GIMM 110 Game Analysis

By JC Ho

When I was first thinking about my game, I wanted to use the main element of the base stub. Collecting the coins and the coins move from platform to platform. But I also wanted it to be much more than that. I also wanted it to be a platformer, but with aspects of a shooter, so being able to shoot. I took both stubs and took the code from both of them to use in my project. I have the movement script which is now highly modified, and the shooting script which I also highly modified to fit with my game. When thinking about my game, the best thought I had was I of a mini-game type game inspired from the game WarioWare. I also wanted there to be a great sense of randomness in my game, similar to WarioWare. In the first mini game “Collect”, it randomly selects from a pool of platform choices, and you collect a certain amount of coins based on the difficulty. I used a ton of random generators in my code. I also have my code organized into an Enemy folder for the zombies in the game, a level management folder for anything that involves levels in the game, a player folder that involves all things the player can do, and a UI folder which controls the UI of the game such as scoring. I took a lot of code such as scoring and the camera follow scripts from the book and used it to my advantage, but I also wrote a lot of my own scripts with many methods and variables. The hardest part of the coding is being able to transfer info from one object or one script to the other. However, I have gotten a lot better at that after working on this game. I will now go into detail on my code. The folder named “Enemies” has “Zombie” which is what controls the zombie prefabs. It has a lot of different blocks of code in there. It has collision code, enumeration, and methods for how the zombie acts. The folder also contains “Zombie Spawner” which moves an empty game object around and performs the “Spawn Zombie” method which spawns a zombie prefab in that location. The last in the folder is “Zombie Walls” which only moves a set of invisible walls down surrounding the main area after about 7 seconds. The zombies collide with this wall, and it tells them to turn around. These all apply to the second mini game. The second mini game is “Shoot” where you shoot a randomly generated amount of zombies. They also have a randomly generated speed. It first randomly selects what type of zombie it will be. There is a fast and a slow zombie. Then it will randomly generate its speed. The zombies have a ray cast in front of them, so if the player is close to them, they will speed up to a max speed. Both the slow and fast zombies have different levels of max speed. If the zombies hit the edge of the screen, they will turn around and continue walking. They are the only thing truly animated. When I first started the project, I wanted to make the character fully animated. However, this was a little out of my realm for now, and I will continue to work on it in the future. It was hard to make it in Adobe Animate when I have little experience with the software so far. I realized it was taking up more time than actually coding and making the game, so I scrapped it to fully focus on creating and coding the game. The next folder is the “Level Management” folder. It contains “Diamond”, which is a script that has a simple collision block that activates when the player collides with the diamond. The second is “Diamond Spawner”. It is the same thing as “Zombie Spawner” except for choosing where the diamond prefab will be spawned in. These scripts are used in the third mini game. The third mini game is “Find” where you navigate a big maze filled with spikes to find a diamond. The diamond has 8 places that it can spawn, some easier, some harder. Each level has a little animation play with text that shows what level it is when the new level starts. These animations are controlled by the “Level Loader” script. This script holds the methods that a couple other scripts call in order to change the scene and play the animations. Each mini game is its own scene, which actually was a bit of a pain to make with the UI. To keep the UI consistent on each minigame, I had to use the Keep UI script from the book. I also had to make the camera a prefab so it would keep the same settings and stay in the same place. A few other scripts in the folder are the “Main Menu” and “Play Button” scripts. The main menu script controls the main menu, and the play button script just starts the game after pressing the play button. The last script in the folder is the “Platform Manager” script which controls the platform locations in the “Collect” level. The next folder is the “Player” folder. It contains scripts such as “Shoot” and “Bullet” which were stub files from the shooter stub. However, these were modified so that the bullet prefabs were able to spawn according to the direction the player was facing. I also made 5 different bullet prefabs with different gravities and speed so that the shotgun pellets were more unpredictable. “Coin Collect” is from the platformer stub. I was able to modify this code so that the coin never transforms to the same platform (except for on random platform selection 5 2 platforms are combined). It seemed hard at first, but the code is really easy when you look at it. It sets the value of the current program and then checks to see if the new value is the same as the current value. If it is, the program runs again to make a new value that isn’t the same. “Movement2D” was also modified. I added a “surprise” and also a Boolean for “IsFacingLeft” and manually changing the rotation of the character instead of flipping the x of the sprite like the stub did. The last folder is the ”UI” folder which contains UI related scripts. “Level Counter”, “Player Health”, and “Score Counter” are all copies of each other that just increment a value represented by text on screen whenever something happens. “Show Hide” is a script that “shows” and “hides” game objects on the main menu by enabling and disabling them. This happens when the credits and the story buttons are clicked. And that is all the code in this game. All 3 of these mini games randomly select and get harder the more levels you beat. For example, the coin level requires more coins for you to collect for the level to be beaten, and the zombie levels spawns in more zombies. The shotgun does a good job of keeping them at bay, however. In the game you play as a stick figure, a character I made up named S4M. He is a clone and there are many of him. They all get the chance to wield the legendary Shell, a creature that takes the form of a shotgun with infinite ammo. I actually had a different plan for these characters, and the game would be more in the roguelike category, however, I realized that there wasn’t enough time for that project and shoved it aside for what I made. The title screen of my game tells the story of these characters and the story of the game. It also has a credits page that lists all of the assets I used from the asset store that were free to use. They were a big help. Some notable ones were the sound effects which I used for the coin, the shotgun, the level transition, and the play button. Another is the coin animation for the coin in the collect levels. The zombie for the shoot levels, and the diamond for the find levels. I was sick for a full week, so I lost a bit of time, but given the time limit, I am proud with what I was able to produce. I was able to learn a lot and am starting to know things about Unity and C# off the top of my head, so I am able to help other people. One last fun fact about the game is, if you press “O” in the game, you will get a special surprise. It was a great experience making this game and I had a lot of fun. I’m excited to learn and create more as the years go on. The thought of making games with a team and having a longer time than the time presented to slow down and make sure everything is perfect and crafted the way it was first envisioned sounds amazing. It is definitely what I want in my future, and this project solidified that feeling.