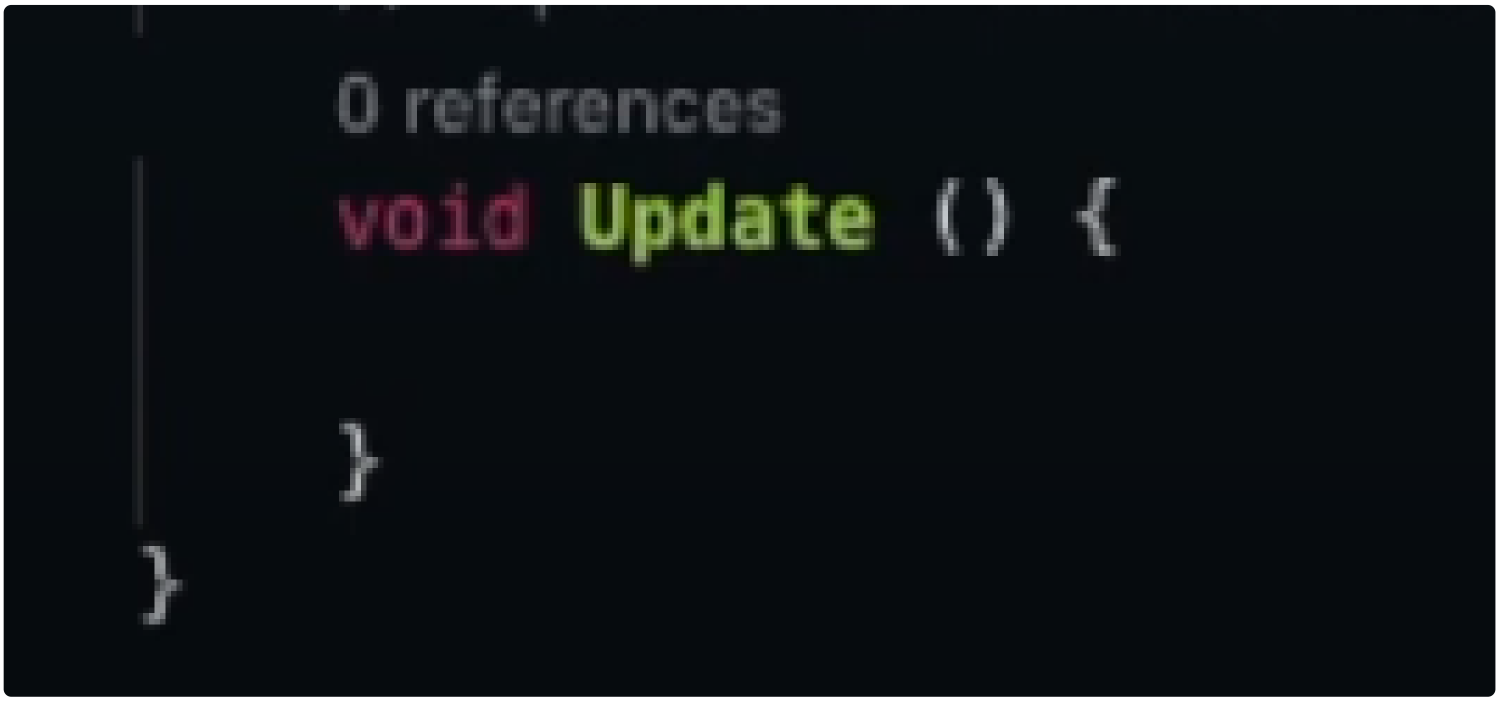
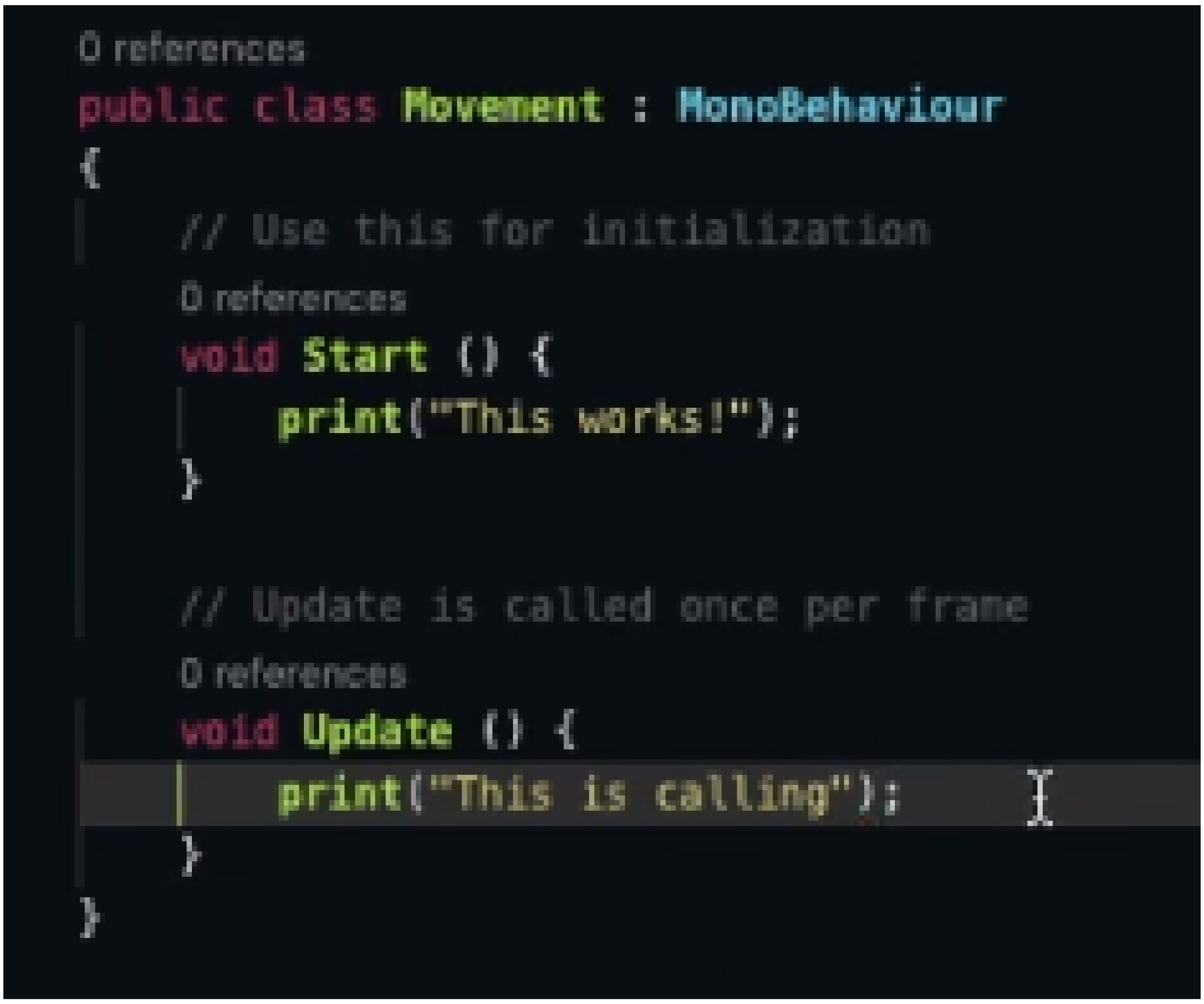


