COS 426: Final Project Report
Story of the Starfighter

Susan Wang, Humza Azam

May 15, 2018

1 Introduction

1.1 Goal

In our project, we are trying to create a game where the player uses the keyboard to navigate a spaceship, Starfighter, through 3D space. It has to avoid the asteroids spawning in space in order to survive because a single collision with the asteroids means game over. To increase Starfighter's odds of survival, we equipped it with two blasters which constantly shoot out energy bullets to destroy the asteroids.

We aim to use this game to engage children's interest in space exploration and also to practice their hand-eye coordination. Both of us working on this project are big fans of Star Wars, so to be able to implement a game like this is like a dream come true.

1.2 Previous Work

We are inspired by the game Space Conflict Marathon found here. In this game, players rotate the plane in the center which shoots the surrounding objects to avoid collision. This game is in 2D. For our project, We are trying to recreate this concept in 3D, and instead of just rotating the plane, players navigate the spaceship and change its positions in space. While brainstorming how our game could look in 3D, We also referred to this YouTube video of Gummi Ship Missions in Kingdom Hearts. It provided us with some ideas on the UI of our game.

1.3 Approach

In Assignment 5, we created a particle system in animated scenes. We borrowed the framework from the assignment and loaded a 3D mesh of a spaceship into the scene. To enable navigation of the spaceship, we added code for keyboard control such that the players could control the position of the spaceship with WASD or arrow keys. In order to create the feeling of space, we loaded a plane into the background of the scene and added a texture of an image of space we found online. For the asteroids, instead of simply using the particles from assignment 5, we spawn 3D rock-shaped objects with a realistic texture from the back of the scene and implemented the velocity updater so that they move towards the spaceship. To make the asteroids look more realistic, we also gave them a rotation factor so that they rotate along the y-axis as they move towards the spaceship.

After creating the setup of the scene, we then implemented the feature of Starfighter shooting energy bullets by spawning particles from the spaceship. We found a cool-looking texture of the bullet particles online and used it in the scene.

The final part of the game is collision detection. This comes in two parts. Firstly, we need to be able to kill the asteroid objects shot by the energy bullets. Secondly, we need to alert the Game Over message once Starfighter is hit by an asteroid.

This approach should be able to function in all circumstances.

2 Methodology

The following pieces had to be implemented to bring the game alive. We referred to the three.js library [1] for several parts of the implementation.

2.1 Spaceship

The first component was the spaceship. We found a simple spaceship model online [2] and then used Three.js to load that into the scene. We scaled the spaceship in order to fit the environment, and then textured it using the materials that came with the spaceship model. This spaceship is placed in the foreground, close to the user, away from the asteroids, which spawn from the background.

2.2 Keyboard Control of the Spaceship

The keyboard is important for two features in the game.

The first feature is position navigation. To implement keyboard control of the position of the spaceship, we added an event listener in system settings such that the position of the spaceship gets modified once certain keys are pressed. Currently, the spaceship can move in four directions: up (W or upward arrow key), down (S or downward arrow key), left (A or left arrow key) and right (D or right arrow key).

Secondly, we also enable the user to use the keyboard to shoot particles. To do so, the user simply holds down the spacebar to continuously shoot two energy bullets from the spaceships blasters. One problem with that still remains is that the current implementation does not handle multiple keypresses. This means that the user could not shoot and move at the same time, or simultaneously move in two directions, which can be quite annoying. This may be added in the future to provide a smoother gaming experience.

2.3 Space Background

There are several possible methods to load an image into the background to create the feeling of space and make the scene appear more realistic. We could potentially set the background image in index.html by modifying the body tag or adding extra style specifications in the css file. Another method is to add the image as the texture of a plane in the scene. We resorted to this method instead of the previous one because in this case we have more control over the position and size of the plane. We used THREE.PlaneBufferGeometry to create a plane mesh and added the image texture as material of the mesh. After rotating it around we found the optimal angle and location for the background to create the best visual effect.

2.4 Asteroids

Figure 2 shows a close up of the asteroid models that we used for the game. To create these models, we used Assignment 2. We first started with the cube mesh and performed a single round of loop subdivision. We did not use Catmull-Clark in order to keep the mesh a triangle mesh. Then, random vertices were selected and translated along their normals inward to break the convexity of the subdivision surface and add some interesting geometry to look more like an asteroid. We then took an image representing stone texture from online [3] and added it to the asteroid using Blender. This was then added into the game as a base object reference. Multiple asteroids based on this model spawn in the game with different size, orientation, speed, as well as rotation, giving the feeling as if the gamer is in an asteroid field.

2.5 Energy Bullets

To create the Energy Bullets, we simply switched out the texture for the blank particle in the original assignment 5 with a new green energy bullet found online [4].

