No changes were needed to meet the requirements of the original challenge.

To meet the requirements for the visual changes, I created eight variables across two scripts – four for the background and four for the starfield. Both required a public bool, a public float, and two public vector3 variables. For the starfield, I set two transform.positions, one in start and one inside an if bool statement in Update(), then moved the starfield from one vector3 to another using a public time variable for speed. The background scroller was similar, but instead of position, I scaled from one size to another. I then put calls for these two scripts in the GameController() so they would trigger after the game was won.

To meet the requirements for the audio changes, I added two more AudioClip variables, victory and defeat music, respectively. For the defeat track, I stopped the background music, set that music source to be the defeat music, and played that music. For the victory music, I set it inside a variable for when the score was between 100 and 101 then swapped the music tracks. I did this to prevent the music from restarting each time the player got a score above 100.

For the first gameplay change, I added an invincibility powerup. I created three variables in GameController(), a public bool and two private floats. The first private float was the duration of the powerup, the second calculated the remaining time that the player was invincible. In Update() I created an If function that would start the timer, and when the remainingInvincibleTime hit 0.0f, it would set the invincibility to false. I then created a public void bool that set the player to the invincible status, if true it would turn the remainingInvincibleTime to the duration of the status, and if false it would set remainingInvincibleTime to zero. I then created a DestroyByPowerUp script, which was similar to the DestroybyContact script, but set the player to invincible if they hit the object. I then created a simple capsule 3d object, applied a rigidbody, the Mover and RandomRotator scripts, an explosion effect, and an audio source on the explosion effect. I then added this as a prefab, and moved that prefab to the hazards list for the GameController().

For the second gameplay change, I added a hard mode toggle. I first changed the speed from public to private, and created a Vector3, which became a transform.forward in the Start() function. I then added the two keycodes as GetKeyDown, one for a normal mode and one for hard, and individually adjusted their rigidbody velocities by multiplying the transform.forward by speed, which were adjusted inside each GetKeyDown If function.