**Spike:** 14

**Title:** Component Pattern

**Author:** Ben Holmes, 103024841

**Goals / deliverables:**

Create part of a game that demonstrates the component pattern with

* Game objects that receive attributes from component objects rather than inheritance
* Game objects that receive actions from component objects rather than inheritance

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

* Visual Studio 2022
* Draw.io
* Word
* The discord

**Tasks undertaken:**

* Make sure to use the ComponentTest.Json in the command line rather than the Adventure.Json as Adventure.Json has not been updated to use components
* Also the code will also contain the messaging system as I developed them simultaneously.
* Made a potential plan for the spike
* Created component manager and component adding
* Created new small sample data to just test the components
* Attempt to create the component actions
* Realise the method I was using was bad and begin implementing spike 15 simultaneously as it made the component actions easier (explained below) (spike 15 report will cover the messaging system)
* Finish creation of component actions/attributes

**What we found out:**

I made a potential plan for the spike but ended up changing it while overhauling to use spike 15, here is the UML diagram

A screenshot of a computer

Description automatically generated

As shown, there are 5 components I created, two attribute components (Attack and Heal) and 3 action components (Open, Health and Use).

The way that components are assigned to entities is through the component managers createAdventureComponents(Adventure&) which is:

A computer screen shot of a program code

Description automatically generated

This cycles through all entities in the player inventory and adds any relevant components and then cycles through every location and does the same to any entities in those locations.

This runs just after the Json file has been read and transferred to the Adventure:

A screen shot of a computer code

Description automatically generated

The sample data was updated (created smaller one for ease of testing) with the new entites in the Json looking like this:

Open:

A screen shot of a computer program

Description automatically generated

Use/Heal or Attack:

A screen shot of a computer code

Description automatically generated

Health:

A computer screen shot of text

Description automatically generated

The major point of difficulty for this spike came up when I tried to implement the execute functionality of the components.

Creating one was fine, but I realised (at the time) that each of the components would use such different data as inputs that I could not easily get the functions to overload/use properly, and some of them were trying to return different values as well, which just was not working.

So instead I began to develop the Spike 15 messaging as it would unify both the input and output if I did it correctly, which I think I did.

This was the old attempt at the parent component class:

A screenshot of a computer program

Description automatically generated

And this is the final:

A black background with white text

Description automatically generated

This is the message structure for ease of understanding (which I will go into more detail in Spike 15s report where it is more relevant):

A screenshot of a computer screen

Description automatically generated

The entity receives a message with type = component name via this function

A computer screen with text

Description automatically generated

And here are the relevant execute functions:

Actions (deliverable 2):

A computer screen shot of a program code

Description automatically generated

The open component takes the message, checks if the parent entity is open, which if it is, returns message with already open.

If chest is locked, it checks the additional information is equal to “key” (this is dependent on the OpenCommand) and returns a message/changes values.

If chest is not locked, it opens the chest.

A screen shot of a computer code

Description automatically generated

The health component takes in the message (source is use component), checks if the entity is not dead (health > 0) and either changes the health by the additional information converted from string to int, or returns message stating its dead.

It needs to use try catch as a just in case a message somehow ends up here with an additional info of a string that cant be converted to int.

Command for the Use component is: USE entity1 ON entity2 (entity 1 needs use and heal/attack, entity 2 needs health)

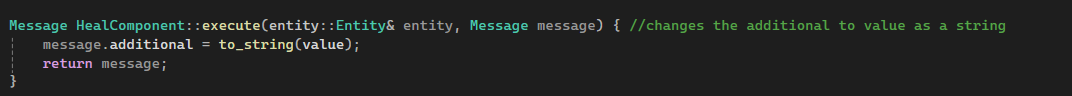
A computer screen with text

Description automatically generated  
Use component is the component that must be paired with a heal or attack component to successfully work, where it checks which of those components it has and executes that component and then sends a message with the result which ends up at the health component.

Attribute:

A screen shot of a computer code

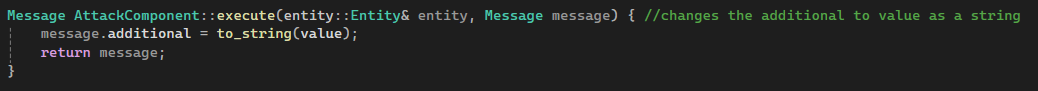
Description automatically generated



The heal component just changes the message additional to its value converted to string, currently it has a preset value of 2

A screen shot of a computer code

Description automatically generated



The attack component just changes the message additional to its value converted to string, currently it has a preset value of -5

**Open issues:**

Currently the heal and attack components use preset values meaning there cant be more complicated items like a sword that does more damage or a potion that heals less. This was for ease of creation as I did not want to have to change the Json file and deserialization more.