Tank Simulator 2020 - Zombie Edition Production Diary

Members:

Tom Pennington and my will to live

Roles:

- My will to live: slowly die. That is its only responsibility.
- Tom Pennington: Everything else excluding making the 3D and a large majority of 2D assets.
- Synty Studios: Responsible for all 3D assets excluding the drone and bullet from the class and the conglomeration of Unity cubes that make up the zombies.
- Kevin Macleod: All music.

Levels:

Tutorial

This is the first level that I wanted the player to experience not only to teach them the basic controls, but to teach them additional concepts. The concepts I needed to cover were ambitious to say the least, considering the time constraints I ended up having to work with. Essentially, I wanted to have a tutorial level because those are extremely important. The production of this first level really laid the foundations of what the rest of the code and game would look like through production.

Level 1

I am a big fan of level design and absolutely enjoyed working with a post-apocalyptic vibe, as provided by Synty Studios. Specifically, the pack I used from Synty Studios is called the “Apocalypse pack”. The large packs like this are extremely pricey, but they are completely worth it for level designers, like what I want to learn more about.

The goal with this map is to follow through with the phrase “throwing me into the deep end”. The music is intense, the zombies are numerous, and there is minimal cover to break “line of sight”, and I believe that this serves a good purpose to a topdownshooter. It is good because it takes all the concepts that were taught before in the tutorial level and see if any of it stuck with the player.

With both levels, the game tester that I had run through the game said the light was far too low for this game, so I changed it so that it was lit up a lot more than what I previously had.

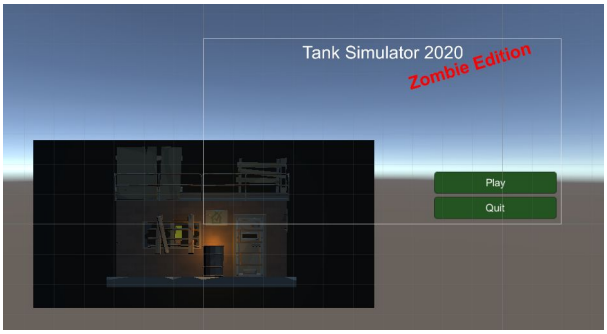
Level 2

Admittedly, I had a lot more fun designing the other level, but I also had fun designing this level. Unfortunately, a lot of the detail put into the map will not be seen by a player running through the level, which is entirely fine. At least the decorative particle effects can be seen. Through my own testing, I had to remove some of the props in the area on the right side of the image above. I had to do this because it was simply far too hard to move around while evading the zombies.

The goal of this map was to have a less of a city level, but also a more difficult level. At this point, I knew that the score system that I had planned to implement to prevent the player moving from area to area would not work as intended and I honestly don't know why the score system and preserved variables simply do not work. In order to work around this, I implemented far denser groups of enemies. The entrance to the market/motel area is covered by a roof where I put some zoomer zombies to not only lead the player to that area, but also to see if their guard is still up at that point. It is also the last level before the victory screen, which meant I was near the end.

The feel of this area is different from the city level, which is why the lighting and music is different.

Main Menu



This is another spot where I wanted to be different from the tutorial. In order to do this, I did not put a static image there. Instead, I designed a small set with a building with a zombie inside it and a barrel fire as a light to provide a little atmosphere. This is where I learned that designing a menu in Unity is not as hard as I thought it would be. At first I wanted it to be a gif, but then I wondered the point of that and why not design an atmospheric main menu. Some other titles in consideration were: Escape the City, Zombie Killing Simulator 2020, Just Another Zombie Game and Just Another Post-Apocalyptic Game.

Victory and Death Screen

I took a similar approach to both the loose condition screen and the victory screen in that I did not want to have a static image. I decided to make an animation for both, thematic for death and victory respectively. I also wanted to include related music, so thanks to Kevin Macleod for coming in handy once again.

