CS171 Final Project Process Book

Pandemic Map

By Armaghan Behlum and Shahyan Sajid

Table of Contents

Overview and Motivation:

Zombies are cool. Everyone is fascinated by zombies. Given that zombies are such a big part of our lives (in the entertainment industry; thank God we don't have to actually deal with them yet), we figured this would be a great way to look at how a pandemic, such as a zombie viral/attack, would spread and affect the world.

Related Work:

A recently published paper by researchers at Cornell served as the primary inspiration behind this project. We figured that taking their model and building upon it seemed like a great way to do pandemic mapping in an interesting way. Their visualization (Fig. 1.1) for a zombie attack also gave us some further ideas for the project. The nukemap visualization (Fig. 1.2) that we had seen before--used to map a nuclear explosion of different yields, anywhere in the world--gave us some other ideas for the interface as well. Our goal is to expand what was done at Cornell and map diseases on an international level.

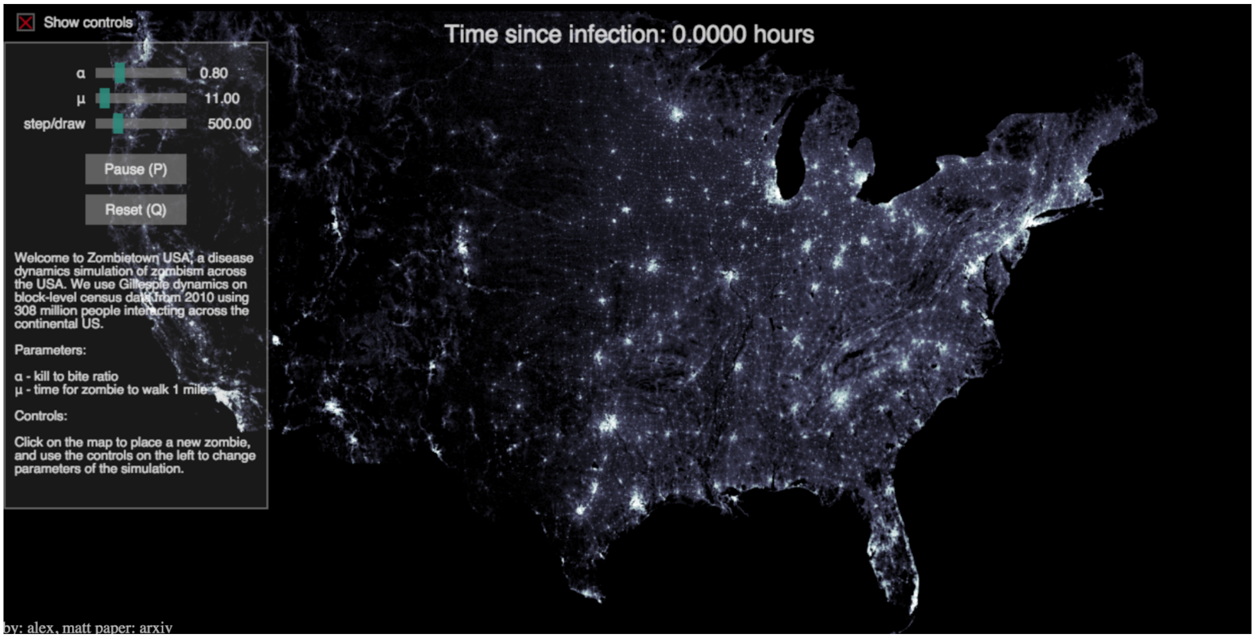


Fig. 1.1: Zombie visualization inspiration (http://mattbierbaum.github.io/zombies-usa/).

