1.协同程序实现原理，多线程和协程是一个概念吗？

协程Unity文档解释：

允许您将任务分散到多帧中执行。在Unity中，协程是一种可以暂停执行并将控制权返回给Unity的方法，但随后在它离开的地方继续执行。

在大多数情况下，当您调用一个方法时，它会运行到完成，然后将控制权返回给调用方法，以及任何可选的返回值。这意味着在方法中发生的任何动作都必须在单个帧更新中发生。

在希望使用方法调用来包含过程动画或随时间变化的事件序列的情况下，可以使用协程。

然而，重要的是要记住协程不是线程。在协程内运行的同步操作仍然在主线程上执行。如果希望减少主线程上花费的CPU时间，避免在协程中阻塞操作与在任何其他脚本代码中一样重要。如果你想在Unity中使用多线程代码，可以考虑使用[C# Job System](https://docs.unity.cn/cn/2022.3/Manual/JobSystem.html).

如果需要处理长时间的异步操作，比如等待HTTP传输、资产加载或文件I/O完成，那么最好使用协程。

协程是一种方法，您可以声明一个返回类型为IEnumerator的方法，并在程序体中包含yield返回语句。yield返回结果是执行暂停并在下一次执行中继续执行的点。要开始一个协程，需要使用StartCoroutine函数:并且是要继承 MonoBehaviour  的脚本才可以通过 StartCoroutine 开始一个协同程序

IEnumerator Fade() {

Color c = renderer.material.color;

for (float alpha = 1f; alpha >= 0; alpha -= 0.1f)

{

c.a = alpha; renderer.material.color = c; yield return null;

}

}

Fade函数中的循环计数器在协程的整个生命周期内保持其正确的值，并且在yield语句之间保留任何变量或参数。

IEnumerator Fade()

{

Color c = renderer.material.color;

for (float alpha = 1f; alpha >= 0; alpha -= 0.1f)

{

c.a = alpha;

renderer.material.color = c;

yield return new WaitForSeconds(.1f);

}

}

默认情况下，Unity在yield语句之后的帧上恢复协程。如果你想引入一个时间延迟，使用WaitForSeconds:

可以使用WaitForSeconds在一段时间内扩展效果，并且可以将其用作在Update方法中包含任务的替代方法。Unity每秒调用Update方法多次，所以如果你不需要经常重复一个任务，你可以把它放在协同程序中以获得定期更新，而不是每一帧在 Update 方法中处理相应的逻辑。

bool ProximityCheck()

{

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

{

if (Vector3.Distance(transform.position, enemies[i].transform.position) < dangerDistance) {

return true;

}

}

return false;

}

像上面这个方法，如果在 Update 中调用 ProximityCheck() 则每帧都要执行，如果 enemies 数量比较多，会造成较大性能开销，可以利用协同程序 来控制每 X 秒执行一次，如下

IEnumerator DoCheck()

{

for(;;)

{

if (ProximityCheck())

{

// Perform some action here

}

yield return new WaitForSeconds(.1f);

}

}

这减少了Unity执行的检查次数，而不会对游戏玩法产生任何明显影响。

要停止协同程序，请使用StopCoroutine和StopAllCoroutines。

如果你把调用协同程序的游戏对象GameObject调用方法SetActive(false),那么协同程序也会停止。调用Destroy(example)(其中example是MonoBehaviour实例)立即触发OnDisable, Unity处理协程，有效地停止它。最后，在帧结束时调用OnDestroy。

注意:如果你通过设置enabled为false来禁用MonoBehaviour, Unity不会停止协程。

协程的执行方式不同于其他脚本代码。Unity中的大多数脚本代码出现在单个位置的性能跟踪中，在特定的回调调用之下。然而，协程的CPU代码总是出现在跟踪中的两个位置。

每当Unity启动协程时，协程中的所有初始代码，从协程方法开始直到第一个yield语句，都会出现在跟踪中。初始代码通常在调用StartCoroutine方法时出现。 Unity回调生成的协程(比如返回IEnumerator的Start回调)首先出现在它们各自的Unity回调中。