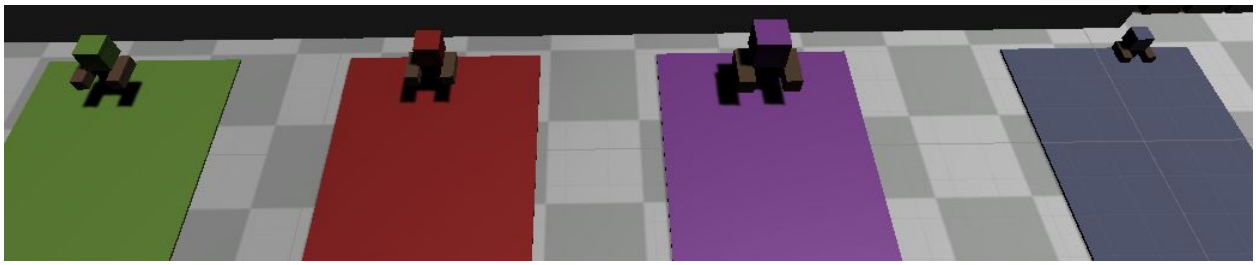
Enemies and Power ups

Concepts Taught in Tutorial Level

From the very beginning of planning, I wanted to go above and beyond with an ambitious game that covers different types of infected, pick-ups, and weapons. I do not

know what the other students did, but I decided to go with the classic zombie shooter aesthetic.

- Types of Infected



Normal Zombie AKA. “Greenie”

The zombie that started it all. This is a generic, not special zombie whose only goal is to walk towards the player to destroy them. This zombie did not cause many problems, but it did establish how the zombies were gonna move. All of the zombies find the player layer and follow the player until they eat your metallic brains.

Something that was funny to me when I was first programming the zombies’ movement was that before I set a limit with a sphere trigger collider (and colliders in general), they would simply sink into the ground and endlessly push the player to their heart’s content. This happened to the point where they pushed the player outside of the physical boundaries of the map, which was pretty silly.

What I did to fix this admittedly funny issue was that I made sure that there was a box collider around each zombie’s prefab which collided with the ground which eliminated the sinking into the ground problem. In order to set a function in place so that the zombies would not just immediately go to the player’s location was to set up a sphere trigger collider. This collider was programmed to trigger a “detectPlayer” bool which made the zombie begin to march after the player.

Spitter Zombie AKA. “gross bastard”

This red zombie is what would be the typical shooter in a typical topdownshooter game that one would find on armor games or addicting games or some other old website like that (jeez, i’ve been on the internet for a long time). The objective of these zombies was to provide a shooter zombie that stood back from the rest of the crowd of zombies that went after you. The movement style is the same, excluding the fact that there is an if statement that detects the range between the player and the zombie. If the zombie got within such a range, they would stop. I programmed it so that they would

shoot whenever they detected the player, not just when they are within that stop moving range.

A problem that arose with this zombie was figuring out how to get the shooting function to work when not on a player. I was not leaning on the tutorial scripts up until the end, which admittedly was not the best decision I've ever made. After figuring that out, I made sure that the zombie spit bullet did not collide with other zombies or themselves and moved on to the next zombie.

Zoomer Zombie AKA. "little bastard"

This smaller zombie is much like greenie in that it is just tasked with running, or sprinting in its case, to the player and kill them. The only thing that is different from greenie is that it is smaller, faster, and has just a touch more health. An exploit that ended up being an interesting mechanic in the game is the way their rigidbody interacted with the player's rigidbody. If the zombie had enough speed, it would latch onto the player and deal damage each time they "collided" to the point where they would appear as if they were on top of the tank. These are a worthy foe.

Boomer Zombie AKA. "Walking nuke"

These were not finished up until a day or so before the project was due. What was difficult about these zombies was the damage being dealt to the player and the surrounding zombies. I wanted the explosions from the grenades, RPG weapon and boomer zombie to not be biased in the way that they dealt damage to each entity. The issue laid in the very brief game testing.

The issue was that before I just had the boomer deal with all the player's health at once if they collided with the player, the game tester just used these purple zombies to clear out hordes of zombies in one go. That is a problem, so after debating about it I decided to just put the damage all the way up to 500, which is what the player's health is.

- Weapons

Pistol

The pistol was the gun that I followed the tutorial for. That caused it to be the basis of the three other weapons. Each weapon was an experiment I wanted to learn how to design. The pistol was the weapon the player started out with and technically had infinite ammo in its entirely realistic 24 bullet magazine. In order to compensate for that, I programmed a 3-second timer in which the player was “reloading” the pistol so that they could not simply shoot forever.

