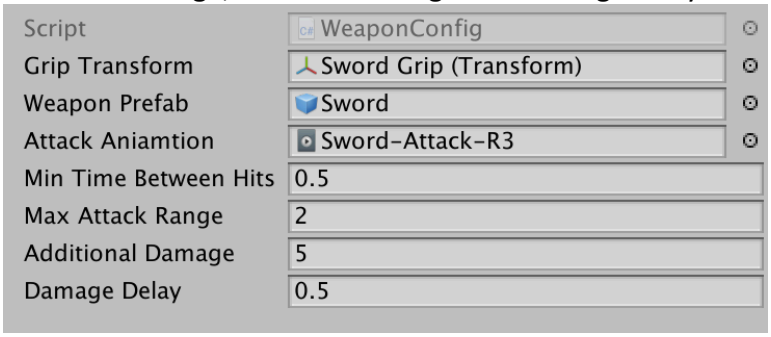
Assignment #4:
Question: How to design the Weapon system for ARPG Game?
Key Words: weaponConfig, Sword, Hoe, weaponGrip

Firstly, I need to create a weaponConfig class inherit from ScriptableObject. This class is used to store the data of weapon settings, like AttackAnimation, TimeBetweenHits, MaxAttackRange, aditionalDamage and damageDelay…..



Here is the code:
WeaponConfig.cs

```
using UnityEngine;

namespace RPG.Weapons_N
{
    [CreateAssetMenu(menuName = ("RPG/Weapon"))]
    public class WeaponConfig : ScriptableObject
    {
        public Transform gripTransform;

        [SerializeField] GameObject weaponPrefab;
        [SerializeField] AnimationClip attackAniamtion;
        [SerializeField] float minTimeBetweenHits = .5f;
        [SerializeField] float maxAttackRange = 2f;
        [SerializeField] float additionalDamage = 12f;
        [SerializeField] float damageDelay = 0.5f;

        public float GetMinTimeBetweenHits()
        {
            // TODO Consider whether we take animation time into account
            return minTimeBetweenHits;
        }

        public float GetMaxAttackRange()
        {
            return maxAttackRange;
        }

        public float GetDamageDelay()
        {
            return damageDelay;
        }

        public GameObject GetWeaponPrefab()
        {
            return weaponPrefab;
```

```
        }

        public AnimationClip GetAttackAnimClip()
        {
            RemoveAnimationEvents();
            return attackAniamtion;
        }

        public float GetAdditionalDamage()
        {
            return additionalDamage;
        }

        //So that Asset packs cannot cause crashes
        private void RemoveAnimationEvents()
        {
            attackAniamtion.events = new AnimationEvent[0];
        }
    }
}
```

After that, I used another class named WeaponSystem to restore the functions, like PutWeaponInHand(), SetAttackAnimation(), AttackTargetRepeatedly(), DamageAfterDelay(), CalculateDamage(),

