Programming Assignment #3

Dylan Stewart

Computer Engineering Major

ECE 407

Site Location: Virginia Beach, VA

Due by: 4/7/21 at 12pm (noon) EST

**Introduction**

This assignment was to create our own 2 Dimensional version of Super Mario Bros inside of Unity. We were given base code from a Platformer game to go off of, but we had to tweak each script to make it work with the logic of Super Mario Bros. To at least have a basic game we were given the minimum requirements and some discussion points:

1. World 1-1: The opening section of Super Mario Bros. was specifically designed in such a way that players would be forced to explore the mechanics of the game in order to be able to advance. Rather than confront the newly oriented player with obstacles, the first level of Super Mario Bros. lays down the variety of in-game hazards by means of repetition, iteration, and escalation [2]. The first level of your game must bear strong similarities to World 1-1 in the original Super Mario Bros. Students in enrolled in MSIM 508/ECE 507 must develop one additional level with similar complexity, but sufficient difference. Replace the ball texture with a ball image created by yourself or downloaded from Internet. You may need to convert the image format from .gif to .png, since .gif is not supported by XNA.
2. Mario, who kills a Goomba by jumping on top of it but is killed by the Goomba if otherwise the contact between them is in a different way, e.g., horizontally.
3. Goomba, which is a type of enemy that does not have any weapons and runs back and forth between obstacles. Goomba must be generated randomly at different locations and times.
4. Two types of coins. One type of coins is hidden in the bricks, while the other floating in the sky (still). Collecting two types of coins results in 200 points and 100 points each, respectively. A new life is awarded when 2000 points are collected.
5. Proper UI with number of points, number of lives, and remaining time.
6. Proper background music and sound effects.
7. Generate animations and particle systems to model various movements and visual effects in your game.
8. Discussion: The background contains a number of different sprites and a technique called background parallax scrolling is used by the game. Explain the concepts of parallax and layers (and Order in Layer) and why they are useful for generating more realistic graphics.
9. Discussion: The game uses two types of techniques for sprite animations. Explain these two types of sprite animations with examples from the game.
10. Discussion: For each script, list its references and explain its main functionalities.
11. Discussion: Discuss different types of colliders used in the foreground (foregrounds) and different types of physics materials

These tasks were all handled and carried out efficiently and sometimes even exceeding the basic expectation of the tasks at hand. All Discussion are addressed at the bottom of the Results section.

**Game Design and Implementation**

For Doxygen, summaries for each task have been added. My .chm file is located inside Super Mario Bros and is called Project\_3.chm, my Doxygen configuration file is also located inside Super Mario Bros and is called Doxygen Config. The Doxygen documentation results are inside Super Mario Bros->Documentation.

For my game, I only went with 1 Scene and tried to add as many features as I could to it. The scene is a close replica of the real game, it contains all the bushes, clouds, pipes and open floors that the original Super Mario Bros had. The only difference is that I had to add floating coins, which I don’t believe are in the original World 1-1, per the minimum requirements listed above (#4).

The player plays as Mario and he can jump and stomp on Goomba (the enemy) or hit question mark blocks to get an item of either a coin, flower or mushroom. The player can be controlled using the arrow keys or the wasd keys and the game can be paused by pressing the p or m button. The Goomba is a character that just walks back and forth, if the player hits him on the head then the Goomba disappears and the player gains 100 points, but if the player touches any other side of Goomba then the player loses a life.

Scripts:

* Background.cs: controls the background music and pausing of the game
* CameraFollow.cs: controls the camera and follows the player throughout the map, taken from the platformer game.
* CharacterController2D.cs: controls the character and allows it to jump up and move side to side using velocity, came from <https://github.com/Brackeys/2D-Character-Controller> because I thought my PlayerControl.cs wasn’t working correctly.
* DataSave.cs: Saves the CoinCount and Number of Lives because at each death, the PlayerMovement script would reset and reset all the variables.
* Enemy.cs: used to automate the Goombas and make them flip if they run into one another or a wall. Also checks if it hit the player or if the player jumped on it, referenced from the Platformer game.
* Item.cs: This is linked to the spawned items and plays an audio clip for some if they are triggered and increases the players score.
* killBox\_Action.cs: Used for the Goombas to check if it hit a player and if so call the function HitPlayer().
* PlayerMovement.cs: The Main code. Here it checks if the user inputs a direction for the player and then calls the CharacterController2D function Move() to move the player, it also updates the score, coin count, time and lives left. It will restart the game if the player is killed and end the game if the lives run out.
* Post\_action.cs: This is used for the end post and is supposed to play an animation when the player comes in contact with it.
* QMark.cs: This is the control for the Question Mark blocks, it spawns an item if the Question mark is hit from underneath and then adjusts the score and coin count if the item was a coin. Referenced from Spawner.cs in platformer game.

