Jonathan Hamstra

Michael Svetlichny

Shane Modena

CS-273-1

12/17/20

Final Project Specifications

**Specifications Overview:**

Specifications

For the purpose of the simulation, there shall be at least 5 districts:

* Downtown
* Soho (safest district)
* The Docks
* Medical Hill (First infected case emerges here)
* The University

In addition to 5 locations, there shall be at least 3 types of denizens:

* Ignorant
* Alarmed
* Zombie

Specifications for people interactions:

* Each person (ignorant and alarmed) has an X% chance of getting bitten during an encounter with a zombie. Ignorant people have a 1-X% chance of becoming Alarmed and escaping the encounter.
* Each alarmed person has an Y% chance to alert a a nearby ignorant person, and a 1-Y% chance of not being able to convince the ignorant of their plight.
* Alarmed people have a lower chance (Z%) of being infected due to their running, with 1-Z% chance of escaping.
* Zombies attempt to bite other humans in the district, with each zombie getting one attempt per game tick.
* All entities have a chance Q% to stay put, and a 1-Q% chance of moving to an adjacent district. District is chosen at random, with a lower chance for Soho due to its island nature. Time to go for a swim! (See Additional Specifications)
* The only communication between districts is travel, so no biting or alarming across borders. Really should invest in phone systems...

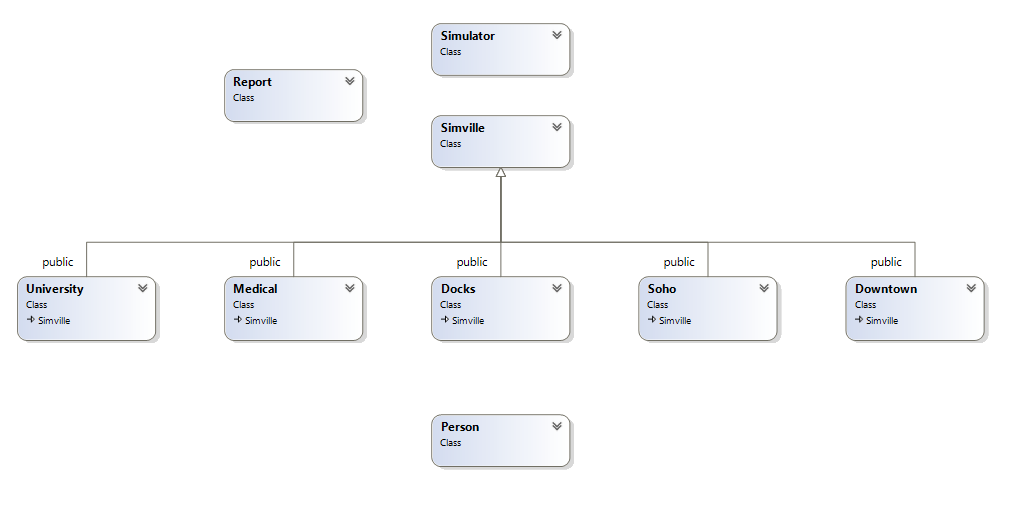
**Use Cases:**

|  |  |
| --- | --- |
| Problem: | The Zombie virus is real and the apocalypse is among us! We need to create a simulator to help us determine what the best outcome is and how we can increase our odds for survival! |
| Input/Output Requirements: | User starts up the program and otherwise has zero interaction with the system. User merely watches the apocalypse unfold before his/her eyes. |
| Inputs: | Population and Probabilities of infection and district moving |
| Outputs: | A simulation displaying the five different districts and their current populations including zombies, alarmed, and ignorant denizens, all in a .csv file |

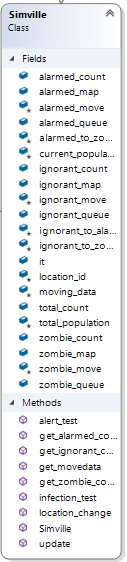
|  |  |  |
| --- | --- | --- |
| **Step** | **User’s Action** | **System’s Response** |
| **1.** | User opens and runs program | System prompts user to input population size, all probability variables for infection or alarmed, and all probability variables for moving districts |
| **2.** | The user either watches in horror or relief depending on the result. | System executes the simulation in Simulator.h/.cpp |
| **3.** |  | Each clock tick, simulator will perform an update on Medical Hill, Docks, Downtown, University, and Soho. Each update in these locations will perform however many required infection tests and alarmed tests, as well as perform how many denizens are going to move from one location to the next. |
| **4.** |  | Before the end of the clock tick, simulator performs the movement for each location and determines where each denizen is going to move. |
| **5.** | User opens .csv file and looks at simulated data in both amazement and grief, realizing there is no hope for humanity and that the zombies will take over. | Once while-loop condition of total zombie population equals total denizen population is met, then the simulation ends and system outputs a class defined string method of all the counts of population per clock tick to a .csv file |

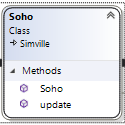
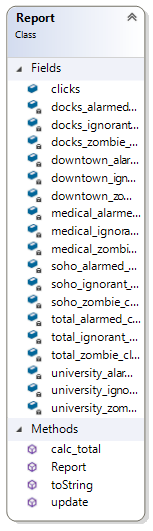
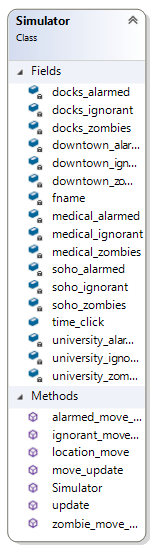
**UML Diagrams:**

**Main Structure:**



**Class Details:**





**Pseudocode:**

Output method

*toString(){*

*string = “Column header,Column header,… \n” + “clock tick population number,… \n”*

*return string*

*}*

Simulator Constructor:

*Simulator(user specified variables){*

*define all Simville objects*

*while(zombie != total\_population){*

*Update locations*

*}*

*}*

Alarmed person encounters zombie – similar for zombie and ignorant, zombie and alarmed

*// Either they become a zombie or they dont*

*// Roll the die:*

*// if(x <= 3){ //person becomes infected*

*// //access a single element of the alarmed map and change id to 3 for zombie*

*// alarmed\_map[alarmed variable]--;*

*// zombie\_map[zombie variable]++;*

*// }*

//potential code for testing for bugs:

// if(tests != zombie\_count){

// std::cout << "ZOMBIE TESTS DONT ALIGN" << std::endl;

// }