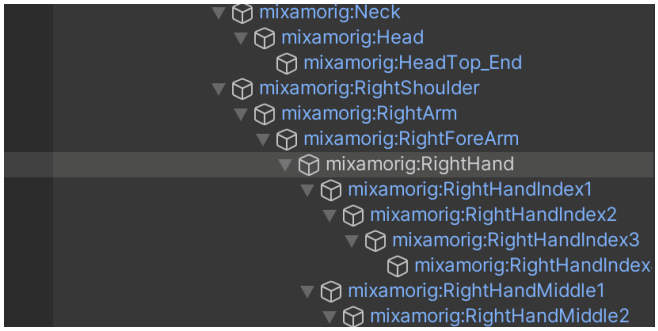
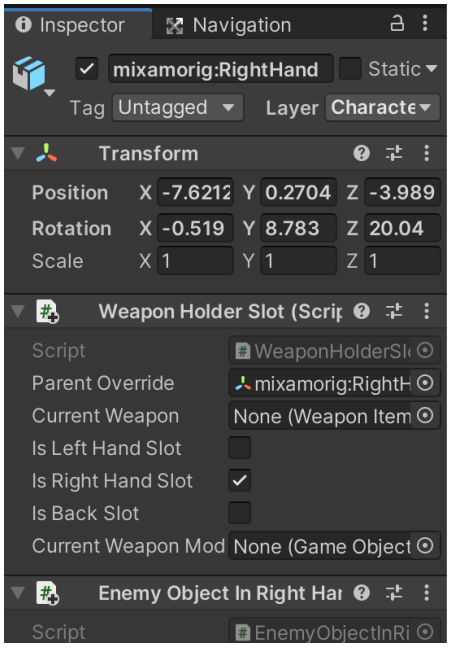
Solving blocking bug of Episode 52 part 2. Receiving damage if the shield is out of a defensive range.

In this tutorial, we solve the bug in the Souls Like Series Ep. 52, letting you receive damage if your shield is out of defensive range. Because of my beginner coding skills, solved the issue in a less clean and simple way, but it works 100%. I hope this tutorial helps you with this blocking issue.

1. In our hierarchy, select the enemy object, then select the right-hand joint. It has to be the joint that was assigned the *WeaponHolderSlot* script. Then create a script called *EnemyObjectInRightHand*.





2. In the script, write the following code:

namespace Name

{

public class EnemyObjectInRightHand : MonoBehaviour

{

[Header("Enemy Game Object for Angle")]

public Transform enemyObjectForAngle;

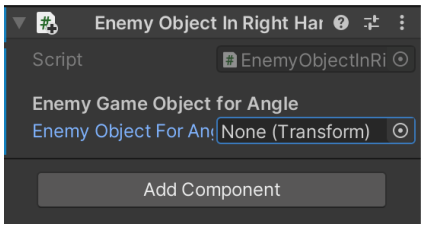
}

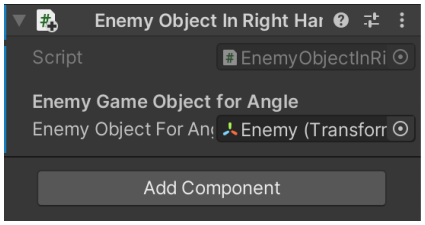
}

In this script, we convert the enemy object to a transform. In order to get the angle information of the object.

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

namespace Basnih
{

    public class EnemyObjectInRightHand : MonoBehaviour
    {

        [Header("Enemy Game Object for Angle")] //NotSGCode
        public Transform enemyObjectForAngle; //NotSGCode
    }
}
```

3. Assign the *Enemy* object In the hierarchy to the *Enemy Object For Angle* slot.





4. Open the *Player Stats* script and write the following variable:

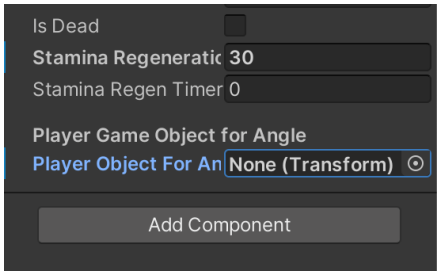
[Header("Player Game Object for Angle")]

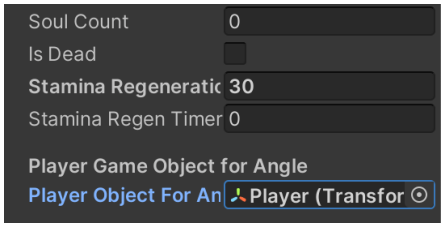
public Transform playerObjectForAngle;

This has the same goal as the previous script, but with the player.

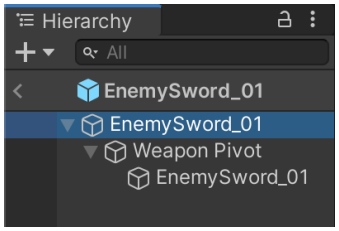
```
public class PlayerStats : CharacterStats
{
    PlayerManager playerManager;
    HealthBar healthBar;
    StaminaBar staminaBar;
    FocusPointBar focusPointsBar;
    PlayerAnimatorManager playerAnimatorManager;

    public float staminaRegenerationAmount = 1;
    public float staminaRegenTimer = 0;

