SWO Technical Manual

by Vladislav Savranschi

When developing this game, I wanted to make it as realistic as possible. I implemented realistic gravity, collisions, and projectiles which will interact with their environment similarly to how they would in real life. Of course, I wanted to implement a modular system of ship, utility modules, and weapon selection, however I couldn't quite figure out how to do that without having to make a separate object for each and every different combination of ships, weapons, and modules. I also wanted particle effects (such as explosions, trails, sparks, and smoke) and even had sprites made by my good friend Raphael, things such as Chaff launchers and Smoke dispensers, but the particles ended up too difficult for me to work with and ended up crashing my game. I wanted each module and weapon to be destructable, with sprites that show blown up parts of the ship that could have only been repaired with nanobots or a pick-up.

As such, I decided to just use 2 different ships, with 2 different weapons, in order to create some differences between the 2 players. The missiles launch around every 1 second, and do 15% damage to the enemy with tracking, but can be shot down by the multicannon that does 2 damage per bullet. I think this creates an interesting balance and allows for both players to win. I did want to make ammo and fuel for each ship, but without making power-ups or drops, I decided to avoid doing this. If I had fully implemented by modular systems, and managed to get them to work, I would have needed to implement these things.

On the other hand, I did do some cool things. When the weapon fires, I used trigonometric functions sin() and cos() to create a circle around the ship by translating 360 degree rotation into x and y components, and used those to create an offset of 55 (on a 100x100 sprite) to have the projectiles leave the weapon rather than the center of the ship, no matter what direction the ship is facing. Additionally, the ship sprite is not rotated by an animation, but by a command that allows sprite rotation to be set to the direction. I only set it to my custom variable "dir", since the built-in "direction" actually looks at the velocity vector, which I couldn't use since I wanted the ship to be travelling one way but facing another as it would realistically.

I also decided to create a main menu, with options and controls, to get some experience with creating UI for future games. I can say this game I created is a very good platform for expansion, and I would love to continue developing it until I reach my true mission with it. I can also change it to a single player game in the future, to be up against smart AI, with dockable space stations and different maps that can be travelled to via jump drive. If I manage to create my own "space simulator" by the end of this class, I will be very happy.