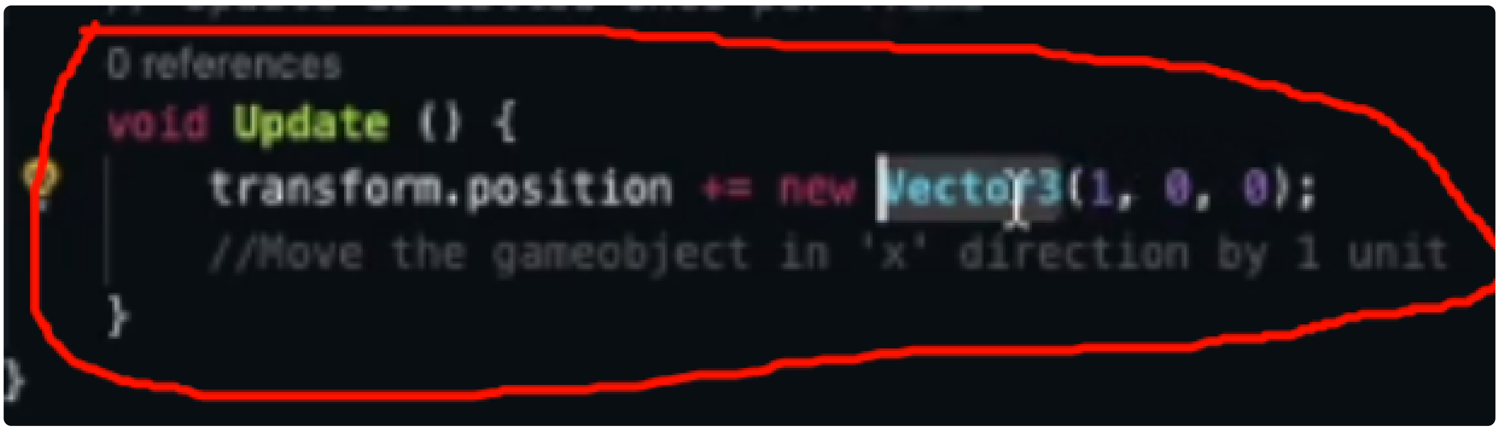
```
    0 references
    void Update () {

    }
}
```

Update method gives you the FPS of the game. game screen refreshes each time update method called.



```
0 references
public class Movement : MonoBehaviour
{
    // Use this for initialization
    0 references
    void Start () {
        print("This works!");
    }

    // Update is called once per frame
    0 references
    void Update () {
        print("This is calling");
    }
}
```

Movement Code:

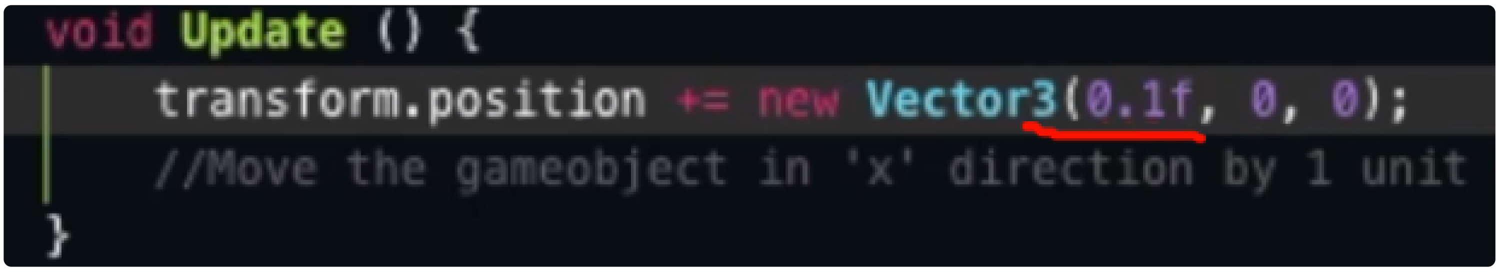


```
0 references
void Update () {
    transform.position += new Vector3(1, 0, 0);
    //Move the gameobject in 'x' direction by 1 unit
}
}
```

transform.position += new Vector3(1,0,0); (**Below is written in more detailed form**)

