# Unity查缺补漏

1. **Image.fillAmount(): 1表示完全填充，0表示不填充**
2. **参数面板默认是欧拉角选择，使用会产生的死锁，进而跳变。**

**当你需要做动画的时候，你想从（90，0，0）插值到（150，90，90）的时候，写出了这样的代码**

**transform.eulerAngles = Vector3.Lerp(transform.eulerAngles, new Vector3(150, 90, 90)**

**因为y和z会突然跳变，就会形成死锁，甚至卡住电脑。那为什么这个问题现在还存在呢因为Unity设置x在中间就是因为一般x不会旋转90度的，常见的旋转插值是对Camera进行的，而如果绕着X轴旋转90度，则意味着正向上，或是正向下，这两种情况都是非常少见的。**

1. **四元数：数据的范围是-1 - 1，每次旋转需要两个四元数**

**例如Quaternion.AngleAxis(float angle, Vector3 axis)，它可以返回一个绕轴线axis旋转angle角度的四元数变换。我们可以一个Vector3和它进行左乘，就将得到旋转后的Vector3。在Unity里只需要用一个“ \* ”操作符就可以进行四元数对向量的变换操作。如果我们想要进行多个旋转变换，只需要左乘其他四元数变换即可。**[【Unity】旋转的尽头是使用四元数让物体旋转\_unity 四元数 旋转-CSDN博客](https://blog.csdn.net/leoysq/article/details/136737304)

1. **lambda表达式**

**单参数：x=>x\*x // 传入参数x，返回x\*x**

**多参数：（x,u）=>x == u // 传入参数x，u，返回bool值**

|  |  |
| --- | --- |
| 1. **[GetName(Type, Object)](https://learn.microsoft.com/zh-cn/dotnet/api/system.enum.getname?view=net-9.0" \l "system-enum-getname(system-type-system-object))** | **在指定枚举中检索具有指定值的常数的名称。** |

1. **Prioty 点模式调整轴心**
2. **接口**
3. **async 和 await 是 C# 中用于异步编程的关键字，可以让代码以非阻塞的方式执行耗时操作（如网络请求、文件读写、数据库查询等）。**