Machine Gun

The machine gun’s goal was to learn how to have a high caliber gun while compensating with the damage. I believe it was successfully implemented, but to be fair it was not all that complicated now in retrospect. With each gun, they have a set amount of ammo. When this ammo runs out, the player automatically switches back to the pistol. My reasoning behind this is that I don’t want the player to upgrade permanently and I wanted to add to the power up collection I had. This weapon does more damage than the pistol but less damage than the shotgun.

Shotgun

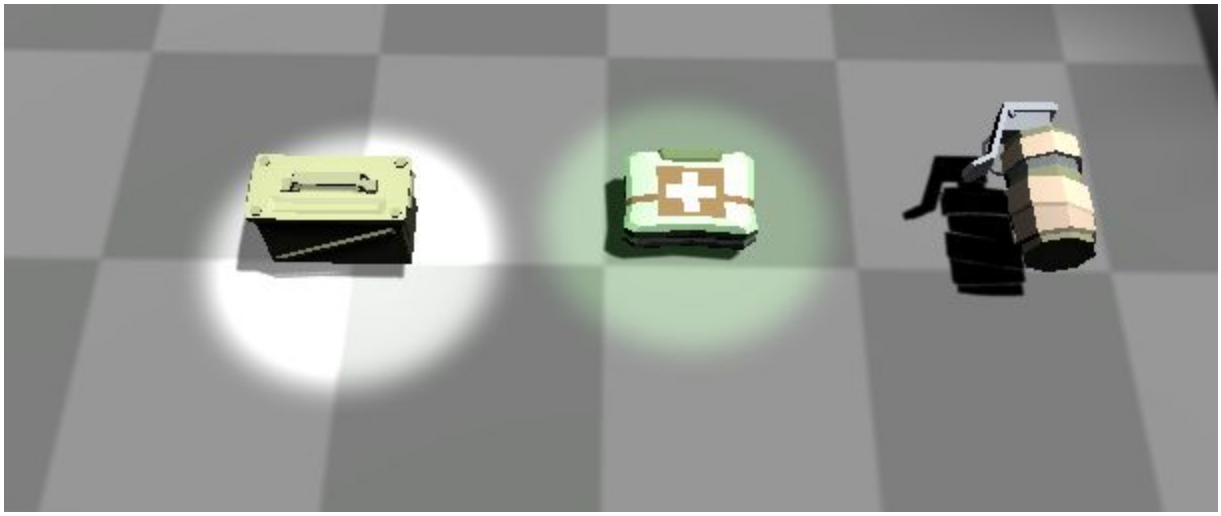
The shotgun’s goal was to have a powerful gun to deal with a cone of enemies. I had a friend of mine help me with implementing the multi-shoot cone function, and I especially like the fact that I got the shotgun to fire in a random assortment of three bullets within the set cone. This weapon does more damage than both the pistol and machine gun, which makes it the most powerful gun that shoots actual bullets.

RPG

The RPG is what happened when I wanted to combine the explosion function with a gun. I made sure it was the most powerful gun/power up but at the same time, I made sure it was hidden away on only the first actual level. At the point when I was working on the RPG, I did not have much trouble combining the explosion script with the explosion particle effect given to us through the topdownshooter assets.

An important note to make about all of these weapon power ups: The player does not need to pick up any of them in order to beat the game.

- Not Weapons



Ammo

Along with the unusable scoring system, the ammo pick up also did not end up working or making the final build. This was the alternative to the forever reloading pistol as mentioned previously. The player would pick this up and reload whichever weapon they had at that current point in time, sort of as an extra use of the weapon power ups. I'm not entirely sure as to why this did not work in the end build.