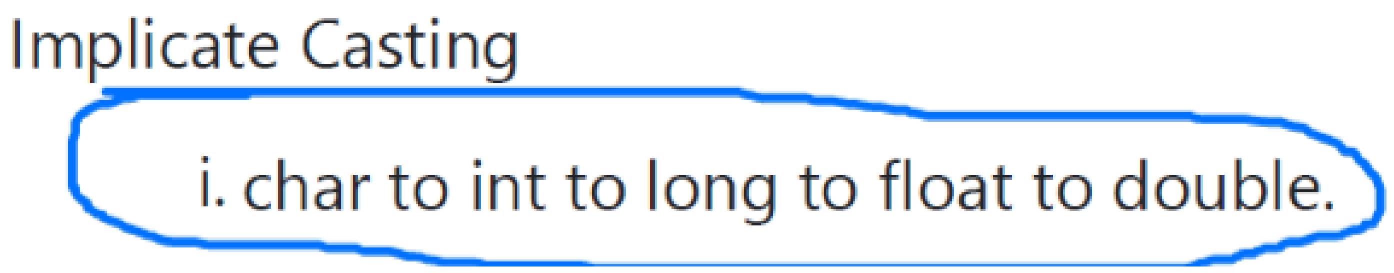
transform.position = current position + new Vector3(1,0,0);

Since it is in update method then this will be called once per frame causing the object to move infinitely in certain direction.



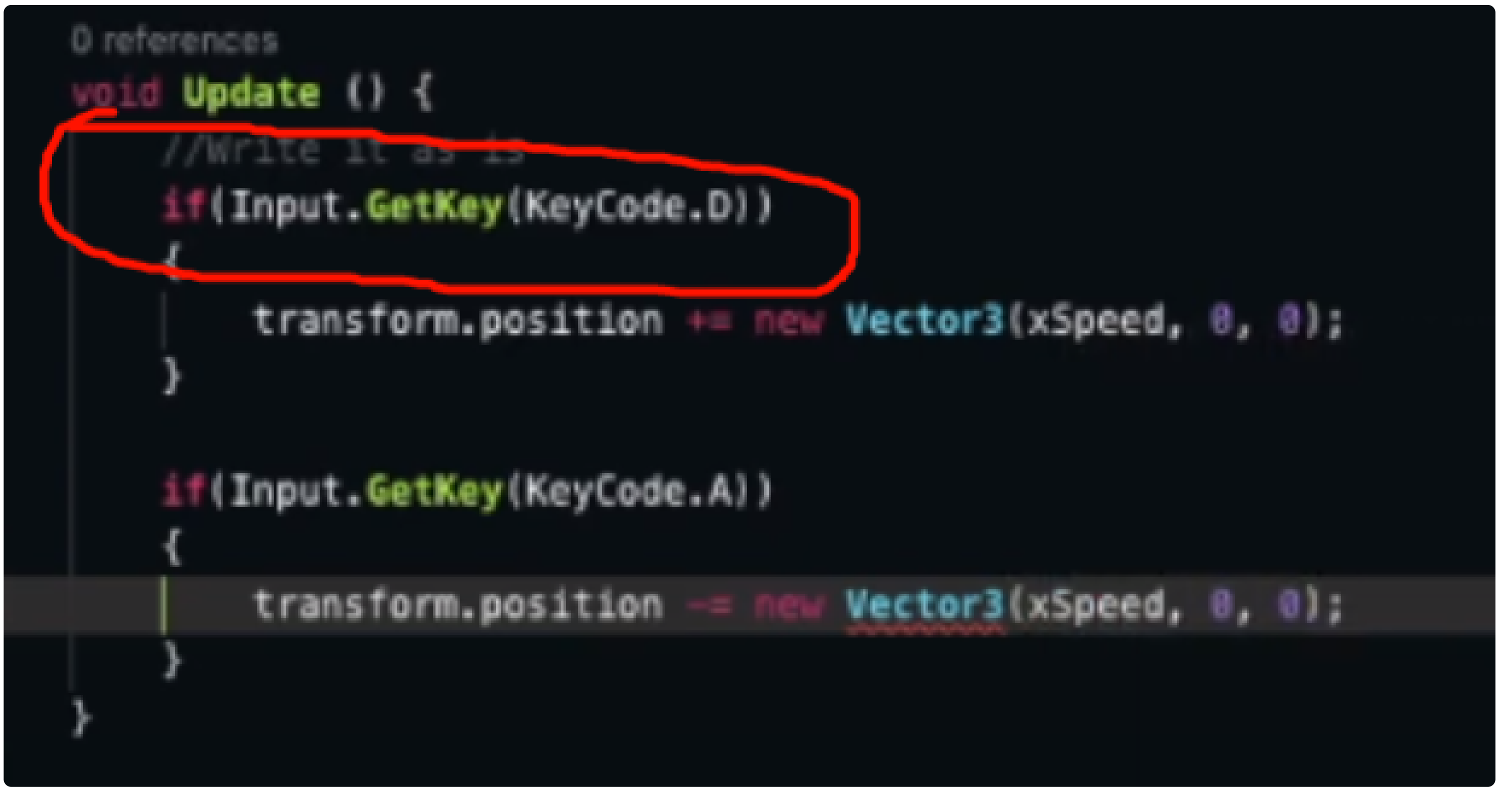
```
void Update () {
    transform.position += new Vector3(0.1f, 0, 0);
    //Move the gameobject in 'x' direction by 1 unit
}
```

keeping in mind the data type. Implicite casting comes into play when coverting int to float.



