Climbing Tutorial

1. Create a scene and add your preferred character controller

Create a new scene.

Create a couple of boxes that will be your floor and walls.

Then create a capsule to that will be your character and add him a material to him so it can be distinguished.

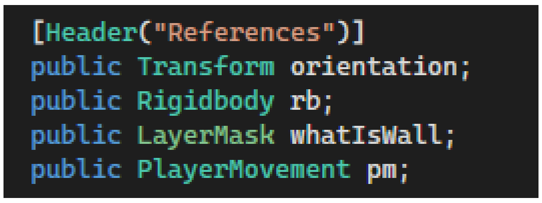
Add any character controller to you capsule to be able to make it move.

Create an empty game object called orientation and add a Rigidbody to the capsule

And finally create a new layer called whatIsWall.

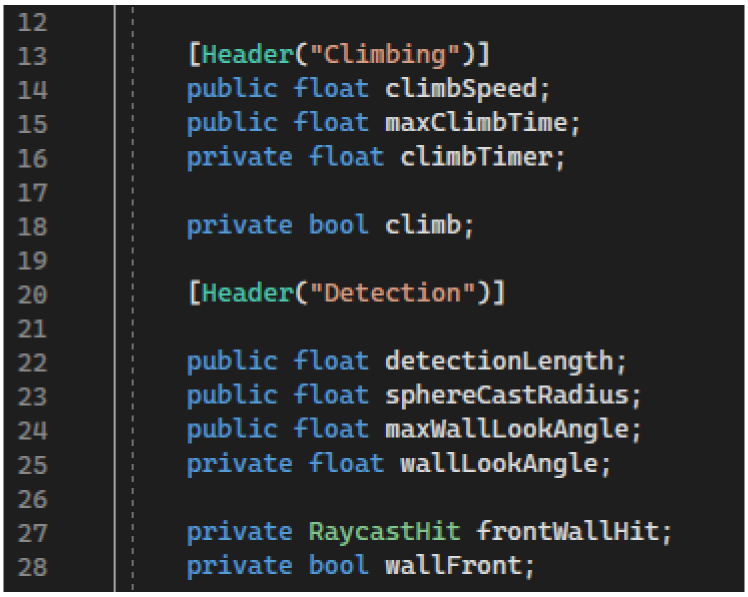
2. Create the Script;

First of all we are going to need some references to the components we have created and also to the character script we have made by doing this:



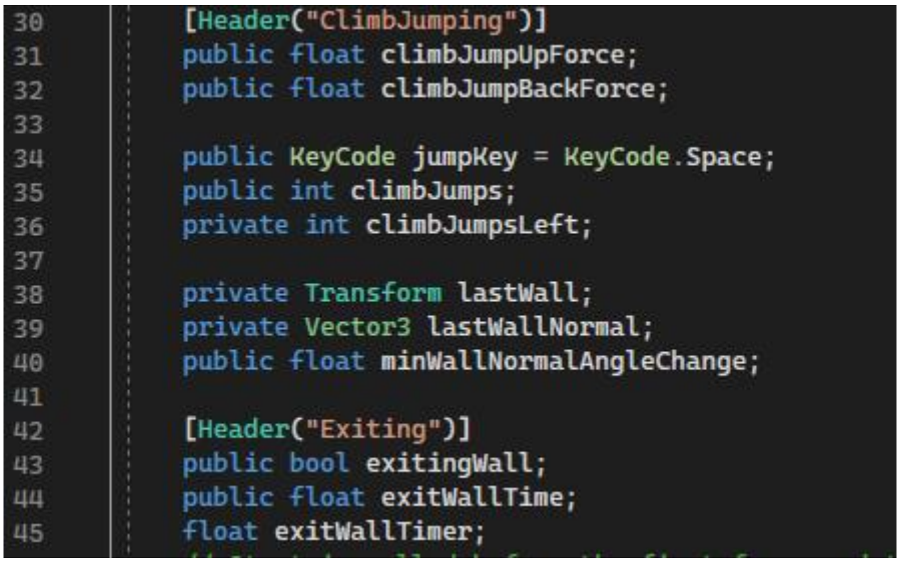
```
[Header("References")]
public Transform orientation;
public Rigidbody rb;
public LayerMask whatIsWall;
public PlayerMovement pm;
```

After referencing other components, we will proceed to add variables that will allow us to select the movement speed and the detection of the wall by using a Raycast.



