**Spike:** 7

**Title:** Composite & Command Patterns

**Author:** Ryan Chessum, 102564760

**Goals / deliverables:**

* Code, see /12 – Spike – Composite & Command Patterns /Zorkish Task 13/
* Spike Report

**Technologies, Tools, and Resources used:**

List of information needed by someone trying to reproduce this work

* Visual Studio 2019
* C plus plus reference (<https://www.cplusplus.com/reference/>)
* Zorkish game specifications
* Code from previous spikes

**Tasks undertaken:**

* Download and install Visual Studio
* Create a new C++ project
* Import code from previous spikes
* Add attributes for entities
* Add functionality for nested entities
* Add ability to load nested entities from file
* Update look command to look at nested entities
* Add take command to take entities
* Add throw command

**What we found out:**

In games programming two important patterns to understand are the composite and component patterns.

The composite pattern is where an object is made up of other objects. Some way this could be implemented is having items in a chest. Or you could even have an enemy made up of multiple parts like a robot. Each limb of the robot could have its own health etc.

The component pattern is where you can add components that give new attributes or actions to an entity. For example, you could make items with the consumable component able to be eaten. Then that item could also have components that tell the game what effects the player receives when consuming the item.

I made a version of Zorkish that shows a basic version of both ideas.

First, I imported the version of Zorkish from task 12 since it has everything we need to get started.

Starting with the composite pattern, I wanted to demonstrate it by nesting entities within one another. I added a vector of entity pointers to the entity class which will allow them to hold entities within.

Graphical user interface, website

Description automatically generated

Next, I created a class for components which I called attributes and added a vector of them to the entity class. Some of the attributes I want to add are “searchable” which means a player will be able to look at the items nested in it, “takeable” meaning an item can be put in the players inventory, “throwable” meaning a player can throw the item at something and “health” meaning the entity will have health.

Text

Description automatically generated

The components I’m adding here will only be very simple tags for the most part and can be expanded upon in the future to hold more data. The advantage for this implementation is that the components can easily be read from file.

Since the health attribute is the only attribute that has data, I simply added an int to entities. If they have a health component it gets set to 100, otherwise it is left NULL.

Text

Description automatically generated

I slightly changed how reading in works. All entities and locations are loaded in first. I tried to implement nesting in the same loading phase however if a vectors size is changed the memory location of all the data stored inside changes. So, we have to store them all first before we can start placing them in locations and other entities.

Text

Description automatically generated

After they are loaded in, they are added to their locations on the second loading phase. If ‘I’ is specified instead of ‘E’ the entity is placed in the entity in the same location that has the same id specified at the start.

Text

Description automatically generated

The in the third phase connections are added like before.

Now everything should load in.

I then updated the look command to be able to look at nested entities by searching the entities list of nested entities for a match. To be able to look at the entities nested in another entity it must have a “searchable” component.

Text

Description automatically generated

Next was the Take command. It works similarly to the look command. Entities can be put into the players inventory if they have a “takeable” component. Entities with the “searchable” component can also have their nested “takeable” items taken out.

Text

Description automatically generated

The last new command I added was the Throw command. If a player has an item with a “throwable” component it can be thrown at an entity with a “health” component in the location. If that entities health reaches 0, they get a “searchable” component meaning players can now look at the nested items.

Text

Description automatically generated

After adding them to the command manager everything should run.

Text

Description automatically generated

Graphical user interface, application

Description automatically generated