Invader of the Void Coding Summary

Invader of the Void is a 2D puzzle platformer created in Unity. The objective of the game is to complete a gauntlet of puzzles, culminating in an epic boss battle. You play as the invader, who starts off in space, before entering a portal and venturing into the void. The project was created in Unity, an engine designed for designing and building video games. In total there are 5 playable levels. Each level comes with its own unique gimmick that the user must figure out in order to advance.

The Space room was created to set the mood of the game. To explore the levels while being patient. Controlling the spaceman can be a little difficult at times. The user can only control the spaceman being jumping and moving side to side. This room has very little gravity for the player, and that is the only way to decline. Being patient while the spaceman declines is very important. The whole room is surrounded in spikes, trampolines, and crystal gems. You must collect the three crystals to activate the portal. Once the portal is activated it takes you to earth where you begin the castle quest. The testing of the room was fairly simple. The script to unlock the door/portal tested with regression and different items. Ensure that not any item that is picked up can activate the portal, and only the three specific gems will work. Testing of the respawn of the player was also made to ensure that no matter what spike made contact the player would be respawned.

The lever room was the first level we created, but it's the second level chronologically. This room is simple, only requiring the user to pull a lever to open a chest containing a key. The user will then have to pull the lever again in order to raise a ladder, and jump onto a platform with the exit. 2 enemy bats are also present in the level, one with a scripted patrol path, and another one that will actively chase the player. The level's win condition is reaching the door with the key. Scripts written for this room include character movement, lever interaction, key interaction, door interaction, and the AI for the enemy bats. When testing this level, there were a few things we wanted to make sure ran smoothly. First and foremost, the collision on the walls, floors, walls, and platforms. 2nd, the lever interacted properly with the chest, the key in the chest, and the ladder. 3rd, the bats properly move, and respawn the user at the start if they hit them. Despite being the simplest in design, it is here where we learned the most about the ins and outs of using Unity. It is also where we learned what is and isn't possible in this engine. Using what we learned in this room, we were able to build bigger and more bombastic levels for the 2nd release.

The Minecraft/Terraria Inspired room uses the concept of building as seen in these inspired by games. In this level, the player has the ability to activate building mode and begin creating blocks to their advantage. The objective is for the player to build their way to the key in order to unlock the door and move on to the next level. They have to be careful with the ghost and water openings as the player will get respawned if in contact with these objects. The functionality of building mode is activated by pressing the F key and then you may begin building by left-clicking on your mouse or mouse pad and right-clicking to destroy a block previously created by the player. The implementation of blocks is done by placing a block sprite onto a blank tile map and if there isn't an object currently at that position then you are free to place it there. It will not allow you to place a block on top of another object and the block sprite

will turn red indicating that you cannot place a sprite there. It was tested by making sure block mode could be turned on and off and check if blocks could be created when block mode was off. Also making sure blocks couldn't be created on top of each other was tested through commands. The respawn points were also tested by commands for the player to come in contact with objects of danger and making sure the player was moved to the initial location of the start of the game.

The Darkness room is built around an extremely simple, but very versatile object in Unity, sprite masks. A sprite mask is as the name implies, a thing that goes over sprites. The darkness room utilized sprite masks to create the illusion that the player is trapped inside a pitch black room. Torches in the room each had a sprite mask attached to them. That would essentially create "pockets" of visibility wherever they were. The objective of this room, similar to the lever room, was to find the key, and use it to escape through the door. One of the differences here is that the player starts next to the exit door, but they don't have the key. The path to the key is pretty direct, the only obstacle is a wall that can be moved when the player moves a torch from one sconce to the correct one adjacent to the wall. From there, it's a straight shot to the key, which is scripted to make the room pitch black dark when it's grabbed by the player. From here, the player must find a way back to the start with the key, all while in complete pitch black darkness. The player can use torches to light their way, but torches also act as "switches", so there will be moments where the player will have to rely on level knowledge alone to navigate. There are multiple ways to beat the level, one cheeky way involves bypassing half the level, and simply gunning it back to the start in total darkness when you have the key. Regardless of how the player does it, once they reach the door while holding the key, that is the end of the level. The test cases for this level were mostly found in the process of its design. One glitch that appeared early was the "infinite" torch glitch where the player would interact with an empty torch sconce, and magically find a torch in their hands. So one of the first test cases I employed was that torches were atomic. The player could not place double torches on one sconce, or grab a torch from a sconce that had no torches. Then, the next test case employed a lot of the ones from the lever room, ie well functioning player movement, proper collision in the walls and platforms, level transitioning when reaching the door, etc. The last test case called for the room to properly blackout when the player grabbed the key. Overall this was a very fun level to design and code.

The Boss Battle room created an augmented bat that is bigger and chases the player with the same script that chases the player around the scene. The player has to defeat the boss bat with bullets from its gun. The bullet has its own script that moves based on where the player is facing. Bullets shoot out of the gun and can hit different objects. The bullets have a cooldown time that the user can specify, and is checked through not being able to shoot bullets during cooldown. If the bullet hits the bat, then its health will decrease. When the bat gets to half health, the bat becomes faster and shoots lasers. Both the player and the enemy have a health bar that updates based on different conditions such as if the player gets hit by the bats hitbox or laser or whether the bat gets hit with the bullets. The enemy's second phase changes the speed of the bat, the sprites color and the UI health bar color. Player invincibility is achieved by ignoring layers that the bat and the player are in for a period of time and changing the alpha value of the player (which makes it slightly transparent).