Git Initiate

(if unity project doesn't exist)

Start Unity 3D project, FirstPersonMovement
Save and exit the Project
Rename project folder to FirstPersonMovementx
Create Repo named FirstPersonMovementin Git
git clone https://github.com/rayhere/FirstPersonMovement.git
cd FirstPersonMovement
Drag the project files from FirstPersonMovementx into FirstPersonMovementfolder
git add .
git commit -a -m "2nd commit, project initiated"
git push

FIRST PERSON MOVEMENT in 10 MINUTES - Unity Tutorial

https://www.youtube.com/watch?v=f473C43s8nE

Summary

This will create a Player in ThirdPerson View
With Move and Jump
With Character Controller
No Animation
No Rigidbody

Step1 Setup a camera

Required Package
Input System
Cinemachine
ProBuilder

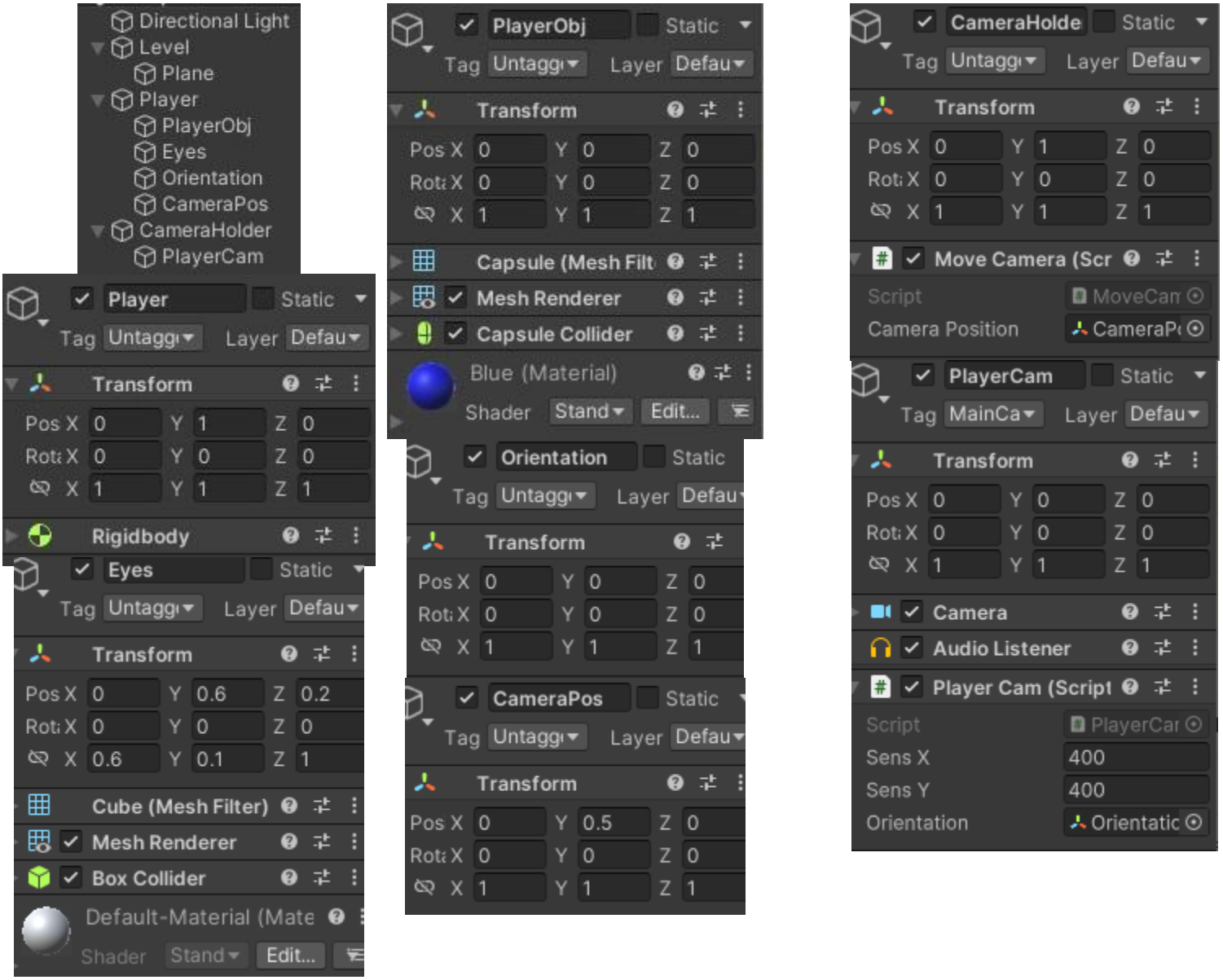
https://youtu.be/f473C43s8nE?si=0GgS2H4KKc4HfyiT&t=131
Create

