Game Engine Suggestions

***Teh_cursed_ones** - FIT2099 Assignment #3 2019-1*

Lack of Documentation

PROBLEM:

Internal documentation of the engine itself is sparse, and does not use Javadoc standards anywhere. This increases the challenge of using the engine greatly, as the code cannot be understood simply by peeking at Docstring via the IDE, which may or may not be of utility to the course design. The engine needs to be quite quickly understood to meet the four day deadline in which Assignment #1 is announced then due however, which does seem a tad unfair and unfun - disassembly of code like that provided could be a fun exercise with sufficient time to complete it in.

PROPOSED SOLUTION:

Addition of Javadoc to public methods and classes in the engine, and/or broadening of the assignment #1 deadline to afford time to study and dissect the internal workings of it. Class interactions were fairly clear, logical, and the provided class diagram was quite useful to understanding their interactions.

Visibility Issues in Engine Classes

PROBLEM:

A number of objects within the world and environment which needed to be accessed for game functionality were quite tricky to work around, and ended up having to be solved with hacky fixes.

For instance, teleportation between maps is effectively impossible as map data is stored and accessible only in world, and can't be modified as world is an element of engine code.

PROPOSED SOLUTION:

Adding public getters (and potentially setters where relevant) for variables of contention, such as maps. An easy workaround for any such issues in the current implementation is to simply extend the class and override its methods where certain processes need to be modified or accessed. It's perhaps a good design from the point of view of railroading students into coming up with creative solutions, but also a tad kludgy and not really production quality code.

Unexpected / Uncaught Internal Errors

PROBLEM:

A variety of internal, uncaught errors occur within the engine occasionally. Of course there is the end of game exception (which seems at least partially intentional), but also some issues occur with despawning the player prematurely (and the world trying to enact NPC behaviours on what is now a null pointer) - this exception in particular was quite elusive, as trapping the calling code did not catch the exception.

PROPOSED SOLUTION:

A full audit and rework of the code may be worthwhile, especially given the fact that the original contributor has since left the unit. No doubt though, the fact that the code is both well built enough to understand and work with within the limited time available for the semester, but also lacking in adherence to proper coding standards and oddly created in certain ways that it's provides a workable challenge for students.