Brief – Networked Resource Management

**What it is:**

It is a modular network resource management system that will be easy to implement by a developer, this can be used for developing online City Builders and Colony Sims. The package will contain pre-made Resource Attribute and Management Systems, Resource Node System, and Networking system the developer can use to speed up their workflow, this will allow the developer to integrate their systems and the other game mechanics, fast tracking their development cycle, rather than spending time developing the underlining systems for the base game to work.

**Third party libraries:**

|  |  |  |
| --- | --- | --- |
| **Name** | **License** | **Version** |
| Mirror | Extension Asset | 66.0.9 |
| Animation Rigging | Unity Companion Package License 1.0 | 1.0.3 |
| Cinemachine | Unity Companion Package License 1.0 | 2.6.11 |
| Universal RP | Unity Companion Package License 1.0 | 10.4.0 |
| Input System | Unity Companion Package License 1.0 | 1.0.2 |
| 2D Sprite | Unity Companion Package License 1.0 | 1.0.0 |

**What will be contained in the Resource Management part:**

* Food:
* Random spawning vegetation
* Farming food
* Whether or not it respawns over time
* The model/s it will use
* Water:
* Well
* Rivers
* Lakes
* Space it takes up
* Customisable resource nodes
* Weather it spawns in clusters or in single nodes
* How spread apart the nodes are
* How rare it is
* Is it farmable
* Is it plantable
* The model/s it will use
* What resource do you get from it
* How much of the resource does it drop
* Does it automatically respawn
* Storage
* What resources can it store
* The model it will use
* Are the resources visualised on/in it
* The model/s the visualised resources will use
* How much can it hold

**Mathematical equations:**

Basic addition, subtraction, division, and multiplication.

The constant rate of decay will just use a lerp function (a + (b – a) \* t) and t will be equal to the current accumulated time divided by the maximum amount of time (t = time / max time).

When adding resources to the storage, there will be an equation to increase the time of decay ((number of resources + amount to add) / max amount of resources \* max time) since time controls how many resources there are.

**How the systems will be modular:**

**Networking:**

Will allow for the developer to specify what variables are to be passed to and from the server. Will have default variables to be passed in for the resource management system that can be overridden.

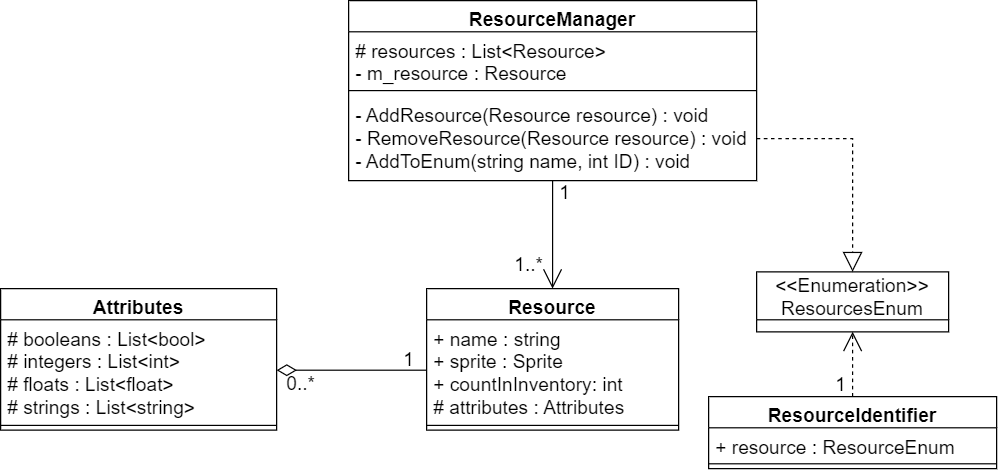
What kind of connection, what to do if no response / unexpected response, every interaction made and if it is expected

**Resource attribute system:**

A scriptable object will be used to create the resources and their attributes as two separate list.

The resources and attributes will be accessible from other components and scripts.

A similar system will be implemented for creating tools.



**Resource nodes:**

Will contain following information:

* Its position (Vector3)
* Rate it grows at (if viable) (float)
* Time it takes to harvest (float)
* What resource it is / what resource it drops (enum)
* How much of that resource it drops (int)
* How depleted it is (if viable) (int)
* What tool is used to harvest (enum)
* If it auto respawns (bool)
* Model it uses (Mesh)

As well as the option to add custom variables.

**How it will be integrated:**

The system will be a unity package downloaded from the asset store.

The developer will be able to create a Resource Manager scriptable object that they can use to create the resources and their attribute their game will contain. There will also be a Tool Manager scriptable object for the tools.

The developer will be able to add a Resource Node component to game objects to make them a Resource Node.