Spike: Task 13
Title: Composite and Component Patterns

Author: Hayden, 104001272

Goals / deliverables:

- Create “Composite” entities, that can contain other entities that can be taken and used
- Create “Component” entities, that have a series of attributes with values, and actions, that can affect the attributes of the player or another object

Technologies, Tools, and Resources used:

- Xcode

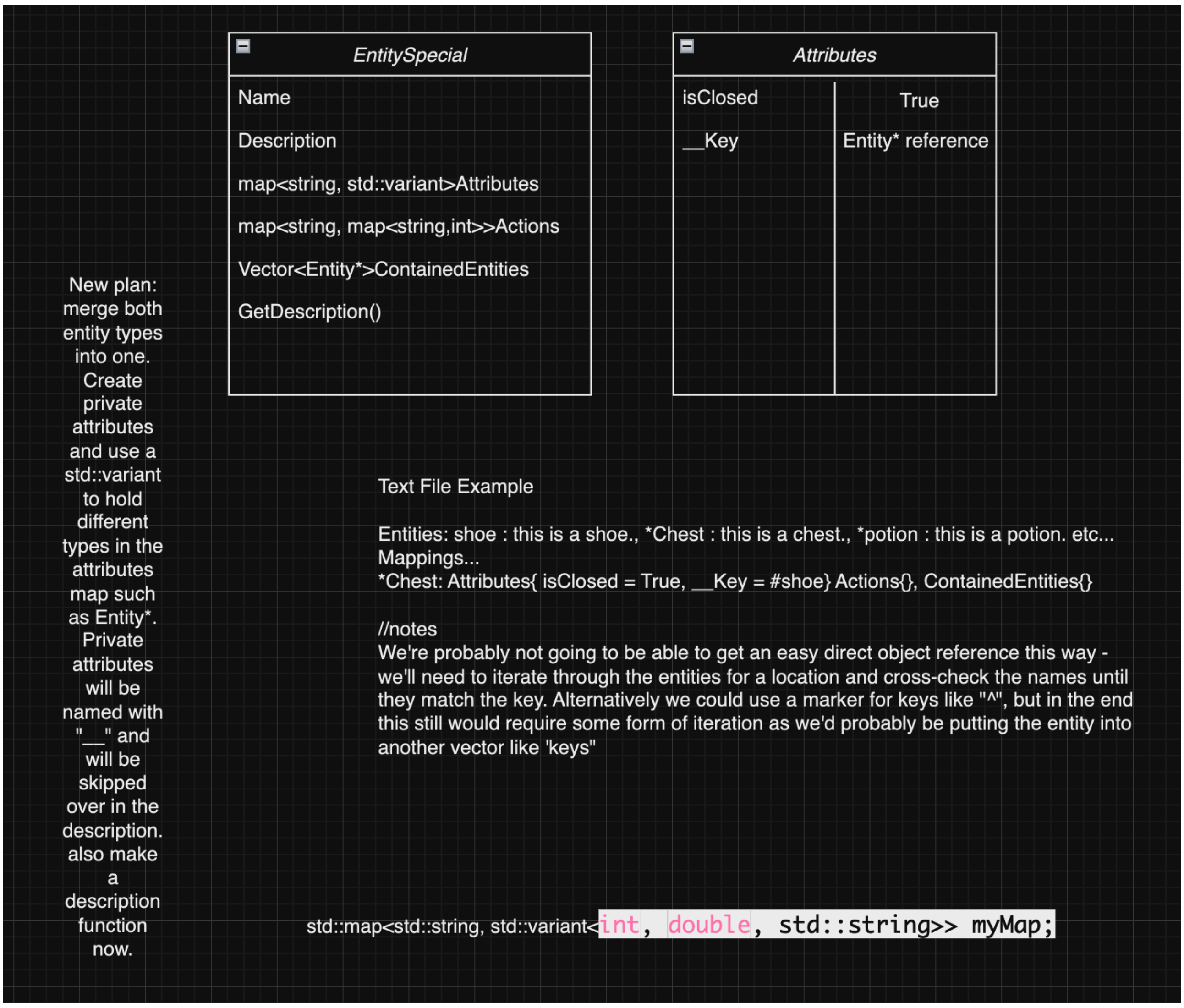
Tasks undertaken:

- **Updated the Graph Constructor extensively to support reading in Composite and Component Entities from the text file**
- **Created a new EntitySpecial object, a type of Entity with attributes, actions, and contained entities**
- **Created a new “use” command that supports self-use and use on another object**
- **Created a new “take” command, that supports taking from the game world and also a Composite Entity**
- **Updated the “look” command to support “look in” for Composite Entities**

What we found out:
The graph constructor is probably overwritten a bunch. If I could go back, I would switch to supporting using json to read in the game world, as it is already formatted. So much of my time was spent on just telling my code how to read in my text file, which seems somewhat useless to the goal at large.

Open issues/risks:

- There are a few commands yet to be implemented, such as “put”, and “open”, but these are not essential to the use of Composite and Component Entities, and are just quality of life improvements for the player.
- There is probably some error checking I have missed, such as my attribute isLocked, I assume that it is a boolean attribute, but perhaps by typo in the text file, I could set this to another data type, and my code would not account for this. I assume there would be other similar errors that could happen.



New plan: merge both entity types into one. Create private attributes and use a std::variant to hold different types in the attributes map such as Entity*. Private attributes will be named with "__" and will be skipped over in the description. also make a description function now.

EntitySpecial

- Name
- Description
- map<string, std::variant>Attributes
- map<string, map<string,int>>Actions
- Vector<Entity*>ContainedEntities
- GetDescription()

Attributes

isClosed	True
__Key	Entity* reference

Text File Example

Entities: shoe : this is a shoe., *Chest : this is a chest., *potion : this is a potion. etc...
Mappings...
*Chest: Attributes{ isClosed = True, __Key = #shoe} Actions{}, ContainedEntities{}

//notes
We're probably not going to be able to get an easy direct object reference this way - we'll need to iterate through the entities for a location and cross-check the names until they match the key. Alternatively we could use a marker for keys like "^", but in the end this still would require some form of iteration as we'd probably be putting the entity into another vector like 'keys"

```
std::map<std::string, std::variant<int, double, std::string>> myMap;
```