Implicate Casting

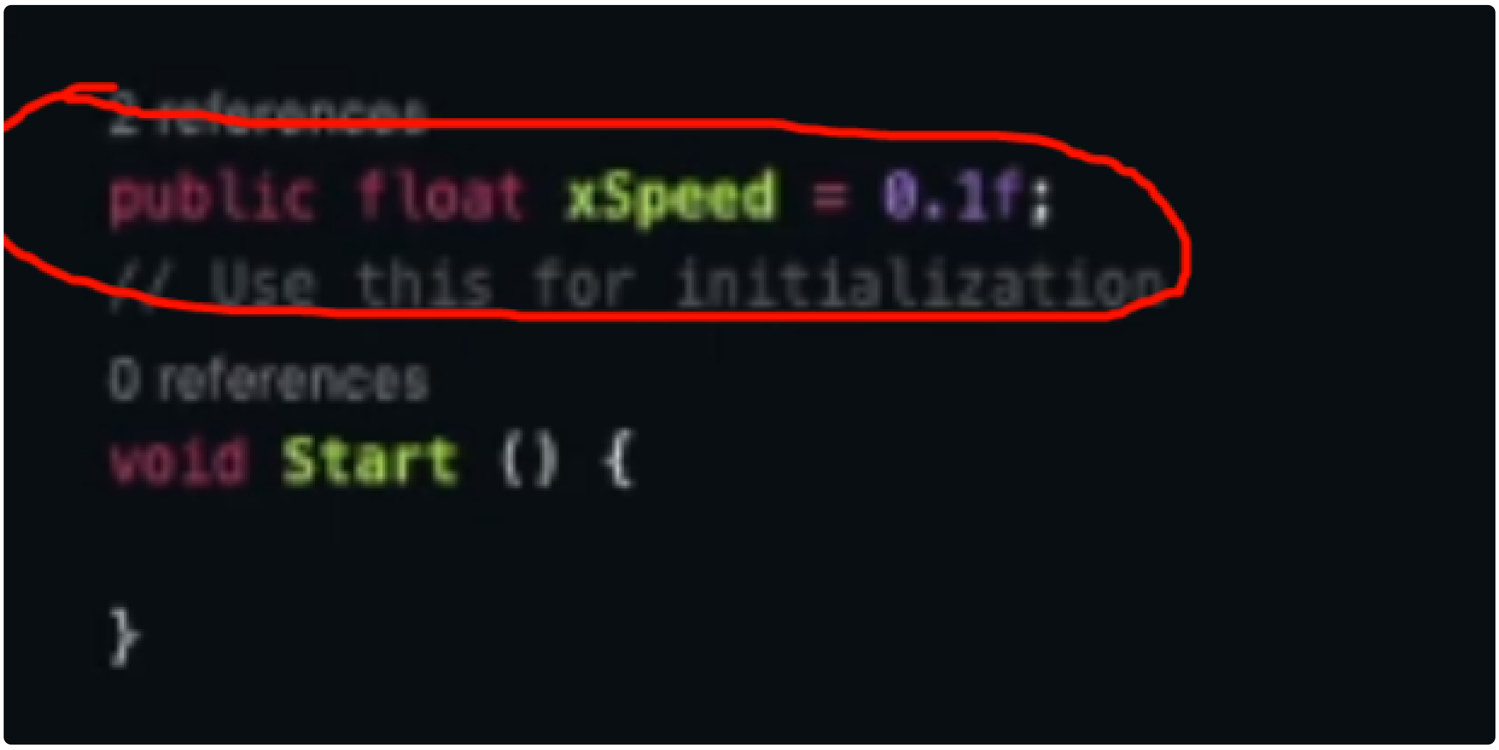
i. char to int to long to float to double.