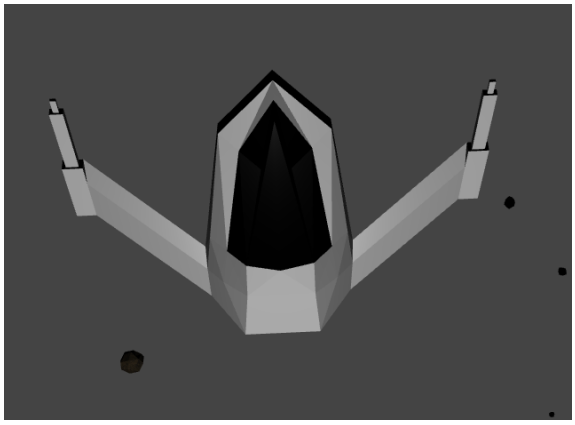

Figure 1: Details on the 3D Mesh of the Spaceship

2.6 Collision Detection

There are two types of collisions in this game. The first is the spaceship colliding with an asteroid, which results in a game over. The second is a bullet colliding with an asteroid, which results in the asteroid being destroyed.

To implement the first, at each time step, we compute the bounding box of the objects comprising the spaceship. Then, for every asteroid within a certain distance of the spaceship, we compute the asteroid's bounding sphere. If any of the bounding boxes of the spaceship intersect with any asteroid's bounding sphere, this counts as a collision, and game over is displayed. We make a simple optimization here by not computing the bounding sphere for each asteroid. We only compute the bounding sphere when an asteroid is within a close enough distance for a collision to happen. This avoids unnecesary and intensive computation.

The second is done in a similar manner. Each bullet is modeled as a point, and whenever the point enters the bounding sphere of an asteroid, we destroy the asteroid. We make the same optimization as in the first type of collision detection to avoid excessive computation.

3 Results

We measure success of our project by its functionality and visual effects. First, the game has to work and allow people to actually play it. This includes functional keyboard control of the spaceship and collision detection. Moreover, the UI of the game has to be visually appealing and realistic to showcase the skills we have learned in COS 426. We would love for the game to be more user friendly and attractive to young players because the goal of the game is to engage their interest in space exploration.

To showcase the results of the functionality and visual effects of our project, we include some screenshots we took of our system below.

Figure 1, 2 and 3 demonstrate the detailed close-ups of the components of the scene. We could see the rendering effects of the 3D spaceship and asteroids, as well as the bullet particles shooting from the spaceship.
Figure 4 demonstrates the setup of the scene. At the beginning of the scene, there is a spaceship at the center of the scene in front of a background wall showing an image of space. The spaceship flies away from the player into space, as a symbol of exploration.

Figure 5 showcases a scene during game action: the asteroids are spawned at a distance and fly towards the spaceship, which shoots energy bullets and tries to dodge the asteroids. After the asteroids cross the vertical plane the spaceship resides without hitting the spaceship or being hit by the bullets, they despawn and disapper.

