

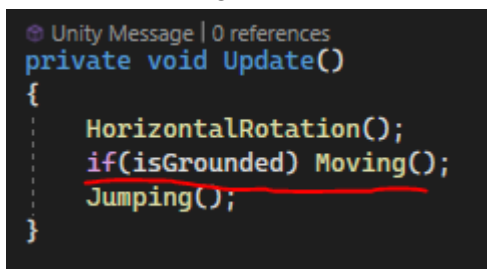
FP Player Movements

31/01/2023

Everything was going fine with this script, but I ran into a problem with the jumping. The raycast was working fine but I noticed I could move the player up in the air. Although some games do allow for total/partial air movement I wanted mine to be limited. The player should not be able to move while in the air, and should retain his velocity he was moving before jumping. I tried to fix it by allowing the player to move only if the `IsGrounded` condition was true, I did this inside the `Moving()` function. This did work but if my player was moving before jumping he would stop to perform the jump, whereas I'd like the player to retain his moving velocity to allow him to perform a jump forward, backwards etc...

07/02/2023

I've finally managed to obtain the result I was looking for: instead of checking `IsGrounded` inside the `Moving()` function I had to check it in the `Update` function.



```
Unity Message | 0 references
private void Update()
{
    HorizontalRotation();
    if(isGrounded) Moving();
    Jumping();
}
```

Destructible Environments

14/02/2023

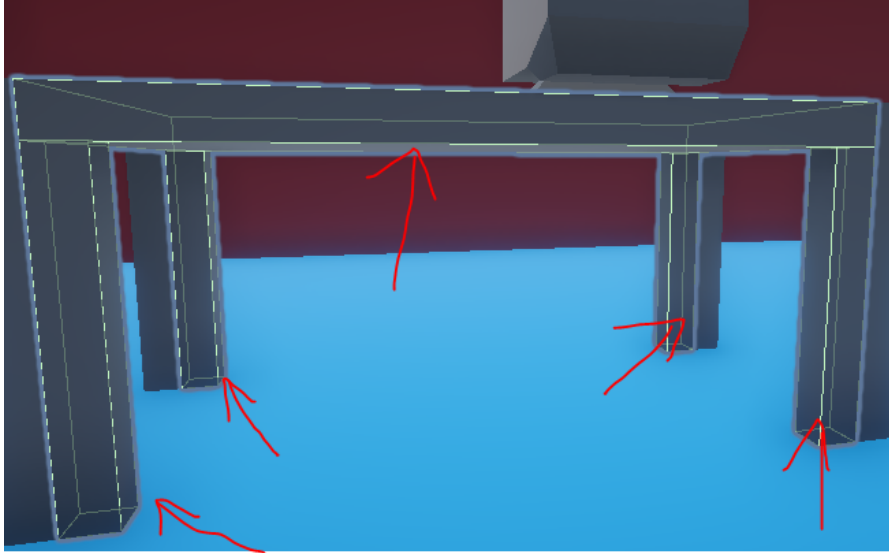
The script seems to be working fine but I encountered a problem:

"Non-convex MeshCollider with non-kinematic Rigidbody is no longer supported since Unity 5. If you want to use a non-convex mesh either make the Rigidbody kinematic or remove the Rigidbody component. Scene hierarchy path"

To solve this problem I simply tick "Convex" on each mesh collider of each destructible game object. The issue doesn't pop up anymore, but I've noticed a strange bug. When I destroy a table for example, the code will instantiate 2 broken mesh of the table instead of one.

21/02/2023

I've finally uncovered the issue: because I'm using a convex mesh collider, unity creates a collision for each component of the mesh, so the table will have a collider and each leg of the table will have a collider too.



This means that when the Player attacks the baseball collider will sometimes hit both the table top collider and the leg collider, it will instantiate the broken mesh twice.

I fixed this by creating a condition called `isDestroyed`, this condition becomes true when the baseball bat collides with the table, and if the condition is true then it will delete the game object and instantiate the broken mesh. This means that only one hit will actually be registered.

```
public class EquipmentDamage : MonoBehaviour
{
    public GameObject destroyedVersion;
    public bool isDestroyed = false;
    2 references
    public void DestroyObject()
    {
        isDestroyed = true;
    }
    Unity Message | 0 references
    private void FixedUpdate()
    {
        if (isDestroyed)
        {
            Destroy(gameObject);
            Instantiate(destroyedVersion, transform.position, transform.rotation);
        }
    }
    Unity Message | 0 references
    private void OnCollisionEnter(Collision collision)
    {
        if (collision.collider.tag == "Range Weapon")
        {
            DestroyObject();
            Debug.Log(collision.transform.name);
        }
    }
}
```

Pick Up System

27/02/2023

Everything is working as it should be, but there's a problem when the player picks up a weapon. The weapon is parented to the Weapon Container which is a parent of the camera. The weapon follows the player's direction, but not its "distance". The weapon will rotate a move with the player, but it will move further away from them.

28/02/2023

I've identified the issue to the rigid body of the weapon, by default its Interpolate is set to "Extrapolate", this is fine when the weapon is not equipped, but it causes the issue discussed above when the player picks it up. To fix this I simply went to the pick up script and said to change the Interpolate to "None", when the weapon is equipped.

```
private void Pickup()
{
    weaponCam.gameObject.SetActive(true);
    IsEquipped = true;
    IsSlotFull = true;

    weapon.transform.position = weaponContainer.transform.position;
    weapon.transform.rotation = weaponContainer.transform.rotation;
    weapon.transform.parent = weaponContainer.transform;

    rb.isKinematic = true;
    rb.interpolation = RigidbodyInterpolation.None;
    coll.isTrigger = true;

    WeaponScript.enabled = true;
}
```

Rigid Body Steps Climb

01/03/2023

The script works, however the player can only move on a staircase only when facing forward. This is because the script shoots a ray from the player's camera to check if it collides with the stairs and then moves the player slightly upwards. Instead of using a raycast the best solution will be to shoot a sphere cast at the player's feet, a sphere is more ideal as it can cover more area.

```

private void Stairs()
{
    origin = transform.position;
    RaycastHit hit;

    if(Physics.SphereCast(origin, sphereRadius, Vector3.down, out hit, maxDistance, stepMask, QueryTriggerInteraction.UseGlobal) && rb.velocity.magnitude >= 1)
    {
        currentHitObject = hit.transform.gameObject;
        currentHitDistance = hit.distance;

        rb.position -= new Vector3(0f, -stepSmooth * Time.deltaTime, 0f);
    }

    else
    {
        currentHitDistance = maxDistance;
        currentHitObject = null;
    }
}

```

I'm also checking the rigid body velocity, that is the speed at what the player is moving. If the velocity is higher than 1 and the spherecast collides with a step then the player will slightly move upwards. The reason why I'm the player's velocity as well is because of this: if the spherecast hits a step and the player just stands next to it without moving, they will start to jitter upwards and downwards, this is because the script has detected a step and it moves the player's upwards.