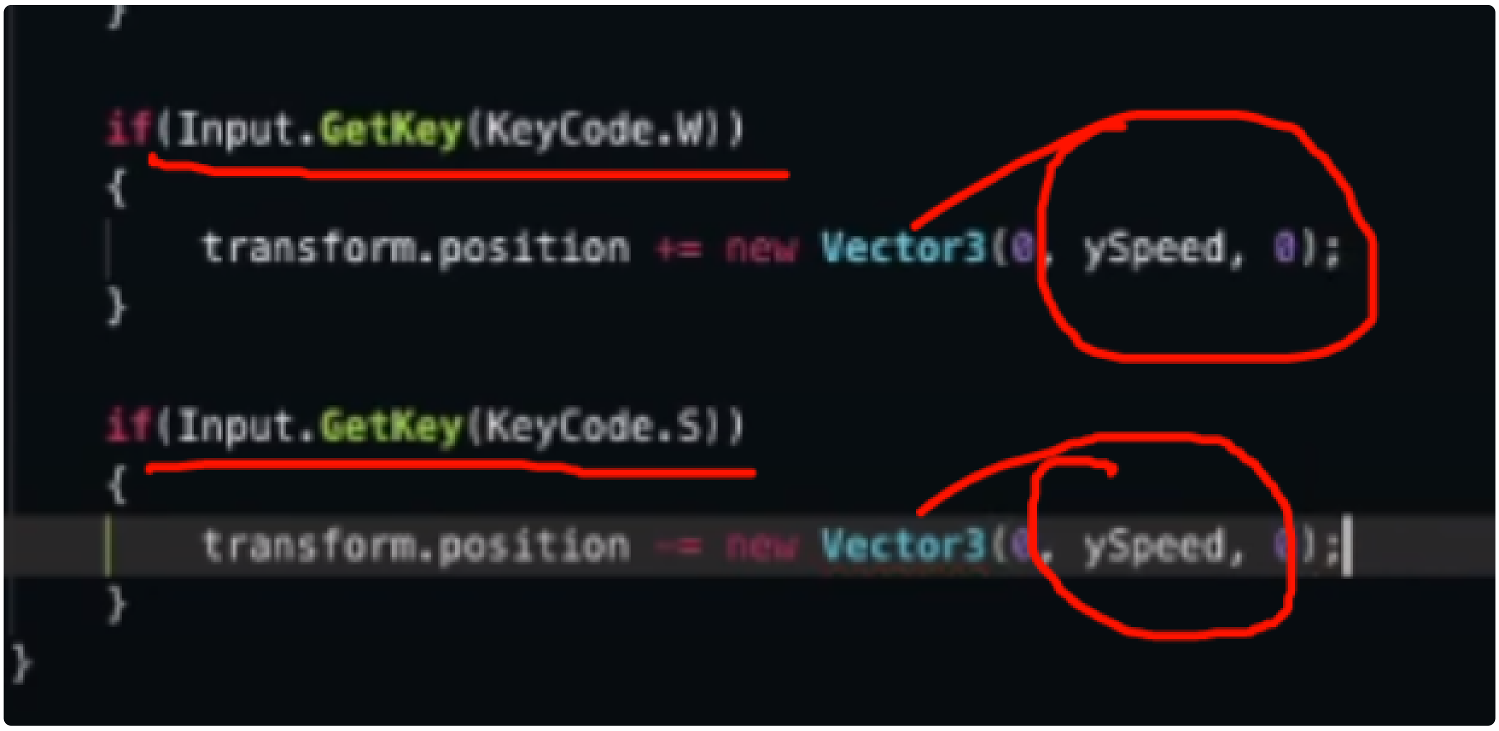
```
0 references
void Update () {
    //Write it as is
    if(Input.GetKey(KeyCode.D))
    {
        transform.position += new Vector3(xSpeed, 0, 0);
    }

    if(Input.GetKey(KeyCode.A))
    {
        transform.position -= new Vector3(xSpeed, 0, 0);
    }
}
```

IF Statement for moving as per player button (Follow Syntax)

Declaring Variable outside the pre-defined methods.



```
2 references
public float xSpeed = 0.1f;
// Use this for initialization
0 references
void Start () {

}
```

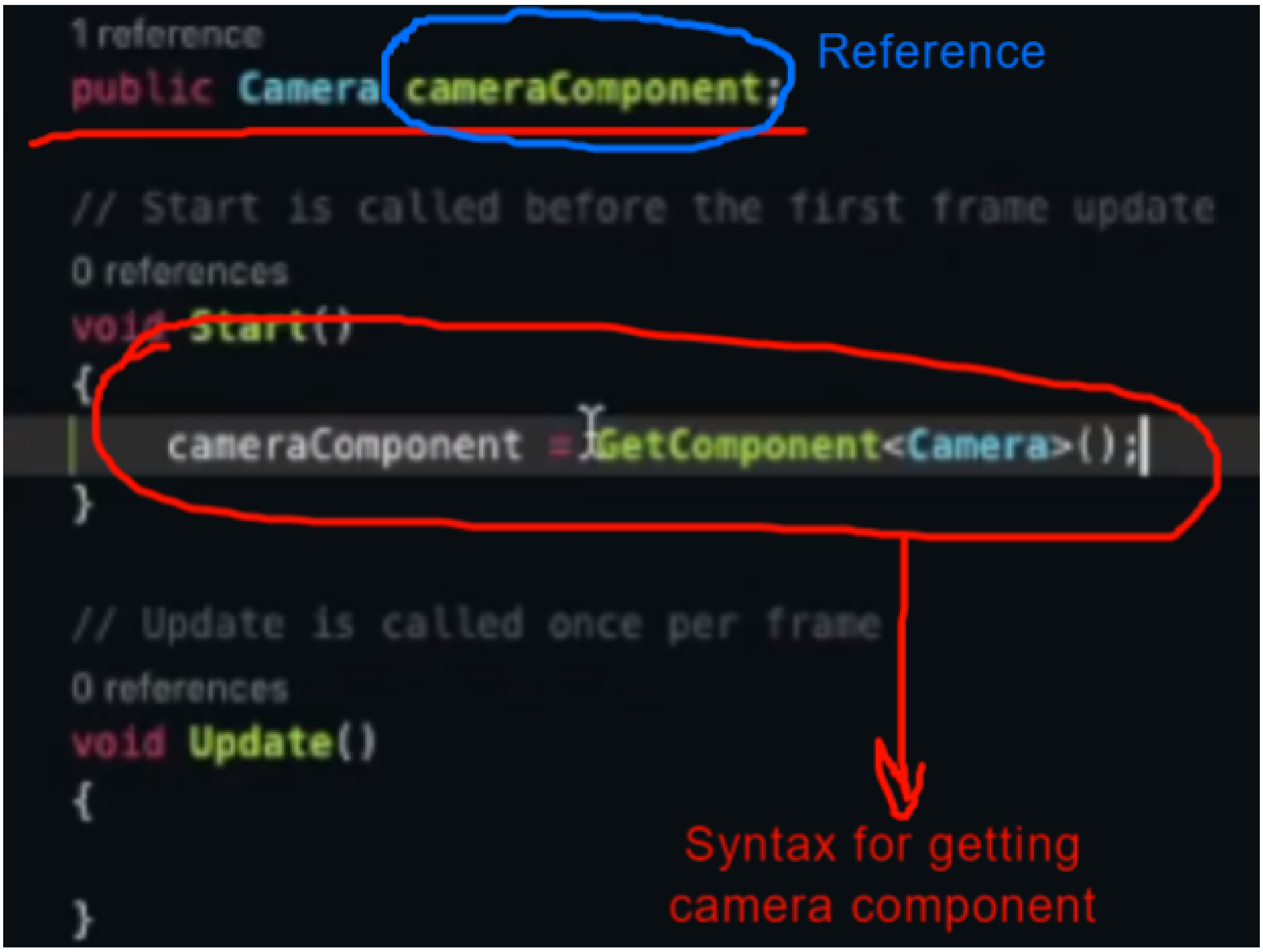


```
    }

    if(Input.GetKey(KeyCode.W))
    {
        transform.position += new Vector3(0, ySpeed, 0);
    }

    if(Input.GetKey(KeyCode.S))
    {
        transform.position -= new Vector3(0, ySpeed, 0);
    }
}
```

