```
using UnityEngine;

public class BossController : MonoBehaviour
{
    public static BossController instance;

    public BossAction[] actions;
    private int currentAction;
    private float actionCounter;

    private float shotCounter;
    private Vector2 moveDirection;
    public Rigidbody2D theRB;

    public int currentHealth;

    public GameObject deathEffect, hitEffect;
    public GameObject levelExit;

    public BossSequence[] sequences;
    public int currentSequence;

    private void Awake()
    {
        instance = this;
    }

    // Start is called before the first frame update
    void Start()
    {
        actions = sequences[currentSequence].actions;

        actionCounter = actions[currentAction].actionLength;

        UIController.instance.bossHealthBar.maxValue = currentHealth;
        UIController.instance.bossHealthBar.value = currentHealth;
    }

    // Update is called once per frame
    void Update()
    {
        if(actionCounter > 0)
        {
            actionCounter -= Time.deltaTime;

            //handle movement
            moveDirection = Vector2.zero;

            if(actions[currentAction].shouldMove)
            {
                if(actions[currentAction].shouldChasePlayer)
                {
                    moveDirection = PlayerController.instance.transform.position -
transform.position;
                    moveDirection.Normalize();
                }

                if(actions[currentAction].moveToPoint && Vector3.Distance(transform.position,
actions[currentAction].pointToMoveTo.position) > .5f)
```

```
                {
                    moveDirection = actions[currentAction].pointToMoveTo.position -
transform.position;
                    moveDirection.Normalize();
                }
            }



            theRB.velocity = moveDirection * actions[currentAction].moveSpeed;


            //handle shooting
            if(actions[currentAction].shouldShoot)
            {
                shotCounter -= Time.deltaTime;
                if(shotCounter <= 0)
                {
                    shotCounter = actions[currentAction].timeBetweenShots;

                    foreach(Transform t in actions[currentAction].shotPoints)
                    {
                        Instantiate(actions[currentAction].itemToShoot, t.position,
t.rotation);
                    }
                }
            }

        } else
        {
            currentAction++;
            if(currentAction >= actions.Length)
            {
                currentAction = 0;
            }

            actionCounter = actions[currentAction].actionLength;
        }
    }

    public void TakeDamage(int damageAmount)
    {
        currentHealth -= damageAmount;

        if (currentHealth <= 0)
        {
            gameObject.SetActive(false);

            Instantiate(deathEffect, transform.position, transform.rotation);

            if (Vector3.Distance(PlayerController.instance.transform.position,
levelExit.transform.position) < 2f)
            {
                levelExit.transform.position += new Vector3(4f, 0f, 0f);
            }

            levelExit.SetActive(true);
```

```
            UIController.instance.bossHealthBar.gameObject.SetActive(false);
        }
        else
        {
            if(currentHealth <= sequences[currentSequence].endSequenceHealth && currentSequence
< sequences.Length - 1)
            {
                currentSequence++;
                actions = sequences[currentSequence].actions;
                currentAction = 0;
                actionCounter = actions[currentAction].actionLength;
            }
        }

        UIController.instance.bossHealthBar.value = currentHealth;
    }
}

[System.Serializable]
public class BossAction
{
    [Header("Action")]
    public float actionLength;

    public bool shouldMove;
    public bool shouldChasePlayer;
    public float moveSpeed;
    public bool moveToPoint;
    public Transform pointToMoveTo;

    public bool shouldShoot;
    public GameObject itemToShoot;
    public float timeBetweenShots;
    public Transform[] shotPoints;


}

[System.Serializable]
public class BossSequence
{
    [Header("Sequence")]
    public BossAction[] actions;

    public int endSequenceHealth;
}
```