TODO

Software Architecture

I am seriously considering utilizing an ECS, however due to the complexity of the issue I kind of want to plan it out, as I can feel the scope creep lurking around the corner just thinking about it.

Each entity should be able to have an arbitrary amount of components

The main design issue is if I should have a list of entities that store their own components or have the manager contain a list of all the specific components and make sure the index of all of them is maintained. Potential use of a hashmap is very likely here

Option 1:

The main problem here is that each component is not contiguous in memory, which would likely slow down the system for processing each component quite a bit due to CPU cache misses.

```
template<type T>
struct Component {
    std::unordered_map<std::string, T> values;
} component_t;

struct entity {
        std::string id;
        std::vector<component_t> components;
}entity_t;
```

Option 2:

Here the problem is just a slight bit more code/complexity, however each component, (which a system will be accessing each one in a row) will be contiguous in memory, speeding up the game super fast!

```
template<type T>
struct Component {
    std::unordered_map<std::string, T> values;
} component_t;

struct entity {
        std::string id;
}entity_t;

class EntityManager {
```

```
    public:
        void addEntity(...);
        void addComponent(int index, component_t component);
        ...
    private:
        std::vector<int>                                    entities;
        std::unordered_map<std::string, component_t<>>  components;
}
```

Here is a list of example components

component name	behavior
stats	contains different stats (str,hp,dex, etc.)
transform	contains entity location, rotation, size
drawable	contains render information

REMINDER!

In an ECS, Entities are just an ID, Components are just data, and Systems are where the actual logic happens!

Here is an example for the structure of a component

```
template
struct compnent_t{...}component_t;

const int COLOR_BLUE = ((255<<24) + (200<<16) + (10<<8) + 20;

component_t<int> stats = {{
    {"STR", 10},
    {"CON", 11},
    {"DEX", 14},
    {"INT",  4}
}};

component_t<int> transform = {{
    {"x", 20},
    {"y", 30},
    {"size", 15},
    {"weight", 10}
}};

component_t<int> drawable = {{
    {"tile_coord_x", 5},
    {"tile_coord_y", 3},
    {"color", COLOR_BLUE}
```

```
}};

component<bool> controllable = {{
    {"in_control", true}
}};
```

So now we have the data

Next should be the systems but right now, I am a bit tired so I will finish out the systems planning tomorrow