**C# Job System**

What:

Job System通过创建Jobs而不是Threads更加简单和安全地管理多线程代码。

How：

1， Job system把jobs放入Job队列。

2， Worker threads从Job队列取出Job并执行。

3， Job system管理依赖并保证Jobs按照正确的顺序执行。

Advantage：

1， 可以和ECS配合使用

2， 配合Burst complier使用提升生成的代码质量

3， 避免创建多于CPU核心数量的线程数量，每个线程运行在一个CPU核心上，避免上下文切换的资源消耗

4, 可以同时利用CPU多个核心

**A job**

What:

A job 接收参数并对数据进行操作, 类似于调用一个方法。Job可能是独立的, 也可能依赖于其他Jobs完成之后才能开始进行。

How:

Create:

• 创建一个实现了IJob接口的Struct

• 添加Job使用的成员变量（值类型或者NativeContainer）

• 实现Execute方法

注意: 在主线程中访问job中的数据的唯一方法是向NativeContainer中写入数据。

编程：声明一个Job

Schedule a job:

• 实例化一个job

• 填充job的数据

• 调用Schedule方法

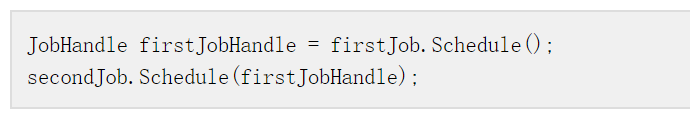
编程：An example of scheduling a job

Job dependencies

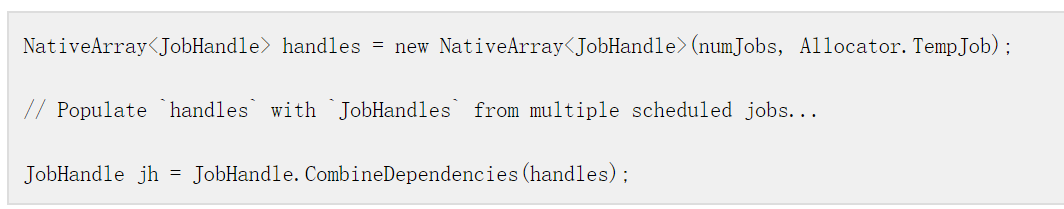
What:

如果 jobA 依赖于jobB, job system 确保 jobB完成之后，jobA才会开始执行。

How:



Combining dependencies



在主线程中等待job结束可以使用job.complete方法。

编程：An example of multiple jobs and dependencies

**Race Conditions**

What:

当一个操作的输出依赖于另一个操作的执行时间时候会发生Race Conditions

Answer:

1,通过传递数据的复制，而不是引用

2，使用NativeContainer

NativeContainer

What:

NativeArray

NativeList

NativeHashMap

NativeMultiHashMap

NativeQueue

How：

DisposeSentel检测内存泄漏

AtomicSafetyHandle检测到冲突时会引发异常，并显示一条清晰的错误消息

如果Job不需要向NativeContainer写入数据，就标记为ReadOnly。

[ReadOnly]

public NativeArray<int> input;

NativeContainer Allocator：

The allocation type depends on the length of time the job runs.

• Allocator.Temp

• Allocator.TempJob

• Allocator.Persistent

ParallelFor job

What：

IJobParallelFor类似于IJob, 不同于单个Execute方法,它为数据源中的每个物体调用一次Execute方法。

How:

1. 编写一个实现IJobParallelFor 接口的struct。
2. A ParallelFor job 使用一个 NativeArray 作为数据源.
3. 当你schedule ParallelFor jobs, 你必须制定NativeArray 的长度
4. 为了优化性能，你需要指定batch数量。从1开始，增加batch数量直到性能合适。