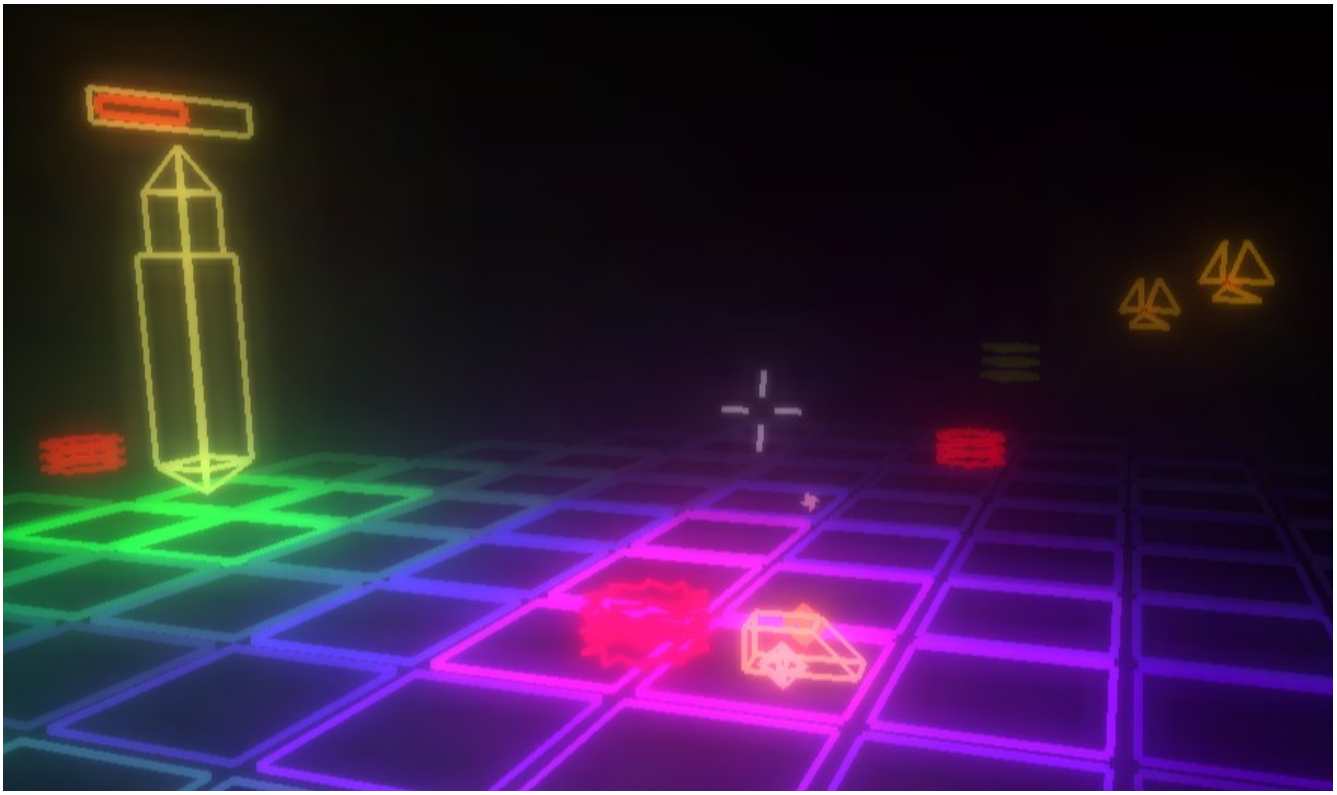


Daniel MacLeod
May 2022
Final Project Report
‘Neo Invaders’
Demo at: https://danielbentleymacleod.github.io/test/

Introduction

For this project I constructed a third person shooter using wire-frame graphics to give it futuristic, cyberspace ascetic. In the game, the player tries to charge up pillars while defending themselves against infinitely spawning waves of enemies. This continues until the player dies. The player is given a score at the end of the game. Players get points by destroying enemies, but the amount of points they get is determined by how many pillars they have charged. The size of the waves the player is attacked by is determined by how long the player has been alive for. This encourages the player to try to charge the pillars as quickly as possible in order to maximize their score.

