**Musings: Random thoughts and notes**

* **Object child sorting?**
  + Pro: Really quick to search and find a child object based on UID
  + Con: Each time you add a child it has to re-sort the children list
    - Work Around A: Object UID gets progressively larger after each object is created, (Increment the UID)
      * Pro: Sorting wouldn’t require any time unless the new child object was created before any other child objects already added
      * Con: Maximum of 4,294,967,295 objects can be created before the UID wraps around, which will cause catastrophic problems
        + Stop making particle effects you fuck
    - Work Around B: Add a flag to enable/disable child sorting and disable searching for a specific child based on UID
* **Object Manager Object-list sorting?**
  + **1:** Sort based on UID using a linked-list?
    - Pro: Really simple to implement
    - Con: Finding objects based on archetype will be slower (Linear)
      * No work-around thought of
    - Con: Finding objects based on UID will be slow (Linear)
      * No work around thought of
    - Con: **Really** slow to sort the list every time a new object is added
      * Work Around A: Object UID gets progressively larger after each object is created (Increment UID)
        + Pro: The list would sort itself
        + Pro: Finding objects based on UID will be slightly faster (Linear)

Stop if we go past the UID

Worst case scenario still is the same

* + - * + Con: Maximum of 4 billion object before catastrophic failure

Bees?

* + **2:** Sort based on Archetype using a linked-list?
    - Pro: Quicker to get a list of all objects of a certain archetype (Linear)
      * Stop searching once passed archetype id
    - Pro: Really simple to implement
    - Con: Finding objects based on UID will be slow
      * Work Around A: Set Object UID’s based on their archetype by modifying the first 4 bits (Hex) or 8 bits (2 Hex letters) and use the insert method and actually sort by UID
        + Pro: Still a linear search, but will exit once archetype id is passed
        + Con: **Limit of 16-255 archetypes** based on the number of bits used for archetype bits
        + Con: Limits the number of unique object ID’s to 16,777,216 – 268,435,456 (4 bits – 8 bits)
        + Con: Object UID’s will be forever linked to archetype ID
    - Con: **Really** slow to sort the list every time a new object is added
      * No work around thought of
  + **3a:** Sort using a Map which has archetype ID’s as keys and Linked Lists of objects as values
    - **Use Work Around A from 1**
    - Pro: **Really** quick to get a list of all objects of a certain archetype
    - Pro: Finding a certain specific object belonging to an archetype will be faster (linear)
    - Con: Slightly harder to implement
    - Con: Finding objects based on UID will be slower (Linear still)
      * No work around thought of
  + **3b:** Sort using a Map which has archetype ID’s as keys (*half-*byte = 4 bits, or byte = 8 bits) and Linked Lists of objects as values
    - **Use Work Around A from 2**
    - With the remaining bits/bytes increment them on every object created
    - Pro: **Really** quick to get a list of all objects of a certain archetype
    - Pro: Finding a certain specific object belonging to an archetype will be faster (Linear)
      * Will stop if passed UID
    - Pro: Finding objects based on UID is pretty much the same as finding an object based on its archetype and UID since they are linked together.
    - Con: Still has a limit on the maximum number of archetypes and objects
      * Work Around A: Make the UID an 8-byte integer, and have the first 4 bytes by the archetype while the later 4 bytes be the actual unique identifier
        + Pro: More archetypes than we will every need
        + Con: Max of 4,294,967,295 objects still
        + Con: Each object now costs 4 more bytes (that’s one whole pointer)
    - Con: Harder to implement
  + **4:** Use both 1 and 3a in tangent to each other. Archetype searches will use 3a and Object searches will use 1.
    - **Use Work Around A from 1**
    - Pro: Finding a certain specific object belonging to an archetype will be faster (Linear)
      * Will stop if passed UID
    - Pro: **Really** quick to get a list of objects from an archetype (constant)
    - Con: Would be somewhat hard to implement
    - Con: Requires that two separate lists of objects must be maintained, in the end this means it costs twice as much to manage objects