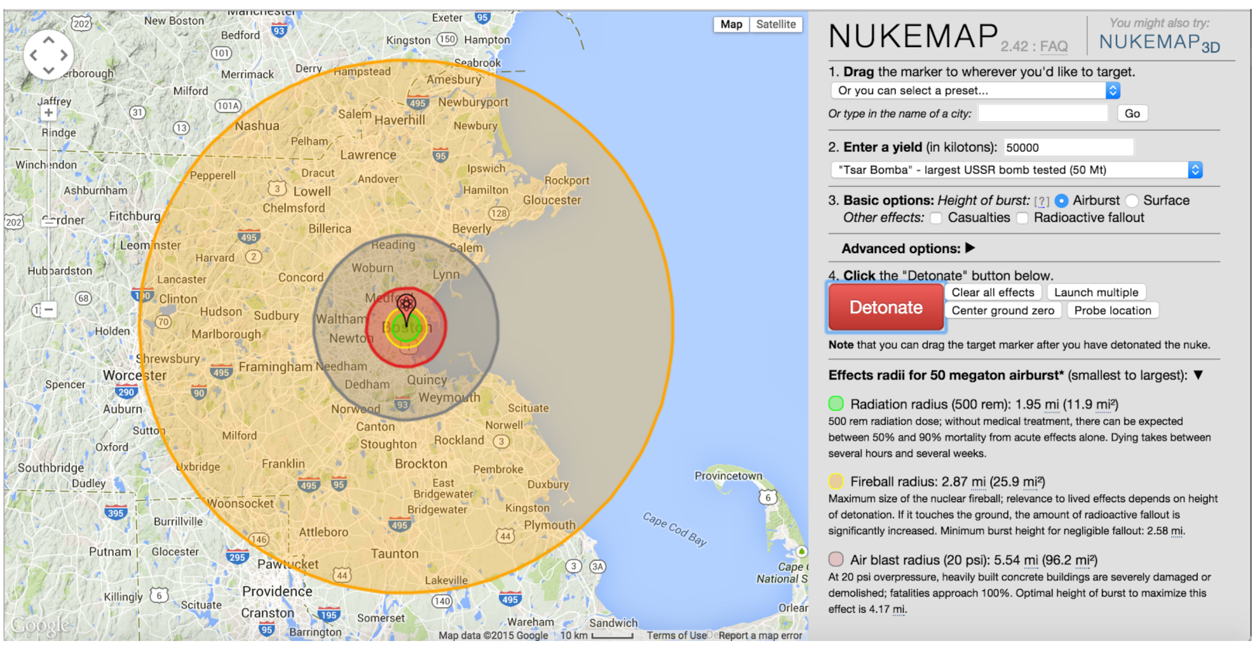


Fig 1.2: Inspiration for how to map and control the visualization (http://nuclearsecrecy.com/nukemap/).

Questions:

The primary question that we were looking to answer is how a zombie virus would spread throughout the world, based off of different starting points and their population densities, around the world. We generalized this to the idea of diseases being spread across the world and the patterns that we saw. While working on the project, we decided that it would be interesting to see how air transport might factor into how diseases spread across the world, with the assumption being that the zombie virus has an incubation period during which one can take a flight. We mostly stuck to this question through the course of the project, though we did have to adjust our rate of spreading the zombies due to limitations of APIs.

Data:

There are seven types of data that we will, primarily, need:

- Country population densities
- Migrational/International travel data
- SIR model variant
- Country geojson data
- Worldwide airport data
- Country area information
- Reverse Geocoding

To load in data we mostly use D3's csv loading function. We needed to mark the titles/keys for some of the columns, like the airport data, since d3.csv assumes that the first line of a csv is the key for the column. The cleaning and loading is done once the the HTML page is loaded. We load each data value into a javascript object so, for example, area goes into an area object, airport data into airport object, etc. The keys are, in general, names of countries so that if a lat/long point is selected on the map, we can use a reverse geocoder (from Google Maps, which is another form of data that we are using) to find the country. Then, using the above data to model how many people are in the country, how many "blocks" of people we can have, and how to model the SIR dynamics within the blocks.

Exploratory Data Analysis:

As mentioned above, the Cornell data visualization was our primary inspiration for what we were aiming to accomplish. In the beginning we were thinking about using a simple map and not showing data at a more granular level. However, we realized this was in conflict with some visualization ideas we had, like using Google Maps Street View, which would only be useful if we had very granular data. We decided to switch away from Google Maps for this reason. We also decided to use airport data after thinking about how zombies should hop continents and countries. Just moving along land would be very slow, but we thought about what would make a reasonable entry point for a zombie going to a different country. After finding some airport data and seeing those points visualized, we realized it would be perfect entry points for new zombies to enter countries through an airport.

