Gebze Technical University

Computer Engineering

CSE443 - Object Oriented Analysis and Design

Fall 2020 - 2021

Final Project - Report

Student:
Nevra GÜRSES
161044071

1 PROBLEM DEFINITION

There is a society and this society will be modeled as 1000x600 pixels canvas.There will be also people in this society of population Po and every person will be modeled as 5x5 pixels on canvas.

Each person has numerical values that are mask value (M=0.2 if that person wear mask or M=1.0 if does not wear), speed S that value in [1,500] of movement in pixels/second, the social distance D value in [0,9] in pixels and C seconds in [1,5] that is spend time with every individual they collide with.

At the beginning there will be one random infected person in the population.The disease will have a constant spreading factor R in [0.5,1.0] and a constant mortality rate Z in [0.1, 0.9].

When two individuals collide with their own C values, they stay together at collision position for time max C value to simulate interaction and then continue their randomized courses. If another person is in collision course with either of them in the meantime, that person can not interact with them and ignores them as if they weren't there.For instance, if two people collide with their own mask statuses(M1 and M2), and their own social distances and first person is infected and second person is healthy, they stay together for a duration max C value before parting, and the social distance between them is min D value. The probability of first person infecting second person is P = min(R * (1+C/10) * M1 * M2 * (1-D/10),1)

An infected person will die after 100 * (1-Z) seconds and disappear from the canvas.And also every infected individual, 25 seconds after her/his initial infection will be assumed to be at the hospital and will be removed temporarily from the canvas. The hospital however is assumed to have only B=Po/100 ventilators. After staying at the hospital for 10 seconds that person will return to the society at a random position as healthy. If the hospital ventilators are all full the individual will remain and continue moving/infecting in the society, until a ventilator becomes available or that person will dies.

Our first goal are modeling these all mentioned cases, providing a timer and show the total count of infected, healthy, hospitalized and dead on can-

vas. Our other goals, creating individuals using flexible and cost reducing design pattern , providing interaction between individuals using Mediator design pattern, providing multi-threaded GUI,using producer/consumer paradigm to implement the hospital functionality, and producing graphical plots with changed numerical values.

1.1 METHOD

1.1.1 Which Design Patterns are Using for These Goals?

For providing given goals, I use basically **Model-View-Controller** architectural pattern that is actually Compound Design Pattern. Now I will explain why I'm using this pattern in basically.

Model-View-Controller architectural design pattern is commonly used for developing user interfaces that divides the related program logic into three interconnected elements.This pattern is used in applications with a visual interface to reduce maintenance costs and to ensure that the application is flexible. It ensures that the changes in the visual part do not depend on the data part, and the reactions in the data part do not affect the visual part. The parts of MVC pattern are providing below goals:

View: Gives us a presentation of the model. The view usually gets the state and data it needs to display directly from the model.

Controller: Takes user input and figures out what it means to the model.

Model: The model holds all the data, state and application logic. The model is oblivious to the view and controller, although it provides an interface to manipulate and retrieve its state and it can send notifications of state changes to observers.

Since the project given to us is a visual modeling project, I basically using MVC pattern in order to provide flexibility, low cost, loose coupling. Because of MVC pattern is actually Compound design pattern, it is using three design pattern. Now I will explain which inner design patterns are using and how I using this inner patterns in my project.

Composite Design Pattern: Composite Design Pattern is using in View part of MVC pattern.The display consists of a nested set of windows, panels, buttons, text labels and so on. Each display component is a composite (like a window) or a leaf (like a button). When the controller tells the view to update, it only has to tell the top view component, and Composite takes care of the rest. In my project, there is a top panel and in this panel, there are 1000x600 pixels canvas, inner information section, many buttons. I provide this view by using Composite Design Pattern in my project.