Taking inputs from keyboard.

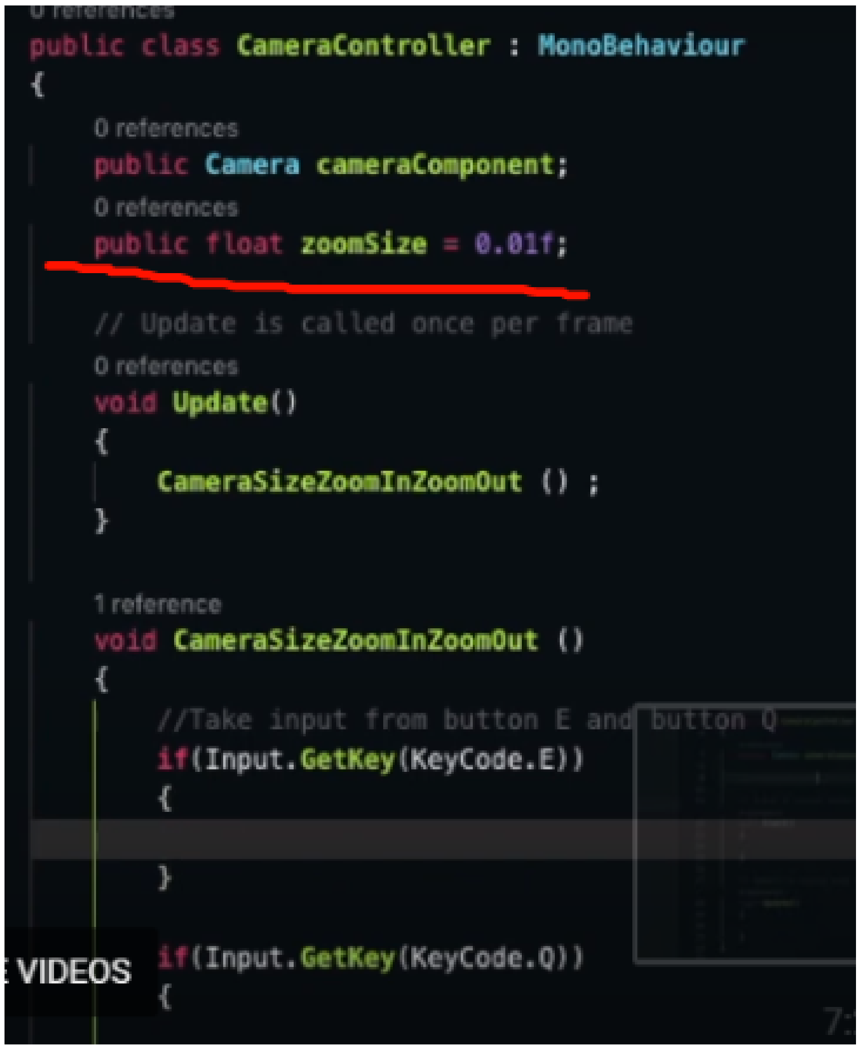
Camera Controller :-

reference value will help us to access all the attributes of a class named as Camera.

