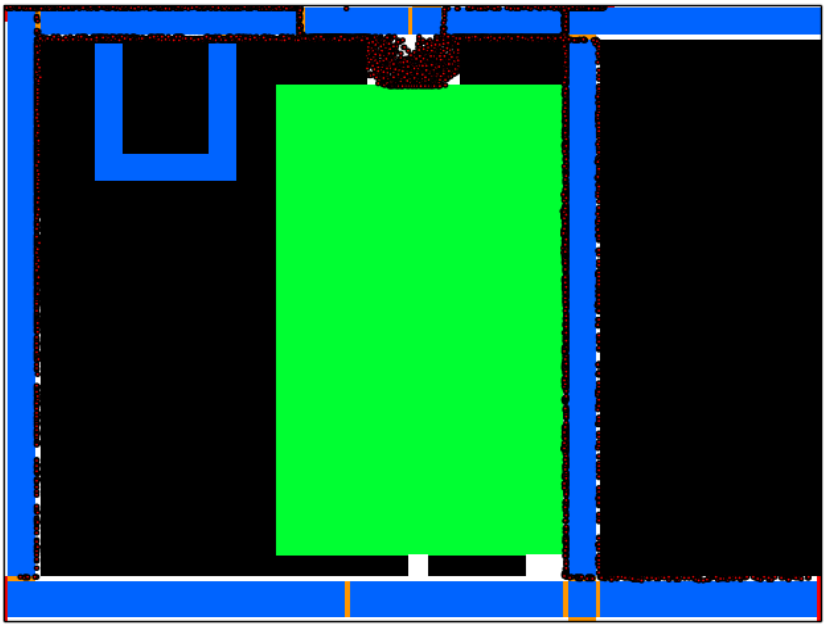

Figure 5.7: Simulation state at 300th time step. Note: The dots representing pedestrians are larger than the grid cells.

5.4. Egress Timing

Multiple runs of the simulation were conducted with the results tabulated below in Table 5.1. Each simulation contained 55,000 pedestrian objects; however only one stadium exit was modeled (the exit was 60 cells wide). The simulations were run on a MacBook Pro with a 2.4 GHz Intel Core 2 Duo processor and 4 GB of RAM.

Table 5.1: Simulation statistics for the egress of 55,000 pedestrian objects from a single, 60 cell wide, stadium exit location.

	Average
Egress Time	49.1 minutes
Execution Time	4.63 hours

The simulation times above represent minimum egress times since all street crossings were left open (allowing pedestrians to cross streets without waiting). The software currently lacks the ability to implement crosswalk signals. As with any modeling and simulation project time constraints limited the number of features added to the simulation. Due to the limit resources (in this case time) decisions had to be made on feature priority. The crosswalk modeling was put lower on the priority list since they are not required to achieve a lower bound on the egress time.

It is desirable to compare simulated results with actual data for simulation verification. However, there was not any such data available to the author (since the author has never actually been to Bobby Dodd stadium). However, the author has attended several large sporting events and the produced result seems to be within the expected order of magnitude.

6. Summary

In this report a Cellular Automata (CA) simulation was developed for simulating the egress of pedestrians from a stadium. A brief CA background and literature review were presented prior to the development of a conceptual model, and simulation software. Code to generate random numbers was developed and validated (see Appendix A). Finally, pedestrian egress from Georgia Tech's Bobby Dodd stadium was simulated and the results presented and validated.

This project was very challenging but provided a great introduction to the realm of modeling and simulation. Due to time constraints, several additional features and updates to the simulator were omitted. They are summarized below in Future Work.

6.1. Future Work

There were several features that were not implemented in the software due to time constraints: path finding for pedestrians traveling in different directions, more sophisticated pedestrian moves (diagonal), and crosswalk signals. Additionally, increases to the simulation speed need to be investigated. The conceptual work for crosswalk signaling logic was developed and is presented here for reference. Loop through the grid once it is created and generate lists of objects representing sets of cells with the same type (stadium exits, destinations, street crossings, etc.). For instance, a list of objects representing the crosswalks would be generated (the objects would contain information about the location, states of cells, etc.). This would allow for modifying the grid on the fly during the simulation. For instance, transient destinations (e.g., bus stop) could be simulated by changing the attributes of the destination cells periodically during the simulation.

