Components：

What：

与实体关联但由数据本身而不是实体组织的数据。

How：

ECS中的组件是一个实现以下“接口”之一的结构体：

IComponentData

IBufferElementData

ISharedComponentData

ISystemStateComponentData

ISharedSystemStateComponentData

Blob assets

IComponentData

What：

How：

实现IComponentData接口

IComponentData不定义行为，只定义数据。

将ICComponentData实现为Struct而不是Class

改变数据的方法

var transform = group.transform[index]; // Read

transform.heading = playerInput.move; // Modify

transform.position += deltaTime \* playerInput.move \* settings.playerMoveSpeed;

group.transform[index] = transform; // Write

IComponentData结构不能包含对托管对象的引用。

这是因为ComponentData存在于简单的非垃圾收集跟踪块内存中，这具有许多性能优势。

Managed IComponentData

Drawback：

它不能与Burst Compiler一起使用

它不能在job structs中使用

它不能 Chunk memory

它需要垃圾收集

如果不需要托管组件支持，menu: Edit>Project Settings>Player>Scripting Define Symbols声明UNITY\_DISABLE\_MANAGED\_COMPONENTS。

ISharedComponentData

What：

共享组件是一种特殊的数据组件，您可以使用它根据共享组件中的特定值细分实体

How：

将共享组件添加到实体时，EntityManager会将具有相同共享数据值的所有实体放置到同一块中。

如果过度使用共享组件，可能会导致块利用率低。

还可以使用EntityQuery.SetFilter（）对具有特定SharedComponentData值的实体进行特定的迭代。

可以使用EntityManager.GetAllUniqueSharedComponents检索添加到任何活动实体的所有唯一SharedComponentData。

Advantage：

ISharedComponentData的每个实体的内存成本为零。

ISystemStateComponentData（待续）

What：

当Entity被销毁时，ECS通常：

查找引用特定实体ID的所有组件。

删除这些组件。

回收实体ID以供重用。

但是，如果存在SystemStateComponentData，ECS不会回收ID。这使系统有机会清理与实体ID关联的任何资源或状态。

ECS仅在删除SystemStateComponentData后才重新使用实体ID。

How：

ECS是没有回调的。那么我怎么知道某个实体是不是被删除了？可能在以前的话，是订阅某个实体的死亡消息，在实体死亡的时候回调。

现在的话，需要利用状态组件（SystemStateComponent）。在实体被销毁时，状态组件是不会被删除的。

比如，EntityA有三个组件：ComponentA、ComponentB、SystemStateComponentC。

当EntityA被销毁时，SystemStateComponentC仍然存在，而此时EntityA实际上是没被完全销毁的。

我们筛选SystemStateComponentC组件并且排除ComponentA和ComponentB。

当实体只剩下SystemStateComponentC组件时，我们就成功筛选到数据，

成功筛选到数据就代表实体已经被”销毁“了，它剩下一个状态组件。

再总结一下，状态组件就是一个不会因为实体销毁而被删除的组件，除非我们主动删除。

IBufferElementData（待续）

What：

IBufferElementData是一个表示数组元素的接口。

ECS无法将同一类型的多个组件与实体关联。因此，如果要拥有相同类型的多个数据，则需要在组件中包含数据数组，但由于基本组件不能使用引用类型，因此它不能包含可变长度数组。

因此，Unity ECS提供了一种称为动态缓冲区组件的机制。动态缓冲区组件是表示可变长度数组的组件，IBufferElementData并指定实现元素类型的类型。

How：

public struct BufferComponent : IBufferElementData

{

public int num;

}

//

DynamicBuffer<BufferComponent> buffer = entityManager.AddBuffer<BufferComponent>(entity1);

buffer.Add(new BufferComponent { num = 1 });

buffer.Add(new BufferComponent { num = 2 });

buffer.Add(new BufferComponent { num = 3 });

//

public void Execute (Entity entity,int index,DynamicBuffer<BufferComponent> buffer){

int sum = 0;

foreach (int number in buffer.Reinterpret<int>())

{

sum += number;

}

Debug.Log("Sum of all buffers: " + sum);

}

ChunkComponent（待续）

What：

块组件其实也是普通组件（仍然继承IComponentData），只不过它有专门的函数来新增、修改、删除等。

块组件和共享组件有点相似，块组件也是所有实体共用的组件，块组件也遵守组件的一些基本规则。

但是，块组件和共享组件的最大区别是：修改块组件的值不会导致实体被移动到新的块（Chunk），而是将实体所在块的所有实体的块组件的值也一起修改。

修改块组件的值，不会导致实体被移动到其他的块，而是块中所有的实体的块组件的值都改变了。