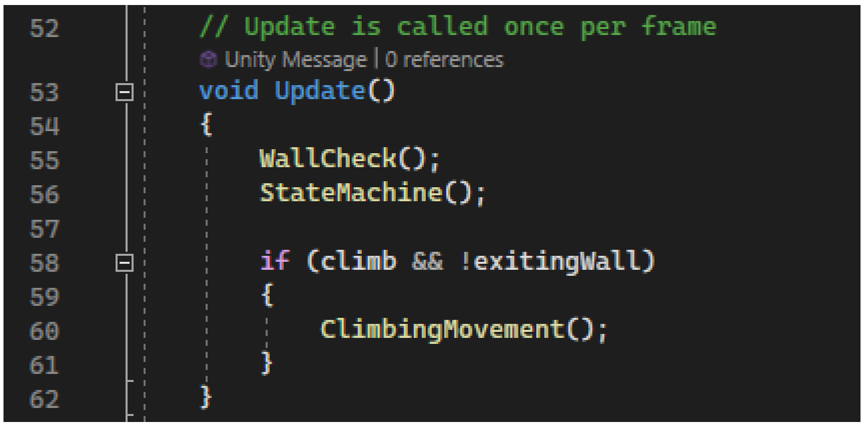
```

    [Header("Climbing")]
    public float climbSpeed;
    public float maxClimbTime;
    private float climbTimer;

    private bool climb;

    [Header("Detection")]

    public float detectionLength;
    public float sphereCastRadius;
    public float maxWallLookAngle;
    private float wallLookAngle;

    private RaycastHit frontWallHit;
    private bool wallFront;
```

To finalise the variable we will make variables that will help us jump up the wall so the climbing feels faster and we will add variables that will allow us to exit the wall once we have started to climb.



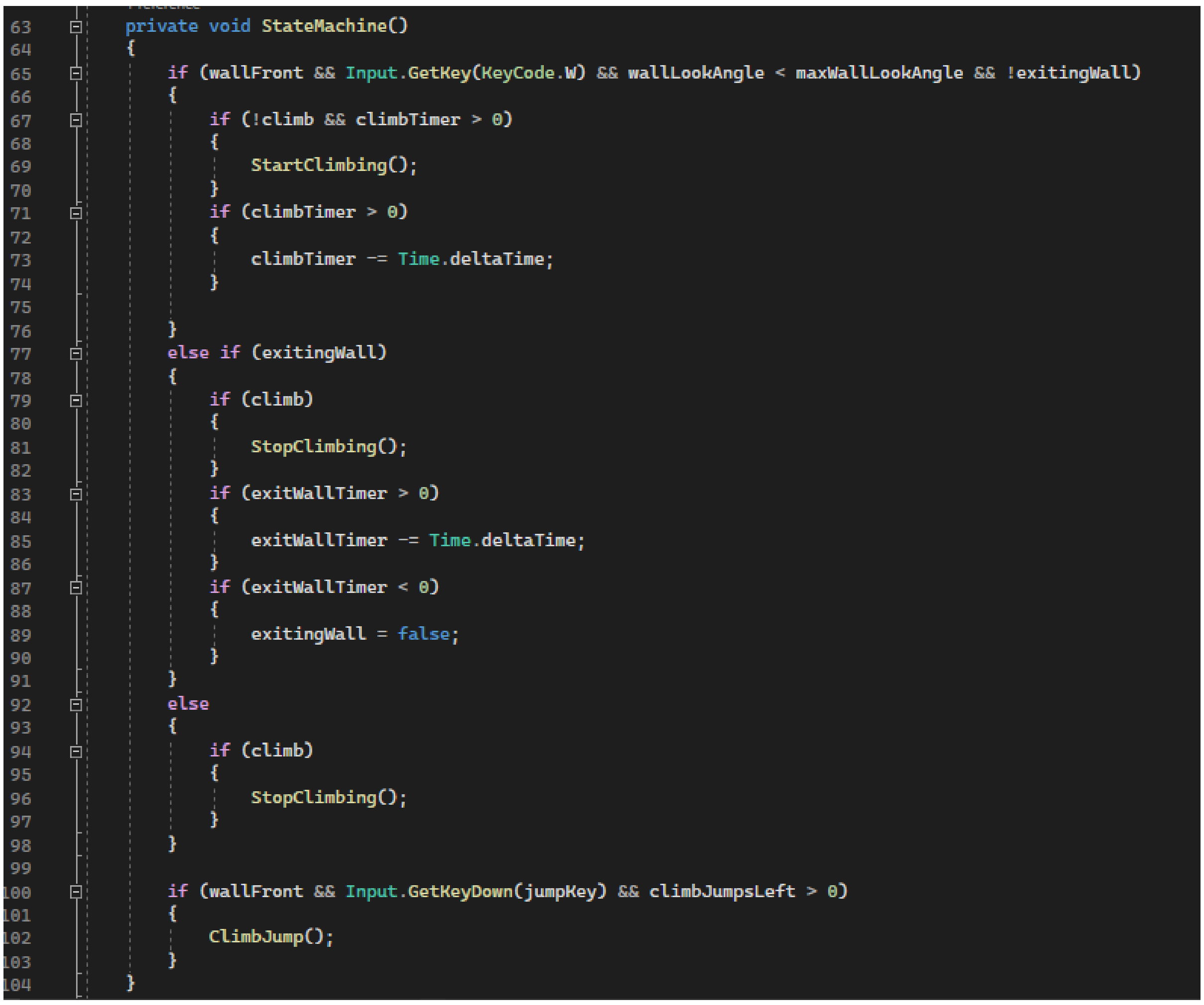
```
    [Header("ClimbJumping")]
    public float climbJumpUpForce;
    public float climbJumpBackForce;

    public KeyCode jumpKey = KeyCode.Space;
    public int climbJumps;
    private int climbJumpsLeft;

    private Transform lastWall;
    private Vector3 lastWallNormal;
    public float minWallNormalAngleChange;

    [Header("Exiting")]
    public bool exitingWall;
    public float exitWallTime;
    float exitWallTimer;
```

