1. Inventory Architecture – Theory

Everything starts with an Item, Item should have Static Data and Instance.

Static Data: this data never changes like wait time for a gun or basic damage for a sword.

Instance: This instance of an item may have runtime data. For example, how much ammo do you have left in the magazine or how much your sword is damaged.

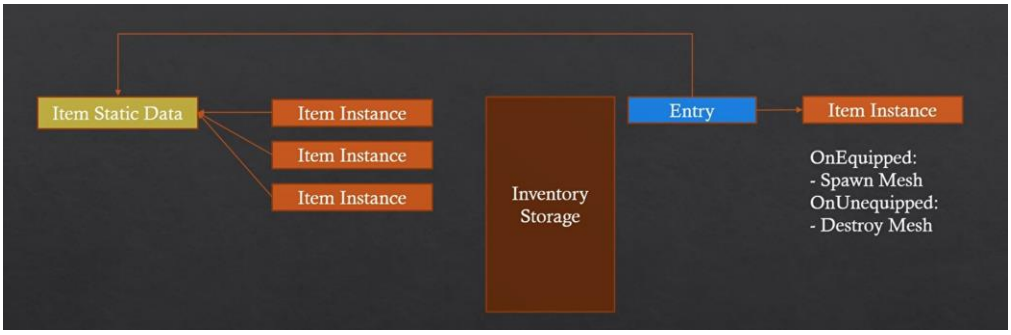
Both instances of the same sword, they would reference a same static data. And each instance of this item might have different runtime data.

We use some sort of list for inventory storage. This list should consist an inventory Entry. each entry might represent one item, or perhaps a stack of items if this inventory supports stacking, if you wish to do that.

each entry will also reference the static data. So for example if we equip a sword, we will have a record, an entry in our inventory storage (in our list) that will reference the static data so it'll know what kind of sword it is. and then each entry will be responsible for handling an Item instance.

for example, we equip this sword or this rifle, we know the static data which it reference, so we know what kind of rifle or would kind of sword it is, and it has a runtime instance that represents how many bullets we have left in the magazine or how damaged this item is. So this is what we need the item instance for.

Item instance will have functions like OnEquipped and OnUnequipped and we will trigger them from the inventory for example this item instance is a sword or gun we will try to spawn the mesh of a gun on OnEquipped and destroy that mesh on OnUnequipped.



Worth mentioning that because we are learning how to make multiplayer games. We need all this stuff to be replicated and I will show you how you can do that replication with UObjects that are dynamically instance, which is a nice trick you might want to use later. The last thing.

the last thing that can exist because we have like some item and this item can also be placed in the world and we can equip it from the world and when you throw item from your inventory storage it should appear in the world. So each be an actor, let's call it right now Item Actor that will also represent our item.

And as you may have guessed, this Item Actor should have both reference to the static data to understand which kind of item it is and the runtime instance.

In general, for the inventor's storage then we might want to implement basic functionality to actually add item to the inventor it can be made by data.

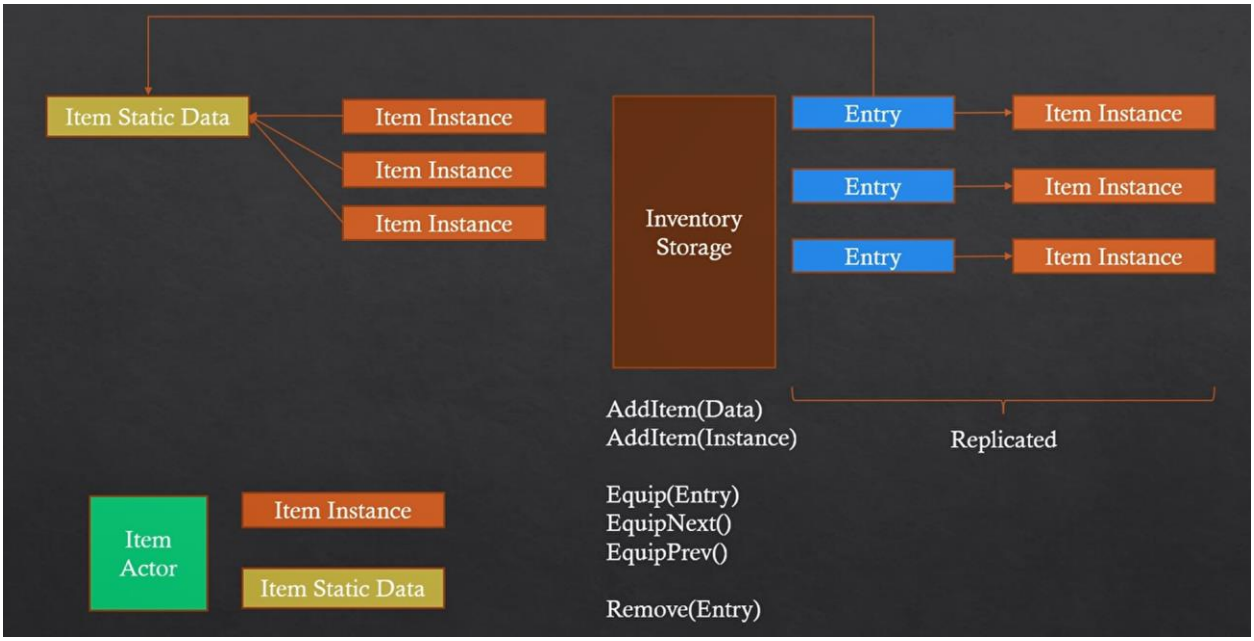
So let's say we are creating a new instance of an item and we just want to create it from the static data. Or we might want to add an item that already has an instance like that sword we throw before and we damage it.

This is not exactly the same as equipping an Item, you may have stored many items in your inventory but only one you want to equip and maybe some stuff you want to do only when this item is equipped. Like a rifle when you equip it applies several effects to the character for example it increases the speed of the character.

And that's why we want to have several functions exactly for equipping and unequipping our item.

some items might not support being equipped. This happens for example, maybe some key for a door but that key can be stored in the inventory.

And also the function that helps us to remove something from inventory. In our case because everything is stored as and Entry we will remove an Entry.


Item Static Data
Item Instance
Item Instance
Item Instance
Inventory Storage
Entry
Entry
Entry
Item Instance
Item Instance
Item Instance
AddItem(Data)
AddItem(Instance)
Replicated
Equip(Entry)
EquipNext()
EquipPrev()
Remove(Entry)
Item Actor
Item Instance
Item Static Data