Final Game Report for AC31009
by Erik Jeny

Original Plan & Their Changes

Originally I had the idea of making a VR asteroid shooter to appease to an older audience that may have played either Asteroids or Moon Defender or a similar game. I started to practise and learn Unreal Engine as early as the 1st of April and it took until mid-May for me to complete the game “BattleTanks” as it was provided in the course. Many of the files you will see in the source code will have the word “Tank” in it and all of these were made with guidance from the course.

Because it took me so long to get a solid grasp of Unreal I had to scrap a few ideas. One of the first things to go was the custom meshes I was going to build as I didn’t have the time to learn a piece of software that was able to export in .fbx or .obj for Unreal to recognise, as Fusion360 and 123Design all are unable to export in said formats. I decided to use what I was provide din the course as well as a nice asteroid pack* I found on the Unity store.

The second thing that had to go was the AI as I was struggling to make the asteroid themselves behave and function properly. I still have yet to figure out how to make the asteroids themselves rotate, but to keep their trajectory and not rotate their health bar along with them.

The third thing I had planned was adding additional types of weaponry, as well as a secondary weapons that would mostly consist of missiles and heavy weaponry with high reload rates. Again, due to time constraints and the way BattleTanks was built I did not have the time to adapt and rewrite the code to have this ability. Interchangeable types of projectiles also had problems and I was forced to hardcode a single type into the aiming component, breaking the original game.

Another thing I wanted to add was droppables that asteroids/enemies would drop on random that could enable things such as up to 2 additional AI turrets helping you fight off the hordes, better reload rates, more health, weapons upgrades, etc. , but, yet again, because of time constraints and the struggle to get the base game working these were scrapped.

*https://www.assetstore.unity3d.com/en/#!/content/83951

Used Design Patterns

In the code I was able to implement the Factory and Locality patterns, and used them as well as Unreal's own implementation of the Observer pattern.

In the SpawnFactory class you can see I have developed a system where the factory spawns asteroids at a random location (currently 9 are coded in) headed for another random location (again, 9 are coded in), close to the player. The factory calculates the direction and force it spawn the asteroid and upon spawning adds listeners to the factory to hear when an asteroid has either died or scored. In it's header file you can observe that the arrays used to store the start and end locations are stored in TArrays that are Unreal's own implementation of vector arrays (that can also be turned into multiple other types of arrays, like heaps, and stack, making it pretty universal).

Unreal's implementation of the Observer pattern is quite intricate and it took me a while to understand how it works, and Is till have a lot to learn from it to use it, but it seems like quite the useful system to generate custom events and have triggers in Unreal's blueprints and call things accordingly in them. I have yet to find out if I can actually send values along the broadcast as this would make it much easier for inter-class communication without making them dependent.

The Aiming Component class is also a factory, even though it is technically a gun, as it also spawns projectiles at the end of the barrel, much like the Spawn Factory does (almost exactly, in fact).

The locality in Unreal seems to be so efficient it is even overefficient as sometimes the values passed through hard coded getter methods were actually the values that were declared below or above the said value and it would take a restart to fix the editor issue.

Gaming Aspects

In the game in its current form I am using the physics engine and collision detection aspects. In code and the blueprints you will see that Unreal stores collision presets all around which can make it quite hard at times to get desired behaviour because, for example, the actor is set to BlockAll (colliding with everything) whilst the Static mesh assigned to the actor is set to NoCollisions making you scratch your head when they don't collide.

The asteroids in their current state score when they hit an invisible wall behind the player generating a hit event. The projectiles themselves only generate overlapping events with the asteroids as a hit would make them score prematurely. I was also having problems with asteroids colliding with each other and all of the parties generating a score. This would be fixed in a later iteration to make it more realistic.

For the physics engine, everything is handled with how it would be handled in the real world where the factory calculates a vector by subtracting the endLocation vector from the startLocation vector to get the desired location and then applies force to get the asteroid moving when it spawns.

Testing Strategy

In the beginning of the actual Asteroid Gunner game the first thing I wanted to make is the turret to be able to aim and follow the reticule. The problem was as it inherited from the Tank Aiming Component the barrel would aim in a hyperbolic trajectory as how it would be if we weren't in zero gravity. This required me to build a new projectile and override all the gravity presets for it upon creating for it to fly in a straight line. After I had implemented the new projectile into the aiming component it worked as desired. For it to constantly aim and not freeze when it didn't collide with anything I made a huge invisible wall a meter behind the spawn factory and the spawn locations. Not the most elegant, but it works.

Then I wanted the asteroids to move in the direction I desired and after a painful 4-5 hours of work I realized the start and end locations had bad coordinates making the asteroids fly off in weird angles and sometimes even score when they would spawn in/behind the wall behind them.

For playtesting I had Javier, Vivii, and Louis-Marie test out my game. I performed the playtesting as per instruction, with minimal intervention. On the lab computers the game performed subpar, even though on my machine that doesn't have the best components it was capped at 120 FPS, which is Unreal's limit. The mice also had strange sensitivity issues and if I had more time I would add an input sensitivity slider in an options menu in the main menu.

Issues aside, they liked it, but they suggested I put a wider angle for the asteroids to spawn in (I made the startLocations start at further away places), slow the asteroids down (they start off slower, but gain speed with each passing level), make health packs and droppables (had to be scrapped due to time constraints), cap the rotation for the turret camera so you can't see below it (was unable to find a way to do so), to increase speed and angles at which the turret can be fired (modified defaults to greater angles and more degrees per second), and adding a tutorial (added to main menu).

Retrospective

In retrospective, I actually don’t mind my game in the state that it currently is, simply because it allows for a fun arcade sort of game where all you do is shoot asteroids and try to achieve the best score. I probably wouldn’t do anything differently in the development itself, but if I knew the learning would take so long I would’ve started earlier to give myself more time for actually making the game.

I will definitely try to continue working on this game, as I think it has a lot of potential, especially if I manage to get the VR section working properly. I am a bit unsatisfied that I wasn’t, but with the time constraints I understand it was a necessary sacrifice.

Erik Jeny