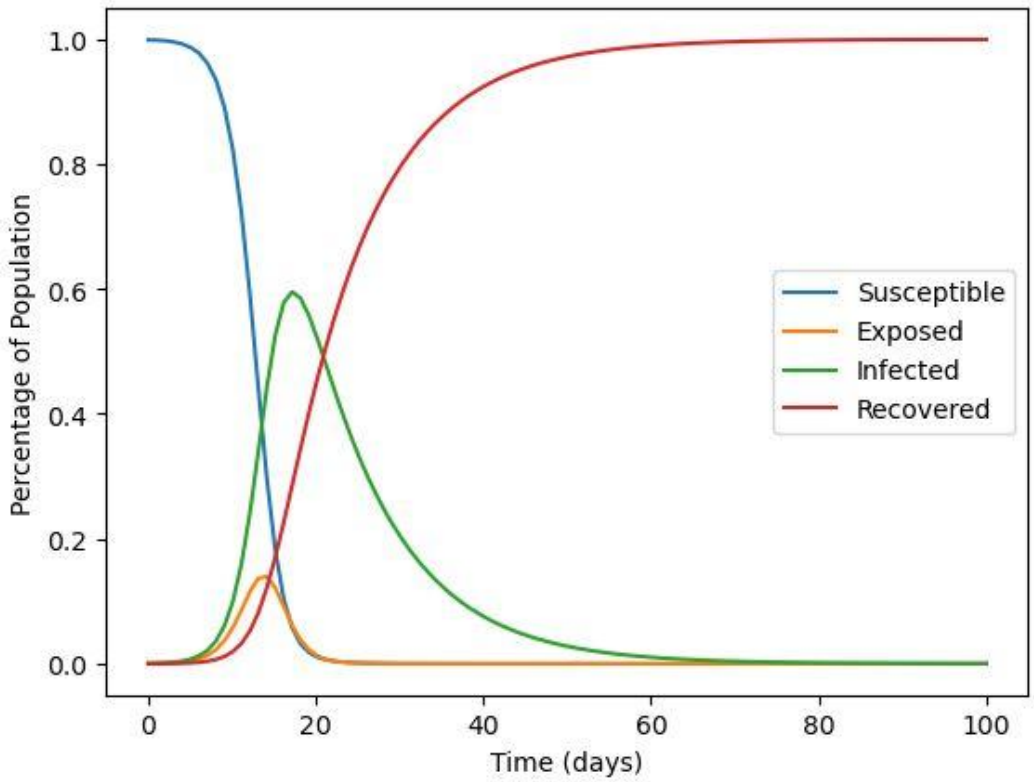
Results of SEIR Modeling:

After implementing the SEIR differential equations into python, and defining every variable as a percentage of the initial population, I ended up with this graph:



This graph is the result of running the SEIR model with the following initial conditions:
N = 7,000,000 S0 = 0.999 E0 = 0.001 I0 = 0 R0 = 0

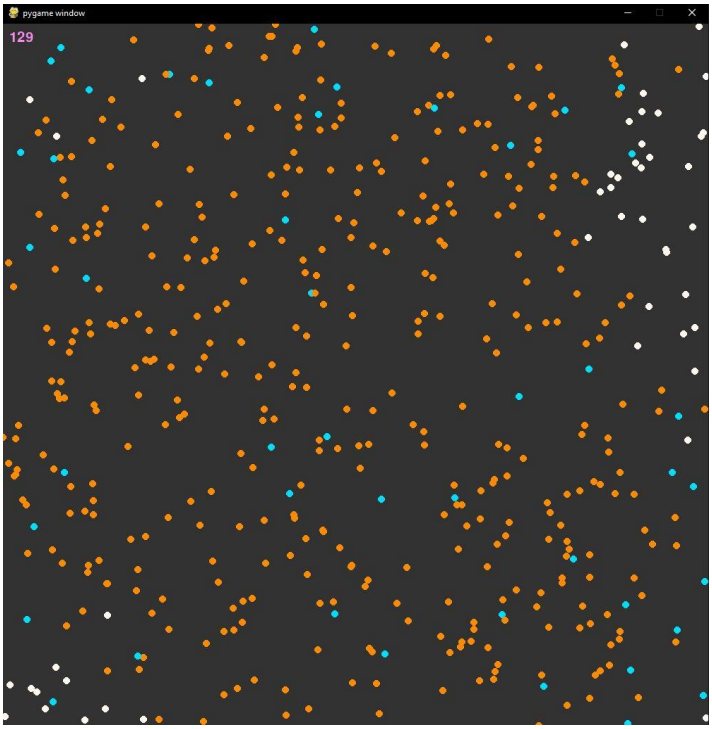
And the following parameters detailing the ODE constants:
β (Transmission Rate) = 1 γ (Incubation Rate)= 1 σ (Recovery Rate) = 0.1

This graph is very similar to the one provided in the project details, though everything happens at a much earlier time than what happens in the online graph. I tried changing all time variables to seconds rather than days, but it produced the same exact graph. In either case, i'ts clear to see that, with a relatively infectious disease, the susceptible population will very quickly become exposed and subsequently infected, leaving no remaining susceptible people after around 20 days. The recovery rate isn't too long, though, so the number of infected people is quickly counterbalanced by the recovered population, leaving no infected people left at around 75 days.

