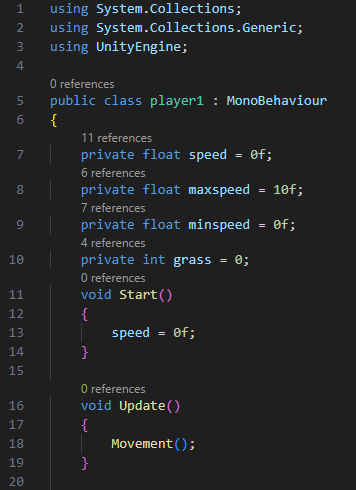
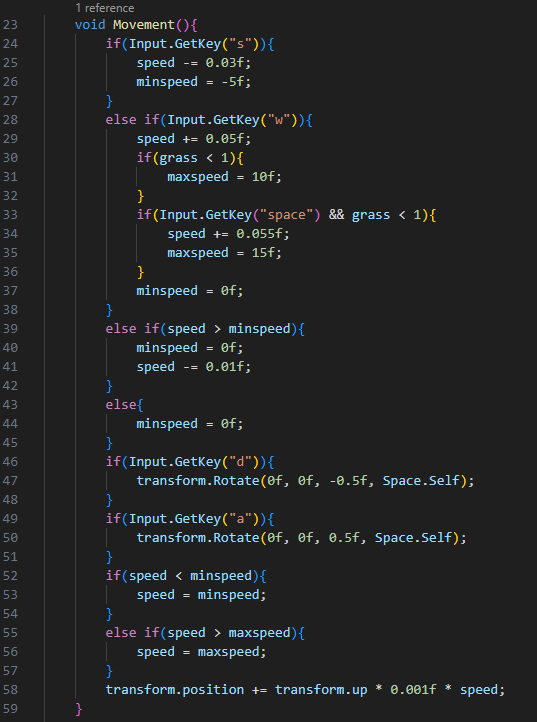


The hierarchy has two player objects, which are 2d sprites, tilemap with multiple different layers for different obstacles, an empty object called GameManager for housing the GameManager code, two text fields made with TMPro, and a few invisible checkpoints.



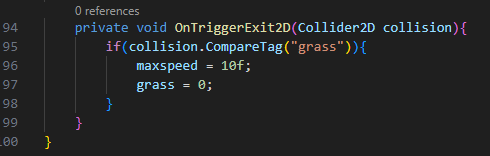
The player 1 script has a few variables, the first one is the speed. This is set to 0 by default and is used to determine how fast the player is going at any given time. The next two are maxspeed and minspeed, these determine the maximum and minimum speed, so that the layer can’t move too fast. Then there is grass, which is used to check if the player is touching grass or not. Void Start is called at when the game is opened. In start speed is set to 0 to prevent a bug where the player moves when the game is started, despite not pressing any keys. Void Update is called every frame, so as often as the computer can update the game state. In it we call movement, in which we do all of our movement for our player.



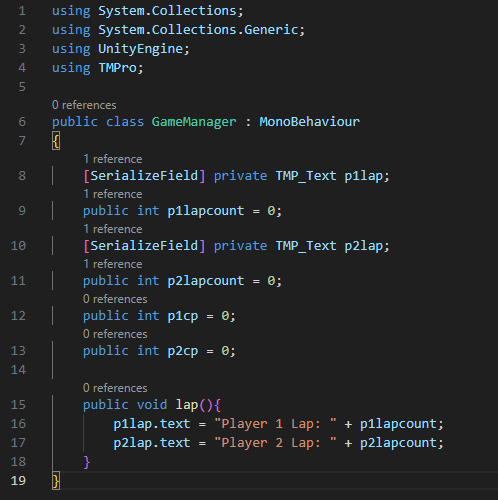
The Main player movement works through a use of a 2d layer object that gets it’s position directly manipulated with the use of transform.position. When the s key is pressed, the speed variable gets reduced by 0.03 units and the minspeed variable gets set to -5, so that you can reverse. When the w key is pressed, the code will increment the speed value by 0.05 units, and if the player is not touching grass it will also set the maxspeed variable to 10. If the space key is also pressed the speed will increment by another 0.055 units and the maxspeed variable will be set to 15. At the end it will also set the minspeed variable to 0 in order to avoid the player automatically going backwards even when you are not pressing the s key. If the d key is pressed, the player's rotation is changed by -0.5 in relation to itself, and the same is done in reverse for the a key. If speed is less than minspeed, it gets set to minspeed, and the same is done with maxspeed. At the end the players position is changed, according to the speed variable. transform.up makes it to where it moves the player in the direction that is designated as up for the player.



Void OnTriggerEnter2D checks for when the player collides with a trigger, these can be designated in unity in the rigidbody settings. At the start the code calls for objects from the GameManager script, so that it can manipulate variables and call functions from it. Next the code compares the triggers tag to see what it is. if it is grass, the maxspeed variable will be set to 3, and the grass variable will be set to one. The grass variable being set to one makes sure the code knows when the player is touching grass. If the tag is oil, the player will be randomly rotated in a range of -30 to 30 degrees in relation to itself. If the tag is border, the players position will be set to the starting position, which in this case is manually set to 1 on the x axis and -3.5 on the y axis. If the tag is one of the checkpoints, the code will check to make sure the p1cp variable is correct in order to prevent skipping part of the course, if it is, the p1cp variable is set to the appropriate value. If the tag is finish, the code will check that the p1cp variable is equal to 3 to make sure the player has hit every checkpoint. If that is the case, it will set the p1cp variable to zero to reset the checkpoint counter and it will increment the p1lapcount variable by one, this is used later to change lapcount text. Finally it runs the lap function from the gamemanager script.



When the player exits a trigger, it checks if it is grass, and if that is the case it will set maxspeed to 10 and grass to 0. This is to make sure the player has the correct maxspeed when getting off the grass.



The GameManager script manipulates text fields made with text mesh pro. It needs to have the using TMPro line at the top or it won’t work. It has a few variables. The serialize field is used to see the variable under the script in unity, which allows it to manipulate different unity objects. In this case it is a text field that is being manipulated. The integers are used to keep track of lap counts between the two players. There are also two variables to keep track of the players checkpoints. When the lap function is run by the player1 and player2 scripts, it sets the text fields to a preset Player x lap: and adds the lap count to the end. Player two is almost exactly the same as player one, except for the fact that a few variables are labeled differently and the starting position is one unit lower.