 Now we will add the following functions to the update method that will help us later to determine various factors like weather we are code to a wall or not or determine if we are climbing and not exiting the wall. This will be done with next code:



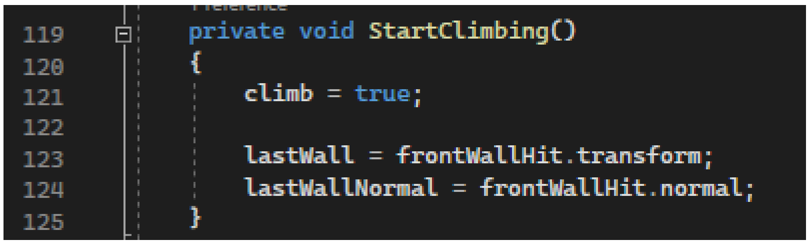
```
    // Update is called once per frame
    Unity Message | 0 references
    void Update()
    {
        WallCheck();
        StateMachine();

        if (climb && !exitingWall)
        {
            ClimbingMovement();
        }
    }
```

Within the state machine function we will determine weather the player is able to climb due to the timer or cannot climb due to the timer also or also will check if the player has jumps left to be able to jump while climbing and this is done within the StateMachine function just like this:



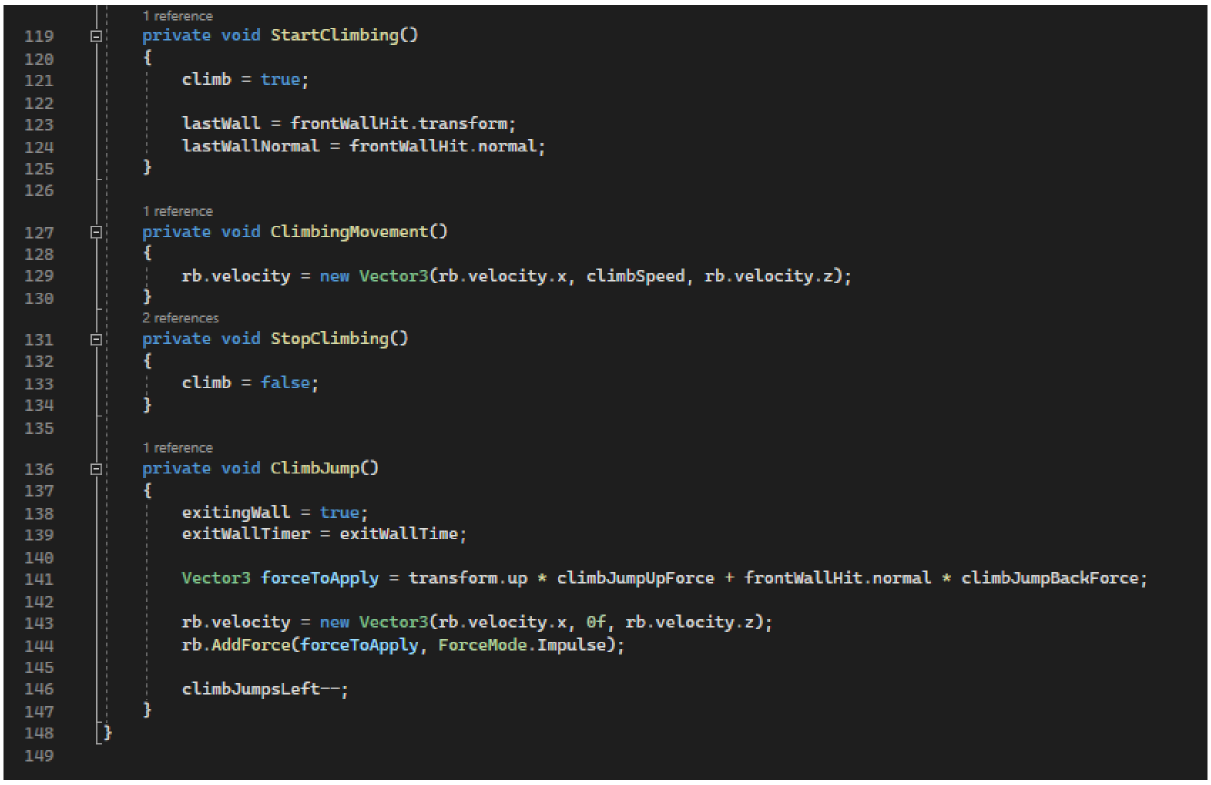
```
    private void StateMachine()
    {
        if (wallFront && Input.GetKey(KeyCode.W) && wallLookAngle < maxWallLookAngle && !exitingWall)
        {
            if (!climb && climbTimer > 0)
            {
                StartClimbing();
            }
            if (climbTimer > 0)
            {
                climbTimer -= Time.deltaTime;
            }

        }
        else if (exitingWall)
        {
            if (climb)
            {
                StopClimbing();
            }
            if (exitWallTimer > 0)
            {
                exitWallTimer -= Time.deltaTime;
            }
            if (exitWallTimer < 0)
            {
                exitingWall = false;
            }
        }
        else
        {