As is evident from the results shown in Table 5.1, the simulations took a very long time to run. This indicates speed improvements to the code are desirable before utilizing the simulation for large scale research (many simulations). Efforts were made to develop code that ran efficiently (estimated $O(n*m)$ running time where $m*n$ are the total number of cells in the grid). Never the less, improvements can always be made to software including porting to a compiled language (e.g., C/C++) for improved speed.

6.2. Acknowledgments

The author would like to acknowledge Mr. Guru Srinagesh for the much appreciated review and feedback as well as his willingness to answer many questions during the development of this project.

APPENDIX A RANDOM NUMBER GENERATION

As discussed in the main body of this report, the pedestrian simulation utilizes randomness in several places. Both uniform and normal distributions were required for the simulation. Software algorithms have been created for the purpose of generating pseudo random numbers. Since the generation is deterministic (software algorithm) the numbers are not truly random; however, the algorithms are designed to generate streams of numbers that are not correlateable to the previous samples and hence appear random. The Random Number Generators (RNGs) implemented are discussed in this appendix.

A.1 UNIFORMLY DISTRIBUTED RANDOM VARIANT

The "Minimal" random number generator developed by Park and Miller [11] was selected to generate a uniform random deviate between 0.0 and 1.0. The algorithm is presented in [11] written in C. It was adapted to Python for this project and is presented below.

Listing 1: Minimal Random Number Generator

```
def min_rand(seed=0):

    # Define constants
    ia = 16807
    im = 2147483647
    am = (1.0/im)
    iq = 127773
    ir = 2836
    mask = 123459876

    # Only allow the generator to be seeded once
    if "seed" not in min_rand.__dict__:
        # XORing with MASK allows use of zero and other simple bit patterns
        # for seed.
        min_rand.seed = seed ^ mask

    # Compute idum=(IA*idum) % IM without over-flows by Schrage's method.
    k = min_rand.seed/iq
    min_rand.seed = ia*(min_rand.seed - k*iq) - ir*k
    if min_rand.seed  < 0:
        min_rand.seed += im
    rand_val = am*min_rand.seed  # Convert to a floating result.
    min_rand.seed ^= mask  # Unmask before return.

    return rand_val
```

The generator was tested for the desired behavior using the *Chi-Squared* test [12] as discussed in class (see equation A.1).

$$A_m = \sum_{i=1}^{k} \frac{O_i - E_i}{E_i} \tag{A.1}$$

Where O_i is the number of values in bin i and E_i is the expected value. The calculated value of A_m is 27 which is much less than χ^* of 38.885 for 26 degrees of freedom (27 bins) and $\alpha = 0.05$. This indicates acceptable performance of the RNG. The set of generated random values used for this test is shown graphically in Figure A.1.