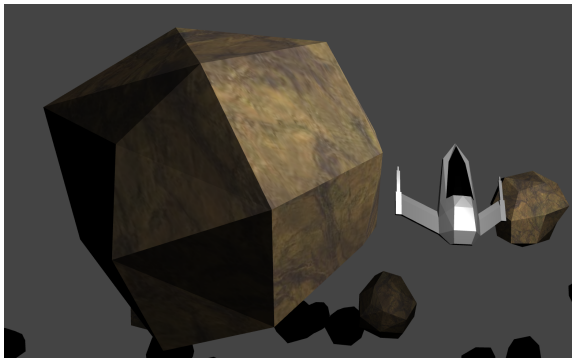

Figure 2: Close-up of the 3D Asteroid Objects

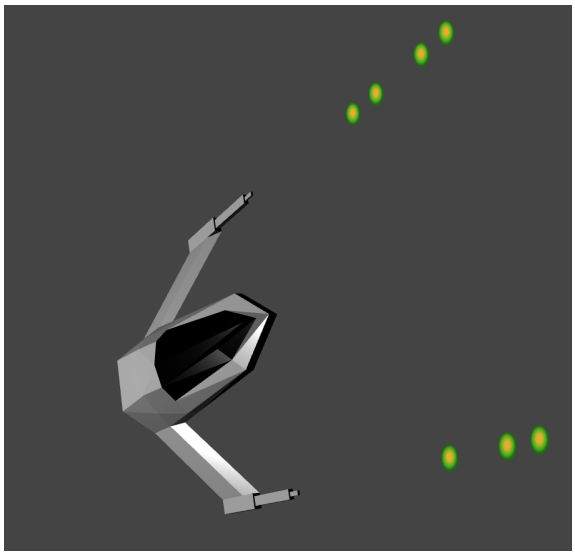

Figure 3: Close-up on Starfighter Shooting Energy Bullets

Figure 4: Setup of the System with Space Background and 3D Mesh of Starfighter

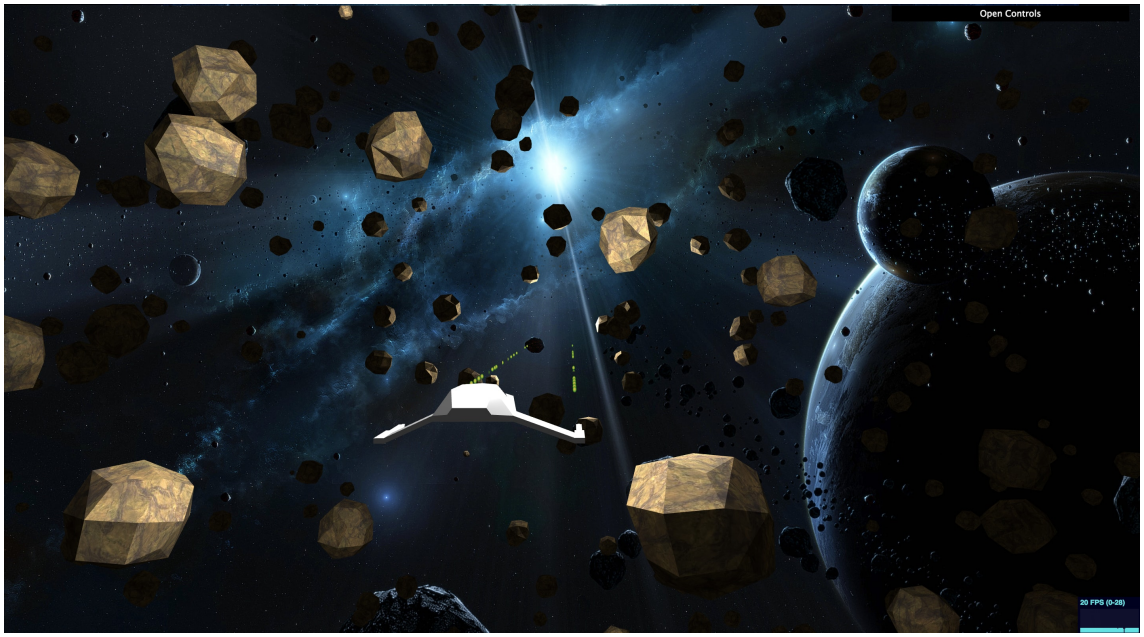

Figure 5: Scene during Game Action

Figure 6 demonstrates the collision functionality. If the spaceship does not dodge properly and gets hit by an asteroid, the system is stopped and an alert message pops up with the message "Game Over".

4 Discussion

Overall, the approach we took was promising and allowed us to create the game system effectively. There are a few possible improvements we could implement as follow-up work to make the game more polished:

- Keyboard Control: Currently the model does not allow the spaceship to shoot bullets and move at the same time, because only one key is registered at a time. In the follow-up work, we would like to parallelize the event handler so that both could be done at the same time.
- Exploding Effect: When the bullets hit the asteroids, they merely disappear without any special effects. We would like to implement some exploding effects and spawn smaller rock particles from the explosion when asteroids get hit by the bullet particles.
- Sound Effect: It would not be a cool game without sound! We could include some space-themed music as background music of our game. There could also be sounds effects during explosion and collision.
- Bounding Box of the Scene: Although the system is only meant to be accessed from the forward direction and a background wall should suffice, we still like to create a bounding box of the scene so that there is space effect in all directions and not just in the forward direction.
- Multiple Lives Allowed: Instead of only allowing one life, we could increase the number of lives of the players to three, so that the user could stay longer in the game.
- Different Items spawning: We could make the game more interesting by adding different objects that the spaceship could catch to boost its chance of survival. For example, we could randomly spawn cannons that the spaceship could use to blow up all the asteroids in sight. Another possible item could be space shields that protects the spaceship in the case of collision.

Figure 6: Game Over Alert Message after Collision

5 Conclusion

During this project, we gained a deeper understanding of the particle system framework. We implemented our own object system to spawn the asteroids. We also learned to interact with the system better in terms of loading meshes and object files.

References

[1] three.js documentation. https://threejs.org/docs/. (Accessed on 05/15/2018).

[2] Free3d website. https://free3d.com/3d-model/low-poly-spaceship-37605.html. (Accessed on 05/15/2018).

[3] Opengameart.org website. https://opengameart.org/node/7874. (Accessed on 05/15/2018).

[4] Donald Carling. Star strike game build. https://donaldcarling.wordpress.com/2016/03/10/star-strike-game-build/. (Accessed on 05/15/2018).