Strategy Design Pattern: The view and controller implement the classic Strategy Pattern.The view is an object that is configured with a strategy. The controller provides the strategy. The view is concerned only with the visual aspects of the application, and delegates to the controller for any decisions about the interface behavior. Using the Strategy Pattern also keeps the view decoupled from the model because it is the controller that is responsible for interacting with the model to carry out user requests. The view knows nothing about how this gets done. In my project I using strategy pattern with mentioned way. When user click on button, it transmits this request on controller and controller does what should do by using model.

Observer Design Pattern: The model implements the Observer Pattern to keep interested objects updated when state changes occur. Using the Observer Pattern keeps the model completely independent of the views and controllers. It allows us to use different views with the same model, or even use multiple views at once. In my project I using observer pattern when every state changes. If an individual is changing position, model that is State class in observer design pattern is notifying observer that is view class. And after this notifying view that is observer class is changing position of individual. In same way, if previous position should be deleted, model class is notify view and in view class deletes previous position of individual. And also in every second, Model class is notifying view for printing up to date healthy,infected,hospitalized and dead people numbers.

And also in my project, I'm using Mediator Design Pattern, now I will explain how I'm using Mediator Design Pattern:

Mediator Pattern: Mediator Design Pattern is using to centralize complex communications and control between related objects.With a Mediator added to the system, all of the appliance objects can be greatly simplified: They tell the Mediator when their state changes.They respond to requests from the Mediator.In my project I using Mediator Design Pattern for interaction between individuals. In my project Model class is Mediator class. When position changing is occurring,every individual that are actually threads tell Mediator class that is actually Model class by using positionChange method.In same way, when previous position should be deleted, every individual tell Mediator class by using deletingPrevPos method. For this way, interactions between individuals are provided via Mediator class.

Also, In our goals, that is said "Create the individuals using some design pattern. Choose wisely and justify your decision. Some are more appropriate than others. Strive for maintenance cost reduction, and flexibility. The user should be able to add them in bulk as well as one by one." .**For providing this goal I use composite pattern.** Because composite pattern allows us to compose objects into tree structures to represent part-whole hierarchies. Composite lets clients treat individual objects and compositions of objects uniformly.Using a composite structure, we can apply the same operations over both composites and individual objects. In other words, in most cases we can ignore the differences between compositions of objects and individual objects. And in this way,the user can be able to add individuals in bulk as well as one by one and also it provides flexibility and cost reduction.

1.1.2 Classes and Interfaces that I Implemented in My Project

- **Observer Interface:** This is Observer Interface in observer design pattern.The Observer interface is implemented by all observers, so they all have to implement the update() methods. In my project the observer is EpidemicView Class and this class implements update methods in observer interface.There are 2 update methods in my project. One update method is called every second via model for up do date information. Other update method is called when state change occur in model.

- **EpidemicView Class:** This is Concrete Observer Class in Observer

Design Pattern. And this class also View part in MVC design pattern.In view there are Model and Controller class references.In view class, view scene is created such as buttons,canvas,panes and much more with using Composite Design Pattern.This Class is using also Composite Design Pattern with feature of adding individuals as bulk or one by one. This class implements update methods of observer interface.

- **ControllerInterface Interface:** In this interface, there are all methods the view can call on the controller. This interface is using Strategy Design Pattern.

- **EpidemicController Class:** This class concrete Controller class. Methods of this class using is by view. And in this class requests are transmitted Model Class.

