# COS 426 Final Project Write-up

Bensu Sicim & Daniel Pallares

## Introduction

We began this project with the idea to use the idea of a traditional 2D game and build a 3D game from it. Our original plan was to take the idea behind a planet defense game in which a character must fire projectiles at different objects that aim to collide with the planet. Using our knowledge from the course, we had the intention of creating different levels in which environments and gravities could differ. Thus, using our knowledge of textures and animation to create an experience. Originally, we had planned to also incorporate models we found that our character could use to walk around the map.

Looking at traditional 2D planet defense games, we felt that the gave the experience a good feeling of an arcade game. The continued participation affected the score and the activity was engaging and easily repeatable. We looked at a three-dimensional game called Nemesis. The game implemented the mechanics of shooting objects in a way that created an organic shooting mechanism. However, the game lacked three-dimensionality in the way it asked the character to engage with the environment. We took lessons from these two games to incorporate their successes and avoid their shortcomings in our execution.

In our approach, we wanted to create an interesting environment for the player to be surrounded by while playing the game. This would ultimately allow us to create different environments and enrich the differences between ‘levels’ or ‘stages.’ After looking at different ways to implement diverse terrain, we attempted an implementation that created a mountain environment around the player. After extensive modifications to the implementation, we found that there is a far more complicated process associated with the implementation of such environment. The terrain model that we tried to use placed the floor far above the head of the player, and had no clear way of modifying it. We then began questioning the value that the terrain gave to the overall experience based on the cost. An uneven terrain entails different ways in which a character has to interact with the ground. The controls that we had set up player movement were set up with the intention that the player moves along a horizontal plane. Instead of changing the floor to create the environment, we decided that we could create a vivid environment for the player by adding models on the ground. Given the replicability of trees in a forest, we decided that the planet that would be defended by the player would be earth, and that the environment of a forest would create the experience for the player.

After our first iteration, we decided that the complexity of the game didn’t need different gravities to succeed. We could have built several stages with different gravities. However, in the context of a space shooter, that addition would not add significant value to the experience. Given that the environment is a flat plane, there is no incentive built for the player to jump and experience the different gravitational pulls. The only objects affected by gravity are the spaceships. Having them increase the speed in which we approach earth, we felt, was a much

more engaging way to create a game. Otherwise, planets with smaller gravities would have very slow-moving ships and bore the player. Likewise, planets with larger gravities would have very fast-moving ships, and be too difficult for the player to shoot down.

From our observations, we found the game to work better in devices with stronger wireless network connections and computers with better processing capabilities. The game shows to slow when played on devices that don't have the graphics processing capability than the devices we built the game on. Given that our game loads multiple instances of models in different positions, while the player as well as some of the models move in the environment, we find this to be a condition for certain devices to naturally struggle with.

# Methodology

We started implementing this project by first looking at how we could create a 3D environment in which we could control our movements. Our process is outlined below by the explanation of the main building blocks of the game. We knew that we wanted to make a shooting game, but we decided to figure out the details of the game in an organic fashion, as we worked more on it.

- **Controls:** We went through the example projects in three.js website, and saw the PointerLock example, which generates an environment with boxes in which the player can walk around using WASD keys. So we copied the source code of that example as a starting point. Initially, we wanted to make the game in 3rd person view, but we liked the simplicity of the 1st person view and decided to stick with it.
- **Enemies:** Seeing the boxes that were created in the source code gave us the idea of using them as targets, and we modified the box generation code from being called in the beginning to be executed periodically in the animate() function. We added a speed variable that made the boxes descend over time. Then we thought we could improve the design by moving away from boxes and loading 3D models instead. We went on Sketchfab.com, and found a cartoony GLTF UFO model. This model gave us a vision on how we wanted our game to look like, graphically, which we will talk about later.
- **Bullets:** We looked at the example of the game Nemesis for inspiration on how bullet mechanics worked. We generated one each time the user clicked the mouse. Each time step we checked every bullet in the world for a collision with the hitbox of an enemy. If we found one, we removed the bullet and the enemy from the world. Then, we triggered an explosion at the position of the enemy. Initially, we found that bullets were going through the enemy models and not triggering explosions. We increased the size of the hitbox of the enemies to allow the player to take out the enemy targets by hitting any part of their model. In order to make the bullets seem more interesting than just solid spheres, we used a glow texture on them using Threex.GeometricGlow library.
- **Score/Lives:** We knew that we had to add some type of challenge and progression to the game, because we want users to be able to see how they are performing. So we made it so that the player loses lives any time a UFO reaches the ground, and gains 1 point anytime they successfully shoot one.
- **Explosions:** Before we decided to go for the overall cartoon-y visuals of the game, we were considering generating realistic looking fire anytime a UFO disappeared. However,

for some reason we had trouble getting the Fire example from three.js website to work. Instead, we decided to make our own explosions. Any time a UFO explodes, we generate four spheres with different sizes and colors, centered around the explosion, and they rapidly expand and get more transparent until they disappear. In order to make the explosions seem more interesting than just solid spheres, we used a glow texture on them using Threex.GeometricGlow library.

- **Graphical improvements:** As mentioned earlier, we decided to use a cartoon-y low-polygon style on the graphics of the game. Initially we considered adding walls around the player to obscure a view, but after browsing Sketchfab.com, we decided to go with GLTF trees instead. We load the trees in the beginning using a GLTF loader, and randomly spread them around the map. This is also when we decided to change the objective of the game from defending a different planet to defending Earth. We also wanted to make the sky look more pleasing, and ended up adding a color gradient and stars that are made of randomly spread Point objects.

# Results

Throughout our development process, we reached phases in which we felt comfortable testing the product for experience beyond build quality. At these stages, we reached out to peers and had them test the game. We listened to their concerns and incorporated their feedback into subsequent versions of the game.

In their testing of the game, we evaluated how engaged they got with the game. We looked to see their interest in continued participation in the game. We also looked to see whether they would continue playing longer than we required them to for testing. In their tests, we also looked to see that the intent to shoot down an enemy was seamlessly executed by the game in terms of organic aiming and firing behavior. In our latest iterations we found that the peers that tested our game responded positively, and in many cases wanted to continue playing subsequent rounds.

# Discussion

We believe we came up with a decently fun and visually pleasing game. People we showed our game to seemed to enjoy it. We tried to implement concepts from lecture by using the Three.js library, using different types of lights (directional light and hemisphere light), using GLTF models which are scene graphs, and using components such as a camera, raycaster, and renderer to generate the scene. By working on this project, we improved our skills using three.js and WebGL, and we learned how to apply our skills gained from COS 426 to create a game.

However, there is always room for improvement. Since the UFOs get faster and the game gets progressively harder, it becomes virtually impossible to keep playing after a certain point. Additionally, there isn't that much replay value, because the setting and the goal does not change. Moving forward, this project could be improved by adding different levels, different

types of weapons, and different enemies. Adding a leaderboard could also be a nice improvement.

## Conclusion

In conclusion, we are proud of our overall progress with this game, and we hope that it is enjoyable for others who try it out too.

## Sources


- https://threejs.org/examples/?q=pointer#misc_controls_pointerlock
- https://stackoverflow.com/questions/50965025/three-js-shooting-bullet
- https://threejs.org/examples/?q=sky#webgl_shaders_sky
- https://threejs.org/docs/#api/en/materials/PointsMaterial
- https://github.com/jeromeetienne/threex.geometricglow
- https://sketchfab.com/dammne
- https://sketchfab.com/reymarch
- https://github.com/IceCreamYou/Nemesis/blob/master/index.html
- https://fonts.googleapis.com/css?family=Quicksand