SMOL

Cyberspace Liberation - Computer-Network-Takeover-Operator delivers Net Neutrality
Triankolos Edition

Alpha Release

14.06.2021

Mehmet Dereli

Felix Kosian

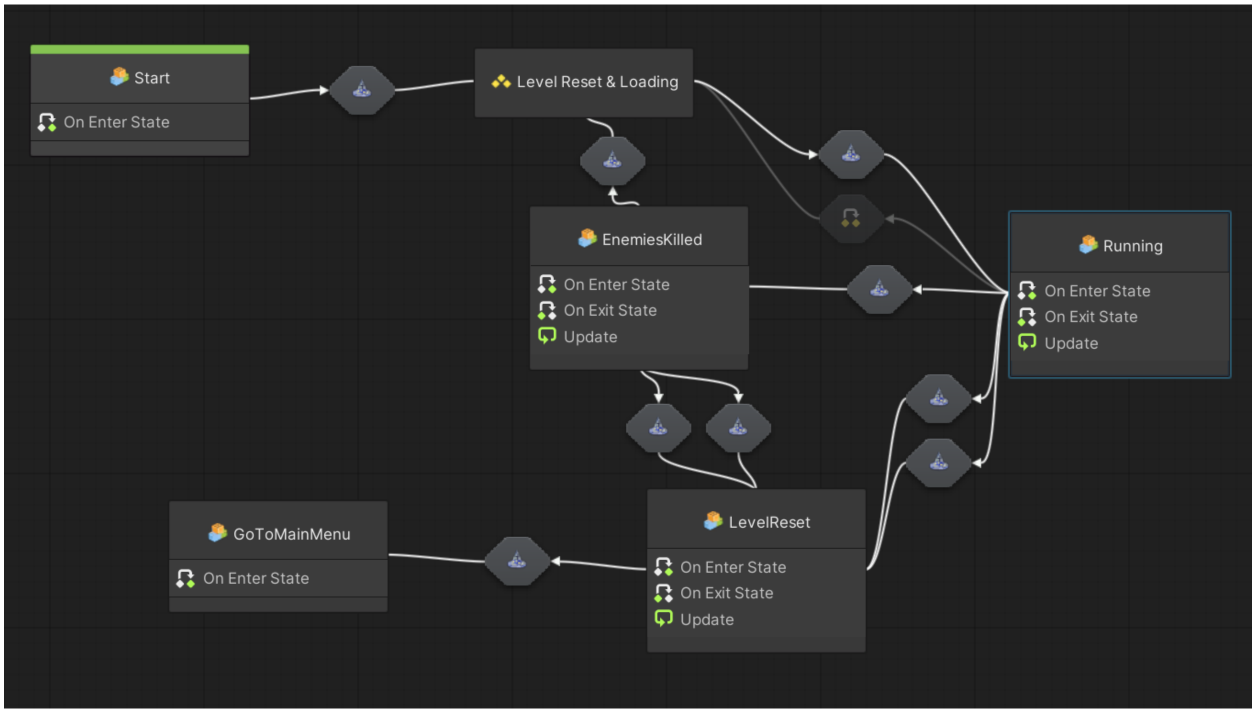
Julius Krüger

Louis Hötzl

Since the interim demo, we came up with a proper title for our game and managed to implement complete a lot of the missing functionality that we had planned for our game. We finally completed the missing features of our functional minimum. Namely, we finished the UI and added a tutorial level. We also added a boss enemy and implemented the rest of the abilities that we had planned. This completes the low target as well. As stated in our desirable target we extended the UI, the procedural level generation always results in playable levels, and we added a trap. We also added an additional enemy, which gets us to a total of two enemies (boss not included) as opposed to our target of three total enemies. We did not have time, however, to integrate sound effects and music, as well as visuals for some of the abilities. In summary, we hit most of our goals for the desirable target. From our high target, we only got started on the progression system, but ultimately we did not have enough time to integrate the feature. In its current state, the game is also rather unbalanced, but we will tackle that during playtesting.

Game Loop

The Game Loop was changed to allow opening the door after a level is cleared, detect the player walking through the door and loading the correct scenes. To allow this more fine tuned Game Loop, we implemented a state machine which decides the correct behaviour.



Level Generation

The procedural Level Generation is one of the more important aspects of the game as it directly influences the player experience. Ever since the Interim Demo, we have made a few changes to the Level Design, including a rescaling of the tile and total level size.