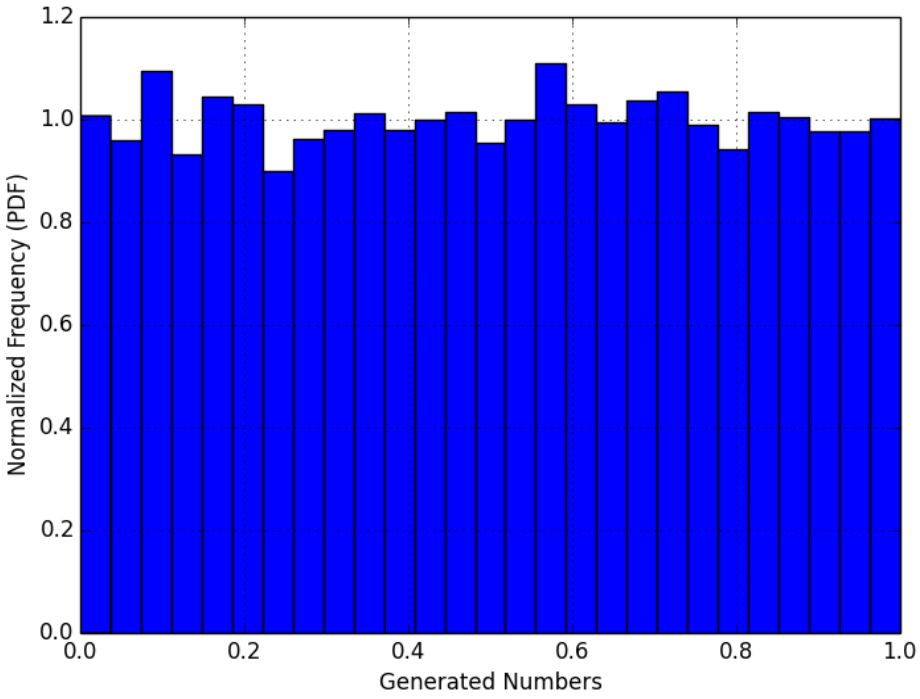

Figure A.1: Histogram of Uniformly Distributed Random Numbers Generated

A.2 Normally Distributed Random Variant

The *Box-Muller* method [11] was selected to generate a normally distributed random variate for use in the simulations developed in this report. A similar method was also briefly discussed in class. The algorithm is presented in [11] written in C. It was adapted to Python for this project and is presented below (note the use of the RNG from Section A.1).

Listing 2: Normal Random Number Generator

```
def gaussian(mu=0, sigma=1):
# Generate a random number from a Gaussian distribution.
# Adapted from Numerical Recipes in C 2nd ed.

    if 'iset' not in gaussian.__dict__:
        gaussian.iset = 0
    if 'gset' not in gaussian.__dict__:
        gaussian.gset = 0

    if gaussian.iset == 0:
        v1, v2, rsq = 0.0, 0.0, 0.0
        while rsq >= 1.0 or rsq == 0.0:
            v1 = 2.0*min_rand() - 1
            v2 = 2.0*min_rand() - 1
            rsq = v1*v1 + v2*v2
        fac = np.sqrt(-2.0*np.log(rsq)/rsq)
        gaussian.iset = 1
        gaussian.gset = v1*fac
        xi = v2*fac
    else:
        gaussian.iset = 0
        xi = gaussian.gset

    # Scale from standard normal distribution to that desired.
    # See page 242 in Statistics for Engineers and Scientists 2nd ed.
    # by Navidi
    return mu + xi*sigma
```

Due to time constraints, the performance of the above algorithm was not verified as throughly as the uniform random number generator (presented in Section A.1). However, the method is reliable since it was presented in class and in [11]. In order to ensure the code was written correctly the output was verified visually (see figures below) by plotting the theoretical PDF over the normalized histogram (visual goodness of fit test).

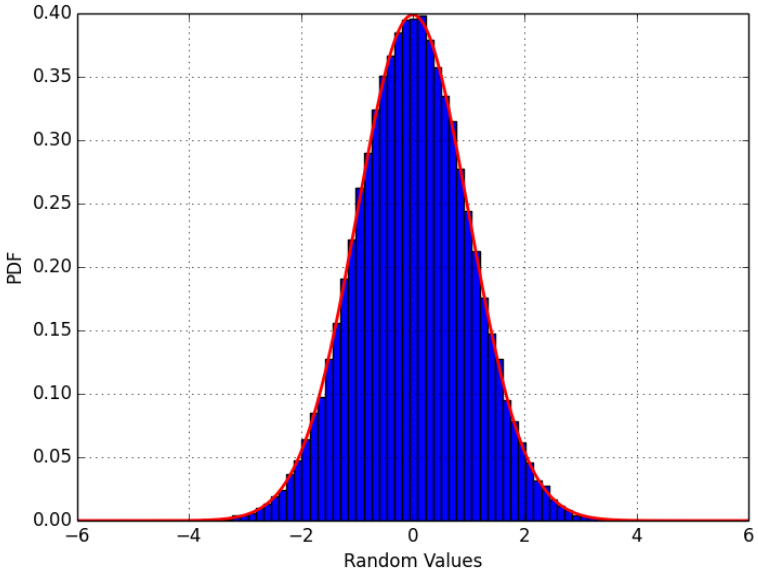

Figure A.2: Histogram of normally distributed random numbers generated ($\mu = 0$ and $\sigma = 1$)

