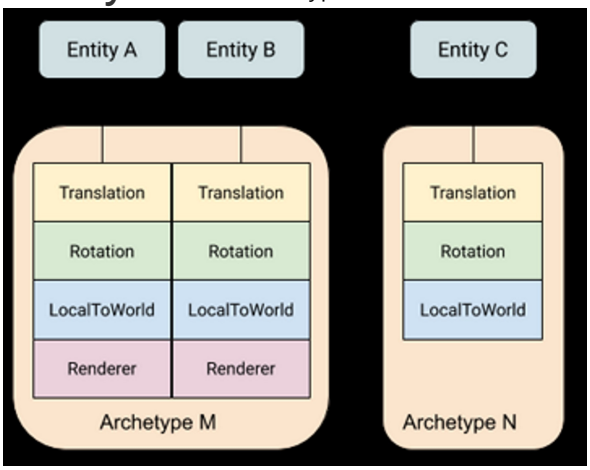
Unity Entity (ECS)

2023년 2월 21일 화요일 오후 5:17

ECS : Entity Component System

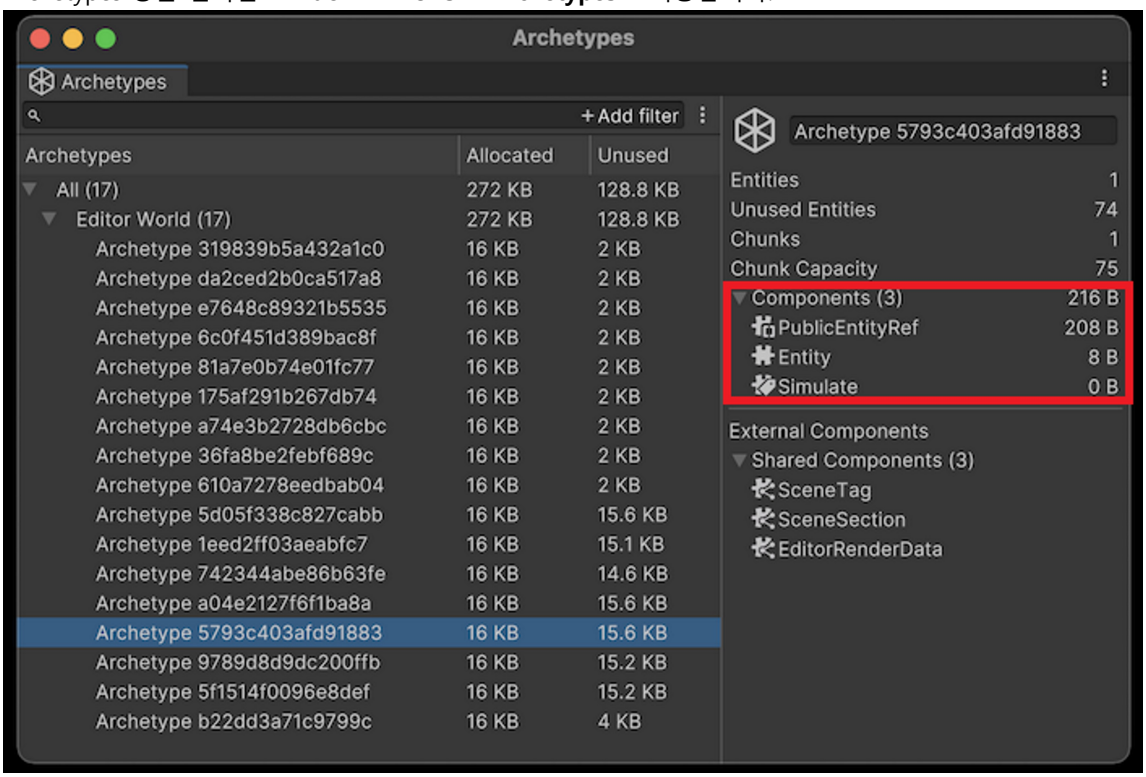
Entity: 엔티티는 어플리케이션을 채우는 것입니다. 데이터나 행동을 가지지 않으며, 어떤 데이터가 같이 있는지 식별합니다.
Components: 엔티티가 아니지만, 엔티티와 연관된 데이터 자체로 구성되어 있습니다.
System: 구성 데이터를 현재 상태에서 다음 상태로 변환하는 논리입니다. Entity와 Component를 조회해서 로직을 수행합니다.

Entity는 사실상 Archetype을 담고 있는 껍데기라고 이해하면 좋을 것 같다.



엔티티A와 B는 같은 아키타입M을 공유하며, Renderer가 없는 EntityC는 아키타입M 과 구분된 아키타입N을 가진다.
런타임에 엔티티의 아키타입 구성 요소를 추가/제거 할 수 있는데, 만약 엔티티B의 Renderer를 제거하면 아키타입N으로 이동하게 된다. 엔티티에 구성요소를 변경하려면 Component를 사용하면 된다. (add component)

Archetypes 창을 열려면 **Window > DOTS > Archetypes**로 이동합니다.



