Before we setup

1. **Overall goals and purpose. What does the program do and why?**
    - Demonstrate an efficient evacuation of the building during a fire drill or an actual fire
    - The program finds the shortest route to an emergency exit so people always know the most effective way to get out in case of an emergency. The routes are adjusted if there is a fire blocking their fastest route out.

After setup

2. **Entities. What do patches represent and what do agents represent?**
    - Patches represent a part of the floor plan - a room, wall, doorway, exit, outdoors or a part of the fire
    - The agents are people in the building during the drill / fire
    - There is also one agent that is the fire when it is not a fire drill
    - The patches that are within a certain radius of that fire agent (depending on the size of the fire) are colored grey and given a class on-fire

3. **Spatial and temporal scale. Size of the world. Time of a tick.**
    - Tick is a second
    - The size of the world depends on the floor plan used, however they are typically 300-500 pixels wide and 200-300 pixels tall.
    - To give you an idea, the walls are typically 2-3 patches wide
    - World represents a building and a bit of its surroundings on the outside
    -

During Simulation

4. **Process and behaviors. What happens at each step?**
    - At each tick each person moves forward once to the next patch that is along their closest route out of the building.
    - The speed and behavior of the people are adjusted based off “chaos-levels”
    - The user has an option to choose whether the simulation is a fire drill (finds the most efficient exit) or a real fire (finds the best way out avoiding the fire).
    - User can choose between a randomly placed fire and a specific location for a fire

5. **Techniques. How did you do it?**
    - We have a setup button that sets up the world and variables
    - Underlying in the setup button we store the distance to the exit and closest exit for each patch
    - We used a mix of BFS and a Priority queue
    - We have 4 separate go functions: one for patient people, one for worried people, one for chaotic people and one for people once they have exited the building and are outside
    - Patient people wait if someone in front of them, worried people move around them, chaotic people push them aside - each with varying speeds
    - Model runs until everyone is safely out of the building or trapped inside

After - with code for dist-to-exit function

6. **Challenges: What made it hard?**
   - Our first challenge came from noisy data in our floorplan images because our rooms, walls, exits, doorways and outside come from the colors. The issue we had was that there were a lot of pixels with colors not matching our classes, so we decided to clean the image up to make it more effective - used shade-of netlogo function
   - Our next biggest challenge was finding the evacuation routes because we were not able to use the built in distance function because people can not go through walls. So, we combined a BFS, Dijkstra's algorithm and a Priority Queue to optimize the exit routes. (mix of island function and BFS)

7. **What did you learn?**
   - I learned how chaotic a fire drill/fire can be, especially when people are not being respectful of others
   - There are also many nuances that come with determining the best path through a building and complications that arise from real life floor plans
   - Planning and organizing your ideas and trying to foresee issues before you start coding is important
   -
   - We learned that a lot of code can be used across different projects and also be combined with more code to make a more efficient solution