Resource Manager:

* Must be general and accept the resource, resource handle, loader and the allocator types (sparse or pool) as template parameters
* Must know how to load, unload and update (save) resources using the provided resource loader
* Must return a resource handle for every new resource
* Must keep handle reference count
* Must have means to increase reference count when the resource handle is copied
* Must unload the resource when the reference count reaches 0
* When provided a resource handle, must return the resource if already in memory or load it from source. If no resource is found, notify of failure and create a default resource in its place that displays the error visually (an ERROR texture)

Resource (General):

* Contains an immutable UUID and mutable string name
* Must be serializable using Cereal

Resource (Specific):

* Inherits from the general resource
* Contains all the resource-specific data and nothing but data
* Must be serializable using Cereal
* Cannot be modified by the user directly (internal functionality)

Resource Handle (General):

* Accepts resource type as template parameter
* Contains resource UUID and a reference to the appropriate resource manager
* Must increase reference count on the resource manager when copied and decrease it when released
* Must be able to return a resource reference that is temporary and non-copyable
* Must be serializable using Cereal (serialize only UUID)

Resource Handle (Resource Specific):

* Inherits from the general resource handle
* Contains functionality to operate on the resource that is obtained through the general handle
* Must be stateless
* Must not provide the resource reference to the user
* Must be serializable using Cereal (only serialize the general handle)

Sparse Allocator:

* STL compliant allocator
* Must be able to allocate objects of a give size on the heap using TBB overloaded “new” for now, and later implement incremental defragmentation
* Must be able to perform aligned allocations

Pool allocator:

* STL compliant allocator
* Must be able to perform aligned allocations
* Must be able to allocate objects using TBB pool

Resource Loader:

* Must be able to load, unload and save the resource from/to its source (file, network, database, etc)