해당 창에서 사용중인 아키타입들을 볼 수 있는데, 그중 하나를 클릭하게 되면 우측에 상세 정보가 표기된다.

우측에 Components 항목을 보면 어떤 구성 요소들(Components)이 들어있는지 확인 할 수 있다.
Entity라는 항목에는 해당 아키타입에 들어있는 엔티티의 수를 알려준다.

만약 위의 Entity ABC의 예시로 들면, Archetype M의 Components 항목엔 [Translation][Rotation][LocalToWorld][Renderer]가 포함되어있을 것이다.

에디터 아이콘 의미

이미지	구성 요소 유형
	버퍼 구성 요소
	청크 컴포넌트
	관리되는 구성 요소 (Managed)
	공유 구성 요소
	태그가 지정된 구성 요소

구조적 변경 사항:

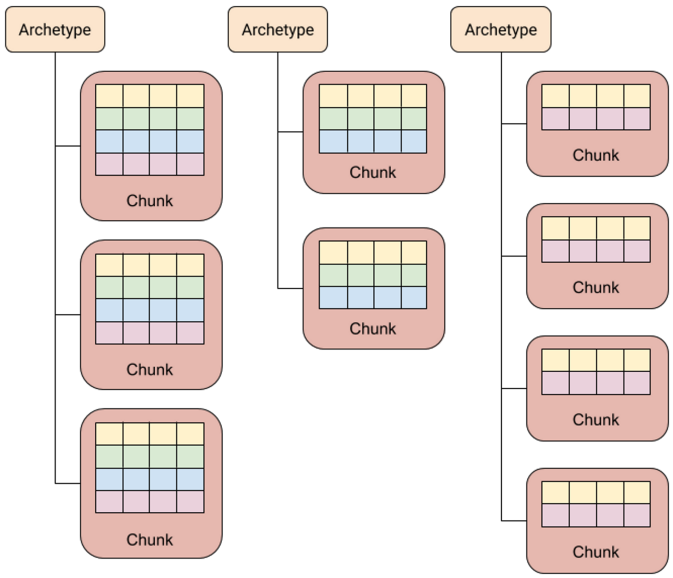
- 청크 생성
- 청크 파괴
- 청크에 엔티티 추가
- 청크에서 엔티티 제거
- ISharedComponentData 값을 설정하는 경우

구조적 변경 사항은 작업에서 수행할 수 없으며 주 스레드에서만 수행해야합니다. 이를 위한 해결책은 EntityCommandBuffer를 사용하여 변경 사항을 기록하는 것입니다.
EntityCommandBuffer는 기록된 변경 사항을 나중에 다시 주 스레드에서 실행하여 변경 사항을 실행할 수 있습니다.
작업에서 컴포넌트를 즉시 추가하거나 제거 할 수는 없지만 EntityCommandBuffer를 사용하여 나중에 컴포넌트를 추가하거나 제거할 의도를 기록할 수 있습니다.

Memory Chunks

Memory Chunk는 동일한 Archetype내 속한 Entity의 Component 데이터를 메모리상에서 연속적으로 저장합니다.

Unity ECS에서 청크의 크기는 16KB로 고정되어 있으며 청크가 가득 차게되면, 같은 아키타입 내 똑같은 크기의 청크를 새로 생성합니다. 특정 순서대로 저장하지 않으며, 한 청크의 모든 엔티티는 공유 구성 요소에 대한 정확히 동일한 값을 가집니다.



Entity Query

시스템이 처리해야 할 엔티티를 식별하기 위해 엔티티 쿼리를 사용합니다. 엔티티 쿼리는 요구사항화 일치하는 컴포넌트를 찾기 위해 검색하며, 아래와 같은 요구사항을 지정 할 수 있습니다.

All — 아키타입은 All 범주의 모든 컴포넌트 유형을 포함해야 합니다.
Any — 아키타입은 Any 범주의 적어도 하나의 컴포넌트 유형을 포함해야 합니다.
None — 아키타입은 None 범주의 컴포넌트 유형을 포함해서는 안 됩니다.

엔티티 쿼리는 쿼리가 필요로 하는 컴포넌트 유형의 종류가 포함된 청크의 목록을 제공합니다. 그런 다음 IJobEntityBatch를 사용하여 해당 청크에 직접 반복적으로 액세스할 수 있습니다.

