Archetypes：

What：

组件类型的一个不同的组合称为原型。

Memory Chunks

What:

ECS以“块”的形式分配内存，每个块由一个ArchetypeChunk对象表示。块始终包含一种原型的实体。

How:

当一块内存变满时，ECS会为使用相同原型创建的任何新实体分配一块新内存。

如果添加或删除组件，然后更改实体原型，ECS会将该实体的组件移动到其他块。

当一个实体被创建或更改为一个新的原型时，ECS会将其放入存储该原型的第一个具有空间的块中。

为了使得块仍然被紧密地打包，当实体从原型中移除时，ECS会将块中最后一个实体的组件移动到组件数组中新腾空的插槽中。

块中的所有实体对于任何共享组件都具有完全相同的值。如果更改共享组件中任何字段的值，则修改后的实体将移动到其他块，就像更改该实体的原型一样。如有必要，将分配一个新块。

Advantage:

这种组织方案在原型和块之间提供了一对多的关系。这也意味着，查找具有给定组件集的所有实体只需要搜索现有的原型，这些原型通常数量很少，而不是所有实体，它们的数量通常要大得多。

World

A World同时拥有一个EntityManager和一系列ComponentSystems.你可以创建任意数量World对象。通常你可以创建一个simulation World和一个rendering或presentation World.

默认情况下，当您进入播放模式时，会用项目中所有可用的组件系统对象填充它，您将创建一个单独的世界。但是，您可以禁用默认的World创建，并通过global defines将其替换为您自己的代码

Job dependencies

System的依赖项属性是一个JobHandle，它表示系统与ECS相关的依赖项。

使用Entities.ForEach和Job.WithCode的重载版本，它们将job dependencies作为参数，并将更新的依赖项作为JobHandle返回。

EntityCommandBuffer

EntityCommandBuffer（ECB）类解决了两个重要问题：

在一个job中，你不能访问EntityManager。

执行结构更改（如创建entity）时，将创建同步点，并且必须等待所有作业完成。

EntityCommandBuffer抽象允许您将更改（来自作业或主线程）排队，以便它们稍后在主线程上生效。

ECS规定，以下行为都不能在Job中处理：

创建实体（Create Entities）

销毁实体（Destroy Entities）

给实体添加组件（Add Components）

删除实体的组件（Remove Components）

上面的四种行为都不能在Job中处理，于是，就有了EntityCommandBufferSystem。

EntityCommandBufferSystem可以让我们在Job里添加一些任务队列，然后在主线程中执行这些任务。

编程：实例LifetimeSystem

同步点（sync point）是程序执行中等待迄今已调度的所有作业完成的点。同步点限制您在一段时间内使用作业系统中可用的所有工作线程的能力。因此，通常应避免同步点。

ECS中数据的结构变化是导致同步点的主要原因。以下都是结构变化：

创建实体

删除实体

向实体添加组件

从实体中删除组件

更改共享组件的值

这些结构更改只能在主线程上执行。

您可以使用entity command buffers（ECB）将结构更改排队，而不是立即执行它们。

每个标准组件系统组实例都提供一个EntityCommandBufferSystem，作为组中更新的第一个和最后一个系统。

Write groups（待续）

Version numbers（待续）

Job extensions

These interfaces include:

IJob

IJobParallelFor

IJobExtensions

IJobParalllelForExtensions

JobHandle

The Jobs package extends the job system to support ECS. It contains:

IJobParallelForDeferExtensions

IJobParallelForFilter

JobParallelIndexListExtensions

Job​Struct​Produce<T>