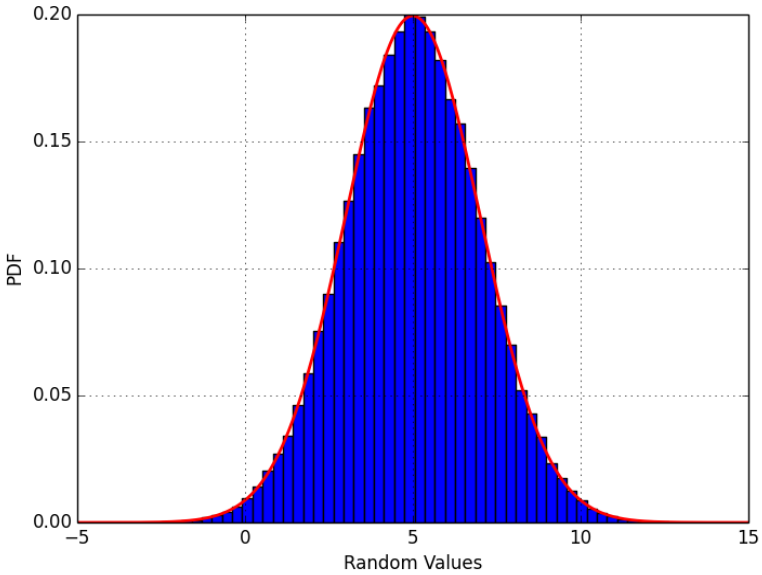

Figure A.3: Histogram of uniformly distributed random numbers generated ($\mu = 5$ and $\sigma = 2$)

APPENDIX B MAP FILE FORMAT

A custom map file format was developed for getting map information into the simulation. The format defines a grid on the first row and then allows blocks of cells to be given a type attribute. The simulation assumes any cells not specified to be prohibited. An example map file and its corresponding map (from the simulation) are shown below.

B.1 EXAMPLE MAP

```
# Test Map File
#
# Grid size (row,col) and size of cell area (arbitrary area unit)
100 100 1
# Define a block (start_row,start_col) to (stop_row,stop_col) of type t
# Cell types:
# 0       = Prohibited (default)
# 1       = Walkway
# 2       = Street
# 4       = Crossing (can also be defined by overlapping 1 and 2)
# 3 to 99 = Arbitrary (for specific use by simulator if needed)
# 1XX     = Destination (can be multiple destinations)
# 2XX     = Starting location (can be multiple starting points)
45 00 50 99 1
00 00 99 05 1
00 48 45 50 1
00 00 02 50 1
00 09 99 19 2
00 60 99 70 2
00 00 00 05 100
00 00 05 00 100
99 00 99 05 101
45 98 50 99 200
```