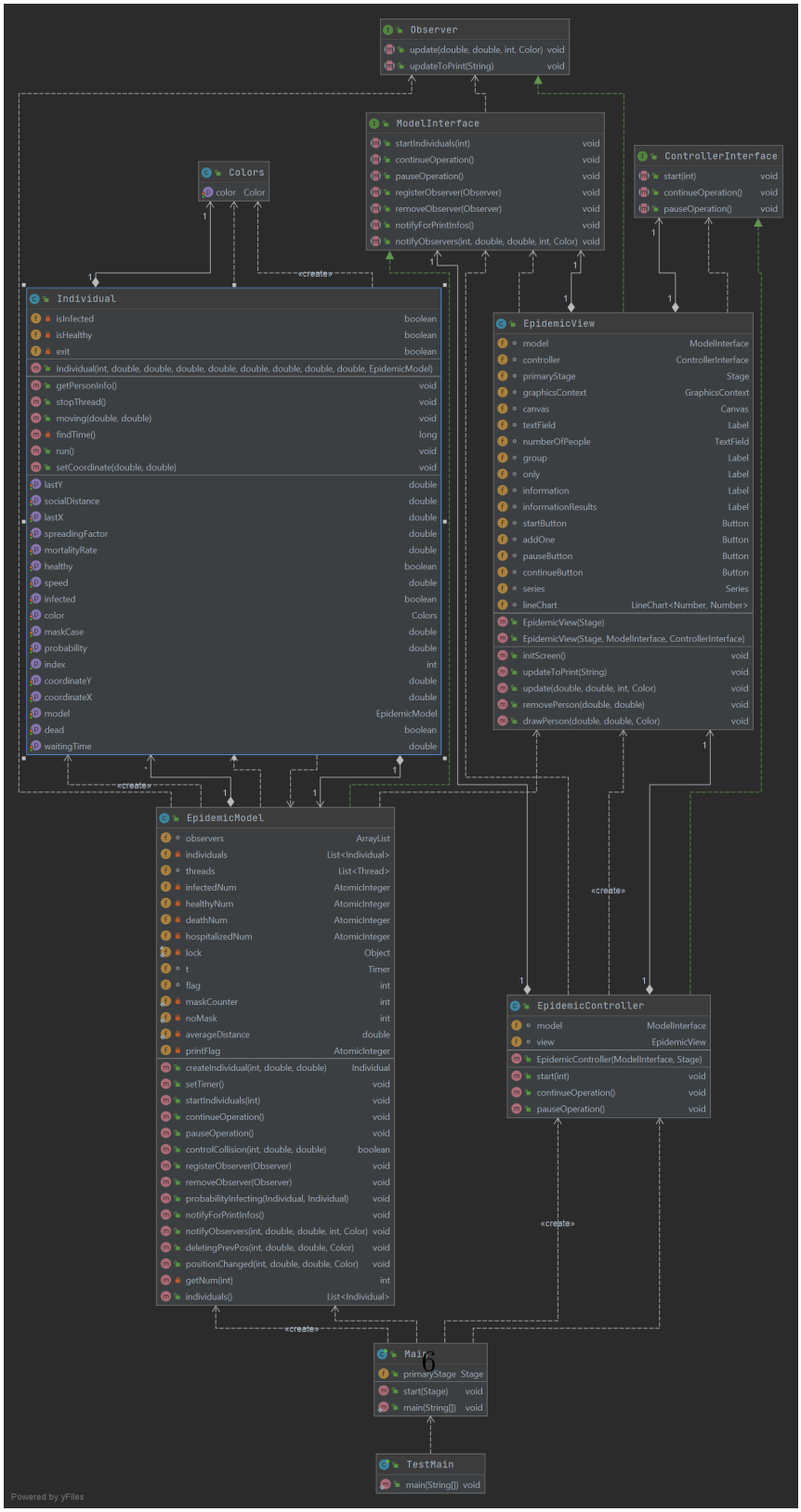
- **ModelInterface Interface:** Model Interface Class is Model part in MVC pattern. This interface is also Mediator Interface for individuals and also Subject Interface for Observers.

- **EpidemicModel Class:** This class concrete Model Class.And also concrete Mediator and concrete subject class.This Class holds all the data, state and application logic. When a state change occur, it notifies observer that is view class.This class also Mediator Class.Because every individual says change of states to this model class and model class comments state changes.Communications between individuals are provided via Mediator Model Class.

- **Individual Class:** This class is runnable class.It keeps all features of an individual such as position,mask case and much more.

- **Color Class:** This class for getting and setting colors of people.

