Task 13 - Spike Summary Report

> 💡 **Spike:** Task_13
> **Title:** Composition and Component Patterns
> **Author:** Thomas Horsley, 103071494

Goals & Deliverables

Aim: Modify Zorkish such that it supports Entities composed of other Entities. Additionally, Entities can contain components which provide them functionality and the ability to be manipulated within the game world.

Deliverables:

- Spike Summary Report
- Functioning Entity Component Code
- Git Commit History

Technology, Tools and Resources

Tech and Tools

Resources

- Echo360 Lectures "*Lecture Title*"
- Title of YouTube video that helped? See: https://www.youtube.com/watch?v=axsplPtoQF0&list=LL&index=1&t=40
- Vector Performance Management: https://www.acodersjourney.com/6-tips-supercharge-cpp-11-vector-performance/

💡 The project was scripted in C++ 17 using Visual Studio Community 2022.

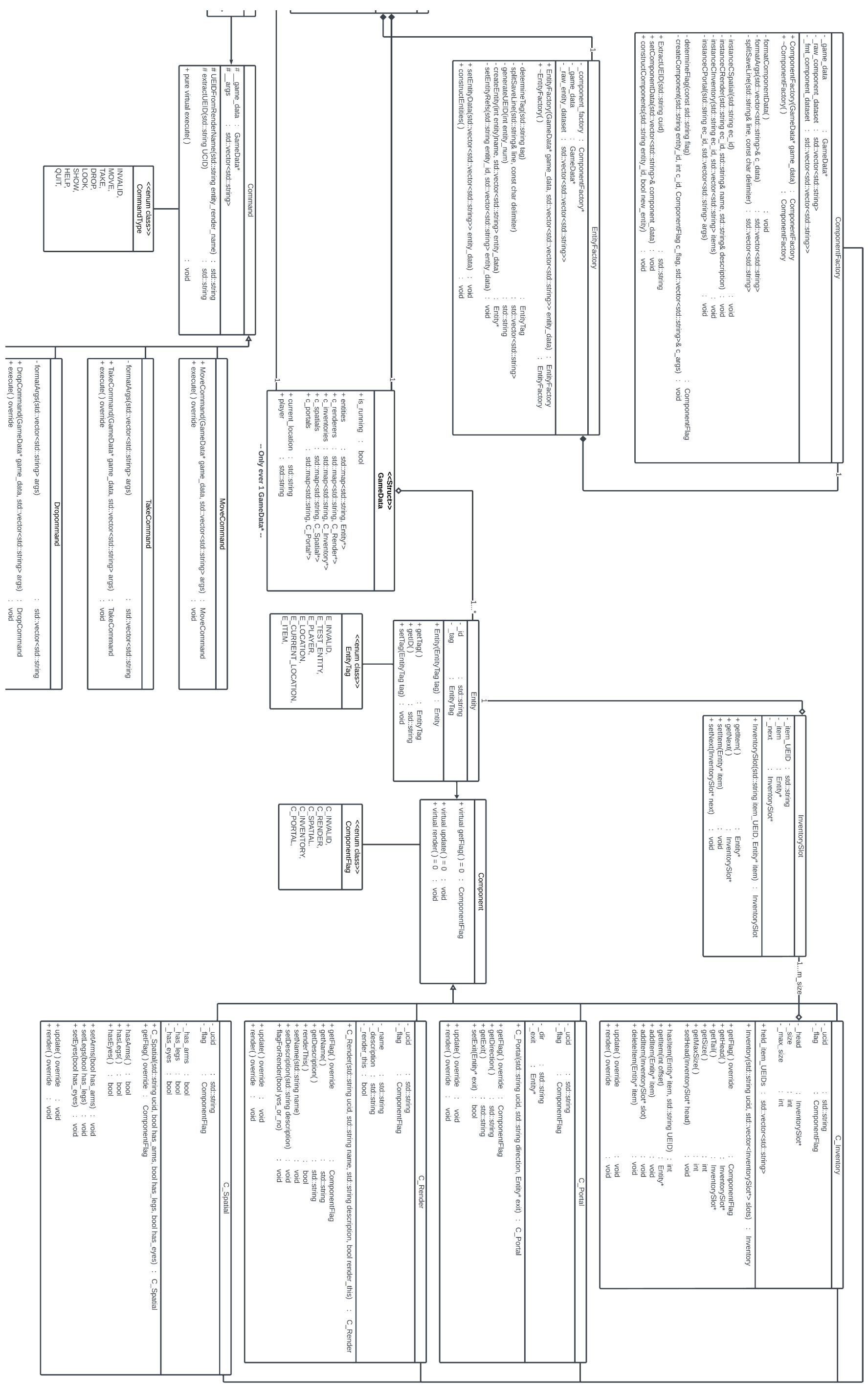
UML’s and charts are made with *www.Lucidchart.com*

Source control is handled using Git.