- ☐ Create Empty named Level >
  - ☐ Create 3DObject > Plane > apply Material named Ground
- ☐ Empty named Player > Pos.y 1 > add Rigidbody > Interpolate Interpolate [Rigidbody] > CollisionDetection Continuous

- [ ] 3DObject > Capsule named PlayerObj > add PlayerInput > Create Actions [PlayerInput] named PlayerInputAction > apply PlayerInputAction in Action [PlayerInput]
    - [ ] 3DObject > Cube named Eyes > Pos 0, 0.6, 0.2 > Scale 0.6, 0.1, 1
- [ ] Create Empty Object named Orientation

- [ ] Create Empty named CameraHolder > add MoveCamera.cs > Transform same as Player [GameObject] > Drag CameraPos child of Player [GameObject] to CameraPosition [MoveCamera.cs]
    - [ ] Drag MainCamera here, named PlayerCam > Pos 0,0,0 > add PlayerCam.cs > set value 400 for SensX, SensY > Drag Orientation child of Player [GameObject] to Orientation [PlayerCam.cs]

P.S. You can directly adjust PlayerCam Pos to have better view



Change:

Empty Object named Orientation is for keeps track of the direction you're facing
Orientation [EmptyObject] stores the direction your facing

Put the camera into a separate camera holder
https://youtu.be/f473C43s8nE?si=KI9Zczq1Wht0kLeX&t=154
Because having a camera on a rigidbody object can be a bit buggy

In order for this to work, you just need this CameraPos [EmptyObject] inside the player.
Drag it up a bit, CameraPos Pos.Y is about Player [GameObject] Pos.Y

Then on the camera holder, you can add this really simple script MoveCamera.cs, to make the camera always move with your player

Create 2 script
MoveCamera.cs
PlayerCam.cs

Code

MoveCamera.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class MoveCamera : MonoBehaviour
{
    public Transform cameraPosition;

    private void Update()
    {
        transform.position = cameraPosition.position;
    }
}
```

PlayerCam.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEditor.Experimental.GraphView;
using UnityEngine;

public class PlayerCam : MonoBehaviour
{
    public float sensX; // Sensitivity for mouse X axis
    public float sensY; // Sensitivity for mouse Y axis

    public Transform orientation; // Reference to the player's orientation
transform

    float xRotation; // Current rotation around the X axis
    float yRotation; // Current rotation around the Y axis

    private void Start()
    {
        // Lock the cursor in the middle of the screen and make it
invisible
        Cursor.lockState = CursorLockMode.Locked;
        Cursor.visible = false;
    }

    private void Update()
    {
        // Get mouse input
        float mouseX = Input.GetAxisRaw("Mouse X") * Time.deltaTime *
sensX;
        float mouseY = Input.GetAxisRaw("Mouse Y") * Time.deltaTime *
sensY;

        // Update rotation values based on mouse input
        yRotation += mouseX;
        xRotation -= mouseY;

        // Clamp the rotation around the X axis to prevent looking up or
down more than 90 degrees
        xRotation = Mathf.Clamp(xRotation, -90f, 90f);
```

```
        // Rotate the camera and player orientation along both the X and Y
axes
        transform.rotation = Quaternion.Euler(xRotation, yRotation, 0);

        // Rotate the player's orientation along the Y axis
        orientation.rotation = Quaternion.Euler(0, yRotation, 0);
    }
}
```