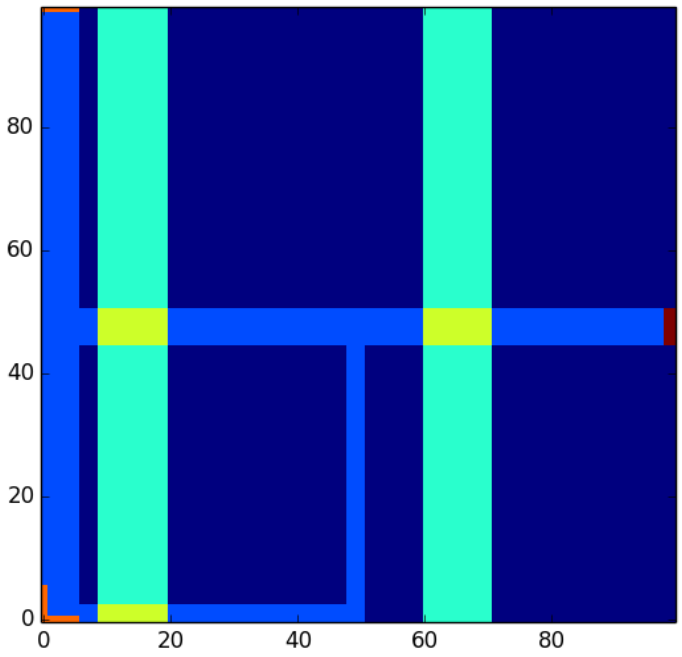

Figure B.1: Example map read from data file

B.2 Bobby Dodd Stadium Map

```
# Grid size (rows, cols) and size of cell area (arbitrary area unit)
700 900 0.5
# Define a block (start_row,start_col) to (stop_row,stop_col) of type t
# Cell types:
# 0       = Prohibited (default)
# 1       = Walkway
# 2       = Street
# 3       = Crossing (can also be defined by overlapping 1 and 2)
# 4 to 5  = Reserved
# 6 to 99 = Arbitrary (for specific use by simulator if needed)
# 1XX     = Destination (can be multiple destinations)
# 2XX     = Starting location (can be multiple starting points)
#
# Walkways
#
50   0  46 899   1 # North sidewalk along North Ave
4   0   0 899   1 # South sidewalk along North Ave
699   0  46   4   1 # West sidewalk along Cherry St
699  36  46  40   1 # East sidewalk along Cherry St
699   0 695 899   1 # North sidewalk along Bobby Dodd Way
663  36 659 899   1 # South sidewalk along Bobby Dodd Way
699 615   0 619   1 # West sidewalk along Techwood Dr
663 651   0 655   1 # East sidewalk along Techwood Dr
658 400 608 500   1 # Walk area at North stadium entrance
```

```
75 574  50 614   1 # Walk area at south east stadium entrance
75 445  50 465   1 # South stadium entrance walkway
#
# Streets
#
45   0   5 899   2 # North Ave
694   5  45  35   2 # Cherry St
694   5 664 899   2 # Bobby Dodd Way
699 620   0 650   2 # Techwood Dr
699 100 499 130   2 # Powerplant Dr
529 100 499 255   2 # Powerplant Dr
663 225 499 255   2 # Powerplant Dr
699 450 664 480   2 # Brittain Dr
699 200 690 230   2 # Substation Dr
699 295 690 325   2 # Fowler St
#
# Crossings
#
694 325 664 330 3 # Bobby Dodd Way Crossing at Fowler St
45 375   5 380 3 # North Ave crossing
694 445 664 449 3 # West crossing at Brittian Dr
694 481 664 485 3 # East crossing at Brittian Dr
#
# Destinations
#
699   0 679   4 100 # Side walk opening in north west corner
50   0   0   4 101 # South West corner of map
50 894   0 899 102 # South East corner of map
699 660 695 670 103 # Housing at the North east Corner of the map
#
# Stadium Exits/Entrances
#
# 75 574  75 614 200 # South-East stadium entrance
# 75 445  75 465 201 # South stadium entrance
607 420 607 480 202 # North entrance
#
# Other
#
607 300  75 614   6 # Show Stadium Outline
```