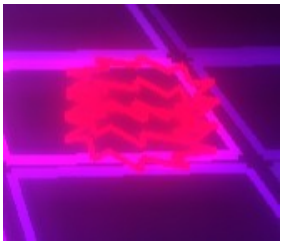
The code for this project is structured as follows: The main program is app.js. This program initializes the Three JS scene, the camera, the music, and other necessary system functions. It also is responsible for reading the users inputs and adjusting the state of the scene in response to those inputs. This program contains the renderer, which is notable as it is overlaying a bloom effect on the entire game in order to

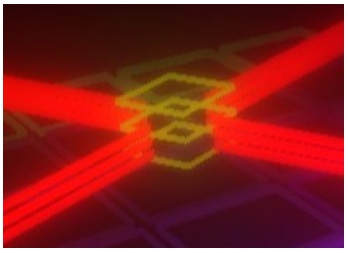
provide a glow like effect without having any real lighting. The scene contains all the relevant variables for the current game in its state, such as the players position and velocity, the direction of the camera, the current time, etc. Every object in the scene is added to it as a child node. All the objects in the game are Three JS groups, and have their individual information stored inside their state. Their graphics are either a mesh or a set of lines added to the group for that object.

Components

- The Player. The player is controlled though the WASD keys and has the ability to jump with the space button. The camera and aim direction can be moved around with the mouse, and the player can fire with any mouse button. Power ups are activated with the Q button. The jump experiences greater gravity when moving downwards than upwards to allow for greater player control. The design for the player is a basic shape to represent a ship with diamonds on either side which the players shots originate from. The cross-hair for shooting is an object in the world which is fixed in place some distance above the player, and is rotated to look at the camera every frame. The camera is located a fixed distance from the cross-hair. It is rotated around the cross-hair when the view direction is changed but is always pointed directly at the cross-hair. The camera points at a location above the player rather than the player itself so that the player can look upwards without moving the camera under the floor. The players attacks originate from the camera instead of the player so that the cross hair always lines up with the target that will be hit, but the visual effects originate from the player to give the illusion the player is shooting.

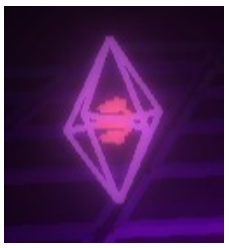
- The Arena. The arena is an array of squares all attached to a group object. It is square and non-infinite, and the player will fall off the edges if the go outside of it. The original concept for the arena was to have it light up in a trail as the player moved around it. The final version does this, but in a different way than expected. Every square in the arena calculates a brightness value based on its distance from the player, with closer squares being brighter. Each square moves towards this value over time. This does result in a trail effect, but it is more diffuse than what was originally conceived. This also makes only a small area of the arena visible at once, which benefits performance.






- The Enemies. There are currently four different enemies that attack the player. They are designed to each make the player evade in a different way. They are all different colors to be distinct against the dark backdrop, and are split between different elevations to make the shooting more diverse. In order to make it clear how each enemy damages the player, everything that will hurt the player is the same shade of bright red. Destroying the enemies increases the players score, by 1xM for the small enemies and 2xM for the larger enemies where M is the score multiplier.

- The Pillars. In reality, there is only one pillar which moves around after it is charged. The pillar have a meter on top of them which fills up if the player is close enough and depletes if the player moves out of the charging area. The player is lit up when in the charging area, to make it clear where the boundary is. These were not originally planned, but were added in order to prevent the players from constantly backtracking from groups of enemies. A style of play where the player was constantly running away was not very fun, especially since the enemies tended to group up if you ran circles around them for long enough. The pillars encourage the player to stay in a smaller area and actually fight and dodge enemy attacks, while still giving them to option to run away if they are getting overwhelmed. Charging the meter on top of a tower will double the players score multiplier, which starts at 1, making them essential in order to get high scores.



