Task 20 Spike: Game Resource Management (Flyweight Pattern)

EXTENSION

Context

Games often use a large number of class instances. Instanced classes are allocated resources in memory, but some of these resources (e.g. some values, almost all methods) are identical to those used in other instances. Using the Flyweight pattern allows for this shared data to be implemented only once, then shared amongst many instances of an object.

Knowledge/Skill Gap:

The developer wants to create the largest Zorkish game world in history - tens of thousands of locations (maybe more) - but can't spare the necessary memory. The developer need to know how to use the Flyweight Pattern to reduce the memory footprint of a large number of objects.

Goals

Building on the work of earlier Zorkish Spikes, split the Location class into instancable and shared data, with the shared data contained in a single location referenced by all the instanced Location objects. Implement a large Location-graph using your old code and your new Flyweight code, and note any memory savings that result.

Expected Output

Repository

1. Code
2. Spike Report

Canvas

1. Spike Report