```
using UnityEngine;
using System.Collections;

public class MoveGUI : MonoBehaviour {


    //moveChar is necesary for the actual movement
    private MoveChar moveChar;

    //at wich point does the move is recognized as such
    private float smooth = Screen.width/18;
    private float runSmooth = Screen.width/6;
    private bool canMove = true;

    void Start () {
        moveChar = GameObject.FindGameObjectWithTag("Player").GetComponent<MoveChar>();
        if(!back)
            back = GameObject.Find("IniMovement");
        if(!actual)
            actual = GameObject.Find("MovedMovement");
    }

    // this must be a global variable, as it must retain its value through time
    private Vector2 initMove = Vector2.zero;

    /*  GetMove recieves a Direction by reference, and saves in that variable the new Direction.
            It could be a good idea to make it return the direction, but maybe we'll need to pass
            more arguments in the future (and I thought this could be a nice demostration of
            reference arguments).
    */
    private Direction dirAux;
    private bool isRunning;
    public void StopMove(){
        ShowMovement(Vector2.zero,Vector2.zero);
        canMove=false;
        moveChar.Move(Direction.none,false);
    }
    public void EnableMove(){
        canMove=true;
    }
    public void GetMove(ref Direction dir){
        Vector2 pointer = initMove;
        //Direction dirAux = Direction.none;
        dirAux = Direction.none;
        //bool isRunning = false;
        isRunning = false;

        if(Input.GetMouseButtonDown(0)){
            initMove = Input.mousePosition;
        }
        if(Input.GetMouseButton(0)){
            pointer = (Vector2) Input.mousePosition;
            pointer -= initMove;
        }
        if(Input.GetMouseButtonUp(0)){
            initMove = Vector2.zero;
            pointer = Vector2.zero;
        }

        ShowMovement(initMove,pointer);
        if(Mathf.Abs(pointer.x) < Mathf.Abs(pointer.y)){
            isRunning = (Mathf.Abs(pointer.y) > runSmooth);
            if(pointer.y < -smooth){
                dirAux = Direction.down;
            }
            if(pointer.y > smooth){
                dirAux = Direction.up;
            }
        }else{
            isRunning = (Mathf.Abs(pointer.x) > runSmooth);
            if(pointer.x < -smooth){
                dirAux = Direction.left;
            }
            if(pointer.x > smooth){
                dirAux = Direction.right;
            }
        }

    }

    void Update(){
        if(canMove)
            moveChar.Move(dirAux, isRunning);
    }

    [SerializeField] GameObject back;
    [SerializeField] GameObject actual;

    private void ShowMovement(Vector2 ini, Vector2 moved){
        moved += ini;
        if(Vector2.Distance(ini,moved) > smooth){
            back.SetActive(true);
            actual.SetActive(true);
            back.transform.position = new Vector2(ini.x/Screen.width,ini.y/Screen.height);
            actual.transform.position = new Vector2(moved.x/Screen.width,moved.y/Screen.height);
        }else{
            back.SetActive(false);
            actual.SetActive(false);
        }
    }
}
```