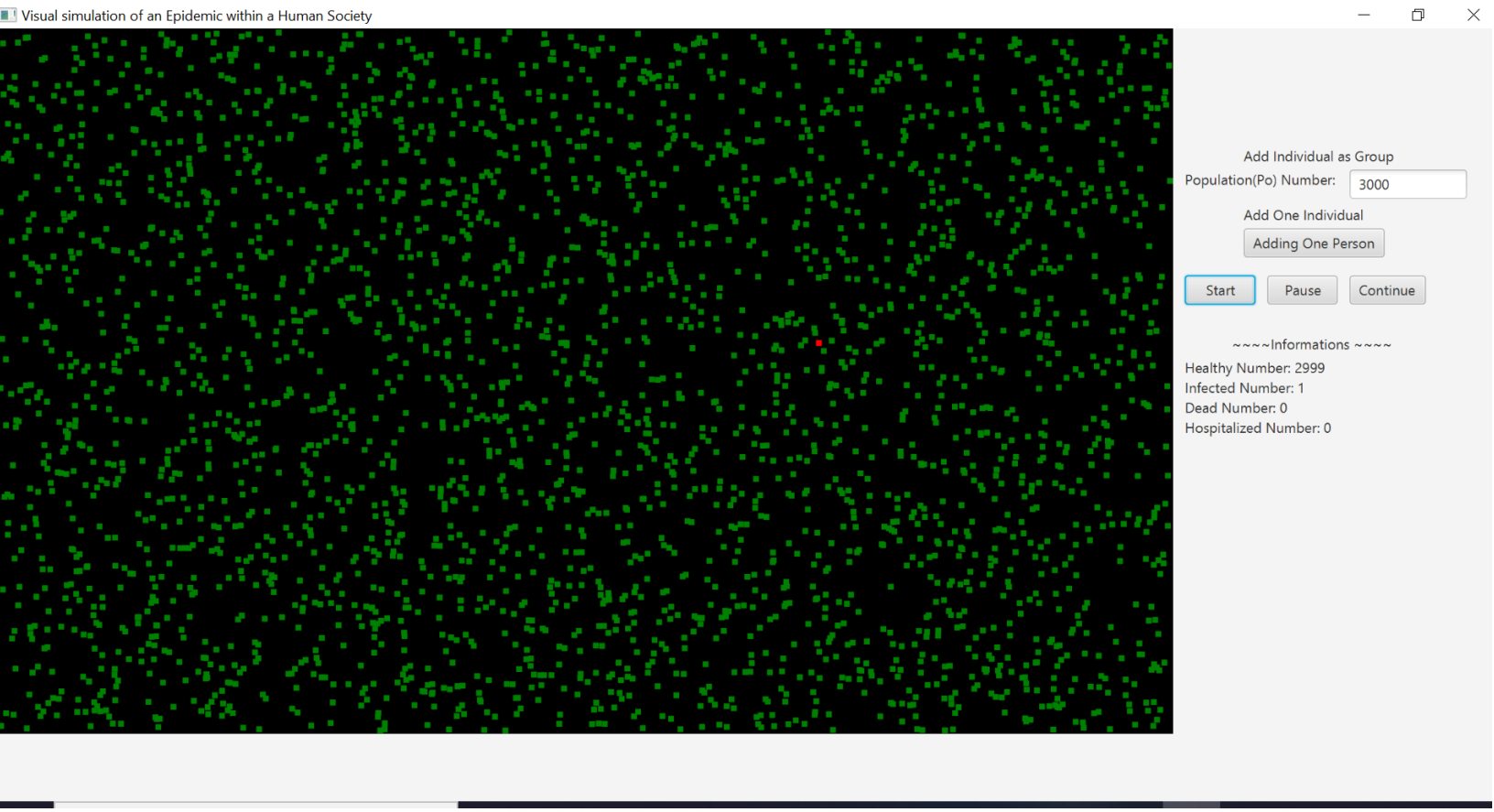
1.2 Class Diagram (UML Diagram)



