NOTE: for optimal gaming experience, use Safari!

Going Viral

COS 426 Final Project
Jeongmin (JM) Cho, Claire Guthrie, Heidi Kim, and Oliver Schwartz
Advised by William Sweeney

Abstract

This paper details the approach, implementation, and results of Going Viral, a COVID-19 themed arcade-type survival game. You, the player, control a healthy white cell while trying to avoid infection by enemy virus particles. You control your cell with the arrow keys and the spacebar (to jump). As you successfully complete levels, the game becomes more difficult, challenging the user to be more daring. Stay Healthy!

1 Motivation and Goal

Given the current state of our world, we were inspired by the global pandemic to create a COVID-19 themed game that was aimed at avoiding the virus and staying healthy -- two things that are quite important in our real lives right now! On a broader level our goal was simple: to make an arcade game that was as entertaining as possible. We achieved this through two main methods: visual experience and game mechanics. We applied our creative talents to mimic the inside of a cell passageway in the body with textures, lighting, and objects. Additionally, we used a physics engine to make the game mechanics as smooth as possible to mimic natural movement as closely as possible.

Our game is set inside the body within a bloodstream. The player of the game takes the point of view of a white blood cell, the main immune defense in our body. As the player moves through the game, they must avoid colliding with viruses or their health will diminish. The goal of the game is to reach the end of the path without losing all health.

Our project is designed to provide light-hearted relief and entertainment for the thousands cooped-up indoors around the world right now. We believe that our game would be fun to play for children and adults of all ages, especially given the relevance of our theme, realistic graphics, and challenging levels!

2 Previous Work

In the creation of this game, we were inspired by a number of “runner” type games, such as Subway Surfers, Cube Runner, and Temple Run. The aim of this type of game is to survive as long as possible (or reach the end of the environment) while avoiding the obstacles in the scene which will either kill the player or cause them to lose “health.” A number of games of this design and logic are set on an infinite pathway. With the requirement of rendering numerous virus objects in the scene, we believed that this infinite pathway approach would potentially fail due to the amount of computing power that it would require. Thus, we instead decided to make ours a finite “runner” game.

With regards to the design of our project, we were inspired by a past COS-426 project, ColoRing, which is how we made our choice to use rolling meshes and a bounded playing space with a colored tiled floor.

3 Approach

Because we were worried about the amount of computing power required to render hundreds of viruses and vein graphics over an infinite environment, we decided instead to make a bounded hallway that would mimic a portion of a human vein. Throughout the environment, we render a set number of virus cells that randomly move about the scene. As they pass over a tile they “infect” it (coloring the tile), which would also impose damage on the player’s health if passed over. Since the length of the game is finite, we decided to create multiple levels of increasing difficulty. As the player completes one level, they are allowed to pass onto the next; however, if they lose, they must restart on level 1. We think this approach makes the most sense for the number of viruses and the detail within the scene that we were trying to achieve. Had we decided to use simpler graphics and a less complex scene, it may have made more sense to make the game infinite.

In our graphical approach, we wanted to create a realistic (but not too gross) scene for our game. Thus, we wanted to use different textures and overlays for the planes and objects in our scene in order to imitate the look of the inside of a vein within the body. We still wanted to maintain somewhat of a fun and polished effect even with the natural textures and imagery.

4 Methodology

We began with the Final Project Starter Code provided to us and THREE.js since we were intimately familiar with it from our work in the course assignments. We then implemented the basic scene, game particles, and game logic to create our game. There were several different components of the game that we had to build in order to assemble the final product. These included the lights, floor tiles, walls, virus particles, the player particle, larger 'boss' viruses, game logic (i.e. additional levels, health, and progress), and music. Each of these had its own unique challenges, and some were more difficult than others. We will limit our discussion to the more complex pieces of the game.

Game Logic

At the core of our game logic is the idea of moving from one end of the rectangular arena to the other end without losing the initially allotted amount of health from collisions. We track the position of the white cell relative to the length of the arena to extract a measure of "progress" which determines when to set the game state to "win" and progress to next levels. We also identify the objects of collision on the white cell, which maps to a unique amount of health to deduct from the initial health (e.g. normal virus deducts 25, boss virus deducts 50) to determine a "gameover" state when the health reaches 0. To visualize these aspects, we created a health bar, progress bar, level label, and varied logic of virus behaviors at different levels.

Game particles

There were several possible approaches we could have taken, including manually programming in constraints and conditionals to handle collisions. To our detriment, in our first attempt we did not integrate any physics engine. This made collision-handling very difficult, compounded by the sheer number of objects in our game world. After we realized how significantly a physics engine would help, we integrated CANNON.js into our project. This made the game logic (and our code) much simpler. As a result, we were able to drastically improve our development speed. The approach we took leveraged the help of CANNON.js: for every object in the game, we created a CANNON mesh, as well as a THREE mesh. At every timestep, we would update the CANNON world (by using instance methods of a CANNON class), and then update the positions of our THREE meshes with the corresponding CANNON mesh position.

Game design: Meshes and textures

In the beginning, we struggled in creating a game board that would be able to render quickly enough, while still creating the dynamically colored “tile” effect that we originally wanted. At first we were creating each box mesh individually, but when making the long “hallway,” this ended up being too slow and laggy. To accommodate this size, we instead implemented plane meshes, which greatly improved this problem.