Design Evolution:

Initially, our visualization was simple and limited to just a map view. We had a discussion with our TF and realized that multiple coordinated views would greatly enhance the user experience by giving users access to a lot more information about how diseases are being spread. We had already concluded that a timeline that would allow users to move back to different time steps during the progression of the disease would be a great feature to have but it is also a decent statistic to have it show how many people are infected in the world. So even though we may not add the ability to rewind back in time, we still show useful time based information. A bar chart would serve the purpose of delivering the more granular details that people might want to see in terms of how diseases spread.

The final design that we settled on has a map component as the main part of our visualization, supplemented with a timeline and a bar chart. One thing about the map projection that we did think about was something the accuracy of the mercator projection that we had initially decided to go with. We know that the project is not an accurate representation of area in the world and therefore we considered other projection models. In the end, we stuck to our original visualization, since the mercator projection is the one that most people are comfortable with and used to seeing. That familiarity would make the user interface less foreign to users (requiring less time to get used to what they are seeing).

We dropped some off the optional features that we proposed because we realized they did not contribute well to our goal of making an interesting and useful tool for modeling a zombie apocalypse. For example, one of our extra features was multiple starting points but we realized our justification for wanting that did not make sense. We also wanted to fast-forward the animation but realized that it would cause issues with the Google Maps API Geocoder because we have a limit to how often we can query it. In fact, the query limit was so severe that the zombies spread to new locations a lot slower than we originally wanted. Because of that, we increased the block size of where the zombies are spreading so that overall the simulation runs faster. What we have now is a simulation that may start off a little slow but escalates into an apocalypse quickly.

The model we used has a flaw in that mathematical errors can cause numbers to explode. Because this is a density based model, we multiply the number of susceptible (S) people against the number of infected (I) to calculate a value that will be used to change S, I, and R (recovered or in this case dead). This SI value will larger than S and I individually and so multiplying it with a high beta or kill parameter will still produce a large number. Therefore, we might accidentally calculate more infected people than there are people in general at that point on the map. Due to that and issues with calculating numbers, we decided to modify the paper’s model slightly so that it scaled well with total number of people in the area. This corrected the issue.

Another issue with the model is that it does not represent natural diseases well. We had anticipated finding a few diseases that spread like the zombies do, but most diseases are modelled as following a frequency-based SIR model as opposed to the density-based SIR model that the paper followed. Because of that, we could not justify our additional feature of preset parameters.

A final issue is that we are using Google Maps Geocoding API to find out what country a point belongs to but we are limited in the queries we can make per second. We could allow a lot of zombies to spread at once, but that causes us to get blocked by the geocoding API which also slows down our model. This issue becomes unmanageable later in the simulation as the number of infected regions explodes, so to avoid these issues we only allowed a small number of zombies to spread to neighboring regions

We did try many options for reverse geocoding, but we needed quick responses and we found Google Maps provided the best service. At first we thought we had a daily query limit but we in fact do not when using their client side Javascript API. We did consider using a local reverse geocoder but none of the options we saw performed adequately.

For a bit of fun, we added a zombie game as an additional feature. That can be played by entering the Konami Code which will launch the game. Enjoy!

Implementation:

After considering many different options, we decided that what we were aiming to accomplish was best completed using d3.js. We went with our idea of using “discrete population blocks” for our model to spread the disease and visualize it. What we mean by that is we have data that tells us population densities for various countries. We can try to break the country down into one square kilometer blocks, where, since we say the country had 50 people per square kilometer, we can now say that there are 50 people in the block. The block itself will follow the normal SIR dynamics which will be simple to implement but infected individuals will spread into neighboring blocks with a certain probability.