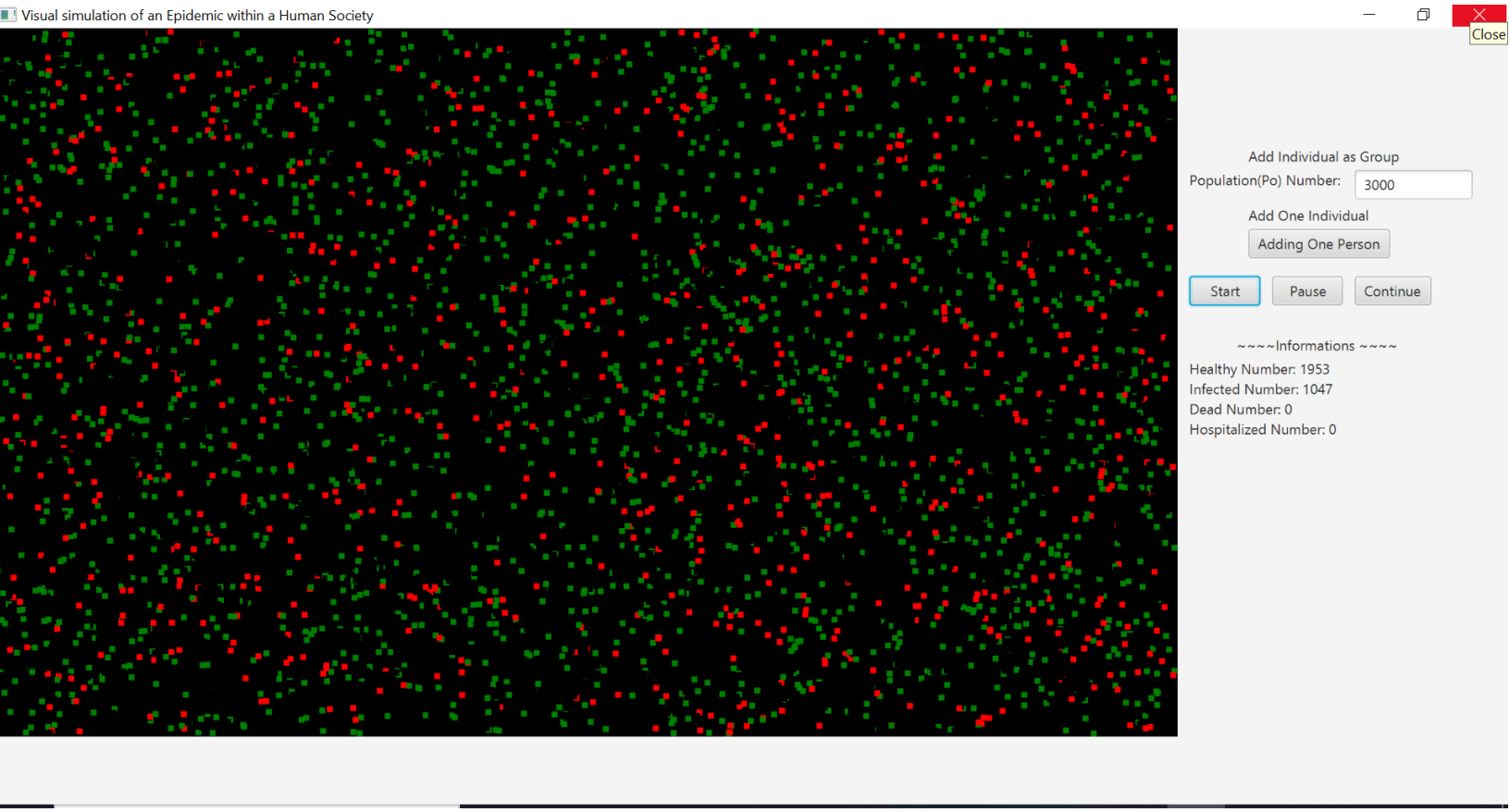
2 Running Results

NOTE: You can add individuals bulk as well as one by one. If you want to add them as bulk, write Population Number then click start button. If you want to add one by one, you can click Adding One Person button.