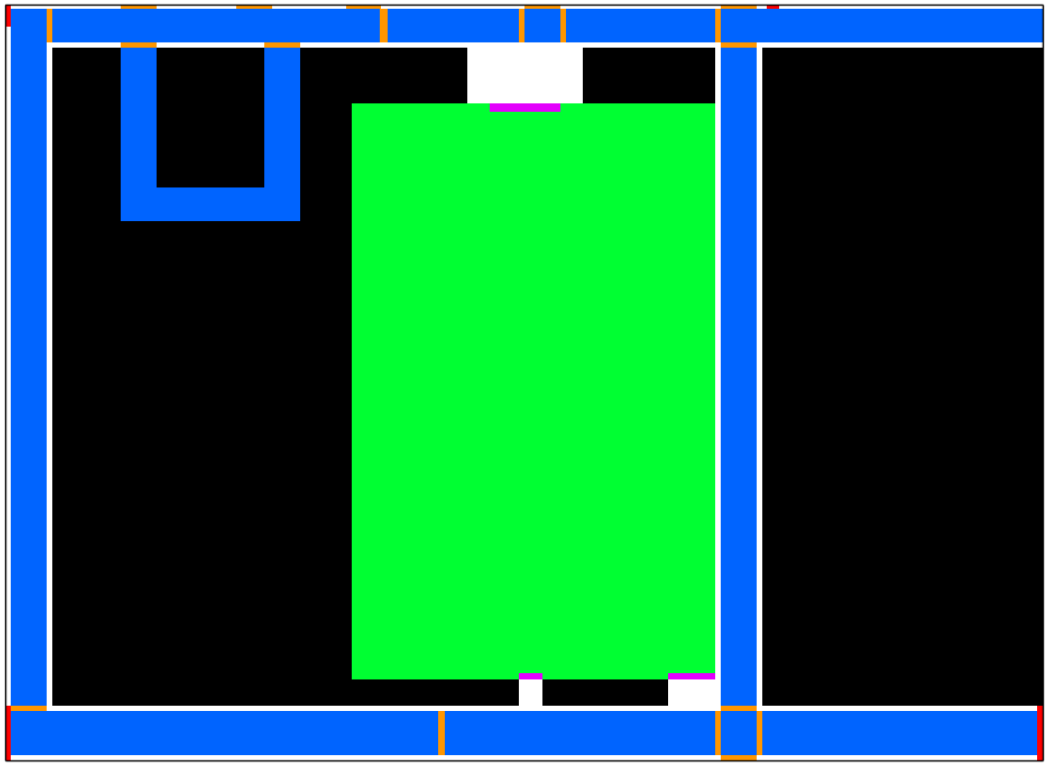

Figure B.2: Simplified map of Bobby Dodd stadium and surrounding area implemented in the simulation. Legend: red = exit, purple = stadium exit, white = walkway, orange = crossing, black = prohibited, green = stadium

APPENDIX C FLOOR FIELDS

A "floor field" [6] is calculated for each destination by assigning a value of 1 to the destination cells and then incrementing the value of neighboring cells for each step they are away from the destination. This "field" is used to "force" pedestrians through the grid to their desired destination. The calculation of this field is similar to a breadth-first search in graph theory and runs in $O(m+n)$ time (where n is the number of gird rows and m is the number of grid columns). Example floor fields for the map simulated (area surrounding GT's Bobby Dodd stadium) are presented below.

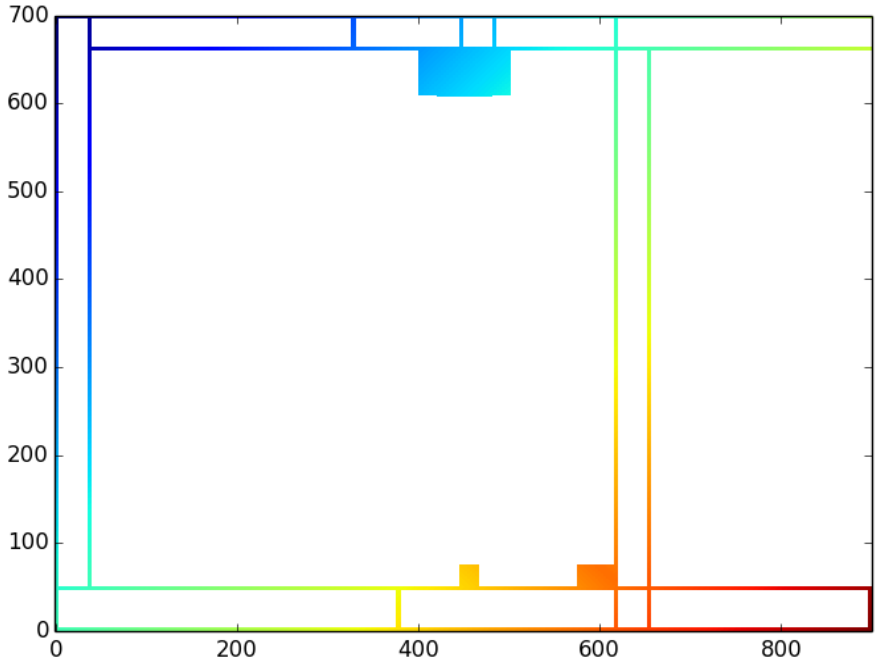

Figure C.1: Calculated floor field for destination #1 (sidewalk at North West corner of the map).