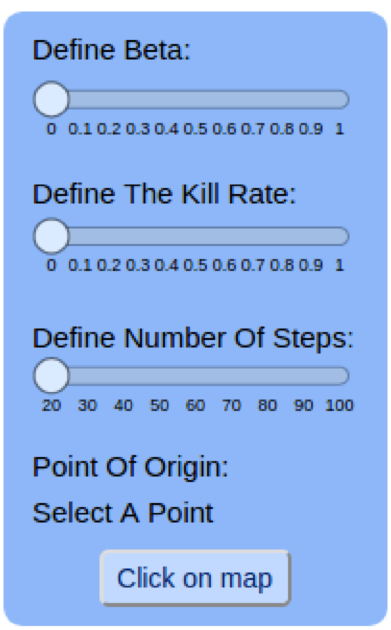
When it comes to spreading the disease outside of an infected block, we tried to think about what would be the most faithful way to do it with the model? We decided that there were two ways for infections to come into new blocks:

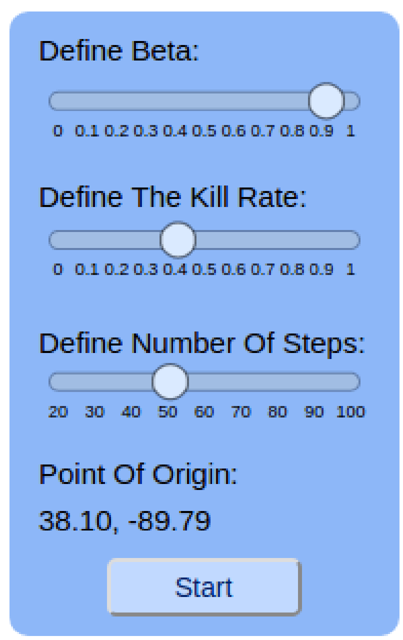
1. They can spread from a neighboring block
2. They can come from a whole other country through an airport point

For the neighboring block version of spreading the disease, we were thinking that we would find the frontier of unaffected places around a set of infected blocks. Together, this mega-block (the blocks with infections and the most immediate neighbor blocks without infections) would be the most important base for modeling the spread of the disease. We can run the SIR model for a step on the megablock and any new infected people we might spread probabilistically across the megablock. This way, we can have the disease spread with accordance to the SIR model and also, with respect to this understanding, we can have diseases spread locally.

We did not specify the need for airport data in our proposal, but we thought our model would be better if a disease enters a new country that does not neighbor any infected countries by way of airports. We already have migrational data so we can model how many people might travel from one country to another, but now we'll only let infected people enter a new country through its airport.

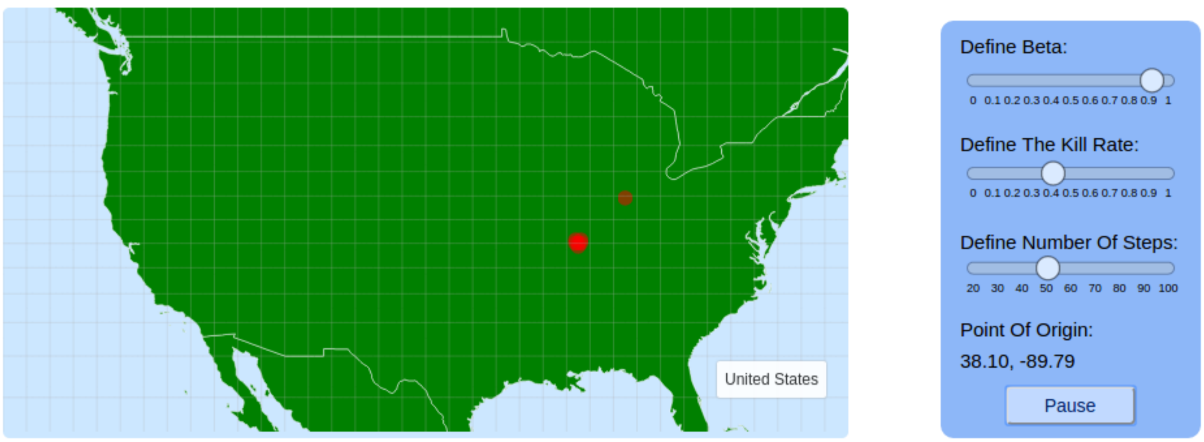
For input parameters, we decided to use sliders after our partner group at the design studio recommended it as a better way to interact. It also made sense because we needed to limit the values of the parameters anyways. We extended it to the slider that determines the number of steps for the sake of consistency.