One of the major changes was the thinning of the walls. During the ID (Interims Demo), we felt that having thick walls would make the player feel as if he were in a more enclosed space and strengthen the idea of an enclosed room. During the early stages of development, we discovered, contrary to our first beliefs, that it feels very uncomfortable to have the player move in such a closed room and that the player can vanish behind the thick walls. We initially did not think that it would feel that comfortable, because the player is always around the center of the screen, but it turned out to be a bigger problem. We therefore thinned the walls to counter those two inconveniences.

Another big change was the decision to break out of our grid structure. During our tests, we discovered that the levels did not feel as natural and comfortable to play as we expected. One of the major reasons was the inorganic spread of obstacles. We initially thought that by placing obstacles in an orderly fashion would further strengthen the aspect of order in contrast to the chaotic enemies. After a revision, we decided to move the obstacles into the theme of chaos and spread them more randomly. By doing so, we managed to incorporate both aspects of the theme (Chaos and Order) into the Level Design.

The next big change that we did was to incorporate 2 different level styles, Vapor Wave and Outrun. In the beginning we thought that it would be nice to be able to switch between those 2 styles when the player finished killing all the enemies, to symbolize the swap from Chaos (Outrun) to Order (Vapor Wave). The Outrun style uses a dark color palette, which is associated with Chaos and Evil, while Vapor Wave uses a bright color palette, which is associated with Order and Justice. Our group was rather split about this feature, because while it is a nice feature, a few of us felt that it would be a waste of assets. While fighting the enemy is something that takes time, passing through the door to get to the next level happens rather fast and we wanted to show off both styles as much as possible, to add more variables to the Procedural Level Generation. The main strength of this type of Level Generation is to generate as many diverse levels as possible to keep the player engaged and by adding more color pallets, the effect of diversity is further enhanced. In the end we went with the later option, but we decided to leave the first option open, in case it felt awkward in the final game.

The last change that we did was more of an addition than a change. We decided to add a Tutorial, that consists of a room with different pillars, which allow the player to change his abilities and which provide a short explanation of the abilities. This was done, so that the player can first familiarize himself with the mechanics. We further added a dummy enemy that can be used to practice.

Character

Since the last milestone, we managed to implement the rest of the abilities we had planned for our game. To keep track of channeling times, durations and cooldowns we used coroutines for all of the following abilities.

We implemented a Teleport that allows the player to teleport to any spot on the map. Once the player enables the teleport the time at which the game runs is reduced by (currently) 30% and the player no longer controls the player character but instead controls the camera. The player can then pick any point on the map they which to be moved to and can confirm or abort the teleport with left or right click, respectively. The location the player is currently selecting is communicated to the player with a semi-transparent object and we indicate whether it is a valid teleport by changing the object's color to blue or red. We determine whether a teleport is valid or not by checking whether the location is part of the NavMesh we are using for our AI. Since we used the Unity Standalone Input System we thought we could simply reuse certain controls by having separate action maps for the Teleport and the rest of the character controller. This would allow us to, for example, to use WASD to move the character during most of the gameplay, but also to move the camera and not the camera while the player is in Teleport mode. For some reason, this did not work by enabling and disabling the corresponding action maps, however, so we had to add booleans to check whether the movement or teleport input should currently be used.

The Slow Projectile is similar to the regular projectile. However, it is shot at an angle and over a fixed distance. Once the projectile collides with any obstacle or the floor it stops moving and slows down all enemies that are in a fixed area around the projectile by changing the movement speed of the enemies within its trigger collider. Once an enemy leaves the area its movement speed is restored. The projectile is destroyed a few seconds after it hit the floor and a cooldown is started.

We also implemented a Hack ability that allows the player to take control of one enemy for a short amount of time. We get all enemies within a certain range of the player and then we change the target of the hacked enemy to the enemy closest to it. After a certain time, the enemy is released and targets the player again and the ability starts a cooldown. One restriction of this implementation is that the hacked enemy can only have one target at a time. However, we assume that players won't notice this during gameplay and therefore did not justify the additional workload that would have come with changing our AI system to allow enemies to have multiple targets.

We implemented a Weapon Boost. After a short duration during which the player cannot do anything, the fire rate and weapon damage of the player weapon is increased, while its reload time is decreased for some time. Then the ability starts a cooldown. This ability is mostly tweaking values and starting a couple of coroutines.

The Revive on Death ability is pretty straightforward. When the player character's health is reduced to zero we simply check whether this ability is enabled and whether it has charges left. If yes, the player character's health is set to a certain amount and we subtract one charge. Otherwise, the player character dies and the game is over. The ability allows for a more defensive playstyle for players who want to take it safe.