协程代码的其余部分(从第一次恢复到完成执行)出现在Unity主循环中的DelayedCallManager行中。

这是因为Unity执行协同程序的方式。

c#编译器会自动生成支持协程的类的实例。

Unity然后使用这个对象来跟踪跨多个调用单个方法的协程的状态。

因为协同程序中的局部作用域变量必须在yield调用中持续存在，所以Unity将局部作用域变量提升到生成的类中，在协程期间仍在堆上分配。

该对象还跟踪协程的内部状态:它记住在代码中的哪个点，协程在产生后必须恢复。

因此，协程启动时产生的内存压力等于固定的开销分配加上其局部作用域变量的大小。

启动协程的代码构造并调用一个对象，

然后Unity的DelayedCallManager在协程的yield条件满足时再次调用它。因为协程通常在其他协程之外启动，

这将它们的执行开销划分为yield调用和DelayedCallManager。

你可以使用Unity Profiler来检查和理解Unity在你的应用程序中执行协同程序的位置。要做到这一点，

在启用深度剖析(Deep Profiling enabled)的情况下对应用程序进行剖析，

它可以分析脚本代码的每个部分并记录所有函数调用。

然后，您可以使用CPU Usage Profiler模块来调查应用程序中的协程。

最佳实践是将一系列操作压缩到尽可能少的单个协程。嵌套协程有助于代码的清晰度和维护，但是它们会增加内存开销，因为协程要跟踪对象。

如果协程每一帧都运行，并且不会对长时间运行的操作产生影响，用Update或LateUpdate回调代替它会更高效。如果您有长时间运行或无限循环的协程，这很有用。

2.Xml,Json,Protobuffer等序列化的区别。如何选型

1. 解释ECS的概念和思想，如何应用，又用在何处

4.如何为 Unity 的 C# 项目添加 dll 引用或安装 NuGet 包

<https://blog.walterlv.com/post/unity-starter-reference-dlls-and-add-nuget-package-for-unity-csharp-projects.html>

5.使用 Visual Studio 开发 Unity C# 脚本，说说根目录的那些 sln 和 csproj 文件

6.堆空间、栈空间 的概念，区别，分别是存储什么数据的？

托管堆、非托管堆IDispose

堆栈在内存中的存放顺序，说明堆栈的区别

7.C# 中值类型与引用类型分别有哪些，有什么不同，装箱拆箱是什么

1. 虚拟内存和物理内存的区别是什么
2. C# List 是数组实现的还是链表实现的
3. C# 中委托是链表还是数组实现的

11.C#中操作两个字符串相加时，会有几次内存操作？

直接写`string str = "a" + "b";`会有几次内存申请操作

1. 如果将C#的代码打包成DLL，DLL中的内容是编译之后的二进制文件吗，还是其他内容？
2. Unity中的C#编译过程是什么样的？

13.解释型语言和编译型语言的区别是什么？

1. Unity上滚动列表ScrollView中的元素如果包含粒子特效，怎么处理元素滚动到被遮挡裁切部分的特效没有被正确裁切问题？

推荐使用 UIParticle 插件

1. Unity手机游戏开发时，通常如何处理不同手机的适配问题？

UGUI Canvas 如何适配，Canvas属性 UIScaleModel、ReferenceResolution、ScreenMatchModel、ReferencePixels Pre Unit 的作用

在参考分辨率为1920x1080的情况下，如果在一个2000x1080分辨率的屏幕上，Canvas会怎么适配？

1920x1080的图片在2000x1080的屏幕上会怎么显示？

1. 如果要保证高度不变，应该怎么计算？
2. Unity游戏接入SDK的流程一般是什么？
3. Unity的Animation和Animator的区别是什么？IK是什么？
4. Unity的UGUI界面的构建过程是什么？在制作UGUI界面Prefab时，有什么技巧或者方式来降低性能开销？UGUI中什么操作会导致重绘？UGUI的渲染流程是什么？
5. 解释型语言和编译型语言的区别是什么？
6. 一个相机中如何做分层渲染？底层原理是什么?

21.Socket TCP、Udp的连接的流程是怎样的，他们的区别？

1. 项目中的打包工具做了什么事情？如何做自动化打包？

23.3D转2D碰撞检测的实现过程，使用什么算法进行检测？

1. 讲讲你对URP的了解
2. 代码安全具体做了什么内容？

26.LineRenderer的底层实现原理是什么？

1. 讲一下二维碰撞检测的算法实现，四叉树。
2. 讲一下UGUI与NGUI有哪些不同的地方
3. 请说明 MonoBehaviour 中 Awake、Start、OnEnable 函数区别

Update、LateUpdate、FixedUpdate 函数区别

1. Unity 提供了哪几种光源，主要区别是什么
2. Unity 怎么使用 Layer 和 Tag
3. MeshRender 中 material 和 sharedmaterial 区别
4. AssetBundle 的打包压缩格式有哪些常用的，区别是什么
5. 前向(forward rendering) 和 延迟渲染（Deferred Rendering）的主要区别和优劣是什么？

什么时候使用现象渲染、什么时候使用延迟渲染

1. 多人游戏的同步方案一般都哪几种，请简述其原理
2. 常用的 AI 系统有哪些？

FSM有限状态机，

Hfsm层次有限状态机，、

GOAP目标行为规划

BehaviorTree行为树，

36.C# string 相加和 StringBuilder.Append 的主要区别是什么？