**Item System Design Document**

Item classes should be prototype for every item in the game

There will have to be two separate definitions for each item, one for the server (pure data) and one for the clients (same data as the server but also with methods etc.).

Example lifecycle for an item:

1. On initialization, server decides to place a piece of gold on the ground
   1. Server creates new GoldItem in its internal item database (gives it a unique network id). So item database is some Dict<id, item>
   2. It sends the item data to all connected clients
2. Client receives a message saying a new item has been spawned in the world
   1. Receives the data from the server regarding object type, e.g. ‘item.resources.gold’ and other parameters, e.g. position
   2. The client can look up the Unity definition of that item with this identifier, some Dict<string, item>. Some manager class containing the lookup dict
   3. Based one the lookup, it can then spawn the item in client world with some Unity-side logic. For example, if it spawns on the ground it should be instantiated as a pick up object, or if it spawns on a player it should be added to their inventory
3. Client can then interact with the object in the game world, e.g. pick it up or consume it
   1. When client interacts with object it needs to send a request back to the server so the server can do validity checks, if checks are returned OK then the action can proceed. Server also needs to update its model of the world. E.g. if item is picked up then the server needs to know it no longer has world position, and is owned by client player.
   2. Item might also be consumed or destroyed, then the server needs to remove it from its model – through the unique network id.

*Dealing with item definitions vs. amounts*

Can be done with two classes, ItemDefinition and Item:

* ItemDefinition is the base prototype e.g. item.resources.gold
* Item has a ItemDefinition and int attribute. So you can specify 2x item.resources.gold
* Then item dicts can be Dict<id, item> and refer to specific amounts of item definitions

*Network messages that must be dealt with*

Server starts with some new objects 🡪 Client spawns them in free containers

Client creates new object 🡪 Server must add it to database

Item is transferred between containers on client side 🡪 server must update its model

*Types of network container events*

Item spawned in the world somewhere not attached to an object 🡪 Drops in a default container

Player joins, needs a network container as their inventory

Building, resource node, chest created 🡪 needs a network container as its inventory