FIRST PERSON MOVEMENT in 10 MINUTES - Unity Tutorial

https://youtu.be/f473C43s8nE?si=YFn1_gT9Xg2Zde3q&t=200

Step2 Set up a movement

Create PlayerMovement.cs > add PlayerMovement.cs in Player [GameObject] >

Drag Orientation Player[GameObject] into Orientation PlayerMovement.cs Player[GameObject] >

Create Layer named Ground > pick Ground Layer in WhatIsGround PlayerMovement.cs[GameObject]

Drag & Speed Control

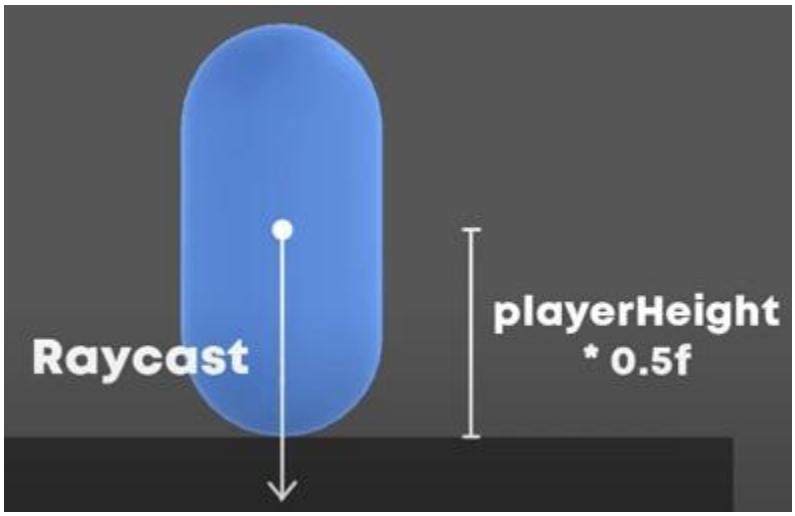
https://youtu.be/f473C43s8nE?si=Ww2eetEaE8hYKGk8&t=305

Apply drag to the player's ridgebody, which will make the movement less slippery and to limit the player's velocity to its movement speed

Apply Drag when Player on Ground

Ground Check

To perform the ground check, you want to shoot the raycast from your current position down, and see if it hits something, the length of this ray will be half of your player's height + a bit more.




```
    [Header("Movement")]
    public float groundDrag;
    [Header("Ground Check")]
    public float playerHeight;
    public LayerMask whatIsGround;
    bool grounded;

    private void Update()
    {
        GroundDrag();
    }

    private void GroundDrag()
    {
        // Perform ground check
        grounded = Physics.Raycast(transform.position, Vector3.down,
playerHeight * 0.5f + 0.3f, whatIsGround);

        // Apply drag when grounded
        rb.drag = grounded ? groundDrag : 0;
    }
```

Jumping & Air Control

https://youtu.be/f473C43s8nE?si=pxDbN1yNIVLISXS5&t=442

```
    public float jumpForce;
    public float jumpCooldown;
    public float airMultiplier;
    bool readyToJump;
```

```
private void MyInput()
    {
        // Check for jump input and conditions for jumping
        if (Input.GetKey(jumpKey) && readyToJump && grounded)
        {
            readyToJump = false;
            Debug.Log("Jump!");
            Jump();
            Invoke(nameof(ResetJump), jumpCooldown); // Allow to
continuously jump if the jump key is held down
        }
    }
```

```
private void Jump()
    {
        // Reset vertical velocity before applying jump force to avoid
accumulating jump force
        rb.velocity = new Vector3(rb.velocity.x, 0f, rb.velocity.z);

        // Apply impulse force for jumping
        rb.AddForce(transform.up * jumpForce, ForceMode.Impulse);
    }

    private void ResetJump()
    {
        readyToJump = true;
    }
```

Code

PlayerMovement.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using TMPro;

public class PlayerMovement : MonoBehaviour
{
    [Header("Movement")]
    public float moveSpeed;

    public float groundDrag;

    public float jumpForce;
    public float jumpCooldown;
    public float airMultiplier;
    bool readyToJump;

