Review the assignment due date

Project 6 - MultiSet Design

The project descriptinon is in the PDF located in this repo.

You will need to add any and all files you use for this project into this repository.

Remember to commit your changes regularly, just like you would with a programming assignment.

RTS Army build pros can set army cap; limited types of troops; not given as a specific type on the pdf, may be more unique cons not given as a specific type on the pdf, may be more unique

Different possibilities for the Data Structure Sequence: more or less a linked list: each change will have to iterate through the array to edit the appropriate one Hash table: unique ID per unit can limit size to total possible Unit count AVLTree cannot be a binary search tree in this instance easily; not the best to do

Using Hash Tables to run (an) Army(ies) for Real Time Strategy

Intro & Philosophy:

Working with a data multiset to create the basis to store something viable for a game, I chose to work with the idea of a real time strategy game (RTS). I want to keep army unit maintenance inside a hash table<string, unsigned int>. For this project, I will need the system to be able to flexibly alter unit status to include upgrades, damage taken, health points, cost to build, and unit value (e.g. a Marine may be worth one while a Tank can be worth 6.) As units are damaged and killed I need them removed from the table. I need to implement this for the production company RipRTSOff so that the program can manage its units inside the game environment for an end user gamer.

Core Operations:

The core operations of this bag should work as follows (somewhat previously explained): damage progression/ability, unit cost (both monetary and time), unit army value, total army count, and unit position. This will likely need to be a hash table of hash tables. Most operations should be O(1) as one can, " The indexes of the array are computed using a hash function. hash table structures have a remarkable property: most of their operations are (amortized, average, etc.) constant time, that is, their theoretical complexity is $\mathcal{O}(1)$."[1] For example, first determine the type of unit to be attacked, like a tank. Then, from that hash table find the specific ID of the one being attacked and alter the unit as needed. Problems may occur when 8 units attack at the same time one unit and do way more damage than the total health of the unit being attacked. Also, the unit could already be dead when the damage vector is on route. However, there is one

factor that will be O(N), that is the total army count (if counting the whole army each time.) this can be mitigated with a increment/decrement upon each unit creation and deletion.

Set Operations:

There are several different set operations that may be of use. For example in many games one can creat specific subsets of units in one's army. Also, to compare different strengths, one must also be able to compare where the army's intersect and where they have difference It is also useful to determine what upgrades one army may have that the other does not. For example if there is a scenario where the enemy is supposed to achieve final victory no matter what the player does, the difference one will reveal weaknesses for further unit generation to wipe out the player.

Extension Feature:

In order for game play to be satisfying to the player, there needs to pleasing mecanics, dynamics, and aesthetics. As described in "MDA: A Formal Approach to Game Design and Game Research":

> > Aesthetics What makes a game "fun"? How do we know a specific type of fun when we see it?...Dynamics work to create aesthetic experiences. For example, challenge is created by things like time pressure and opponent play. Fellowship can be encouraged by sharing information across certain members of a session (a team) or supplying winning conditions that are more difficult to achieve alone (such as capturing an enemy base)...Mechanics are the various actions, behaviors and control mechanisms afforded to the player within a game context. Together with the gameís content (levels, assets and so on) the mechanics support overall gameplay dynamics.[2]

Using one unit and another with the addition of an appropriate amount of time, there can be two separate units that merge into a single better unit. For example, if the player has a marine and robotic "AI" armor, the two units could merge to form the "starship trooper ape unit or at least a differently named facsimile of one. With this action, it gives another avenue for the player to progress in power without the creation of new units when unit capped for total supply. With the addition of more types of armies, this ability can be added in different versions to each. For example, one could be available early in the game to an army, but it would not have as much added strength as that for the army(ies) that have to wait longer to gain the ability.

[1] S. Tapia-Fernández, D. García-García, and P. García-Hernandez, "Key Concepts, Weakness and Benchmark on Hash Table Data Structures," Algorithms, vol. 15, no. 3, p. 100, Mar. 2022 [2]R. Hunicke, M. LeBlanc, and R. Zubek, "MDA: A Formal Approach to Game Design and Game Research," in Proceedings of the AAAI Workshop on Challenges in Game AI, San Jose, CA, USA, Jul. 2004, vol. 4, no. 1, p. 4