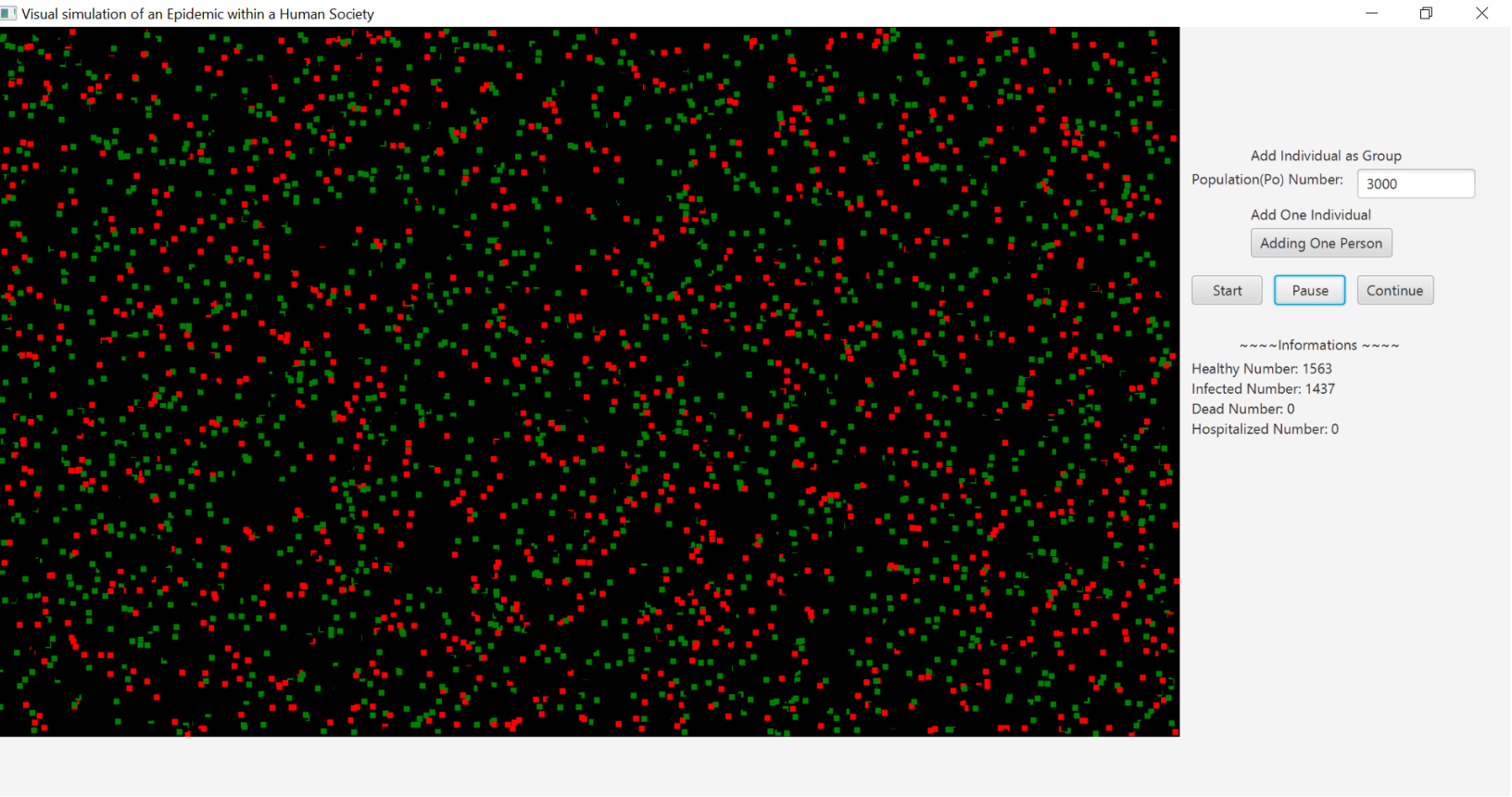
At the Beginning, One Random Infected Person in Population:



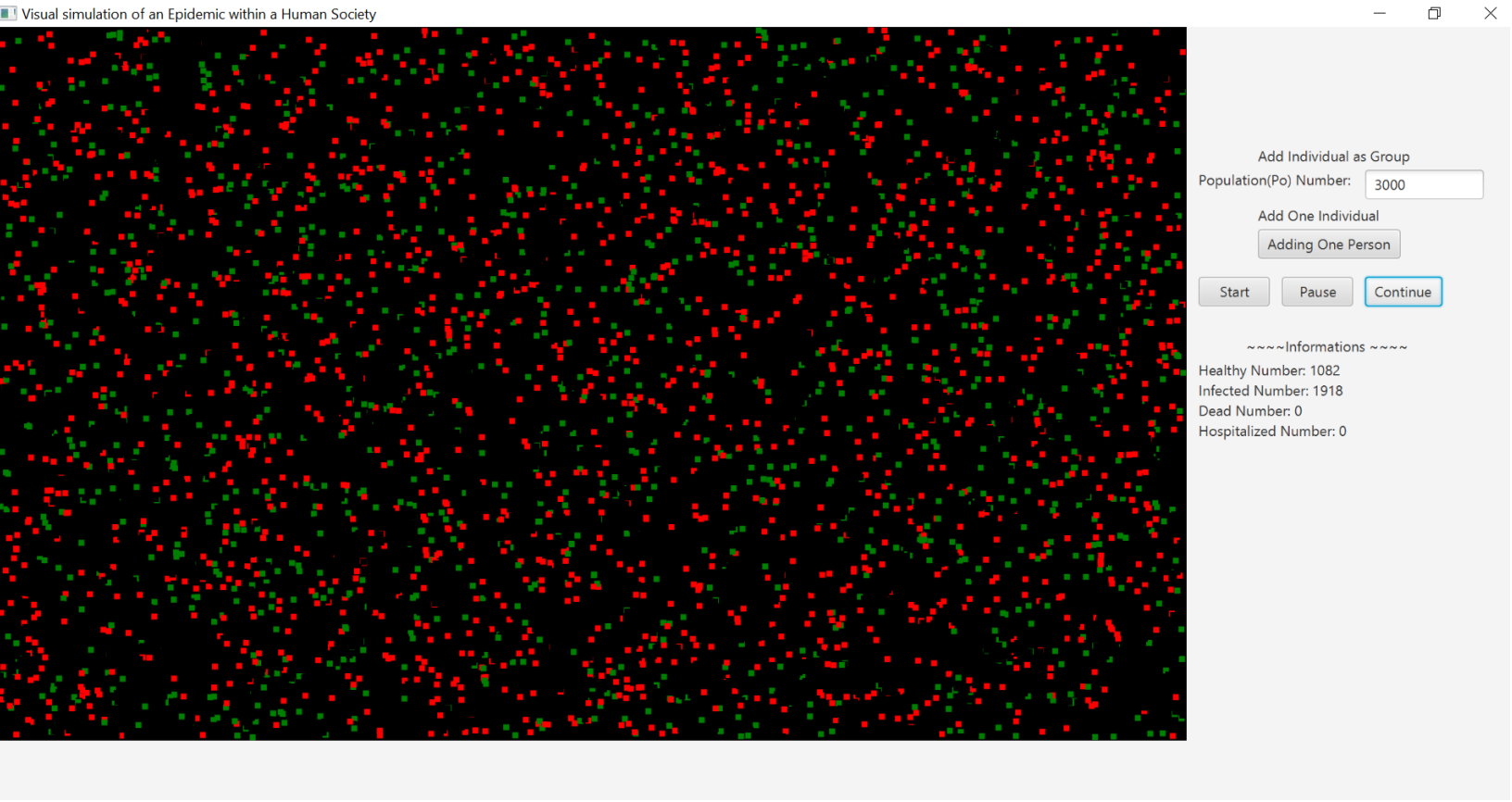
Some Time After Clicking the Start Button:



Pause Button is Clicked:



Continue Button is Clicked:



Add Three Person One By One Testing Using Adding One Person Button:

2.1 What I did Via Visual Simulation:

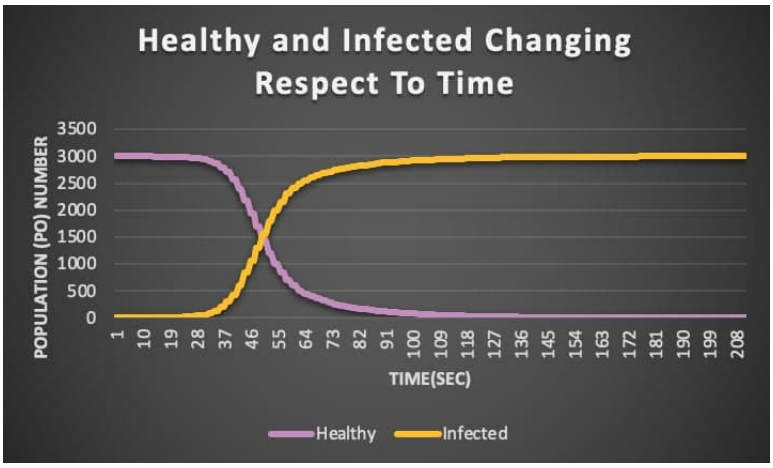
After entering people collectively or individually through the interface, numerical values for each person are randomly assigned at the intervals specified in the project, and people start to work as a thread. Healthy people look green, infected people look red. Initially, 1 random person is determined as a patient. If 1 infected and 1 healthy person collide, if the probability of being infected is greater than 0.5, the healthy person also becomes infected and the color is changed from green to red.

3 Graphical Plots

Graphical Plots with different Po,Z,R, Average Social Distance D and also Mask Use and No Mask Use Cases:

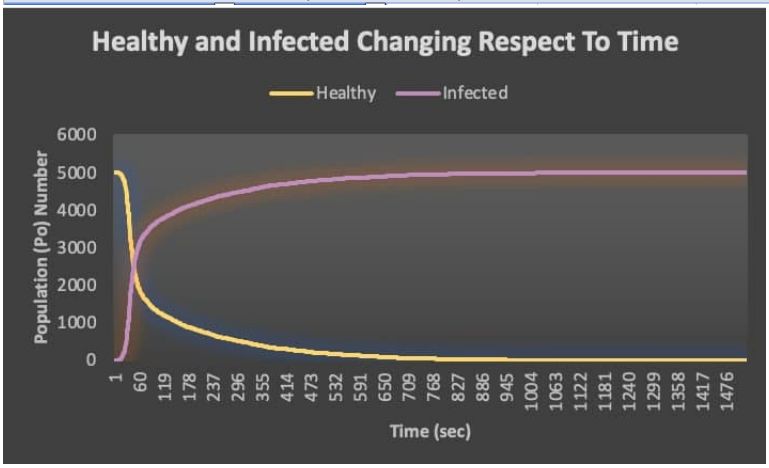
Graphical Plot - 1

P	Z	R	D	Total_MaskUse	Total_NoMaskUse
3000	0,171228012	0,971905322	4	1491	1509



Graphical Plot - 2

P	Z	R	D	Total_MaskUse	Total_NoMaskUse
5000	0,650363283	0,607584399	5	2467	2533



Graphical Plot - 3

P	Z	R	D	Total_MaskUse	Total_NoMaskUse
8000	0,579619374	0,803625504	4	3943	4057

3.1 References:

Eric Freeman & Elisabeth Robson with Kathy Sierra and Bert Bates ; Head First Design Patterns.