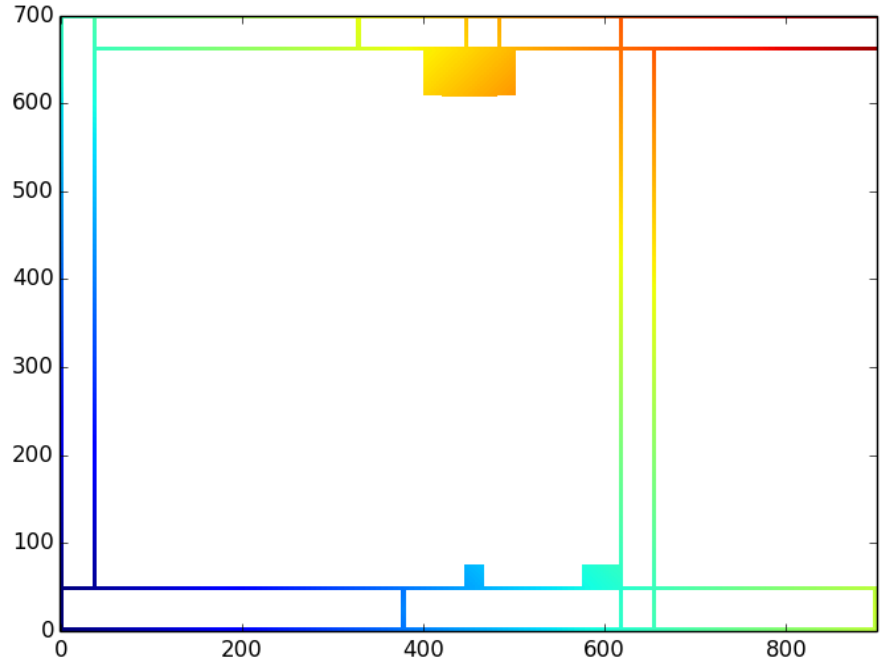

Figure C.2: Calculated floor field for destination #2 (South West corner of the map).

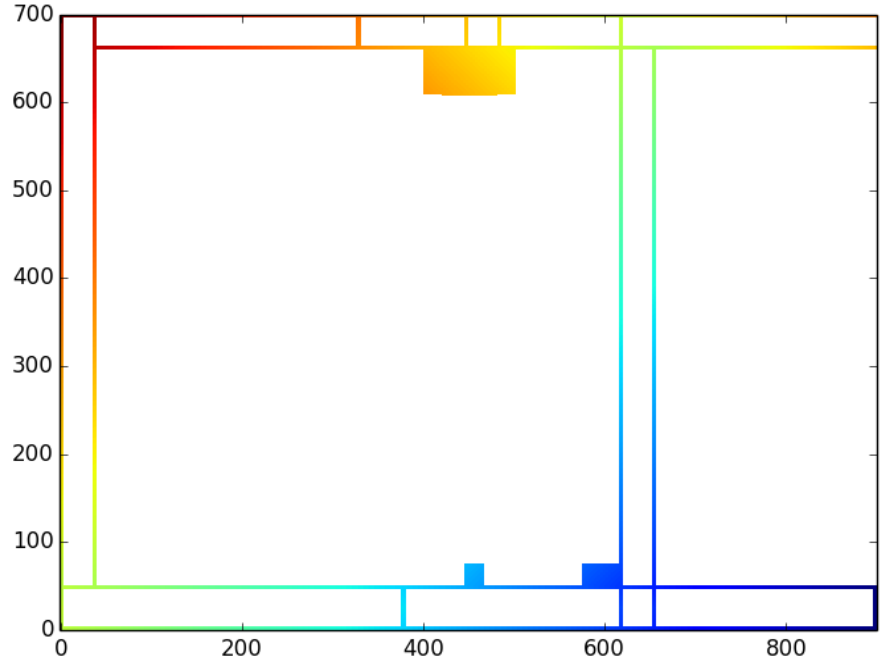

Figure C.3: Calculated floor field for destination #3 (South East corner of the map).