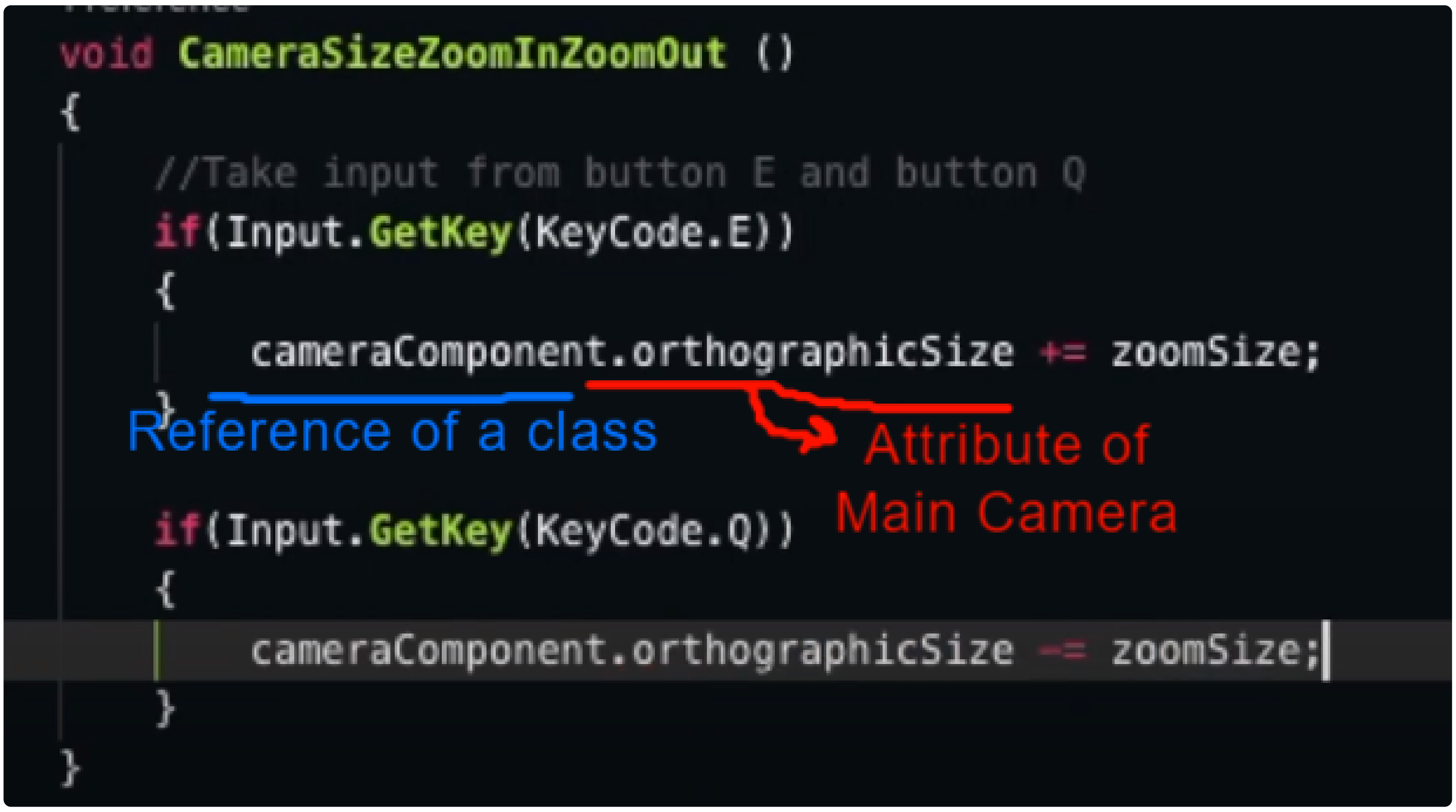


```
1 reference
public Camera cameraComponent;

// Start is called before the first frame update
0 references
void Start()
{
    cameraComponent = GetComponent<Camera>();
}

// Update is called once per frame
0 references
void Update()
{

}
```

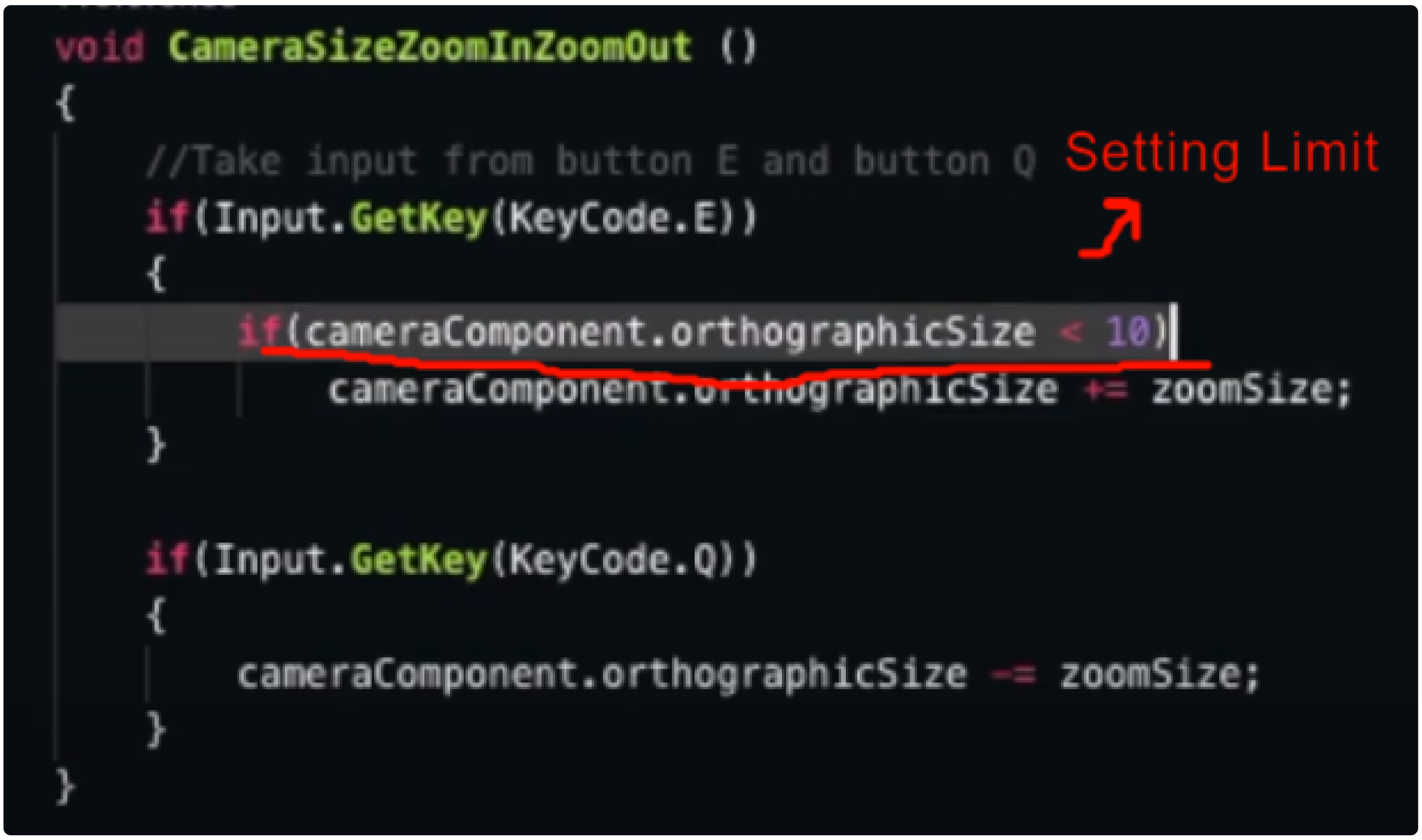
cameraComponent reference/variable created as public.



```
0 references
public class CameraController : MonoBehaviour
{
    0 references
    public Camera cameraComponent;
    0 references
    public float zoomSize = 0.01f;

    // Update is called once per frame
    0 references
    void Update()
    {
        CameraSizeZoomInZoomOut () ;
    }

    1 reference
    void CameraSizeZoomInZoomOut ()
    {
        //Take input from button E and button Q
        if(Input.GetKey(KeyCode.E))
        {

        }

        if(Input.GetKey(KeyCode.Q))
        {
```

