```
//Матеріали взяв з сайту           https://null-code.ru/scripts/115-
prostoy-primer-risovaniya-v-unity.html
//На сцені у нас дві камери, перша, дивиться на «чистий аркуш»,
//тобто об'єкт з матеріалом, але без текстури, так само, перед цією
камерою створюються клони кисті, з яких утворюється малюнок,
//а камера в свою чергу передає зображення в рендер текстуру. Ця текстура
причеплено на матеріал іншого об'єкта, який знаходиться перед другою
камерою.
//Тобто коли ми водимо мишкою по полотну(де рендер текстура), то
створюємо клони пензлика там, де знаходиться перша камера, а вона
повертає нам текстуру з малюнком.
//Скрипт вішаємо на головну камеру
using UnityEngine;
using System.Collections;
using System.Collections.Generic;
public class Paint : MonoBehaviour
{
      public SpriteRenderer kistochka; // спрайт кісточкі
      public Color colorKistochky = Color.red;//Колір кісточки
      [Range(0.1f, 0.2f)] public float sizeKistochky = 0.1f;//Розмір
кісточки
      public Camera cameraRT; // дублююча камера
      private int sizeRT = 1024; // Зорзмір текстури
      public MeshRenderer canvasObject; // Головний Quad по якому і
будемо малювати
      public MeshRenderer planeRT; // дуюлюючий Quad на якому
відображається малювання

    private RenderTexture renderTexture;
      private Vector3 position;
      private int counter = 0, maxCount = 1000; // счетчик и макс.
количество объектов
      private bool isSave;
    public static List<Transform> positionCreate = new
List<Transform>();//масив для точок по яким малювати
    void Awake()
      {
            position.z = kistochka.transform.position.z;//стала позиція по
Z
            Clear();
      }
    private void FixedUpdate()
    {

    }
    void Update()
      {
            if(Input.GetMouseButton(0) && !isSave)//якщо ЛКМ нажата і не
було збережено
            {
                  kistochka.gameObject.SetActive(true);
                  Draw();//малювання
            }
            else if(Input.GetMouseButtonUp(0))//Якщо ЛКМ віджата
            {
             //Save();
             Clear();//очистка
            }
            else if(isSave)//якщо щось намалювали можна зберегти але
оскільки очищаєм при віджиманні кнопки то сюди доходити не буде
            {
```

```
            Save();
            kistochka.gameObject.SetActive(false);//Кісточку робимо не
активною, щоб можна було активувати при наступному мазку, і щоб не
заважала
            //Clear();

        }
           else //Якщо нічого не намальовано
           {
                 kistochka.gameObject.SetActive(false);//кісточка не
активна (Brush)
           }
      }
      void Save() //Метод для збереження (але оскільки нам зберегтии
треба лише координати то це можна буде зробити і перед очисткою)
      {
         Debug.Log("Save Done");
         counter = 0;//Обнуляємо оюмеження вводу
            RenderTexture.active = renderTexture;
            Texture2D tex = new Texture2D(renderTexture.width,
renderTexture.height, TextureFormat.RGB24, false);//в змінну tex
записуємо параметри зображення
        tex.ReadPixels(new Rect (0, 0, renderTexture.width,
renderTexture.height), 0, 0);//зчитати данні з зображення
            tex.Apply();//Фактично застосувати всі попередні зміни
SetPixel та SetPixels.
        RenderTexture.active = null;
            planeRT.material.mainTexture = tex;//присвоюємо зчитані данні
зображення до дублюючого Quad

//////////////////////////////////////////////////////////////////////////
///////////////вивільнення памяті
        foreach (Transform child in planeRT.transform)
        {
            Destroy(child.gameObject);
        }

//////////////////////////////////////////////////////////////////////////
///////////////
        isSave = false;//обнуляємо збереження
      }
      void Clear() // очищення
      {
            counter = 0;//Обнуляємо оюмеження вводу

//////////////////////////////////////////////////////////////////////////
///////////////Записуємо в список і видаляємо
        foreach (Transform child in planeRT.transform)
            {
            positionCreate.Add(child);
            Destroy(child.gameObject);
            }

//////////////////////////////////////////////////////////////////////////
///////////////Перевірка заповнення списка
        if (positionCreate.Count != 0)
        {
            foreach (Transform ch in positionCreate)
            {
                //Debug.Log($"Take=\t{ch.transform.position}");
            }
```

```
        }

////////////////////////////////////////////////////////////////////////
///////////////
        Destroy(planeRT.material.mainTexture);
           Destroy(renderTexture);
           renderTexture = new RenderTexture(sizeRT, sizeRT, 24,
RenderTextureFormat.ARGB32);
           cameraRT.targetTexture = renderTexture;
           canvasObject.material.mainTexture = renderTexture;
      }

      void Draw() // малювання (клонування кісточки(Brush))
      {
           RaycastHit hit;
           Ray ray =
Camera.main.ScreenPointToRay(Input.mousePosition);//Зчитуємо з екрану
координати курсору
           if(Physics.Raycast(ray, out hit))
           {
                SpriteRenderer s = Instantiate(kistochka) as
SpriteRenderer;//Створення дублікату кісточки як спрайту
                Vector2 uv = new Vector2(hit.textureCoord.x,
hit.textureCoord.y);//Координата текстури УФ в місці зіткнення.
            position.x = uv.x - cameraRT.orthographicSize;//Напіврозмір
камери в ортографічному режимі.
            position.y = uv.y - cameraRT.orthographicSize;
            s.color = colorKistochky;
                s.transform.localPosition = position;
                s.transform.localScale = Vector3.one * sizeKistochky;
                s.transform.parent = planeRT.transform;
                counter++;
                if(counter > maxCount)
                {
                     isSave = true;//Переключитись на збереження
                }
           }
      }
}
```