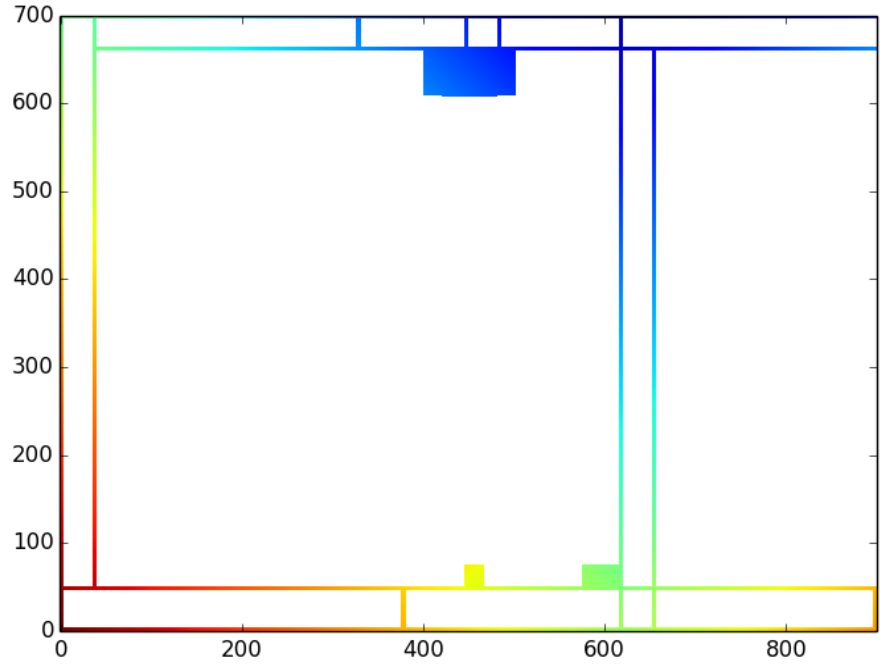

Figure C.4: Calculated floor field for destination #4 (student housing at North East corner of the map).

REFERENCES

[1] H. Sayama, *Introduction to the modeling and analysis of complex systems.* State University of New York at Geneseo: Open SUNY Textbooks, Milne Library, 2015.

[2] V. J. Blue and J. L. Adler, "Cellular automata microsimulation for modeling bi-directional pedestrian walkways," *Transportation Research Part B: Methodological*, vol. 35, no. 3, pp. 293–312, 2001.

[3] L. Ji, Y. Qian, J. Zeng, M. Wang, D. Xu, Y. Yan, and S. Feng, "Simulation of evacuation characteristics using a 2-dimensional cellular automata model for pedestrian dynamics," *Journal of Applied Mathematics*, vol. 2013, 2013.

[4] C. Burstedde, K. Klauck, A. Schadschneider, and J. Zittartz, "Simulation of pedestrian dynamics using a two-dimensional cellular automaton," *Physica A: Statistical Mechanics and its Applications*, vol. 295, no. 3, pp. 507–525, 2001.

[5] A. Kirchner, H. Klüpfel, K. Nishinari, A. Schadschneider, and M. Schreckenberg, "Simulation of competitive egress behavior: comparison with aircraft evacuation data," *Physica A: Statistical Mechanics and its Applications*, vol. 324, no. 3, pp. 689–697, 2003.

[6] A. Varas, M. Cornejo, D. Mainemer, B. Toledo, J. Rogan, V. Munoz, and J. Valdivia, "Cellular automaton model for evacuation process with obstacles," *Physica A: Statistical Mechanics and its Applications*, vol. 382, no. 2, pp. 631–642, 2007.

[7] A. Jolly, R. Oleson II, and D. Kaup, "Pedestrain cellular automata and industrial process simulation," in *Proceedings of the 20th European Modeling and Simulation Symposium*, EMSS 2008, 2008.

[8] S. Robinson, "Conceptual modeling for simulation," in *Simulation Conference (WSC), 2013 Winter*, pp. 377–388, IEEE, 2013.

[9] "Bobby dodd stadium." `http://www.ramblinwreck.com/genrel/071001aaa.html`.

[10] S. Young, "Evaluation of pedestrian walking speeds in airport terminals," *Transportation Research Record: Journal of the Transportation Research Board*, no. 1674, pp. 20–26, 1999.

[11] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical recipes in C*, vol. 2. Cambridge university press, 1996.

[12] W. Navidi, *Statistics for Engineers and Scientists.* McGraw-Hill, 2 ed., 2008.