Tasks Undertaken

Planning

Diagrams and Charts


ComponentFactory
EntityFactory
Command
<<enum class>> CommandType
<<Struct>> GameData
-- Only ever 1 GameData* --
MoveCommand
TakeCommand
DropCommand
Entity
<<enum class>> EntityFlag
InventorySlot
Component
<<enum class>> ComponentFlag
C_Inventory
C_Portal
C_Render
C_Spatial

Class Descriptions and Notes

💡 My primary objective when designing this solution was to decrease coupling between entities and components as much as possible. To achieve this, nearly every aspect of both the AdventureSelectMenu and the GameplayState was modified.

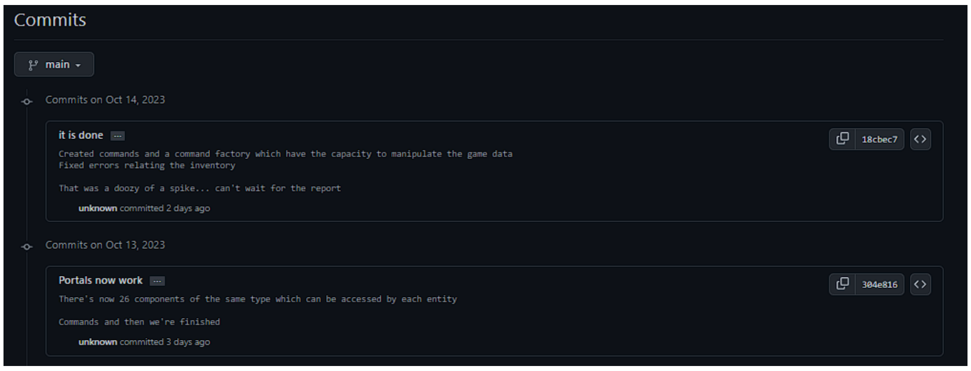
The entities and components are instantiated with a string identifier, this string identifier can be decoded to reference a specific component or entity contained within our game data. No entities know about their components nor do components directly know of their owning entities. However, as this information is encoded within the Unique Entity / Component Identifiers, managing systems are able to interact on specific entities easily.

The GameData struct was initially very simple containing only a reference to the locations, current location and player. However, this has been modified to be a PoD Struct containing a map for the entities, a map per component and a string identifier to both the player and current location.

Something which is to be noted and will be seen frequently within the Commands is the use of the render component to grab the string names of items and locations. After some thought and testing I found it to be the easiest method of translating input args to entities.

Implementation

Git Commit History



Commits on Oct 14, 2023

it is done

```
Created commands and a command factory which have the capacity to manipulate the game data
Fixed errors relating the inventory

That was a doozy of a spike... can't wait for the report
```