            if (climb)
            {
                StopClimbing();
            }
        }

        if (wallFront && Input.GetKeyDown(jumpKey) && climbJumpsLeft > 0)
        {
            ClimbJump();
        }
    }
```

Next up we will check weather we can climb up or not by checking if we are close to a wall or not, within the StartClimbing function we will right up the following code:



```
    private void StartClimbing()
    {
        climb = true;

        lastWall = frontWallHit.transform;
        lastWallNormal = frontWallHit.normal;
    }
```

Lastly we will add our climb speed values to the code as well as the force in which we will jump un in the wall and this is done by doing the following:



```
    1 reference
    private void StartClimbing()
    {
        climb = true;

        lastWall = frontWallHit.transform;
        lastWallNormal = frontWallHit.normal;
    }

    1 reference
    private void ClimbingMovement()
    {
        rb.velocity = new Vector3(rb.velocity.x, climbSpeed, rb.velocity.z);
    }
    2 references
    private void StopClimbing()
    {
        climb = false;
    }

    1 reference
    private void ClimbJump()
    {
        exitingWall = true;
        exitWallTimer = exitWallTime;

        Vector3 forceToApply = transform.up * climbJumpUpForce + frontWallHit.normal * climbJumpBackForce;

        rb.velocity = new Vector3(rb.velocity.x, 0f, rb.velocity.z);
        rb.AddForce(forceToApply, ForceMode.Impulse);

        climbJumpsLeft--;
    }
}
```

3. Assign and Test

To finish up we will assign the script to the player and test the to see if the code works.