User Controls:

-Arrow Keys or WASD to control the players Left, Right and Jump actions.

-“P” or “M” keys to pause the game and music.

**Results**

For this Assignment I got started by looking over all the code for the platformer and figured out what each function’s purpose was and how it was used in this program to control the outcomes. I then started with the design of the Super Mario Game. I knew I needed all the blocks and items so I grabbed the sprite sheets from the websites provided to us and figured out how to separate a sprite sheet into multiple sprites. Once I had that figured out, I created a prefab for the blocks, bricks and question marks and began adding them to the scene and gave them colliders to make sure the player won’t go right through them. After figuring out all the basics of game objects, I did my best to recreate the look of the original Super Mario Bros 2D and this completed the first task:



Figure 1: Personal replica of World 1-1

The next task was to get the player, aka Mario, working and being able to jump, run and collide with objects. It was difficult to get him running and took some research. Originally, I just had a box collider around Mario and every time he ran, he would tip over and I thought this was a problem with the code from the platformer so I did some research and found another Character controller created by a user name Brackeys, the link is: <https://github.com/Brackeys/2D-Character-Controller>, but this one was doing the same thing and after looking back at the original platformer and peoples creations with this script, I saw that there was a circle collider being used and that made it possible to go over small bumps and not flip. So after implement a small circle collider and a few other components, along with a lot of scripting, I was able to get him running and jumping with the proper animations:



Figure 2: Mario Animation

After getting the main character up and running, I had to create multiple enemies to make the game a little bit more difficult. I created a prefabs for Goombas and gave it a basic circle collider to make sure it doesn’t fall through the ground. After that, I added an edge collider that fit the top of Goomba’s head and had to add a box collider so that when the player jumps on top of the Goomba, it won’t fall through and hit the circle collider which would kill the player.

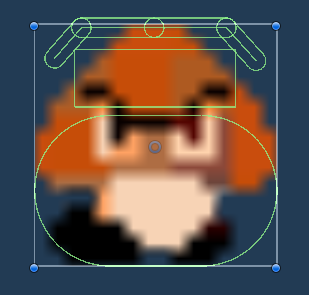


Figure 3: Goomba’s Colliders/Prefab

The next task was to add two types of coins to the game, one that came from the question marks and one that was floating in the air for the player to get and these would give 100 points and 200 points respectively. To accomplish this task, I used the sprites from the sprite sheet of items and created an animation with would go through the different sprites, in this case there were 3 sprites to animate through. The coins would loop through this animation until it collided with the player and then it would add the correct amount of points to the players score and then delete itself. The coins inside the Question Mark blocks would spawn randomly when hit and then would be automatically added to the users score and coin count. The coins floating in air would remain in place, looping the animation until the player comes by and collides with it/collects it.



Figure 4: Coin Animation

Next, I had to add a proper UI that contained the players Score, Coin count, Lives left, Scene name and time left. This was fairly simple to add, I just created a canvas on the Main Camera object that follows the player and then added multiple text blocks for the different UI that needed to be added and then I added code that would update the score, time, coin count and lives left as needed.



Figure 5: Proper UI

And I had some function calls to update each text, for example:

public void UpdateScore()

{

int Sc\_Len = ScoreCount.ToString().Length;

int z\_len = 6 - Sc\_Len;

string score = "";

for (int i = 0; i < z\_len; i++)

{

score += "0";

}

score += ScoreCount;

m\_points.text = score;

Previous\_Score = ScoreCount;

}

public void UpdateTime()

{

m\_time.text = "Time\n" + time;

}

The next task, I had to add some background music and sound effects for the user to listen to while playing the game. This was fairly simple, I just had to add a new Game Object to the Level and give it an Audio Source and then drag and dropped the Super Mario Bros Melody and set it to loop. I created a script for the background audio so that any other script can stop and play the music, this is also where the pausing of the game happens because it was most convenient to have it pause the music and simply set the TimeScale to 0. The sound effects were added to their appropriate game objects; the player controlled the Jump, Die, GameOver, Flag Post, and Stage Clear sound effects and would call them on the correct triggers, the Question Mark controlled the coin sound and the power up appear sound which would spawn when the Question mark was hit and the item appeared, the Goomba’s controlled the Bump, which would happen when the player collided with the Goomba anywhere but on top, and the Stomp Sound, which would happen when the player jumped on the Goomba’s head. I had the last minute Idea that I should add all the sound effects to the background audio and then call them from wherever, but everything was working fine and I didn’t want to risk messing something up before the deadline.