    [Header("Player Game Object for Angle")]
    public Transform playerObjectForAngle;
```

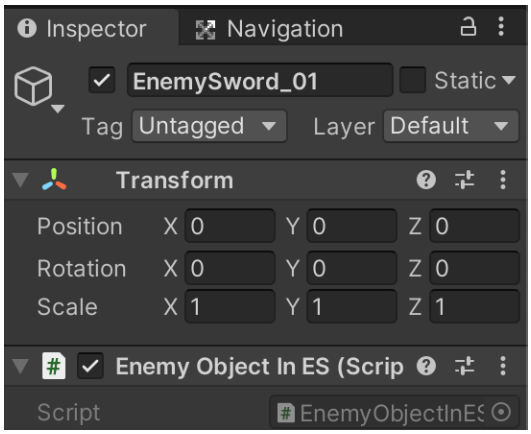
5. Assign the *Player* object In the hierarchy to the *Player Object For Angle* slot.





6. Next, go to our enemy weapon prefab, and in the first object of the hierarchy, create a script called EnemyObjectInES. Write the following code:






```
namespace Name
{
    public class EnemyObjectInES : MonoBehaviour
    {
        EnemyObjectInRightHand enemyObjectInRightHand;
        PlayerStats playerStats;
        public float angleEnemyObject;

        public void Awake()
```

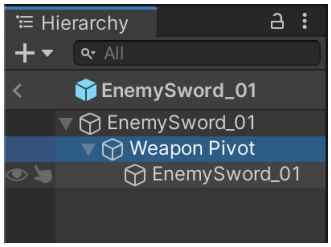
```
    {
        playerStats = FindObjectOfType<PlayerStats>();
    }
    private void Update()
    {
         enemyObjectInRightHand = GetComponentInParent<EnemyObjectInRightHand>();
        angleEnemyObject =
        Vector3.Angle(enemyObjectInRightHand.enemyObjectForAngle.transform.forward,
        playerStats.playerObjectForAngle.transform.forward);
    }
  }
}
```

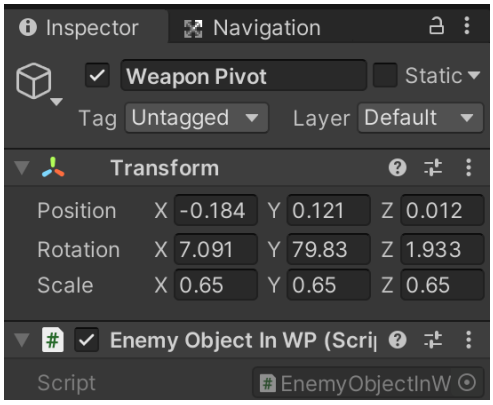
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

namespace Basnih
{

    public class EnemyObjectInES : MonoBehaviour
    {
        EnemyObjectInRightHand enemyObjectInRightHand;
        PlayerStats playerStats;

        public float angleEnemyObject;

        public void Awake()
        {
            playerStats = FindObjectOfType<PlayerStats>();
        }

        private void Update()
        {
            enemyObjectInRightHand = GetComponentInParent<EnemyObjectInRightHand>();
            angleEnemyObject = Vector3.Angle(enemyObjectInRightHand.enemyObjectForAngle.transform.forward, playerStats.playerObjectForAngle.transform.forward);
        }
    }
}
```

In the script, we calculate the angle created between the enemy and the player. If the player is in front of the enemy, it will be 180. If facing away to the enemy will be 0.
The GetComponentInParent<EnemyObjectInRightHand>(); is called in the Update() function because the enemy weapon prefab is not in our scene but is loaded after hitting play, so it has to be updated.

Also is important to use GetComponentInParent instead of FindObjectOfType for the enemyObjectInRightHand variable. The reason is that every enemy game object will get his angle calculation when duplicating them in the scene. FindObjectOfType, instead, will return the angle calculation only to the first enemy game object closer to the player. After that, it will ignore the calculation of an enemy far from the first one.

