Carnival Shooting Gallery

Gabriel Vivas
Gabriel.Vivas001@umb.edu
University of Massachusetts Boston

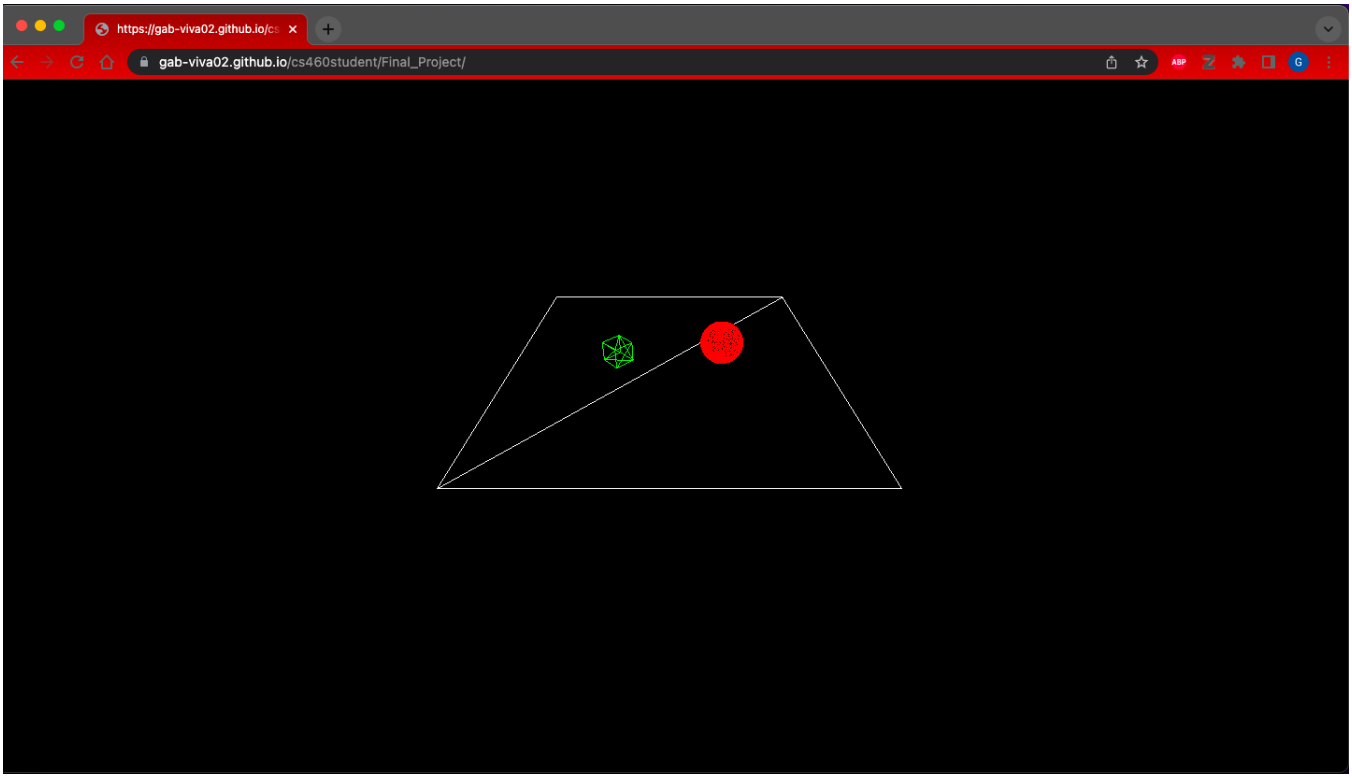

Figure 1: The work I managed to get done before I needed to write this report.

ABSTRACT

This project was meant to be a sort of time-waster-type game, where a limited degree of interactivity coupled with physics-driven interactions would combine to make a mildly-entertaining game that was enjoyable in short bursts, in the same way that a yo-yo or a fidget spinner might be. The "carnival" descriptor serves more so to give an idea as to the number of interactions available to the user (point and click → point and shoot), rather than as a descriptor for the kinds of visual aesthetics the project was meant to achieve. A changing environment was meant to provide visual variety to the user, in addition to providing new situations in which to point and shoot, as it isn't very fun to shoot at the pieces of a structure that has already been toppled. A scoring system was soon taken off the cards, as the primary point of interest was the physics interactions, not a scoring system.

CS460, Fall 2022, Boston, MA

ACM ISBN 1337.
https://gab-viva02.github.io/cs460student/Final_Project/

KEYWORDS

WebGL, Visualization

ACM Reference Format:
Gabriel Vivas. 2022. Carnival Shooting Gallery. In *CS460: Computer Graphics at UMass Boston, Fall 2022*. Boston, MA, USA, 3 pages. https://gab-viva02.github.io/cs460student/Final_Project/

1 INTRODUCTION

This project was meant to be my first real attempt at making a video game-like thing using the experience I've gained so far as a CS student. For my final project, I knew I wanted to make something interactive, as I didn't think I could really make or come up with any kind of visual showcase for my final project. I know that the games I usually enjoy actually take a lot of time and effort to design and program, so I figured that a game that only involved pointing and clicking would be a good place to start. Plus, I also figured doing something like this involving 3D spaces would be a good

opportunity for me to apply what we had learned about raycasting earlier in the semester. Finally, I figured that working on such a project would give me a good excuse to learn how a physics engine works. Ever since I was little, I was always amused by the way that physics objects would fumble about in games, like when a crate would get broken and the pieces of the crate would scatter about. Thus, I figured that doing this project would force me to get hands-on with a technology that I'd been curious about in the past, but had no idea where to begin with before.

The contribution I had been aiming for was to make a little physics-based game with toon-shading, as I have a big appreciation for games with cel-shading, but I rarely see such games release that also have a focus on physics interactions.