We also implemented a feature that allows the player to switch between some abilities in-between levels. The player can now switch between Teleport and Dash, Hack and the Slow Projectile, and between a Weapon Boost and Reviving on Death.

We also implemented the character model as well as animations. We expected that the animations would take a lot of work due to past experiences. For the most part, the art integration was pretty straightforward. One minor hiccup that we had to deal with was that the shooting animation did not play as fast as the player could shoot without looking ridiculous. Therefore, we split the shoot animation into three parts and looped the middle part for as long as the time since the last shot was not above a certain threshold. With this fix, the animations work reasonably well.

Additionally, we changed our camera behavior to be more dynamic. The camera is now moved toward the area into which the player is aiming rather than the camera being always centered on the player character. This turned out to be trickier than expected. The initial idea was to have the camera follow an invisible game object that follows the player's mouse pointer but only up to a fixed distance from the player. However, our camera is slightly tilted, and therefore objects at the bottom of the screen are closer to the camera's near plane than objects at the top. This resulted in the position of the game object being off and some weird camera behavior. Ultimately, we did not figure out the necessary vector math for the calculations, but we shot a ray from the mouse position in screen space into the world and then set the game object's position accordingly.

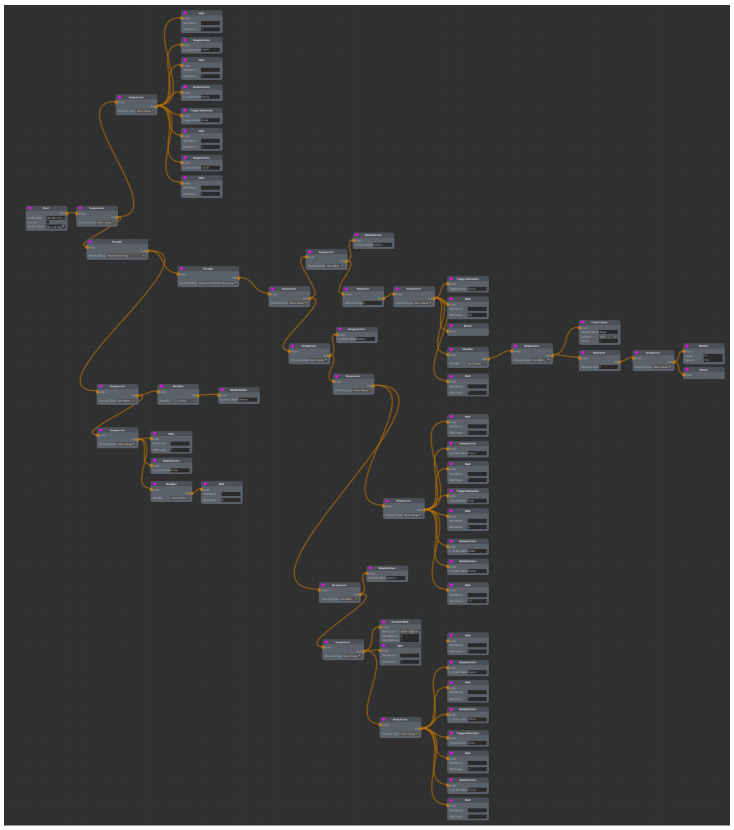
We also wrote a bunch of code for a progression system, but ultimately we did not have time to properly integrate it into the game or set up the necessary UI elements that are required so the player can actually interact with it.

AI

The AiDirector now manages difficulty and the special boss level. Each enemy can have multiple stages with difficulty cost tokens. Each level has a predefined amount of difficulty tokens. With two settings between 0 and 1 giving the mean, we can define the probability (following a normal distribution) of how many vs better stage and easy vs hard enemies will be spawned. Each spawned enemy reduces the amount of difficulty tokens left for this

level.
For the boss level, the spawn behaviour gets overridden and dynamic spawning enemies are supported.

Improvements to Behaviour Tree includes many more nodes (conditional checks, animation trigger, gameobject controls, ...) to allow much more complex behaviours.



User Interface

After a major rework, the UI now covers In-Game, Ability-Select and Menu. The Menu UI allows switching to different scenes (Game, Main Menu, Tutorial, Credits) and is partially available in all Scenes. The In-Game UI includes health bar, enabled and available abilities and ammunition count.

Design Revisions

We made the following design revisions:

- Thinning the Walls (see Section Level Generation)
- Rescaling the Tiles
- Breaking out of the grid structure by placing the obstacles more chaotically (see Section Level Generation)