1. 编写一个通用的unity3d的对象池类

using System.Collections.Generic;

public class ObjectPool<T> where T : class, new()

{

    private Stack<T> \_pool = null;

    public ObjectPool()

    {

        \_pool = new Stack<T>();

    }

    public ObjectPool(int capacity)

    {

        \_pool = new Stack<T>(capacity);

    }

    public T Acquire()

    {

        if (\_pool.Count == 0)

        {

            return new T();

        }

        return \_pool.Pop();

    }

    public void Release(T item)

    {

        if (null == item)

            return;

        if (\_pool.Contains(item))

            return;

        \_pool.Push(item);

    }

    public void Clear()

    {

        \_pool.Clear();

    }

    public int \_Count

    {

        get

        {

            return \_pool.Count;

        }

    }

}

1. 请描述类似LOL中战争迷雾的解决方案，可以用文字结合代码说明

//1.生成表示地图信息，生成一个bool类型二维数组表示障碍物信息，true表示改坐标有障碍物

public void GenerateMapData(float beginX, float beginY, float deltax, float deltay, float heightRange)

{

for (int i = 0; i < m\_Width; i++)

{

for (int j = 0; j < m\_Height; j++)

{

m\_MapData[i, j] = MapInfoUtils.IsObstacle(beginX, beginY, deltax, deltay, heightRange, i, j);

}

}

}

//2.计算视野区域，每次玩家位置变化后需要根据生成的障碍物信息重新计算视野范围.

//射线检测可视范围，将可见区域设置为可见

public void SetAsVisible(int x, int y)

{

m\_MaskCache[y\*m\_Width + x] = 1;

}

//3.生成战争迷雾mask贴图，R通道记录叠加的每次可见区域，G通道记录当前可见区域，B通道缓存上一次更新的可见区域

public bool GenerateOrRefreshTexture()

{

if (m\_MaskTexture == null)

{

m\_MaskTexture = new Texture2D(m\_Width, m\_Height, TextureFormat.RGB24, false);

tex.wrapMode = TextureWrapMode.Clamp;

}

for (int i = 0; i < m\_MaskTexture.width; i++)

{

for (int j = 0; j < m\_MaskTexture.height; j++)

{

bool isVisible = m\_MaskCache[i, j] == 1;

Color origin = isNew ? Color.black : m\_MaskTexture.GetPixel(i, j);

origin.r = Mathf.Clamp01(origin.r + origin.g);

origin.b = origin.g;

origin.g = isVisible ? 1 : 0;

m\_MaskTexture.SetPixel(i, j, origin);

m\_Visible[i, j] = (byte) (isVisible ? 1 : 0);

m\_MaskCache[i, j] = 0;

}

}

m\_MaskTexture.SetPixels(m\_ColorBuffer);

m\_MaskTexture.Apply();

m\_UpdateMark = UpdateMark.None;

return true;

}

//4.投影战争迷雾,将生成的战争迷雾mask贴图投射到场景，直接采用后处理的方式。

//生成投影矩阵并将矩阵参数设置到shader，(xSize迷雾区域宽度 zSize迷雾区域高度)

Matrix4x4 m\_WorldToProjector = default(Matrix4x4);

m\_WorldToProjector.m00 = 1.0f/xSize;

m\_WorldToProjector.m03 = -1.0f/xSize\*position.x + 0.5f;

m\_WorldToProjector.m12 = 1.0f/zSize;

m\_WorldToProjector.m13 = -1.0f/zSize\*position.z + 0.5f;

m\_WorldToProjector.m33 = 1.0f;

m\_EffectMaterial.SetMatrix("internal\_WorldToProjector", m\_WorldToProjector);

//设置生成的mask纹理数据到shader

m\_EffectMaterial.SetTexture("\_FogTex", fogTexture);

//计算interpolatedRay

//绘制

void OnRenderImage(RenderTexture src, RenderTexture dst)

{

RenderTexture.active = dest;

fxMaterial.SetTexture("\_MainTex", src);

GL.PushMatrix();

GL.LoadOrtho();

fxMaterial.SetPass(0);

GL.Begin(GL.QUADS);

GL.MultiTexCoord2(0, 0.0f, 0.0f);

GL.Vertex3(0.0f, 0.0f, 3.0f);

GL.MultiTexCoord2(0, 1.0f, 0.0f);

GL.Vertex3(1.0f, 0.0f, 2.0f);

GL.MultiTexCoord2(0, 1.0f, 1.0f);

GL.Vertex3(1.0f, 1.0f, 1.0f);

GL.MultiTexCoord2(0, 0.0f, 1.0f);

GL.Vertex3(0.0f, 1.0f, 0.0f);

GL.End();

GL.PopMatrix();

}

//战争迷雾Shader编写

…

1. 请用C#编写一段代码，讲下面文件中的数据全部读取出来并加以存储。

[1]

prefab=handsome

scale=1

facecameraoffset=-0.11,-1.9,1.3

[11]

prefab=bird

scale=1

facecameraoffset=-0.11,-1.9,1.3

[51]

prefab=m0001

scale=1

facecameraoffset=-0.11,-1.9,1.3

//======================================================

//从文件中读取数据

//[1]

//prefab = handsome

//scale=1

//facecameraoffset=-0.11,-1.9,1.3

//[11]

//prefab=bird

//scale = 1

//facecameraoffset=-0.11,-1.9,1.3

//[51]

//prefab=m0001

//scale = 1

//facecameraoffset=-0.11,-1.9,1.3

//并将其存在PlayerDataTableManager.playerDataMap中

//======================================================

using System.Collections.Generic;

using System.IO;

using UnityEngine;