    [HideInInspector] public float walkSpeed;
    [HideInInspector] public float sprintSpeed;

    [Header("Keybinds")]
    public KeyCode jumpKey = KeyCode.Space;

    [Header("Ground Check")]
    public float playerHeight;
    public LayerMask whatIsGround;
    bool grounded;

    public Transform orientation;

    // To hold your input
    float horizontalInput;
    float verticalInput;

    Vector3 moveDirection;
```

```
    Rigidbody rb;

    private void Start()
    {
        // Assign your Rigidbody and freeze its rotation
        rb = GetComponent<Rigidbody>();
        rb.freezeRotation = true;

        readyToJump = true;
    }

    private void Update()
    {
        GroundDrag();
        MyInput(); // This will keep checking allowed input for all
movement
        SpeedControl();
    }

    private void FixedUpdate()
    {
        MovePlayer(); // To apply force on the player Rigidbody
    }

    // This method will handle your input, including movement and jumping,
and verify if jumping is allowed
    private void MyInput()
    {
        horizontalInput = Input.GetAxisRaw("Horizontal");
        verticalInput = Input.GetAxisRaw("Vertical");

        // Check for jump input and conditions for jumping
        if (Input.GetKey(jumpKey) && readyToJump && grounded)
        {
            readyToJump = false;

            Jump();

            Invoke(nameof(ResetJump), jumpCooldown); // Allow to
continuously jump if the jump key is held down
```

```
        }
    }

    private void MovePlayer()
    {
        // Calculate movement direction based on orientation
        moveDirection = orientation.forward * verticalInput +
orientation.right * horizontalInput;

        // Apply force for movement based on ground or air
        if (grounded)
            rb.AddForce(moveDirection.normalized * moveSpeed * 10f,
ForceMode.Force);
        else if (!grounded)
            rb.AddForce(moveDirection.normalized * moveSpeed * 10f *
airMultiplier, ForceMode.Force);
    }

    private void SpeedControl()
    {
        // Get the horizontal velocity
        Vector3 flatVel = new Vector3(rb.velocity.x, 0f, rb.velocity.z);

        // Limit velocity if needed to prevent exceeding maximum speed
        if (flatVel.magnitude > moveSpeed)
        {
            Vector3 limitedVel = flatVel.normalized * moveSpeed;
            rb.velocity = new Vector3(limitedVel.x, rb.velocity.y,
limitedVel.z);
        }
    }

    private void Jump()
    {
        // Reset vertical velocity before applying jump force to avoid
accumulating jump force
        rb.velocity = new Vector3(rb.velocity.x, 0f, rb.velocity.z);

        // Apply impulse force for jumping
        rb.AddForce(transform.up * jumpForce, ForceMode.Impulse);
```

```
    }

    private void ResetJump()
    {
        readyToJump = true;
    }

    private void GroundDrag()
    {
        // Perform ground check
        grounded = Physics.Raycast(transform.position, Vector3.down,
playerHeight * 0.5f + 0.3f, whatIsGround);

        // Apply drag when grounded
        rb.drag = grounded ? groundDrag : 0;
    }
}
```

Problem: Rigidbody won’t do rotation follow the direction of PlayerCam

To Fix:
First, create a serialized field for the PlayerCam object:

```
[SerializeField] private Transform playerCam;
```

Then, in your `Update()` or `FixedUpdate()` method, update the rotation of the Rigidbody to match the rotation of the PlayerCam object:

```
private void Update()
    {
        GroundDrag();
        MyInput(); // This will keep checking allowed input for all
movement
        SpeedControl();
        RotatePlayer();
    }
private void RotatePlayer()
    {
        if (playerCam != null)
        {
            // Set the Rigidbody's rotation to match the PlayerCam's
rotation
            rb.rotation = Quaternion.Euler(0f, playerCam.eulerAngles.y,
0f);
        }
    }
```

Code

PlayerMovement.cs Fix update rb rotation follow the roataion of separate object

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using TMPro;

public class PlayerMovement : MonoBehaviour
{
    [Header("Movement")]
    public float moveSpeed;

    public float groundDrag;

    public float jumpForce;
    public float jumpCooldown;
    public float airMultiplier;
    bool readyToJump;

    [HideInInspector] public float walkSpeed;
    [HideInInspector] public float sprintSpeed;

    [Header("Keybinds")]
    public KeyCode jumpKey = KeyCode.Space;

    [Header("Ground Check")]
    public float playerHeight;
    public LayerMask whatIsGround;
    bool grounded;

    public Transform orientation;

    [SerializeField] private Transform playerCam;

    // To hold your input
    float horizontalInput;
    float verticalInput;
```

```
    Vector3 moveDirection;

    Rigidbody rb;

    private void Start()
    {
        // Assign your Rigidbody and freeze its rotation
        rb = GetComponent<Rigidbody>();
        rb.freezeRotation = true;

        readyToJump = true;
    }

    private void Update()
    {
        GroundDrag();
        MyInput(); // This will keep checking allowed input for all
movement
        SpeedControl();
        RotatePlayer();
    }

    private void FixedUpdate()
    {
        MovePlayer(); // To apply force on the player Rigidbody
    }

    // This method will handle your input, including movement and jumping,
and verify if jumping is allowed
    private void MyInput()
    {
        horizontalInput = Input.GetAxisRaw("Horizontal");
        verticalInput = Input.GetAxisRaw("Vertical");

        // Check for jump input and conditions for jumping
        if (Input.GetKey(jumpKey) && readyToJump && grounded)
        {
            readyToJump = false;
            Debug.Log("Jump!");
```

```
            Jump();

            Invoke(nameof(ResetJump), jumpCooldown); // Allow to
continuously jump if the jump key is held down
        }
    }

    private void MovePlayer()
    {
        // Calculate movement direction based on orientation
        moveDirection = orientation.forward * verticalInput +
orientation.right * horizontalInput;

        // Apply force for movement based on ground or air
        if (grounded)
            rb.AddForce(moveDirection.normalized * moveSpeed * 10f,
ForceMode.Force);
        else if (!grounded)
            rb.AddForce(moveDirection.normalized * moveSpeed * 10f *
airMultiplier, ForceMode.Force);
    }

    private void SpeedControl()
    {
        // Get the horizontal velocity
        Vector3 flatVel = new Vector3(rb.velocity.x, 0f, rb.velocity.z);

        // Limit velocity if needed to prevent exceeding maximum speed
        if (flatVel.magnitude > moveSpeed)
        {
            Vector3 limitedVel = flatVel.normalized * moveSpeed;
            rb.velocity = new Vector3(limitedVel.x, rb.velocity.y,
limitedVel.z);
        }
    }

    private void Jump()
    {
        // Reset vertical velocity before applying jump force to avoid
accumulating jump force
```

```
        rb.velocity = new Vector3(rb.velocity.x, 0f, rb.velocity.z);

        // Apply impulse force for jumping
        rb.AddForce(transform.up * jumpForce, ForceMode.Impulse);
    }

    private void ResetJump()
    {
        readyToJump = true;
    }

    private void GroundDrag()
    {
        // Perform ground check
        grounded = Physics.Raycast(transform.position, Vector3.down,
playerHeight * 0.5f + 0.3f, whatIsGround);

        // Apply drag when grounded
        rb.drag = grounded ? groundDrag : 0;
    }

    private void RotatePlayer()
    {
        if (playerCam != null)
        {
            // Set the Rigidbody's rotation to match the PlayerCam's
rotation
            rb.rotation = Quaternion.Euler(0f, playerCam.eulerAngles.y,
0f);
        }
    }
}
```

THIRD PERSON MOVEMENT in 11 MINUTES - Unity Tutorial

https://www.youtube.com/watch?v=UCwwn2q4Vys

Different Orbits value for CinemachineFreeLook
https://youtu.be/UCwwn2q4Vys?si=WsGNXDFBfr4Ftr4o&t=145