# Intro

https://github.com/Hokome/bullet-box

https://trello.com/b/J9GpjUrT/bullet-box

* GDD

# Imported Assets and Libraries

There are some things I didn’t make in this game and some things I made before starting this project. There is of course Unity which I didn’t make, but I also didn’t code some of the packages I used for this game. These packages are:

* Cinemachine, for better cameras and transitions
* TextMeshPro, for better text display
* LeanTween, for UI animation

Cinemachine and TextMeshPro are official Unity packages while LeanTween is third-party.

I also imported assets which I frequently use for all my games such as Audio and UI, but these are made by me.

The first commits on the GitHub repository were all imported from the aforementioned sources.

# The barebones prototype

I have a lot of ideas (some of which won’t make it before the end of the assignment, or won’t make it into the game at all), but first I need to build the crude version of the game to see if there is potential for fun. However, the concept of a top shooter is decades old so the prototype will need to be a bit richer than normal.

With that in mind, I made a GDD (Game Design Document) and wrote some ideas, I set up a Trello board and made a list of what I should do first before any play testing is done. First the player movement, which was something I already had done dozens of time so it went pretty easy. I used a physics-based movement with Unity’s Rigidbody to allow for smooth controls as well as potential use of physics later (such as wind). The next thing was shooting, this is also relatively easy, just a little bit more verbose for the input since I have to detect when the player presses and releases the button for firing.

The firing was working just fine, but then I set up the main menu (which took 5 minutes since I already made a template) and it was not working anymore. The player was moving according to the input, but it was not shooting. Although I don’t know why it is like this, the likely culprit is the PlayerInput component I have on my Canvas (which was necessary for my menu) which meant I had two PlayerInput components in my scene at once which broke something (the Input System package is still relatively new so it’s still a bit funky to use at times). The movement input was using messages and worked fine, while the shooting was using with C# events and didn’t work. I couldn’t use messages for the shooting since I need to detect whether the input was pressed or released so I switched the PlayerInput to use C# events and made sure to change the movement input accordingly. After that, the shooting and movement were both working fine, at the expense of concise code since it probably tripled the amount of line needed for this.

Next, I made a simple seeking enemy, added a health bar and created a spawner for the enemies, it gets a random index from the list and instantiates the corresponding enemy, then it waits a cooldown specified by the enemy spawned (stronger enemies have higher spawn cooldowns). It’s not ideal, but it gets the job done for the prototype. I’ll also create a faster enemy to add a little bit of variety. With that done, it’s difficult to tell when your bullet hit an enemy since it doesn’t disappear immediately (each bullet can hit multiple enemies), so we’ll need to add some feedback. I also fixed a bug where the bullets were hitting the same enemy twice. Finally, I made a dashing ability to make the game a little bit more interesting. This is supposed to be a top shooter but also a bullet hell game, so I’ll need to add a shooting enemy before finishing the prototype.

# Filling up the game

The prototype seems promising (lining up your shots so that each bullet hits multiple enemies is quite fun), so I can start filling up the game and make an MVP (Minimum Viable Product). I’ll add as many things as I can to make the game more interesting as well as playtest a little bit to get a little bit of feedback before releasing the alpha version.

I want to add another weapon, but for that I need to implement selecting the weapons, took a bit of setup, but in about 30 min it was finished. I added a weapon that fires faster but can hit less enemies, but it’s just for testing purposes and both weapons are likely to change later on.

There are a few problems I need to fix like the fact that enemies can spawn right on the player, or that the player doesn’t know when it’s invincible after getting hit. To do that I just made the player translucid for a few seconds after getting hit, and I added particles that pre-emptively shows up where an enemy spawns.

Another thing is that if the player gets hit, they have no way of getting back the health they lost. I therefore made a healing item that randomly spawns. To avoid having to write the same code for spawning the heals, I made a base Spawner class that randomly chooses in a list of Spawnables and then instantiates it, as well as an override EnemySpawner class to deal with the particle effect. This will be useful for all of things I plan to add in to the game that require spawning like this. Also, the healing item overrides the PowerUp class, which will also serve later.

I noticed searching for healing items was quite tedious and stressful, so I added a map at the bottom right corner of the screen, indicating where the player, enemies and power ups were. To achieve this, I set up a camera far at the back of the scene, aligned it with the terrain, put a copy of the terrain back there added a script that instantiates a dot as a child to its transform, and finally set the camera to use a render texture which I can then access for the HUD. Now anything that needs to get render on the map can have that script, and I added a color parameter to use different colors depending on the object.

Next, I made an enemy that spawns enemies near it. I modified the spawner so I could spawn enemies from outside the class. However, this enemy is a little bit stronger than other enemies, and currently my spawning algorithm doesn’t take that into account. I came up with a system of budget and cost, where stronger enemies have a higher cost and smaller enemies have little to no cost. Every few seconds the budget increases until a strong enemy spawns and the budget becomes low again. The problem is that if the gap between the cost of a strong enemy and a very strong enemy is too high, it’s very unlikely to reach the necessary budget because of the other enemy depleting the budget too quickly. This will require balance, but I can always create two separate spawners if it’s too much of a hassle.

I added a timer so that the player can have an objective. It saves the timer in a file, but currently there’s no way to look at them. I also added a game over screen that displays how long you survived as well as a tutorial (it’s just a panel with the controls before you start the game).

I made weapon boxes that spawn.

I made strong special weapons that spawn.

Adjustments

# First playtest

I did the first playtesting session with a friend of mine, and there weren’t too many bugs. However, I saw many design problems with my games, so I will now work on improving them, among with adding more things into the game.

The tutorial I initially made was just a screen with a list of input and actions. The mechanics aren’t too complex so that probably was enough information. However, the player doesn’t really remember anything they see on that screen on the first game, because they didn’t even know what the game was about. Therefore, I added a simple tutorial throws you into the game, shows the same input prompts than before but separately and it waits until you have successfully done what was asked before going on to the next one. This isn’t exactly a great tutorial, but I will see if it’s good enough on the next play test.

Another thing I noticed was that the player got hit many times by surprise and didn’t have time to react. To solve that problem, I made the game slower in general. I decreased the player speed, enemy speeds and projectile speeds. I did this because the point of the game isn’t really to have good reaction time, it’s more about strategic use of weapons and dodging a lot of bullets.

Something else that was apparent is that the terrain felt empty. It will probably be even worse now that the game is slower. I could add things to the terrain to make it more interesting, but that would take a lot of time (adding obstacle means adding a pathfinding algorithm to enemies) to implement but also design. So instead, I just made the battlefield a lot smaller, all that space was useless. I can still add things to the terrain later, but it feels a lot less empty now.

The spawning is also very boring, so I added levels that I can design myself, this will ensure a better difficulty curve at the price of novelty. I will come back to the automated spawner when there will be more enemies. I made a few tweaks to the Mother enemy (the one that spawns other enemies), so that its function is clearer. Now it shoots eggs that hatch after a second.

The next thing would be to add more enemies with different shooting patterns, but currently that means I need to make new scripts for these enemies. Instead I standardized shooting patterns into one class. Everything that shoots (including player weapons and the bullet storm) now have a reference to that pattern instead of having to code it manually every time. The information that has nothing to do with the pattern (such as damage) has been transferred to the projectile. This will make creating new weapons and enemies much easier than before, since I basically only need to create projectiles instead.