This is a player movement Tutorial

This Tutorial will help step by step on how to create Player Movement in Unity.

STEP 1

When you create a new script for the player's movement, you will need to type "private float horizontal;" and underneath that in box brackets type" Header(values to adjust)". By doing this It allows the player to create headers in unity that the player can change making it faster to change the player's mechanics compared to always going into Visual Studios and doing it from there. After creating box brackets type "SerialzeField" in the middle of them, and next to that put "private float speed;" then underneath do the same thing except you change"private float speed" to "private float jumpPower;". By doing this it creates headers in Unity that help adjust the speed and the jump power of the player. Using "SerializeField" makes private variables accessible in Unity without making them public and using float in code is meant to help represent a single precision floating point number. Underneath that type "private bool justJumped = false;" because it will make sure that the sprite wont constantly be jumping by using bool which represents a value that is either true or false. Below that create another header in square brackets and in normal brackets next to it type "objects" then close the square brackets. After type "SerializeField" in square brackets and then "private Rigidbody2D rb;".Next you will need to repeat the instruction before 3 more times however instead of typing"Rigidbody2D rb;", type "SpriteRenderer sprite;", "Transform groundCheck;", and "LayerMask groundLayer;". Because of this it now adds headers to Unity to help customise the game. For example "groundCheck" and "groundLayer" will help the character recognise what is the ground and will make sure that when starting the game the sprite wont just fall through the ground and keep on falling. "Rigidbody2D" is the physics of the 2D sprite and "SpriteRenderer" helps render a sprite for 2D physics.

STEP 2

Underneath "void update()" in the curly brackets, type "horizontal = input.GetAxisRaw("horizontal");". After press enter to start a new lin of code and type "if (Input.GetButton("Jump")) justJumped = true;" then press enter again and type "if (Input.GetButtonUp("jump")) justJumped = false;". Below that type "Flip();". All of this says that when the jump button is being pressed, the character sprite will start jumping and when the button isn’t being pressed down the sprite stops jumping. By using “Flip()” it constantly updates the screen so when I do take my finger off the jump button it stops the sprite from jumping.

STEP 3

Below that Type “private void FixedUpdate()”.After create curly brackets, click enter and then type “rb.velocity = new Vector2(horizontal * speed, rb.velocity.y);”. This means that the sprite will move on horizontally and at the speed of my choosing that I put in the header above. After that click enter and then start typing”if(justJumped && IsGrounded())”, create curly brackets and type “rb.velocity = new Vector2(rb.velocity.x, jumpingPower);” and under that put “justJumped = false;”. By doing this it allows the sprite to jump and also shows how the code will calculate how the player will jump by using the sprite’s speed when running and the amount of jumping power the sprite will have.

STEP 4

After click enter and type “private bool IsGrounded()”.Underneath create curlybrackets and in between type “return Physics2D.OverlapCircle(groundCheck.position, 0.3f, groundLayer);”. This resets all the physics to before when the sprite was on the ground.

STEP 5

Under that press enter and type “private void Flip()” and create curly brackets. Between the brackets, type “switch(horizontal) and then create another pair of curly brackets. In those brackets start typing “case 1:” then press enter and put sprite.flipx = false;” and under type “break;”. Press enter again and put “case -1:” and under that type “sprite.flipx = true;” and then press enter and put “break;”. This means when running the sprite will flip depending on the direction you are running in.

STEP 6

After doing all of the steps above save the code above go back into Unity and add the newly created script to the player. Once you’ve done that click on the player you’ve already made and click on “tag”. When you’ve done this click on add tag and create a new tag under “player” but make sure to spell it the same way you spelled it in the code. By doing this everything relevant to the player in the script will be attached to the sprite. Also, add all of the other scripts that will interact with the player for example the health system or the coin pick-up script will need to be on the player as well since both interact and affect the player.

Once you complete all of these steps you have successfully created Player Movement for your game.