For the viruses and the main white blood cell, in order to create an organic feel, we leveraged several different Google Poly meshes for our particles (including the virus, cell objects, and powerups), and JPGs for texturing the background and the playing surface. At first, we had difficulty when attempting to integrate Google Poly meshes: we tried using several different file formats (GLTF, GLB) to no avail, and ended up using OBJ files as they were the least troublesome. However, OBJ and MTL files proved to be an additional struggle for deployment, hence we resorted back to loading GLB files. Additionally, we used perlin noise to create a textured, vein surface for the boundary walls of our game, in order to make the scene feel more realistic (see image below). To generate this bumpy effect, we referred to a public, three-dimensional Javascript perlin noise generator, which we have linked in our References section.

5 Results

Throughout the project, we measured success by adding features and surprising ourselves with how entertaining the game was to play. Another important success metric was how enjoyable the project was to work on, and how evenly tasks were distributed across members.

We conducted several experiments with our project. All four of us engaged our family members with alpha-testing in order to collect immediate feedback on things that needed improving, as well as ideas for additional features. Our alpha-testing rendered several important results. Claire's brother pointed out that we could use 'boss' particles to add a more competitive flavour to the game. Heidi's parents suggested that we incorporate a progress bar in order to see the distance left to cover in the level. Oliver's brothers suggested adding sound effects and a soundtrack to the game for another dimension of fun. JM's brother prudently suggested adding shadows in order to gauge the height of the particle when jumping.

6 Discussion & Conclusion

Overall, the approach we took was very promising. We were able to achieve our goal of making an entertaining game that included multiple levels of difficulty with realistic movements and a body-like scene. We are very happy with the playability of the game, as well as the overall graphic aesthetic of the scene.

In the future, we think it could be interesting to add more levels and increase the difficulty even further, as well as add options to toggle board size or difficulty level. Additionally, we think it could be fun to add high scores and maybe even a leaderboard! We particularly enjoyed this project as we were able to combine so much of what we have learned throughout this semester into one product, and create something that is really fun to play!

7 Contributions

Overall, we found contributions to be very well-distributed across our members, in terms of both time committed and changes made to different components of our game (brainstorming/ideation, logic, styling, deployment, etc). We made sure to begin

development by brainstorming together to consider all our options, before settling on a product we all felt we could fully contribute our time and ideas to, and derive satisfaction from. We continued with this highly collaborative dynamic in setting up our core programming framework (e.g. our component/class structuring, relative to our main game file 'app.js' and our 'index.html'), before moving on to implementation details that specific members could take on. Throughout the rest of our design and development process, we met daily over Zoom sessions, often pair-programming over Live Share and checking in with each other about any substantial ideas we might have. We are happy to report that the division of labor was not just equal, but highly enjoyable. That said, below we have outlined some of the primary contributions of each group member:

Oliver worked on integrating the CANNON.js physics engine with the THREE.js rendering, in order to create the underlying physics for the game. Working with Heidi, Oliver also incorporated event listeners and other physics logic to add movement to the player particles (including friction, wall collisions, jumping, and back/forth movement). Together, Claire and Oliver created the health bar component complete with dynamic animation. In addition, Oliver created the power-ups for adding health and added logic to collide with them. He also added the various sound effects and soundtracks in the game (i.e. the backing soundtrack, the 'damage' noise, and the 'healing' noise). Oliver also added shadows underneath the player-controlled cell by overlaying a circular planar mesh on the game floor.

After we set up our game's floor, Heidi worked on having our virus particles color different tiles on contact, as well as randomly walk in 8 directions (N, S, E, W, etc.) from their current tile. Heidi and Oliver later worked on having these viruses and infected tiles damage the player cell's health. Heidi also added state logic to our game, in order to allow for more logical, seamless transitions between different divisions of our game (i.e. from menu to play, to reset, and so on). When we met with Will for our final check-in, we received feedback that more realistic, textured walls might improve our game's visual appeal. Heidi made changes to incorporate Perlin noise generation in the initialization of these walls.

Claire, along with JM, worked a lot on designing the aesthetics and theming of the game in order to create a cohesive and fun overall game scene. This included drawing initial ideas and potential prototypes for the game, and then later the objects, color scheme, and textures in the game. She also helped to guide the initial logic of the game when we decided to pivot away from our original color tile logic. In the beginning, Claire helped work on the tiling of the board and coloring the tiles as the virus moved over them. Later, she worked mostly on the logic side of the game, implementing the Menu and

instructions page (prototyping in CodePen & JSFiddle and implementation) and the CSS for the menu and main game scene, the dynamic health bar with Ollie, and leveling for the game.

At the beginning, JM helped guide the direction of the game logic, pivoting from the original idea of conquering a square floor space to a “virus” runner game across a highway. JM and Claire designed the aesthetics of the game by deciding on the overall color scheme and adding CSS to construct the lead-in menu, gameover/win screens, and white cell movement progress bar. JM also designed the variations of viruses across all five levels of the game to add the fun of progressive challenges to the game logic, and implemented it by adding modularity to the virus codebase to simplify the logic of spawning new virus cells. JM also figured out deployment difficulties with static assets by converting OBJ and MTL files to GLB and solving issues with the CSS-loader.

References

“Three.js Shadows.” *Three.js Shadows*, threejsfundamentals.org/threejs/lessons/threejs-shadows.html.

Wwwtyro. “Wwwtyro/Perlin.js.” *GitHub*, github.com/wwwtyro/perlin.js.

Google Poly Objects:

https://poly.google.com/view/d_QDWsT0kBG

https://poly.google.com/view/aoZ4DjiO35h

https://poly.google.com/view/bnAE4wYavQQ

https://poly.google.com/view/40xh2QRX9pA