unknown committed 2 days ago — 18cbec7

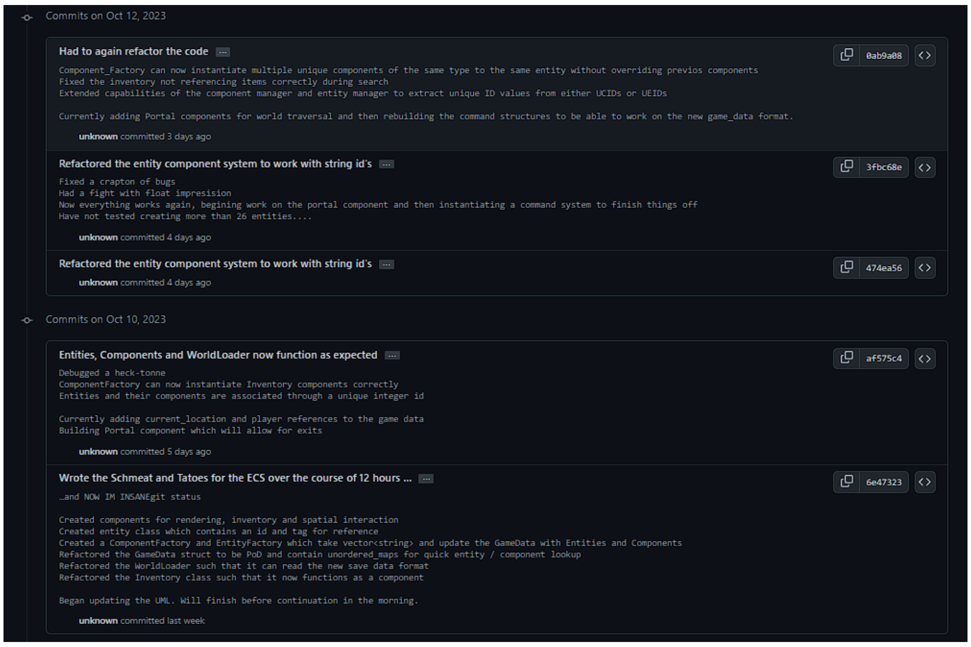
Commits on Oct 13, 2023

Portals now work

```
There's now 26 components of the same type which can be accessed by each entity

Commands and then we're finished
```

unknown committed 3 days ago — 304e816



Commits on Oct 12, 2023

Had to again refactor the code ··· 0ab9a08

```
Component_Factory can now instantiate multiple unique components of the same type to the same entity without overriding previos components
Fixed the inventory not referencing items correctly during search
Extended capabilities of the component manager and entity manager to extract unique ID values from either UCIDs or UEIDs

Currently adding Portal components for world traversal and then rebuilding the command structures to be able to work on the new game_data format.
```

unknown committed 3 days ago

Refactored the entity component system to work with string id's ··· 3fbc68e

```
Fixed a crapton of bugs
Had a fight with float impresision
Now everything works again, begining work on the portal component and then instantiating a command system to finish things off
Have not tested creating more than 26 entities....
```

unknown committed 4 days ago

Refactored the entity component system to work with string id's ··· 474ea56

unknown committed 4 days ago

Commits on Oct 10, 2023

Entities, Components and WorldLoader now function as expected ··· af575c4

```
Debugged a heck-tonne
ComponentFactory can now instantiate Inventory components correctly
Entities and their components are associated through a unique integer id

Currently adding current_location and player references to the game data
Building Portal component which will allow for exits
```

unknown committed 5 days ago

Wrote the Schmeat and Tatoes for the ECS over the course of 12 hours ... ··· 6e47323

```
…and NOW IM INSANEgit status

Created components for rendering, inventory and spatial interaction
Created entity class which contains an id and tag for reference
Created a ComponentFactory and EntityFactory which take vector<string> and update the GameData with Entities and Components
Refactored the GameData struct to be PoD and contain unordered_maps for quick entity / component lookup
Refactored the WorldLoader such that it can read the new save data format
Refactored the Inventory class such that it now functions as a component

Began updating the UML. Will finish before continuation in the morning.
```

unknown committed last week

What was Learned?

💡 Throughout coding, there were many mistakes which lead to a drastically drawn out dev time for this spike. My primary issue was creating a UCID (Unique Component Identifier) where-in the UEID (Unique Entity Identifier) was encoded. However, after all was said and done I do find running into walls the single best way to improve, too bad it sucks.

Maps

Initially I implemented a simple integer system to ID Entities and their Components. A component would inherit it's related UEID, however I never thought what would happen if 2 components shared an ID.

Consider 3 components ($C_0 - C_2$) and

Floating Point Inaccuracy.

After some pondering on a solution to the *integer issue* I decided that using a float value where the non-decimal component represents the UEID and the decimal component the UCID.

Entity E_0. If E_0 requires 2 instances of C_1 and if these components are stored in the same map, under this system they will share the same key. This results in the last instance of C_1 overriding the first.

So that idea sucked…

The Fixes Fix

Now enter the single most frustrating computing conundrum I've come across in a long while.

Whilst I knew about floating point inaccuracy there was something about subtracting a value less than 0.22 that my C++ wasn't having. I tried fancy cut arounds where I multiply the float by a base 10 value, cast it to an integer, back to a float and divide that by the original scalar. Even transferring everything over to use a double didn't work.

> Just use letters — Dad

So that's what I did, the components contain a prefixed lowercase value which is it's related UEID and an appended set of up too two characters. This combination allows for an infinite number of entities each containing a maximum of $26^2 = 676$ of each component. As a result, no Components or Entities are directly related but all can be managed using dedicate systems.