/// <summary>

/// 数据类

/// </summary>

public class PlayerTableData

{

    public string playfabName = string.Empty;

    public float scale = 1f;

    public Vector3 faceCameraOffset = Vector3.zero;

}

/// <summary>

/// 玩家数据管理

/// </summary>

public static class PlayerDataTableManager

{

    public static Dictionary<string, PlayerTableData> playerDataMap = new Dictionary<string, PlayerTableData>();

    public static void LoadFromFile(string path)

    {

        string fileContent = "";

        using (FileStream fs = new FileStream(path, FileMode.Open, FileAccess.Read, FileShare.Read))

        {

            using (StreamReader sr = new StreamReader(fs))

            {

                string text = sr.ReadToEnd();

                if (text.Length > 0)

                {

                    fileContent = text;

                }

            }

        }

        int lineIndex = 0;

        FileTextParser parser = new FileTextParser();

        parser.Init(fileContent);

        string keyStr = "";

        while (!parser.IsEOF())

        {

            ++lineIndex;

            string line = parser.ReadLine().Trim();

            if (string.IsNullOrEmpty(line))

                continue;

            if (line[0] == '[')

            {

                bool bFindChar = false;

                keyStr = "";

                for (int i = 0; i < line.Length; ++i)

                {

                    if (line[i] == '[' || line[i] == ']')

                    {

                        bFindChar = true;

                        if (line[i] == ']') // end?

                        {

                            bFindChar = false;

                        }

                        else

                        {

                            continue;

                        }

                    }

                    else if (bFindChar)

                    {

                        keyStr += line[i];

                    }

                }

                continue;

            }

            if (!playerDataMap.ContainsKey(keyStr))

            {

                playerDataMap[keyStr] = new PlayerTableData();

            }

            int bodyIndex = line.IndexOf('=');

            if (bodyIndex <= 0)

            {

                string msg = string.Format("Text body not exist (line:{0}/{1})", lineIndex, path);

                throw new System.Exception(msg);

            }

            if (bodyIndex > 0)

            {

                string key = line.Substring(0, bodyIndex).Trim();

                string body = line.Substring(bodyIndex + 1).Trim(' ', '\t').Replace("\\n", "\n");

                switch (key)

                {

                    case "prefab":

                        playerDataMap[keyStr].playfabName = body;

                        break;

                    case "scale":

                        playerDataMap[keyStr].scale = float.Parse(body);

                        break;

                    case "facecameraoffset":

                        ParseVector3(body, ref playerDataMap[keyStr].faceCameraOffset, ',');

                        break;

                    default:

                        break;

                }

            }

        }

    }

    public static bool ParseVector3(string content, ref Vector3 output, char split = ',')

    {

        string[] values = content.Split(split);

        if (null == values || values.Length < 3)

            return false;

        try

        {

            output.x = float.Parse(values[0], System.Globalization.CultureInfo.InvariantCulture);

            output.y = float.Parse(values[1], System.Globalization.CultureInfo.InvariantCulture);

            output.z = float.Parse(values[2], System.Globalization.CultureInfo.InvariantCulture);

        }

        catch (System.Exception e)

        {

            Debug.LogError(e);

            throw e;

        }

        return true;

    }

}

/// <summary>

/// 通用的文本解释器

/// </summary>

public class FileTextParser

{

    private string \_fileText = null;

    private int \_currIndex = -1;

    private int \_currLine = -1;

    private int \_cutoffLength = 0;

    public void Clear()

    {

        \_fileText = null;

        \_currIndex = -1;

        \_currLine = -1;

        \_cutoffLength = 0;

    }

    /// <summary>

    /// 初始化

    /// </summary>

    /// <param name="fileText"></param>

    public void Init(string fileText)

    {

        \_fileText = fileText;

        \_currIndex = 0;

        \_currLine = 0;

        if (\_fileText.IndexOf('\r') >= 0)

        {

            \_cutoffLength = 1;

        }

        else

        {

            \_cutoffLength = 0;

        }

    }

    /// <summary>

    /// 每次读取一行

    /// </summary>

    /// <returns></returns>

    public string ReadLine()

    {

        if (\_currIndex < 0)

            return null;

        int nextIndex = \_fileText.IndexOf('\n', \_currIndex);

        if (nextIndex > 0)

        {

            int length = nextIndex - \_currIndex - \_cutoffLength;

            if (length < 0)

            {

                Debug.LogError("[FileTextParser]ReadLine Cannot be negative : " + length);

            }

            string line = \_fileText.Substring(\_currIndex, length);

            \_currIndex = nextIndex + 1;

            ++\_currLine;

            return line;

        }

        else if (nextIndex < 0)

        {

            int length = \_fileText.Length - \_currIndex;

            if (length < 0)

            {

                Debug.LogError("[FileTextParser]ReadLine Cannot be negative : " + length);

            }

            string line = \_fileText.Substring(\_currIndex, length);

            \_currIndex = -1;

            return line;

        }

        else

        {

            \_currIndex = -1;

            return null;

        }

    }

    public int GetCurrLine()

    {

        return \_currLine;

    }

    public bool IsEOF()

    {

        if (\_currIndex < 0 || null == \_fileText || \_currIndex >= \_fileText.Length)

            return true;

        return false;

    }

1. 请描述unity3d项目开发过程中多人针对同一场景进行编辑时的冲突问题的解决思路
2. 请描述C#的委托，并以unity3d项目中的应用情景进行举例说明
3. 请从综合的角度（程序、美术、设计）说明如何提高动作游戏的“打击感”