Ok So we are coding like crazy since the last two , three lessons

And we have not checked anything in the game because there is not any change in the game

So now we are going to finally attach all these scripts together so that our game actually works so lets get started

So we make a Game Manager script and attach it to the Game Manager IOA

And as always we are going to create singleton instance of the Game Manager

So now we are going to create two more function

First function name is StartGame and second function name is GameOver and both are not public

So this will ultimately control all the other managers from here

so when the game starts what do we need to do

As we remember our Score Manger class have function StartScore in the beginning and our UIManager has function GameStart that we need to call in the beginning So in the Game manager we can call both of these function in the beginning when the game starts.

1:55

So inside this StartGame

Through this UIManagerclass we call its GameStart

void StartGame()

{

UiManager.instance.GameStart();

}

Now you can see that’s why we created singleton pattern Now we can access any pubic function from UIManager directly by writing this otherwise we would have to do lot of thing by GameObject

Or findWithTag<who knows> and get access to that and then get component with that and then call any function with that so that is why we use this static singleton pattern

By UIManager we do the code thas need to be done in the start from the UI Manager side. Now we need to call static member of ScoreManager and call the startScore function

Now this will do whatever it needs to be done from the scoremanager side when the game starts

Here’s the code for GameStart

void StartGame()

{

UiManager.instance.GameStart();

ScoreManager.instance.startScore();

}

Now in GameOver

We simply write UIManager.instance.gameOver

So this will do all needs to be done when the game over happens from the UIManager

Now here we are going to write scoreManager.instance.stopScore()

So this will syop our score from incrementing

Next we need to do is declare a Boolean variable gameOver

Inside start method write gameOver = false

Below is our gameOver method

void GameOver()

{

UiManager.instance.GameOver();

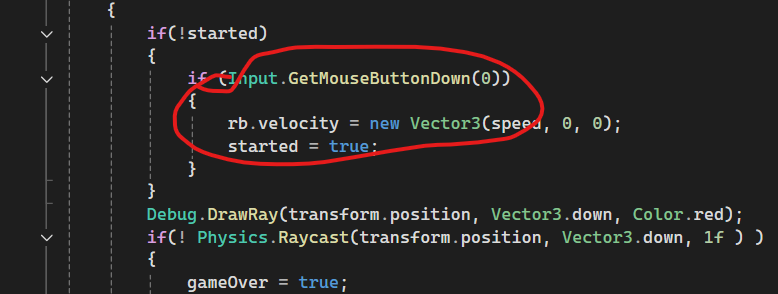
ScoreManager.instance.StopScore();

}

}

Now we go To ball Controller script

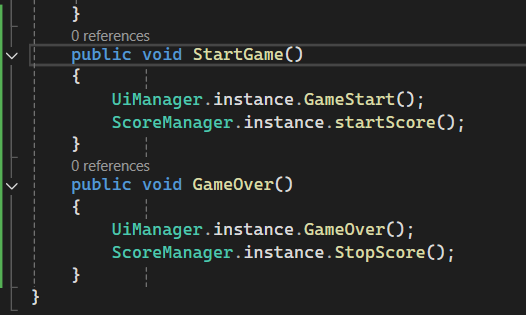
Now you can see in the update we are checking whether the game is actually starting by detecting a tap on the screen



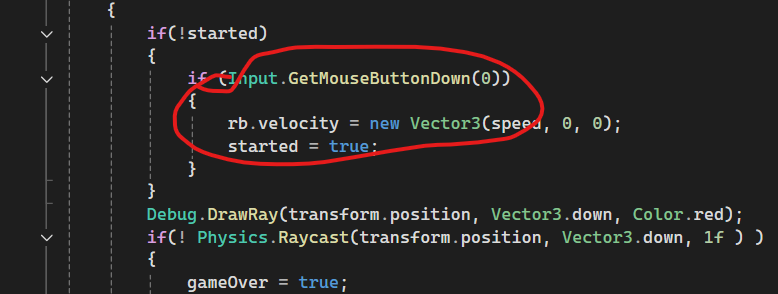
So the above marked area is the area where we actually start the game , where we actually need to start the game

