**Abstract**: the purpose of this document is to explain the underlying connection between items and the inventory system and how the player can store and access items using the inventory.

**Inventory**

The purpose of the inventory system is to allow the player to store and access things within it. But the big question here is how we create an inventory system and store necessary data so that the player can access and store the inventory.

Here is the idea: the inventory system will store things that are of type *item*. Anything that is considered an *item* would inherit the properties of *item*. This way, we don’t need to incorporate everything that needs to be in the inventory (i.e. health potions, shields, swords, etc.) because all of the things being put into the inventory are all *item*.

In order to ensure that the scene doesn’t get overflowed with *item*, after the player picks up an object, the object will be immediately destroyed. But the problem with that is that we lose the instance of that object which will be a problem when we want to access it again. The solution to that (after thinking about it for 10 minutes) is to create an *Inventory Manager*. The *Inventory Manager* will store the *item*’s mesh data. This way, what the inventory actually needs to store is the *item* type, the *item* data that is associated with each item, and the count of the *item* (i.e. how many of the *item* do we actually have). This way, we can successfully delete not needed meshes within the scene to increase productivity and successfully create an *inventory*.

**Item**

The *item* can be thought as a base class or in more specific terms an abstract class. The general purpose of this class is to make it such that the *inventory* only needs to take in one type of data, which is the *item*. The *item* base class will have the following properties: name and type. The purpose of the name is self-explanatory. The purpose of the type, however, requires more explanation. As I said, anything that is considered an *item* will inherit the *item* class. But we still need to know what type of the *item* is. If the *item* was a weapon, then we need to know that the thing is weapon. The name itself won’t tell programmers much of what the item is.

**Conclusion**

With the establishment of the *inventory, Inventory Manager,* and *item* class we, in theory, should be able to create a relatively useful storing and accessing system. While I have no clue whether or not this will work or not, I am pretty certain to say that this should make our lives easier in terms of storing.