**Entities**

What：

填充游戏或程序的实体或事物。

How:

实体既没有行为也没有数据；相反，它标识哪些数据块集合在一起。

一个实体本质上是一个ID，默认情况下甚至没有名字。实体id是稳定的，你可以使用它们来存储对另一个组件或实体的引用。

尽管实体没有类型，但实体组可以按与其关联的数据组件的类型进行分类。

Create entities:

1，设置ECS在运行时将放置在场景中的游戏对象和预设的游戏对象转换为实体。

2，在作业中创建多个实体的生成系统

3，使用EntityManager.CreateEntity

使用EntityManager.CreateEntity：

逐个创建实体：

使用使用ComponentType对象数组的组件创建实体。

使用使用EntityArchetype的组件创建实体。

使用Instantiate复制现有实体（包括其当前数据）

创建没有组件的实体，然后向其添加组件。

创建多个实体：

使用CreateEntity用具有相同原型的新实体填充NativeArray。

使用Instantiate将现有实体（包括其当前数据）的副本填充到NativeArray中。

显式创建由指定数量的实体填充的块，这些实体具有具有具有CreateChunk的给定原型。

Entity queries

What:

要标识系统应处理的实体，请使用EntityQuery。

How:

SystemBase.Entities.ForEach会创建内部EntityQuery实例

可以基于组件类型数组创建EntityQuery：

EntityQuery m\_Query =

GetEntityQuery(typeof(RotationQuaternion), ComponentType.ReadOnly<RotationSpeed>());

Outside of a system class, you can create

a EntityQuery with the EntityManager.CreateEntityQuery()

EntityQueryDesc

How：

可以通过查询指定以下组件要求：

All-原型必须包含All类别中的所有组件类型。

Any-原型必须至少包含任意类别中的一个组件类型。

None-原型不能包含“无”类别中的任何组件类型。

var query = new EntityQueryDesc{

None = new ComponentType[]{ typeof(Frozen) },

All = new ComponentType[]{ typeof(RotationQuaternion), ComponentType.ReadOnly<RotationSpeed>()}}

EntityQuery m\_Query = GetEntityQuery(query);

Defining filters

How：

两种筛选器：

Shared component filter：根据共享组件的特定值筛选实体集。

m\_Query.SetFilter(new SharedGrouping { Group = 1 });

Change filter：根据特定组件类型的值是否已更改来筛选实体集。

m\_Query.SetFilterChanged(typeof(Translation));

Executing the query

EntityQuery执行其查询的方法：

ToEntityArray()

ToComponentDataArray<T>

CreateArchetypeChunkArray()

查看HelloCube IJobChunk例子