7. Select the *Weapon Pivo*t object of the enemy weapon prefab and add a new script called *EnemyObjectInWP* with the following code:





namespace Name

{

public class EnemyObjectInWP : MonoBehaviour

{

EnemyObjectInES enemyObjectInES;

public float angleEnemyObjectWP;

public void Awake()

{

enemyObjectInES = GetComponentInParent<EnemyObjectInES>();

}

private void Update()

{

angleEnemyObjectWP = enemyObjectInES.angleEnemyObject;

}

}

}

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

namespace Basnih {

    public class EnemyObjectInWP : MonoBehaviour
    {
        EnemyObjectInES enemyObjectInES;
        public float angleEnemyObjectWP;

        public void Awake()
        {
            enemyObjectInES = GetComponentInParent<EnemyObjectInES>();
        }

        private void Update()
        {
            angleEnemyObjectWP = enemyObjectInES.angleEnemyObject;
        }

    }
}
```

Here we transfer the angle calculation to the game object. To use in the following, which keeps the weapon collider.

8. In the *Damage Collider* script, write the following variables for the code:

```
EnemyObjectInWP enemyObjectInWP;

[Header("Angles for Blocking")]

public float angleEnemyObjectDamageCollider;

public float maxAngleBlock = 30;
```

The maxAngleBlock; variable is the maximum angle range that the shield has to block the enemy attack.

9. Then write in the *Awake()* function the following code:

```
enemyObjectInWP = GetComponentInParent<EnemyObjectInWP>();
```

Create an Update() function with the following code:

```
private void Update()

{

   angleEnemyObjectDamageCollider = enemyObjectInWP.angleEnemyObjectWP;

}
```

Here we obtain the angle calculation from a previous game object as a float variable.

10. In the *OnTriggerEnter()* function, after the *shield* variable, write this code:

```
float angleForBlocking = 180f - angleEnemyObjectDamageCollider;
```

Here we create angleForBlocking to compare it with the maxAngleBlock, to determine if our player can block or not the weapon attack.

11. On the *else if (shield != null && enemyCharacterManager.isBlocking)* we add:

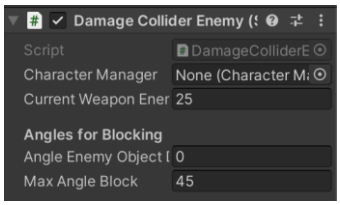
```
&& angleForBlocking <= maxAngleBlock
```

For example, if angleForBlocking is 0, it is less or equal to maxAngleBlock, which value is 30, and the player will block the attack.

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

namespace Basnih
{
    public class DamageColliderEnemy : MonoBehaviour
    {
        public CharacterManager characterManager;
        Collider damageColliderenemy;

        public int currentWeaponEnemyDamage = 25;

        EnemyObjectInWP enemyObjectInWP;
        [Header("Angles for Blocking")]
        public float angleEnemyObjectDamageCollider;
        public float maxAngleBlock = 30;

        private void Awake()
        {
            damageColliderenemy = GetComponent<Collider>();
            damageColliderenemy.gameObject.SetActive(true);
            damageColliderenemy.isTrigger = true;
            damageColliderenemy.enabled = false;

            enemyObjectInWP = GetComponentInParent<EnemyObjectInWP>();
        }

        private void Update() //not SG code
        {
            angleEnemyObjectDamageCollider = enemyObjectInWP.angleEnemyObjectWP;
        }
```

```
        private void OnTriggerEnter(Collider collision)
        {

            if (collision.tag == "Player")
            {
                PlayerStats playerStats = collision.GetComponent<PlayerStats>();
                CharacterManager enemyCharacterManager = collision.GetComponent<CharacterManager>();
                BlockingCollider shield = collision.transform.GetComponentInChildren<BlockingCollider>();

                float angleForBlocking = 180f - angleEnemyObjectDamageCollider;

                if (enemyCharacterManager != null)
                {
                    if (enemyCharacterManager.isParrying)
                    {
                        Debug.Log("Parried");
                        characterManager.GetComponentInChildren<AnimatorManager>().PlayTargetAnimation("Parried", true);
                        return;
                    }
                    else if (shield != null && enemyCharacterManager.isBlocking && angleForBlocking <= maxAngleBlock)
                    {
                        float physicalDamageAfterBlock =
                            currentWeaponEnemyDamage - (currentWeaponEnemyDamage * shield.blockingPhysicalDamageAbsorption) / 100;

                        if (playerStats != null)
                        {
                            playerStats.TakeDamage(Mathf.RoundToInt(physicalDamageAfterBlock), "Block Guard");
                            return;
                        }
                    }
                }
```

12. You can change the *maxAngleBlock* in the inspector to your taste.



Issues with the tutorial:

1. You have an error like this:

   Assets/Scripts/Enemy/EnemyObjectScript/EnemyObjectInWP.cs(9,9): error CS0246: The type or namespace name 'EnemyObjectInES' could not be found (are you missing a using directive or an assembly reference?)

   Make sure to change *namespace Name{}* to the name used in your script.

   Tutorial by: Basnih (@_basnih)

   Credits:

   This tutorial was made with the help of the logic of the following tutorial: Unity RPG Series: Blocking by Alvin Roe https://www.youtube.com/watch?v=L9mPkOtY6TQ&t=1s