Task 14 Spike: Component Pattern

CORE

Context

Game programming often makes heavy use of inheritance, which can be appropriate for their roles as simulations (however stylised) of the real world. In many instances though, this can lead to unnecessarily deep class 'trees' and many abstract objects intended to represent specific properties an object deeper in the tree may have. The component pattern de-emphasises inheritance as the source of object attributes by creating objects out of components, each one contributing some attribute or function to the complete object.

Knowledge/Skill Gap:

The developer needs to know how to create and modify, for a text - based adventure game, game entities that are the sum of their parts, receiving attributes from components rather than inheritance.

Goals

Building on the work of earlier Zorkish Spikes, extend the game world to support game entities composed of components that contribute properties and actions.
Create part of a game that demonstrates the following:

1. Game objects that receive attributes (damage, health, flammability, etc.) from component objects rather than inheritance.
2. Game objects that receive actions (can be picked up, can be attacked, etc.) from component objects rather than inheritance

Expected Output

Repository

1. Code
2. Spike Report

Canvas

1. Spike Report

Notes

- This is a good place to start learning about the component pattern: http://cowboyprogramming.com/2007/01/05/evolve-your-heirachy/
- Also have a look around here: http://gameprogrammingpatterns.com/
- To avoid confusion design first! Think as much as possible before you code. (If you do this, be sure to include your paper/diagrams/notes of your design with your outcome report.)
- Don't try to be too clever. Make a simple version of each separate key idea (mini demo) before you attempt anything interesting with it all combined.
- The visitor pattern has some similarities to the component pattern, and tutorials on implementing the visitor pattern can provide inspiration for implementations of the component pattern.
- For this spike you can hard-code the setup of an entities attributes or actions, or load from a file. Up to you

Note: For the component pattern you'll need to update the command manager to be able to "ask" an entity about its attributes or ask it to "do" a certain action. Some general examples:

- `open mailbox` == the "open" action is something the "mailbox" entity has, and it will be called and then change the mailbox (its owner) attribute of "state" from "is closed" to "is open".
- `use potion on dog` == the "potion" has a "use" action that would modify the target "dog" by checking if the dog has a "health" attribute, and then changing it (by, say, +2).