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

public class Paint : MonoBehaviour, IDragHandler, IDropHandler
{

    private bool isMousePressed;
    public List<Vector3> pointsList;
    private Vector3 mousePos;

    private LineRenderer lineRenderer;
    struct myLine
    {
        public Vector3 StartPoint;
        public Vector3 EndPoint;
    };
    void Awake()
    {
        lineRenderer =
GameObject.Find("LineRender").GetComponent<LineRenderer>();
        lineRenderer.positionCount = 0;
        lineRenderer.startWidth = 0.5f;
        lineRenderer.endWidth = 1f;
        lineRenderer.startColor = Color.green;
        lineRenderer.endColor = Color.grey;
        //lineRenderer.useWorldSpace = false;
    }

        private void Start()
    {
        //lineRenderer.enabled = false;
    }

    public void OnDrag(PointerEventData eventData)
    {
        //Debug.Log("Nazav"+eventData.position);
        //lineRenderer.enabled = true;

        isMousePressed = true;

        lineRenderer.positionCount = 0;
        pointsList.RemoveRange(0, pointsList.Count);
        lineRenderer.startColor = Color.green;
        lineRenderer.endColor = Color.grey;

    }
    public void OnDrop(PointerEventData eventData)
    {
        //Debug.Log("Vidpustiv" + eventData.position);
        isMousePressed = false;
        //lineRenderer.enabled = false;
    }


    private void Update()
    {
        //if (Input.GetMouseButtonDown(0))
        //{
        //    isMousePressed = true;
```

```
        //}
        //if (Input.GetMouseButtonUp(0))
        //{
        //    isMousePressed = false;
        //}



        if (isMousePressed)
        {
            mousePos =
Camera.main.ScreenToWorldPoint(Input.mousePosition);
            mousePos.z = 0;
            if (!pointsList.Contains(mousePos))
            {
                pointsList.Add(mousePos);
                lineRenderer.positionCount = pointsList.Count;
                lineRenderer.SetPosition(pointsList.Count - 1,
(Vector3)pointsList[pointsList.Count - 1]);
                if (isLineCollide())
                {
                    isMousePressed = false;
                    lineRenderer.startColor = Color.red;
                    lineRenderer.endColor = Color.red;
                }
            }
        }
    }
    private bool isLineCollide()
    {
        if (pointsList.Count < 2)
            return false;
        int TotalLines = pointsList.Count - 1;
        myLine[] lines = new myLine[TotalLines];
        if (TotalLines > 1)
        {
            for (int i = 0; i < TotalLines; i++)
            {
                lines[i].StartPoint = (Vector3)pointsList[i];
                lines[i].EndPoint = (Vector3)pointsList[i + 1];
            }
        }
        for (int i = 0; i < TotalLines - 1; i++)
        {
            myLine currentLine;
            currentLine.StartPoint = (Vector3)pointsList[pointsList.Count
- 2];
            currentLine.EndPoint = (Vector3)pointsList[pointsList.Count -
1];
            if (isLinesIntersect(lines[i], currentLine))
                return true;
        }
        return false;
    }
    private bool checkPoints(Vector3 pointA, Vector3 pointB)
    {
        return (pointA.x == pointB.x && pointA.y == pointB.y);
    }
    private bool isLinesIntersect(myLine L1, myLine L2)
    {
        if (checkPoints(L1.StartPoint, L2.StartPoint) ||
```

```
            checkPoints(L1.StartPoint, L2.EndPoint) ||
            checkPoints(L1.EndPoint, L2.StartPoint) ||
            checkPoints(L1.EndPoint, L2.EndPoint))
            return false;

        return ((Mathf.Max(L1.StartPoint.x, L1.EndPoint.x) >=
Mathf.Min(L2.StartPoint.x, L2.EndPoint.x)) &&
            (Mathf.Max(L2.StartPoint.x, L2.EndPoint.x) >=
Mathf.Min(L1.StartPoint.x, L1.EndPoint.x)) &&
            (Mathf.Max(L1.StartPoint.y, L1.EndPoint.y) >=
Mathf.Min(L2.StartPoint.y, L2.EndPoint.y)) &&
            (Mathf.Max(L2.StartPoint.y, L2.EndPoint.y) >=
Mathf.Min(L1.StartPoint.y, L1.EndPoint.y))
               );
    }


}
```