WeaponSystem.cs

```
using System.Collections;
using UnityEngine;

using RPG.Weapons_N;

namespace RPG.Character
{
    public class WeaponSystem : MonoBehaviour
    {
        [SerializeField] float baseDamage = 20f;
        [SerializeField] WeaponConfig weaponInUse;
        [SerializeField] GameObject weaponSocket;

        const string ATTACK_TRIGGER = "Attack";
        const string DEFAULT_ATTACK = "DEFAULT ATTACK";

        GameObject target;
        GameObject weaponObject;
        Animator animator;
        Character character;
        float laslHitTime = 0f;

        void Start()
        {
            animator = GetComponent<Animator>();
            character = GetComponent<Character>();

            PutWeaponInHand(weaponInUse);
            SetAttackAnimation();
        }
```

```
        void Update()
        {
            bool targetIsDead;
            bool targetIsOutOfRange;
            if(target == null)
            {
                targetIsDead = false;
                targetIsOutOfRange = false;
            }
            else
            {
                var targetHealth =
target.GetComponent<HealthSystem>().healthAsPercentage;
                targetIsDead = targetHealth <= Mathf.Epsilon;

                var distanceToTarget = Vector3.Distance(transform.position,
target.transform.position);
                targetIsOutOfRange = distanceToTarget >
weaponInUse.GetMaxAttackRange();
            }

            float characterHealth =
GetComponent<HealthSystem>().healthAsPercentage;
            bool characterIsDead = (characterHealth <= Mathf.Epsilon);

            if (characterIsDead || targetIsOutOfRange || targetIsDead)
            {
                StopAllCoroutines();
            }
        }

        public void PutWeaponInHand(WeaponConfig weaponToUse)
        {
            weaponInUse = weaponToUse;
            var weaponPrefab = weaponToUse.GetWeaponPrefab();
            Destroy(weaponObject); // empty hands
            weaponObject = Instantiate(weaponPrefab, weaponSocket.transform);
            weaponObject.transform.localPosition =
weaponInUse.gripTransform.localPosition;
            weaponObject.transform.localRotation =
weaponInUse.gripTransform.localRotation;
        }

        public WeaponConfig GetCurrentWeaponInUse()
        {
            return weaponInUse;
        }

        private void SetAttackAnimation()
        {
            if(!character.GetOverrideController())
            {
                Debug.Break();
                Debug.LogAssertion("Please provide" + gameObject + "with an
animator override controller!");
            }
            else
            {
```

```
                var animatorOverrideController =
character.GetOverrideController();
                animator.runtimeAnimatorController =
animatorOverrideController;
                animatorOverrideController[DEFAULT_ATTACK] =
weaponInUse.GetAttackAnimClip();
            }
        }

        public void AttackTarget(GameObject targetToAttack)
        {
            target = targetToAttack;
            StartCoroutine(AttackTargetRepeatedly());
        }

        public void StopAttacking()
        {
            animator.StopPlayback();
            StopAllCoroutines();
        }

        IEnumerator AttackTargetRepeatedly()
        {
            bool attackerStillAlive =
GetComponent<HealthSystem>().healthAsPercentage >= Mathf.Epsilon;
            bool targetStillAlive =
target.GetComponent<HealthSystem>().healthAsPercentage >= Mathf.Epsilon;

            while (attackerStillAlive && targetStillAlive)
            {
                float weaponHitPeriod = weaponInUse.GetMinTimeBetweenHits();
                float timeToWait = weaponHitPeriod *
character.GetAnimSpeedMultiplier();

                bool isTimeToHitAgain = Time.time - laslHitTime > timeToWait;
                if(isTimeToHitAgain)
                {
                    AttackTargetOnce();
                    laslHitTime = Time.time;
                }
                yield return new WaitForSeconds(timeToWait);
            }
        }

        private void AttackTargetOnce()
        {
            transform.LookAt(target.transform);
            animator.SetTrigger(ATTACK_TRIGGER);
            float damageDelay = weaponInUse.GetDamageDelay();
            SetAttackAnimation();
            StartCoroutine(DamageAfterDelay(damageDelay));
        }

        IEnumerator DamageAfterDelay(float delay)
        {
            yield return new WaitForSecondsRealtime(delay);
            target.GetComponent<HealthSystem>().TakeDamage(CalculateDamage());
        }
```

```
        private void AttackTarget()
        {
            if (Time.time - laslHitTime > weaponInUse.GetMinTimeBetweenHits())
            {
                SetAttackAnimation();
                animator.SetTrigger(ATTACK_TRIGGER);
                laslHitTime = Time.time;
            }
        }

        private float CalculateDamage()
        {
            //bool isCriticalHit = UnityEngine.Random.Range(0f, 1.0f) <=
criticalHitChance;
            //float damageBeforeCritical = baseDamage +
weaponInUse.GetAdditionalDamage();
            return baseDamage + weaponInUse.GetAdditionalDamage();
            //if (isCriticalHit)
            //{
            //    cirticalHitParticale.Play();
            //    return damageBeforeCritical * criticalHitMutiplier;
            //}
            //else
            //{
            //    return damageBeforeCritical;
            //}
        }

    }
}
```

Another thing is to set a weapon pick up system, here is the code:
WeaponPickUp.cs

```
using UnityEngine;
using RPG.Character;

namespace RPG.Weapons_N
{
    [ExecuteInEditMode]
    public class WeaponPickup : MonoBehaviour
    {
        [SerializeField] WeaponConfig weaponConfig;
        [SerializeField] AudioClip pickUpSFX;

        AudioSource audioSource;
        // Start is called before the first frame update
        void Start()
        {
            audioSource = GetComponent<AudioSource>();
        }

        // Update is called once per frame
        void Update()
        {
            if (!Application.isPlaying)
            {
                DestroyChildren();
```

```
                InstantiateWeapon();
            }
        }

        private void DestroyChildren()
        {
            foreach (Transform child in transform)
            {
                DestroyImmediate(child.gameObject);
            }
        }

        private void InstantiateWeapon()
        {
            var weapon = weaponConfig.GetWeaponPrefab();
            weapon.transform.position = Vector3.zero;
            Instantiate(weapon, gameObject.transform);
        }

        private void OnTriggerEnter(Collider other)
        {

FindObjectOfType<PlayerControl>().GetComponent<WeaponSystem>().PutWeaponInHand(
weaponConfig);
            audioSource.PlayOneShot(pickUpSFX);
        }

    }
}
```