Components

유니티 컴포넌트와 차이점

ECS 컴포넌트	유니티 컴포넌트
보통 구조체의 인스턴스(관리되지 않는 컴포넌트)이다. 클래스의 인스턴스(관리되는 컴포넌트)일 수도 있다.	클래스의 인스턴스이다.
엔티티와 관련이 있다(공유 컴포넌트와 청크 컴포넌트의 경우 여러 엔티티와 관련이 있다)	게임오브젝트에 포함되어 있다.
보통 동작(메소드)을 포함하지 않는다.	보통 동작(메소드)을 포함한다.
다음 인터페이스 중 하나를 구현한다:	UnityEngine.Component에서 상속된다.
- IComponentData	
- ISharedComponentData	
- ISystemStateComponentData	
- ISystemStateSharedComponentData	
- IBufferElementData	

Tag Component
필드가 없는 IComponentData 구조체를 태그 컴포넌트라고 합니다.
이런 Tag Component의 경우, 청크에 컴포넌트 배열을 저장하지 않습니다.
데이터가 있다면 일반적인 Unmanaged 컴포넌트와 같이 작동합니다.

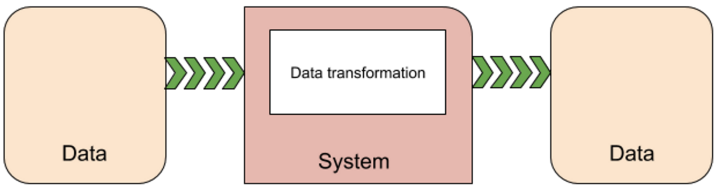
Unmanaged Component 저장소 (?)

Unmanaged 컴포넌트와 달리, Managed 컴포넌트는 직접 청크에 저장되지 않습니다. 대신, 전체 World에 대한 대규모 배열에 관리 컴포넌트 클래스 인스턴스가 모두 참조되고, 청크의 관리 형식 컴포넌트 배열은 이 배열에 대한 인덱스를 저장합니다.

여러 컴포넌트 값 읽기/쓰기

- ArchetypeChunk를 사용하여 청크의 컴포넌트 배열을 직접 읽고 쓸 수 있습니다.
- EntityQuery는 쿼리와 일치하는 청크 집합을 효율적으로 검색합니다.
- Entities.ForEach는 EntityQuery의 생성과 사용을 편리하게 처리해줄 뿐만 아니라 메인 스레드나 작업 내에서 청크의 엔티티를 반복하는 것을 더욱 편리하게 만들어줍니다.

System

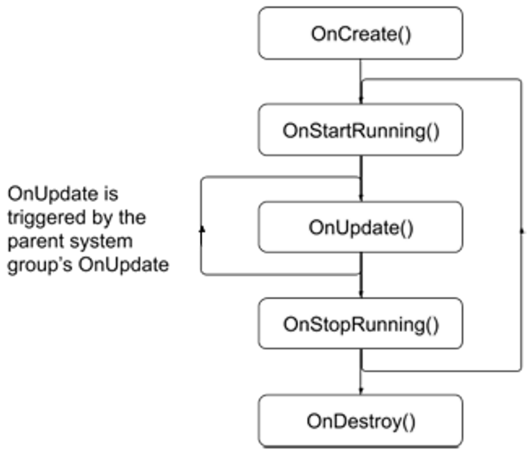


시스템 인스턴스화: ECS가 프로젝트에서 System Class를 찾아 런타임에서 자동으로 인스턴스화 한다.
System Class가 검색되면, 기본 시스템 그룹 중 하나에 추가한다.
속성을 사용하여 자동 생성을 비활성화 할 수 있다.

시스템 업데이트: 부모 **ComponentSystemGroup***에 의해 구동된다.

ComponentSystemGroup: 자식 시스템을 업데이트하는 책임을 지는 특수한 종류의 시스템.
시스템은 실행 중인 World에서 시간 데이터를 가져옵니다.
시간은 UpdateWorldTimeSystem에 의해 업데이트됩니다.

[System의 작동 순서]



- **OnStartRunning():** OnUpdate()가 처음 호출되기 전이나, 시스템이 일시 정지되었다가 다시 실행 될 때 호출됨.
- **OnStopRunning():** OnDestroy()가 호출되기 전에 호출된다. 시스템 Update를 멈추거나, 쿼리에 일치하는 Entity가 없을 경우에도 호출된다.

시스템 Type: 일반적으로 게임/데이터 변환을 위해선 SystemBase에서 확장된 클래스들을 사용하게된다.

- **SystemBase**: 시스템을 생성할 때 구현하는 기본 클래스.
- **EntityCommandBufferSystem:** 다른 시스템을 위한 EntityCommandBuffer 인스턴스를 제공합니다.
- **ComponentSystemGroup:** 다른 시스템을 위한 중첩 구성 및 업데이트 순서를 제공합니다.
  Unity ECS는 기본적으로 여러 개의 ComponentSystemGroup을 생성합니다.
- **GameObjectConversionSystem:** 게임의 GameObject기반 에디터 내 표현을 엔티티 기반 런타임 표현으로 변환합니다.
  게임 변환 시스템은 Unity Editor에서 실행됩니다.