2 RELATED WORK

Three.js [1].

Cannon.js [2]

Cannon-es [6]

3 METHOD

My project has not changed since what I showed during my final presentation. Upon loading the page, a cube and a sphere spawn over an infinite plane, implementing some test functions that were explained in a YouTube video I saw giving a quick demo on Cannon.js. From this point forward, the user can click anywhere in the window, and a projectile ball will be spawned and launched in said direction, using a ray cast out from the camera looking at the entire scene. At this point, no code has been implemented in order to limit the number of projectiles present in the scene, nor has any code been implemented to change the target scenarios as was desired previously.

3.1 Implementation

I tried to build my project iteratively, in that I would create new html files to test out different features of Cannon.js in order to understand them, before moving on to subsequent html test files, that would contain select amounts of the previous files, plus whatever new elements I wanted to try out for the given file.

To get started, I first tried to take some starter code we had used for one of our homework assignments, and I tried grafting some Cannon.js code into it, so in order to get something on the screen that I could quickly comprehend how it worked. I figured this would be a better way to get started, compared to just building all of my code off someone stranger's Cannon.js code, since in that case I would be dealing with a lot of scaffolding that I didn't want to learn and that I wouldn't want to explain later.

After a long detour that involved figuring out whether I should use Cannon.js or Cannon-es, I ultimately settled on Cannon.js, as at least with Cannon.js I could import it into my code the same way we would import XTK, Three.js, and dat.GUI for most of the semester. From there, I managed to make my first program using Cannon.js, using a bit of starter code we were given for Assignment, combined with this [3] example from the Cannon.js GitHub page. The result of this was the index1.html file that is in my Final_Project directory for this class.

Next, I tried making a more complex program using Cannon.js, that would implement the functionality seen in this [7] YouTube tutorial that I found at the very beginning of my research process for this project. This work, while ultimately not too difficult, was admittedly time-consuming, as I tried to research and take note of all the Cannon.js functionality being used for my own future reference. During this process, I found that some of the documentation on Cannon.js coming from Stefan Hedman was lacking. Thus, I started turning to the documentation for Cannon-es as well, as that documentation sometimes explained things more clearly than the documentation for Cannon.js did. Eventually, using index1.html as a starting point, I wrote index2.html to implement this desired functionality, at which point I felt comfortable enough with my knowledge on Cannon.js to write the functionality for launching a projectile upon the user's mouse click.

At this point, I started studying the functionality of this [5] Cannon.js demo to understand how it managed to create new objects at the click of the mouse, while also launching them in a desired direction. After understanding how that worked, I tried combining that functionality with the raycasting information we learned for Assignment 3 in order to create functionality that would create and launch projectiles from right below the static camera used to view the scene in the web page. The result of this was index3.html, which was what I presented during my Final Project presentation.

From here, I had hoped to move on to figuring out how to change scenes as seen in this [4] demo, when picking between 4 different scenes in a dat.GUI window changes what scene is rendered inside the window. However, as I will explain later, this did not come to pass.

3.2 Milestones

How did you structure the development?

3.2.1 Milestone 1. Come up with potential project ideas to present to the instructor

3.2.2 Milestone 2. Set up the development environment for using Cannon.js

3.2.3 Milestone 3. Create an initial program using Cannon.js

3.2.4 Milestone 4. Develop enough confidence with Cannon.js so as to be able to create the projectile-launch-upon-click feature

3.2.5 Milestone 5. Create the projectile-launch-upon-click feature for my project

3.2.6 Milestone 6. Study how other Cannon.js demos instantly load in new scenes to render

3.2.7 Milestone 6. Create the scene-change feature for my project

3.2.8 Milestone 7. Create one additional scene in order to use the scene-change feature

3.2.9 Milestone 8. Create one additional scene that uses as-of-yet unused functionality in Cannon.js (cloth, connected bridge, etc.)

3.2.10 Milestone 9. Change the rendering style to suit my aesthetic desires for the project

3.2.11 Milestone 10. Design as many additional scenes to change to as possible

3.3 Challenges

Describe the challenges you faced.

- Challenge 1: Setting up a development environment for Cannon.js that I could understand (I spent a lot of time trying to figure out how to use Cannon-es, but I couldn't make it work)
- Challenge 2: Familiarizing myself with Cannon.js
- Challenge 3: Having enough work to present during the Final Presentation
- Challenge 4: Finding the time to work on the Final Project after the Final Project Presentation

4 RESULTS

Ultimately, I got the very basic functionality of my program finished, which just involved implementing the point-and-shoot mechanic, but I didn't get to design any of the additional scenes that I wanted to work on, even though I did familiarize myself with Cannon.js that I think I could do that eventually.

5 CONCLUSIONS

Ultimately, this project was a victim of all the additional homework assignments I had to finish after the last day of class. I initially had a good feeling that I could finish up at least up to Milestone 9 within the 4 days leading up to the due date, but the other homework I had to get done took longer than I expected, leaving me in this current situation. In conclusion, the takeaway I'm getting from this project is that I really should have split it into smaller chunks, so I would have felt less daunted to work on it on a regular basis.

REFERENCES

[1] Ricardo Cabello et al. 2010. Three.js. *URL: https://github. com/mrdoob/three.js* (2010).
[2] Stefan Hedman. 2012. Cannon.js. *URL: https://github.com/schteppe/cannon.js* (2012).
[3] Stefan Hedman. 2012. Cannon.js Three.js example. *URL: https://github.com/schteppe/cannon.js/blob/master/examples/threejs.html* (2012).
[4] Stefan Hedman. 2012. collisions. *https://schteppe.github.io/cannon.js/demos/collisions.html* (2012).
[5] Stefan Hedman. 2012. threejs$_fps$.