The final task, was to generate proper animations, which I have shown previously in this section that I have done and the recorded video will show the proper animations as well.

**For the Discussions in Tasks 8 - 11:**

1. – The Background Parallax would move the clouds from side to side and make it feel more realistic to the user and it used Order in Layer to determine which was on top of the other because it wouldn’t look real if the clouds were behind the background.
2. – The enemy spaceship wobbles side to side by animation using the body rotation and the enemy slime guy wiggles his tale back and forth by using Scale
3. – Scripts:
   1. BackgroundParallax.cs: Linked to the backgrounds prefab and its main functionality is moving the clouds in the background to make it more realistic.
   2. BackgroundPropSpawner.cs: Linked to the busCreator, cabCreator, and swanCreator and its main functionality is where to spawn the props, how fast they should be and when they should be destroyed.
   3. Bomb.cs: Linked to bombs that are laid by user and the main functionality is to land and wait until it’s time to explode or explode on collision with enemy.
   4. BombPickup.cs: Linked to bombs and its main functionality is to check if collided with player and if so, add a bomb to the player’s inventory.
   5. CameraFollow.cs: Linked to the main camera and it follows the player around keeping the camera pointed at the player.
   6. Destroyer.cs: Linked to the explosionCircle and rocketExplosion prefabs and its main functionality is to destroy any game object it hits.
   7. Enemy.cs: Linked to the Enemies and it controls their movements from left to right and adjusts their health if the player hurts them and plays a death animation if they die.
   8. FollowPlayer.cs: Linked to the Health Display and it follows the player around so that the player knows their health.
   9. Gun.cs: Linked to the Hero’s Gun child and it checks if the player presses the fire button and if so then it play and animation and launches the rocket.
   10. HealthPickup.cs: Looks like it’s supposed to be Linked to the Health Crates, but for some reason mine isn’t linked. Its main functionality is to check if the player collides with it and if so, give the player health.
   11. LayBombs.cs: Linked to Player and checks if the player presses the fire2 button and if so it drops a bomb.
   12. Pauser.cs: used to pause the game
   13. PickupSpawner.cs: Linked to pickupManager and it spawns in the bomb and health crates in random locations above the player.
   14. PlayerControl.cs: Linked to the Player and this controls the player’s movements left and right and jumping.
   15. PlayerHealth.cs: Linked to the Player and this controls the players health and if they’re hit it decreases
   16. Remover.cs: linked to the killTrigger and it destroys the player object and restarts the game.
   17. Rocket.cs: Linked to the rockets and controls what happens on impact and whether it should destroy what it collided with or not.
   18. Score.cs: Linked to the Score UI and it updates the score constantly.
   19. ScoreShadow.cs: Linked to the Score Shadow UI and adds a shadows that constantly updates.
   20. SetParticleSortingLayer.cs: Linked to different particle effects and sets the sorting layer.
   21. Spawner.cs: Spawns random instances of enemies based on a given list.
4. – The platform bridge, top and tower all use box colliders which act as a box that no object and go through, the Platform UFO has a polygon collider so that the creator could make the collider the exact shape of the UFO sprite. The TowerFull has a Platform End physics material that had 0 friction and 0 bounciness so the player would just slide off of it and not get stuck.

**Conclusion**

To conclude this project, I accomplished so much in this project. I learned a lot about creating 2D games from almost nothing. I learned how to create and use sprite sheets and how to make a character move and jump and crash into things. I ran into a lot of issues that involved extensive research though, like some points my character would only show the Jump animation for ½ a second then go right back to idle animation and this took me almost a full day to come to a resolution that I needed to change it from FixedUpdate to just Update. There were a lot more tedious errors like this that I ran into but I think I smoothed them out good enough for a decent gaming experience. The only part that I was unable to fully get was the final animation when Mario gets to the post, given more time I could probably figure it out, but this was a huge struggle for me. With all the struggles came great knowledge and appreciation for game developers and an even greater appreciation for google and online forums. A couple things I could have done better for this assignment was, to consolidate the audio clips to just the backgroundAudio object and do a call to that when needed, this would have saved a lot of headache in having multiple audio source and having to add the clips to each script. If I had more time I would like to have finished that final exit animation and add more enemies into the game. Overall, I learned a lot about game development in this project and I am excited to see what Project 4 has in store for me.