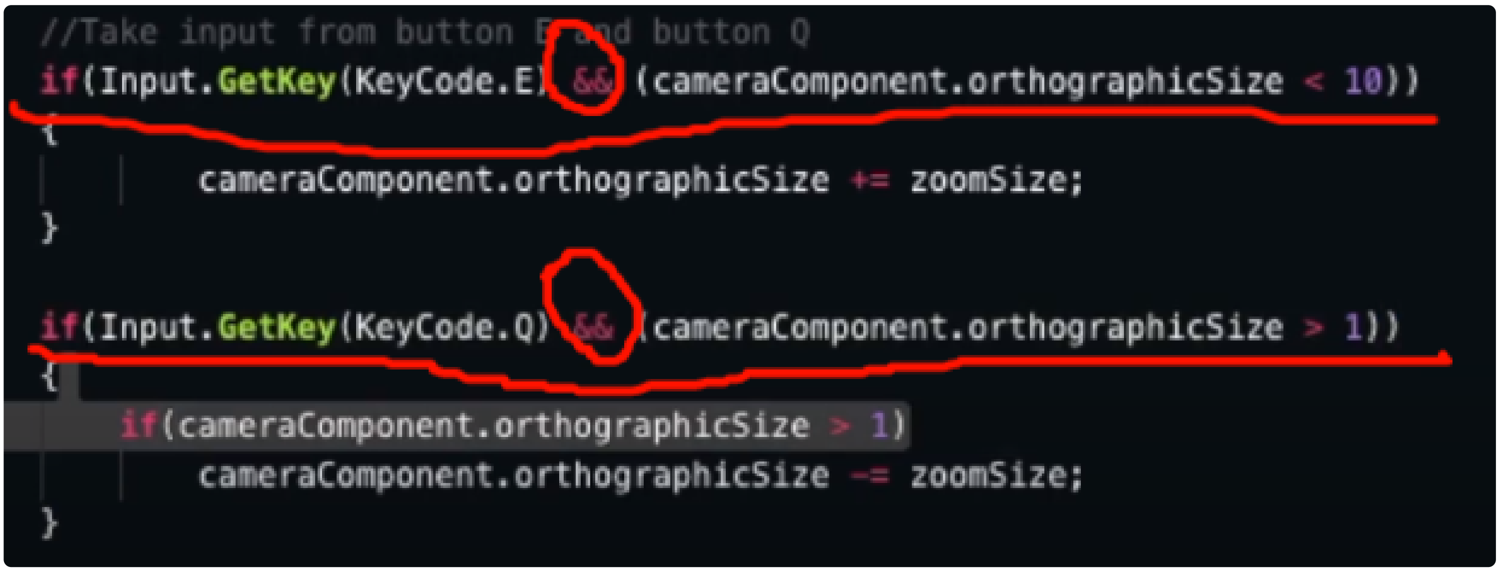


```
void CameraSizeZoomInZoomOut ()
{
    //Take input from button E and button Q
    if(Input.GetKey(KeyCode.E))
    {
        cameraComponent.orthographicSize += zoomSize;
    }

    if(Input.GetKey(KeyCode.Q))
    {
        cameraComponent.orthographicSize -= zoomSize;
    }
}
```



```
void CameraSizeZoomInZoomOut ()
{
    //Take input from button E and button Q
    if(Input.GetKey(KeyCode.E))
    {
        if(cameraComponent.orthographicSize < 10)
            cameraComponent.orthographicSize += zoomSize;
    }

    if(Input.GetKey(KeyCode.Q))
    {
        cameraComponent.orthographicSize -= zoomSize;
    }
}
```

Another way of doing above thing as shown below



```
//Take input from button E and button Q
if(Input.GetKey(KeyCode.E) && (cameraComponent.orthographicSize < 10))
{
    cameraComponent.orthographicSize += zoomSize;
}

if(Input.GetKey(KeyCode.Q) && (cameraComponent.orthographicSize > 1))
{
    if(cameraComponent.orthographicSize > 1)
    cameraComponent.orthographicSize -= zoomSize;
}
```