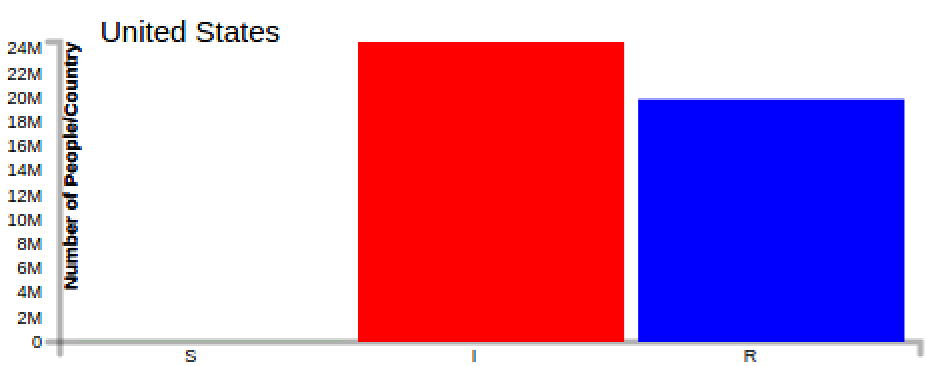


Once a point of origin has been selected, the start button because enabled and the simulation can start running.

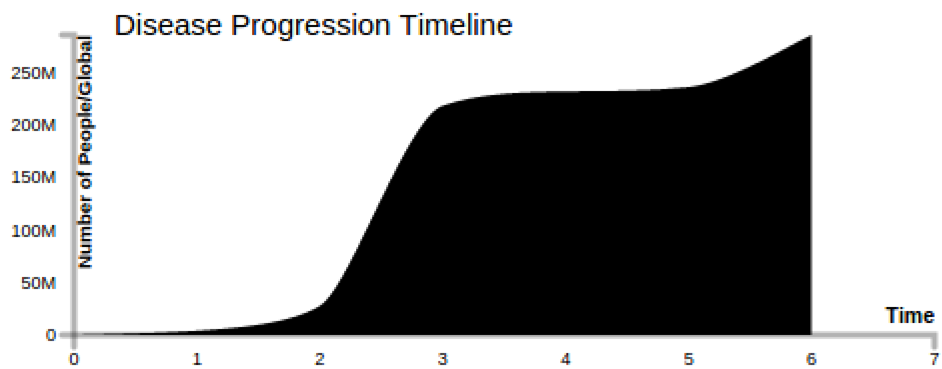
The map is the main portion of our visualization. One can zoom in on it, move around on it, and click on a spot to mark it as the point of origin for the infection.



Hovering over a country displays some useful stats. We have a bar graph showing the number of people in infection critical zones of the country. In this way, we can watch as susceptible people become zombies and eventually die.



Finally, we added a timeline plot on the bottom left. This shows the number of infected people in the world over time. We had planned to also use this as a slider to go back to previous steps, but found doing so broke parts of our model so we decided to avoid it instead.



Evaluation:

We use airport data as a large part of our visualization process. There are a couple of things that we noticed, based off of different starting points for the virus. Some of these discoveries led us to problems within our code as well. One very notable thing was how, when diseases started in China, they spread really quickly to the United States. One bug that we noticed while running our visualization was that if the disease started in Russia, it never spread anywhere. Since diseases never spread into Russia via air routes either (when the disease started in other countries), we think that there may be some naming differences in our data, causing a bug. Another thing that we noticed was that vacation spots and tourist destinations seemed to be overridden with zombies the fastest (sorry Harvard). While running simulations, we noticed that we were not getting as much localized zombie spreading as we wanted. The reason was because of the google maps API’s limits on the number of requests that could be made. We changed our model to adapt for this, but the number of zombies worldwide became unrealistic, so due to the limits placed on us by google maps geocoder API, we had to make design decisions for our model.

Using the visualization we created, we were able to answer the question of how zombies would spread across the world. We saw that places that were more populated and places that got more tourists were more heavily impacted. This showed us the impact of air traffic and these other different patterns that we were hoping to see.

In order to further improve our visualization, we would need a faster API, which does not have a limit on the number of queries you can make. Since we have a bug with Russia, we might want to find out how we can resolve our data so that everything matches. Other than that, our model is accurate to the model that was described in the papers that we used for reference and we got to see how population densities, as well as air traffic, impacted the spread of a disease like a zombie apocalypse.

Now use that Konami code on our site and kill some zombies!!