Results of Epidemic Simulation:

Before getting into the findings from the simulation, I want to stress that the animated simulation **IS** my "extra idea" that I decided to investigate, which is different from the "mitigation methods" I was initially going to do. Code comments are abundantly present in the code window, so any and all explanations of how things actually work can be found there. I will simply explain what the machine does, what it produces, and how to operate it in this write-up.

The simulation looks something like this when ran with 500 individual size-three dots

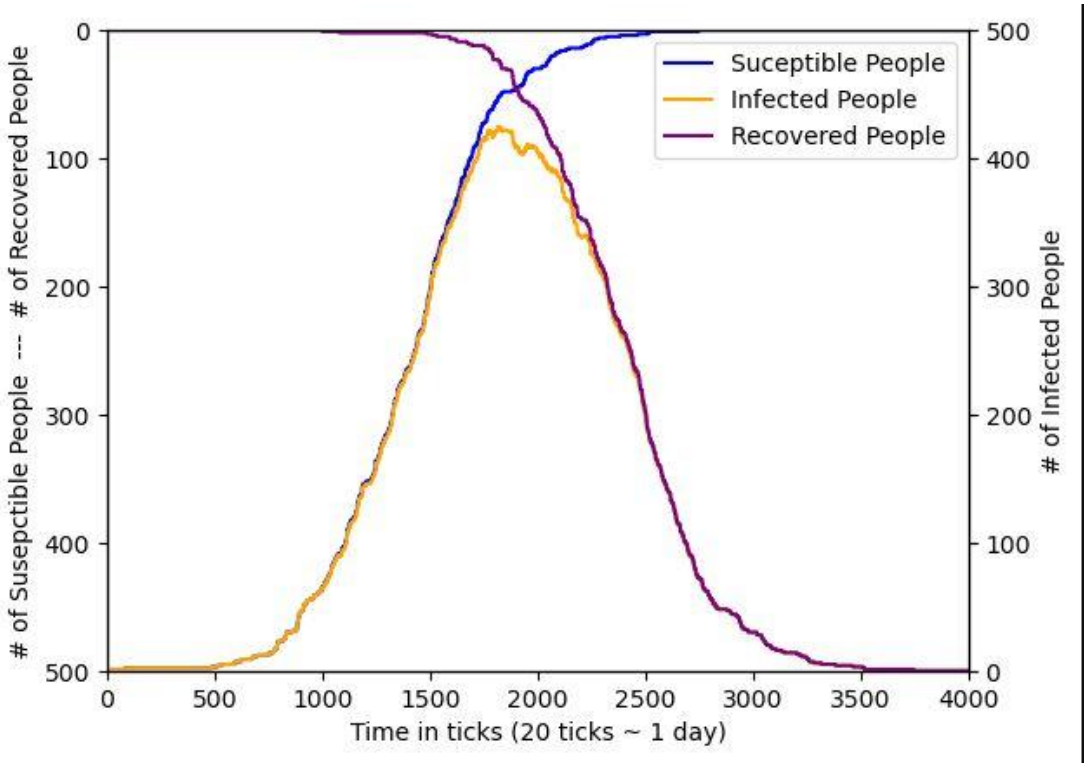


In this simulation, the colored dots represent all of our susceptible people (white), infected people(orange), and recovered people (blue), while the pink text in the upper right is the fps counter.

People are generated in random positions and move in RANDOM directions based on RANDOMLY changing velocity vectors, with one person being randomly selected to be patient zero. If a susceptible person is within "infect_dist" of an infected person, they become infected. To recover, infected people must simply wait out the recovery time set in the simulation parameters

To run the simulation with the given parameters, simply open the code panel and run the cell. If you want to change the parameters, open up the code panel to the "EpidemicSim" class.

The graph produced by the above simulation looks like this:



Which pretty closely follows the graphs on the Washington Post “stimulitus” article. The y-axis for both the susceptible and recovered people is flipped to mimic the format on the WP article.

From this graph we can see that the graphs these researchers publish about epidemics in the real world is VERY similar to graphs produced ENTIRELY through simulation! Infected people peak at the halfway mark, and the slope of the “infected people” line follows that of the susceptible and recovered population as we would expect it to.

In all, this simulation was a huge success and I’m extremely proud of what I made and all the work that went into it. I want to thank you personally for helping me take my first real venture into the wonderful world of coding, this semester has been a blast!