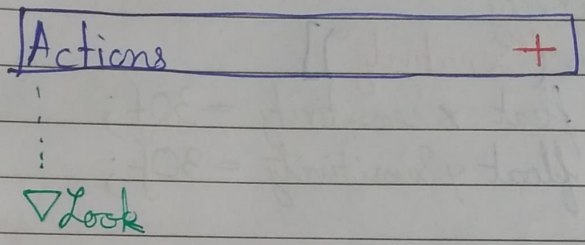
Camera (PlayerLook)

Its great we can move arround the scene. But we are not able to look around the scene with the camera

(i) Let's create a Action for PlayerLook.

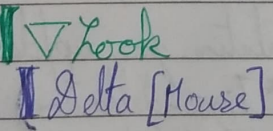
(i) Open Player Input

(ii) Under Actions, click on [+]



(iii) Under Properties of ▽Look Action
- Change Action Type → Value
- Change Control Type → Vector 2

(iv) Add Binding, Path → Delta[Mouse]



(v) Save the asset

If you check Player Input [#]
The Action Look and its binding is already updated

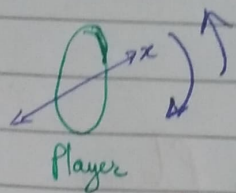
(2) Let's ~~script~~ create PlayerLook script
(manages mouse & camera control)

In this code :-

```
public class PlayerLook : MonoBehaviour
{
    public Camera cam;
    private float xRotation = 0f;

    //Sensitivity
    [Header("Sensitivity")]
    public float xSensitivity = 30f;
    public float ySensitivity = 30f;

    //Process Look
    public void ProcessLook(Vector2 input)
    {
        //Mouse input
        float mouseX = input.x;
        float mouseY = input.y;

        //Camera rotation calculation for up/down look
        xRotation -= (mouseX * Time.deltaTime) * ySensitivity;
        xRotation = Mathf.Clamp(xRotation, -80f, 80f);

        //Apply transform to camera
        cam.transform.localRotation = Quaternion.Euler(xRotation, 0f, 0f);

        //Rotate Player itself to look left/right
        transform.Rotate(Vector3.up * (mouseX * Time.deltaTime)
                         * xSensitivity);
    }
}
```

Let's breakdown the code :-

- xRotation



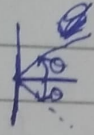
∴ xRotation signifies tilt rotation of x-axis
It is not rotation for left/right

This tilt rotation => up/down movement

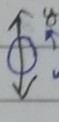
In the code :- xRotation -= (mouse X * Time.deltaTime) * y Sensitivity

(x-axis)

up/down influence

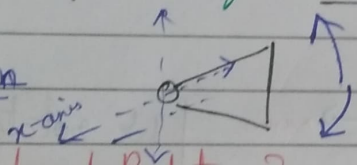


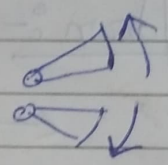
xRotation = Mathf.Clamp(xRotation, -80f, 80f);



Here, Clamp() → forces xRotation value to be bet^n -80f & +80f

- local Rotation





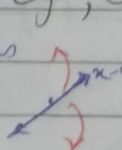
[Between -80f to +80f]

Why we local Rotation?

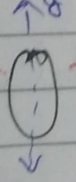
→ Because we ~~may~~ have multiple camera(s) in scene
& we want only the camera attached to Player be rotated

In the code :- cam.transform.localRotation = Quaternion.Euler(xRotation, 0f, 0f);

Here, Quaternion.Euler() → rotate/tilt x-axis
Vector3 used here → (xRotation, 0, 0)



- Rotate



This time to look around horizontal we do tilt rotation of y-axis

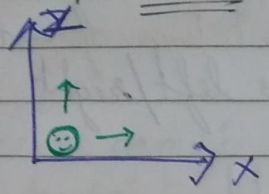
Here, transform.Rotate() → rotate/tilt y-axis
Vector3 used here → (Vector3.up * (mouse X * Time.deltaTime * x Sensitivity));

(0, 1, 0)

Let's understand the workflow

- For Translation :-

We simply have to move Player

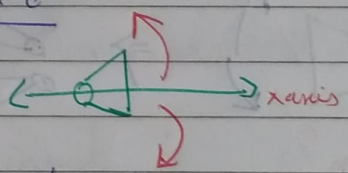


So we had to just add Vector 3

- For Rotation :-

We have to think in terms of rotation of axis

(i) For up/down :-

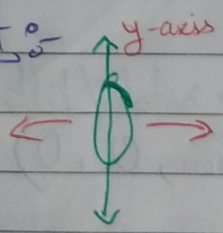


[We rotate the x-axis]

For player's transform to remain unaffected,
we rotate the camera

∴ Player itself won't tilt along x-axis

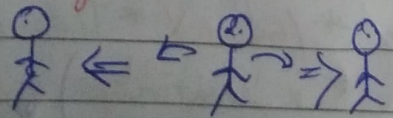
(ii) For left/right :-



[We rotate the y-axis]

For player to look around horizontal,
we rotate the player

∴ Player & Camera will tilt along y-axis



(Ensure Camera is child of Player)

③ Let's modify Input Manager script

In this code :-

```
public class InputManager : MonoBehaviour
{
    : (same as previous)
    • private PlayerLook look;

    void Awake() {
        :
        • look = GetComponent<PlayerLook>();
        :
    }

    • void Update
    • private void LateUpdate() {
        look.ProcessLook(onFoot.Look.ReadValue<Vector2>());
    }
    void ProcessLook(Vector2 input)
    {
        :
    }
}
```

onFoot → Player Input

Look → Action

ReadValue<Vector2>() → Read Vector2 value of Look Binding

Now, attach PlayerLook → Player object

↳ Reference PlayerCam → PlayerLook's Cam

With that, YOU HAVE

~~FPS~~

First Person Movement