So here you will write GameManager.instance.startGame()

Oh we need to make StartGame and gameOver method of GameManager public otherwise we will not be able to access them from our BallController Script



Now go back to our Ball controller script

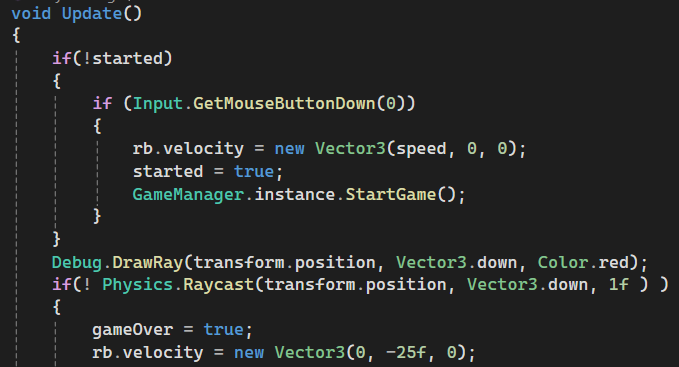


Now in above marked area we write

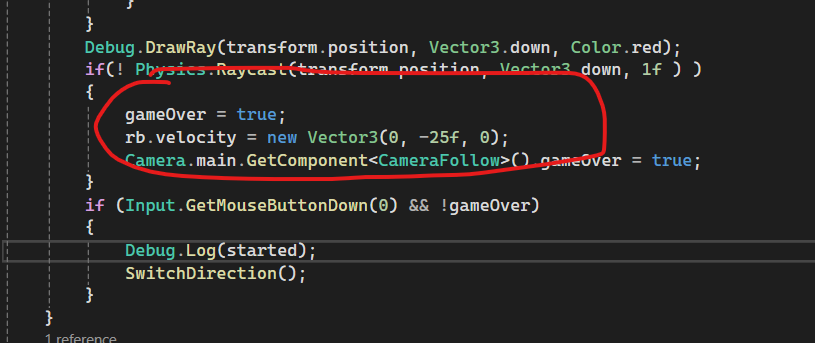
GameManager.instance.StartGame()

So this will take care of everything

We just started it and it will take care of our UIManager, scoreManager and everything



And when the GameOver happens



The above marked area is the area that handles GameOver

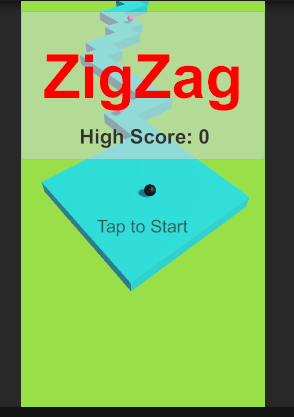
And we are also writing gameOver = true there

So we write here GameManager.instance.GameOver()

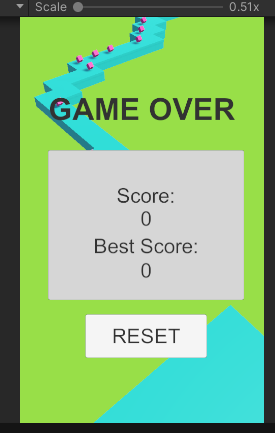
And this will take care of everything we have . we don’t need to bother about anything else

Ok so now if we go back to unity

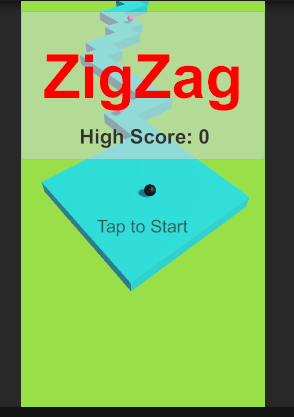
One more thing is that you just need to enable our tapText



Now when we play the unity game then this screen shows if I tap then the screen goes up and the TapTostart if the ball fall down and the game over then we see below screen coming from right side



Now If I click on Reset button then you again see the starting game screen



So the basic mechanism of all our game is complete

In the next lesson we will fix some glitches and we will add some more things so that our game will become complete see you in the next lesson