~\OneDrive - St Paul's Catholic College\Documents\2D Strategy Game - Liberator\Assets\Scripts\Movement\MoveSoldier.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine.UI;
using UnityEditor.Rendering;
using UnityEngine;
using System.Runtime.CompilerServices;
using System.IO;
using Unity.VisualScripting;
//using UnityEngine.UIElements;

public class MoveSoldier : MonoBehaviour
{
    public bool isMoving = false; // enables and disables motion
    public List<Image> path; // images of hexes included in optimal path
    private int totalSteps; // number of hexes included in optimal path
    private int currentStep; // list index defining the current target for movement
    Vector3 targetPos; // coordinates of the hex defined as the current target for movement
    float speedOfAnim = 5f; // determines the speed of the movement
    internal bool lookingToTheRight = true; // determines the rotation of the hero
    SpriteRenderer SoldierSprite; // SpriteRenderer component reference
    [SerializeField] SpriteRenderer weapon; // the rifle the soldier carries

    public GameObject bulletPrefab;

    Controller battleController;

    void Start()
    {
        SoldierSprite = GetComponent<SpriteRenderer>(); // getting the soldier in the game
        battleController = FindObjectOfType<Controller>(); // getting the battle controller script
    }

    void Update()
    {
        if (isMoving) // checks if the player is moving along the hexes
        {
            HeroIsMoving(); // this function initialises all the code and other functions to move the soldier
        }
        if (Input.GetKeyDown(KeyCode.X) && isMoving == false) // detects if the player pressed X
        {
            var bullet = Instantiate(bulletPrefab, weapon.transform.position, weapon.transform.rotation); // creates a new bullet from a bullet prefab
            // positions it in the front of the player's weapon(gun)
        }
        if (Input.GetKeyDown(KeyCode.R) && isMoving == false)
        {
            SoldierSprite.flipX = !SoldierSprite.flipX; // rotates a sprite of the soldier
            weapon.flipX = !weapon.flipX;
            lookingToTheRight = !lookingToTheRight; // sets the opposite value for a variable
        }
    }

    public void StartsMoving()
    {
        battleController.CleanField();
        currentStep = 0; // update the variable value to start with the first hex of the optimal path
        totalSteps = path.Count - 1; // numer of hexes included in optimal path, used as an index
        isMoving = true; // enables movement
        ResetTargetPos(); // switches the elements of the path list defining the next step

    }

    private void ResetTargetPos()
    {
        // defines next step changing the value of currentStep variable
        targetPos = new Vector3(path[currentStep].transform.position.x, path[currentStep].transform.position.y, transform.position.z);
        ControlDirection(targetPos);
    }

    private void ManageSteps() // changes the value of the currentStep variable depending on the distance to the current target
    {
        if (Vector3.Distance(transform.position, targetPos)< 0.1f && currentStep < totalSteps) // compares the coordinates of the soldier's current position
                                                                                                // and the disance to the current target position
        {
            currentStep++; // adds one to the value of the CurrentStep variable
            ResetTargetPos(); // sets a new target hex
        }
        else if (Vector3.Distance(transform.position, targetPos) < 0.1f)
        {
            StopsMoving(); // stops movement if the soldier reaches the end point of movement
        }
    }

    private void StopsMoving()
    {
       UpdatePath(); // updates the path when the soldier has finished his movement
    }

    private void HeroIsMoving()
    {
        // moves a soldier in the given coordinates
        // transform position is the current position of a soldier, targetposition is a hex defined as a current hex for movement
        transform.position = Vector3.MoveTowards(transform.position, targetPos, speedOfAnim * Time.deltaTime);
        ManageSteps();
    }

   private void UpdatePath()
    {
        isMoving = !isMoving; // reverses the value of a variable
        transform.parent = path[currentStep].transform; // setting the parent of the soldier to the new hex he just moved on
        HexData startingHex = Controller.soldier.GetComponentInParent<HexData>(); // getting the startign hex
        startingHex.DefineMeAsStartingHex(); // giving the new starting hex the characteristics of it
        IAdjacentFinder adjFinder = new PositionsForSoldier();
        AvailablePos Soldier = FindObjectOfType<AvailablePos>();
        int stepsLimit = Controller.soldier.steps; // the amount of steps the soldier can take. Right now 2.
        Soldier.GetAvailablePositions(Soldier.GetComponentInParent<HexData>(), stepsLimit, adjFinder); // getting the available positions of the soldier around him
    }

    internal void ControlDirection(Vector3 targetPos)
    {
        // compares the coordinates of the soldier and the coordinates of the target hex
        // rotates the soldier and his rifle if necessary

        if (transform.position.x > targetPos.x && lookingToTheRight || transform.position.x < targetPos.x && !lookingToTheRight)
        {
            SoldierSprite.flipX = !SoldierSprite.flipX; // rotates a sprite of the soldier
            lookingToTheRight = !lookingToTheRight; // sets the opposite value for a variable
            weapon.flipX = !weapon.flipX;
        }
    }
}

```