Health

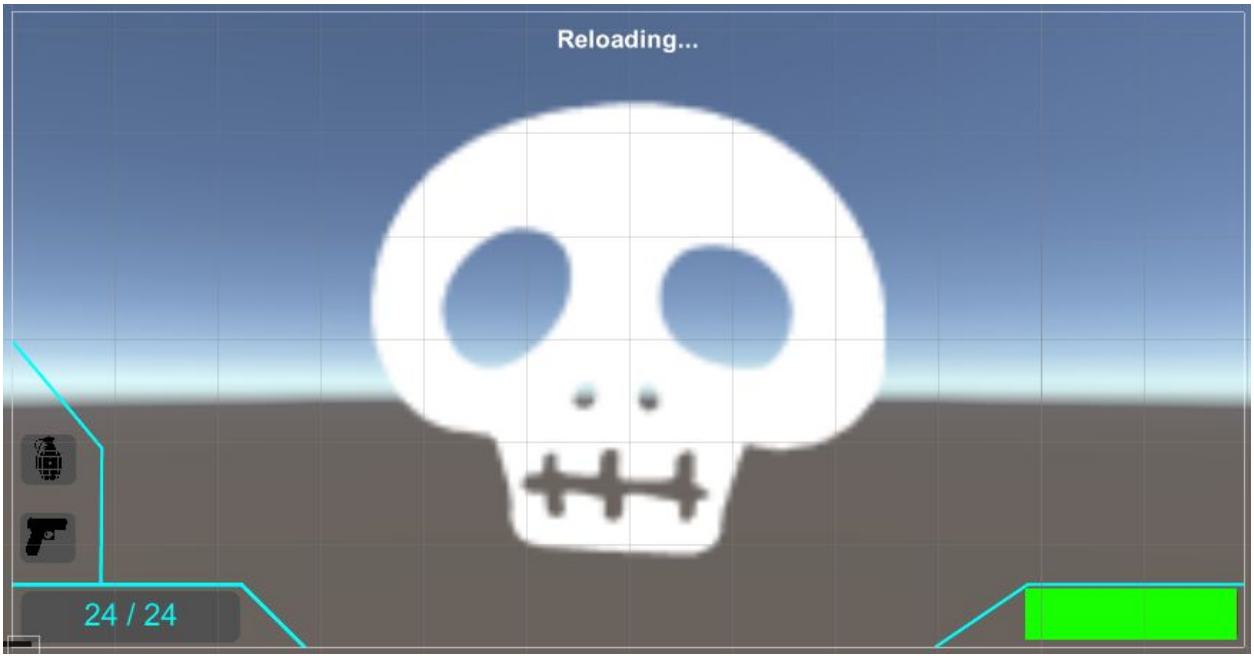
Pretty standard stuff in most topdownshooters, this is the pickup that restores a pitiful amount of health to the health stat for that level. This was also taught in the tutorial if the player had gained any kind of damage by accidentally blowing themselves up with either the RPG or the grenade power ups. It increases the health count by 100, even going above the set amount of 500 points of health.

Grenade

This is an interesting pick up. By interesting, I mean I had made a mistake when I was programming it and decided to run with it in order to have the grenade do more damage to the zombies and minimal to the player. The mistake I'm talking about is that the grenade's explosion sphere trigger (which only appears for around half a second) damages all of the zombies' colliders, which include the unity primitives that make up the prefab. This works similarly to the bullet script, but has a shorter lifetime and

instantiates the explosion effect which then damages the enemies (and player) within the radius.

The User Interface



This was perhaps the very first thing I designed for my topdownshooter project. I created the base lines and gray boxed in Illustrator before bringing it in as an image for the information to be placed upon. Instead of the slider provided by the tutorial, I used a green bar image. In hindsight, I really should I have used the slider method and followed along with the tutorial from the very beginning because that healthbar caused a lot more problems than it ended up worth being.

Another problem that the UI had was on the last day of production. Every time I built the game until I figured it out, the icons, text, and green bar was offset down. What it ended up being, and what I learned about because of this, was an anchor issue. I had not compensated for other resolutions other than 800x600 which was the default. My display and my game tester's display was 1920x1080, and it was causing it to be offset due to an anchor issue. I changed the y value and it ended up functioning as intended.

At the End of It All…

What have I learned after all of this project? Besides learning a lot more about coding in C# and working with Unity, there are some core concepts that I should keep a note of with future projects.

1. Begin the production diary at the beginning and keep track of it through production. Do not leave it for the end
2. Do not take entire weeks from the project for the sake of other papers/classes
3. When offered a teammate, take the opportunity for they could keep you on track and make the project far easier
4. I can make an actually fun to play game

Thank you so much for the experience and for giving the first steps for game creation that I desperately needed to progress anywhere near my goal of being an indie developer. It has been a pleasure to be in the paper and interact in an entirely different country while I was there.

P.S. My code is pretty much programming spaghetti. I apologize for that.