**using System;**

**using System.Threading.Tasks;**

**public class Example**

**{**

**// 声明一个异步方法，返回 Task<string>**

**public async Task<string> FetchDataAsync()**

**{**

**Console.WriteLine("开始加载数据...");**

**// 模拟耗时操作（如 HTTP 请求）**

**await Task.Delay(2000); // 等待 2 秒**

**Console.WriteLine("数据加载完成！");**

**return "这是返回的数据";**

**}**

**// 调用异步方法**

**public async void StartLoading()**

**{**

**string result = await FetchDataAsync(); // 等待 FetchDataAsync 完成**

**Console.WriteLine($"获取到的数据: {result}");**

**}**

**}**

**开始加载数据...**

**（等待 2 秒）**

**数据加载完成！**

**获取到的数据: 这是返回的数据**

**public async Task<int> GetUserAgeAsync()**

**{**

**await Task.Delay(1000); // 模拟网络请求**

**return 25;**

**}**

**public async Task<string> GetUserNameAsync()**

**{**

**await Task.Delay(1500); // 模拟数据库查询**

**return "张三";**

**}**

**// 同时执行多个异步任务，等待所有完成**

**public async Task LoadUserDataAsync()**

**{**

**Task<int> ageTask = GetUserAgeAsync();**

**Task<string> nameTask = GetUserNameAsync();**

**// 并行执行，等待所有任务完成**

**await Task.WhenAll(ageTask, nameTask);**

**Console.WriteLine($"姓名: {nameTask.Result}, 年龄: {ageTask.Result}");**

**}**

**（等待 1.5 秒，因为最长的任务是 1.5 秒）**

**姓名: 张三, 年龄: 25**

**using UnityEngine;**

**using UnityEngine.SceneManagement;**

**using System.Threading.Tasks;**

**public class SceneLoader : MonoBehaviour**

**{**

**public async void LoadGameSceneAsync()**

**{**

**AsyncOperation asyncLoad = SceneManager.LoadSceneAsync("GameScene");**

**// 等待场景加载完成**

**while (!asyncLoad.isDone)**

**{**

**float progress = Mathf.Clamp01(asyncLoad.progress / 0.9f); // 0~1 范围**

**Debug.Log($"加载进度: {progress \* 100}%");**

**await Task.Yield(); // 每帧检查一次**

**}**

**Debug.Log("场景加载完成！");**

**}**

**}**

**using UnityEngine;**

**using System.Threading.Tasks;**

**public class ResourceLoader : MonoBehaviour**

**{**

**public async Task<Sprite> LoadSpriteAsync(string path)**

**{**

**ResourceRequest request = Resources.LoadAsync<Sprite>(path);**

**// 等待加载完成**

**while (!request.isDone)**

**{**

**await Task.Yield(); // 每帧检查一次**

**}**

**return (Sprite)request.asset;**

**}**

**public async void StartLoading()**

**{**

**Sprite sprite = await LoadSpriteAsync("Characters/Hero");**

**Debug.Log($"加载完成: {sprite.name}");**

**}**

**}**

**public async Task<string> FetchDataWithRetryAsync(int maxRetries = 3)**

**{**

**int retryCount = 0;**

**while (retryCount < maxRetries)**

**{**

**try**

**{**

**// 模拟可能失败的 HTTP 请求**

**await Task.Delay(1000);**

**if (UnityEngine.Random.Range(0, 2) == 0)**

**throw new Exception("网络错误！");**

**return "数据获取成功！";**

**}**

**catch (Exception ex)**

**{**

**retryCount++;**

**Debug.LogError($"第 {retryCount} 次失败: {ex.Message}");**

**await Task.Delay(1000); // 等待 1 秒后重试**

**}**

**}**

**throw new Exception($"尝试 {maxRetries} 次后仍然失败！");**

**}**

**// 调用**

**public async void LoadData()**

**{**

**try**

**{**

**string result = await FetchDataWithRetryAsync();**

**Debug.Log(result);**

**}**

**catch (Exception ex)**

**{**

**Debug.LogError($"最终失败: {ex.Message}");**

**}**

**}**

闭包

using UnityEngine;

using UnityEngine.UI;

public class ClosureExample : MonoBehaviour

{

public Button[] buttons;

void Start()

{

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

{

buttons[i].onClick.AddListener(() => Debug.Log("Button " + i + " clicked"));

}

}

for (int i = 0; i < 3; i++)

{

StartCoroutine(DoSomethingAfterDelay(i));

}

IEnumerator DoSomethingAfterDelay(int index)

{

yield return new WaitForSeconds(1);

Debug.Log(index);

}

1. **生命周期，awake》外部调用方法》start**
2. **A**
3. **A**
4. **A**
5. **d**

}

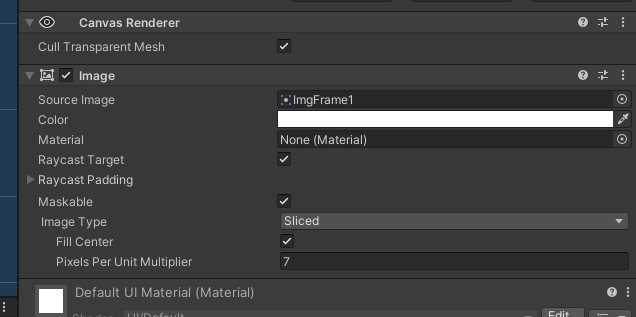
Paras 可变数组

Tmp IMg射线检查

gc释放资源

bindPog 自变化属性

SpriteEditor 修改图片的缩放方式



里氏替换原则]

Api 检查对象是否在渲染内

Ui部分组件会延迟渲染，

Layout group ui

射线队列