- The Power Ups. Charging a pillar to full will spawn in a power up, which is a special enemy that does not damage the player, but will disappear after a short amount of time. If the player destroys it, they will gain a power up which is displayed above the player. The player can hold up to three power ups. Pressing the Q button will activate the most recently held power up. There are currently three power ups in the game. The first causes an explosion which destroys all enemies in an area., the second increases the players speed and damage for a brief period, the last one makes the player invincible for a short time. They are color coded red, green, and blue respecitvely.

- The Music. Special thanks to David Renda for providing royalty free music. The track being used is called ‘Boomerang’

Implementation Challenges

There were several issues that proved to be problematic for this project. The first major issue was in creating the glow effect in Three JS. Implementing proper lighting for this project would have been a major performance drain and not really necessary to get the desired ‘cyberspace’ effect. Instead, a bloom filter was added to the renderer. It was a big challenge to get this renderer working at all, as the documentation was not

great and it was not clear how to properly add the bloom render stage to the renderer. After it was added, there were further issues with the filter as it would apply bloom to everything equally. If the bloom was too high, bright objects would appear blurry. If it was too low, darker object would not appear to glow. After a lot of experimentation, settings that got a pretty good effect were decided on. However, some of the objects in the scene had to be changed to accommodate the filter. For example, the text at the start and end of the game had to be made transparent, because if it was not the bright bloom would make it illegible.

Another major headache was the firing of the players weapon. I knew early on that it was going to be hit-scan. This is because in the third person perspective, it is important for the players aim to be relative to the camera so that they are actually shooting what the cross-hair lines up with, but it still is important to make it look like the player is shooting. With a hit-scan weapon, the point of impact is determined as soon as the player shoots, so direction of the visual effect of the projectile can immediately be calculated. However, it was extremely difficult to get the players projectile to move in the correct manner, as the enemies location, the collision with the enemy, and the location of the diamonds on the player were all in different reference coordinates. Initially, the projectiles were children of the player, but this had to be changed as the player could move or rotate while the projectiles were in motion, which would move the projectiles and look absurd. As a result, the projectiles are their own object in the scene, and when the player fires all the relevant locations are converted into the global coordinate system. There are only 4 projectiles initialized in the scene, and the player iterates though a list of them whenever they fire. This is enough to give the illusion of continuously firing projectiles without the need to repeatedly create and delete new objects.

Current Issues

There are still a couple of issues with the current build of the game. The collision for enemies is not perfect. It was put together in a sort of slapdash way, and as a result can feel slightly off at times. Instead of finding a proper collision, the program only checks if the nearest point to an enemy on the attack vector is inside the enemy. It is intentionally designed to be favorable to the player, with the hit-boxes for every object except the player being a lot wider than the visuals would suggest. The player, on the other hand, has very nearly a point hit-box. This is generally okay, but there is an issue here because some enemies have inconsistent attack hit-boxes. For example, the saw enemies have to be nearly halfway though the player to register a horizontal collision, but will register a vertical collision if the player so much as brushes them from above. Tightening up these hit-boxes could be a big area for improvement.

Even after all the changes to the player projectiles, there is still a minor visual issue. The projectiles disappear after the front of the projectile reaches the target, but since the projectiles are quite long, they can spawn already having reached very near

targets. This causes the projectiles to disappear the frame after they are spawned, which looks terrible. I have no idea how to fix this, it may require shortening the projectiles or just not spawning them at all if the target is too close.

Conclusions

Overall, I am pretty happy with how this project turned out. The end result was very similar to how I envisioned it, and it was fairly fun to play. The ‘cyberspace’ ascetic was not very original, but looked pretty good here and served the game-play well. I would have liked to add more enemies and more mechanics for the player if I had had more time. Having the player die in one hit defiantly limits what can be done with the game as it ensures that the game length must be kept short to limit player frustration. A proper health system would really have to be added to make the game more replayible.