Data Design Analysis: Micro-Assignment 1

Defining data design for this analysis: Data design is a methodology used to design a game with optimal architecture giving the developer the maximum amount of control over a variety of data sets within the game.

My intention was to explore this concept and practice building an efficient game architecture from a coding perspective, as well as to develop a strong understanding of when to group data together and what data sets should a developer have control over.

When tasked to design a turn based combat game around the foundation of data handling and data design, I decided to create a card game. The use of cards seemed to be the obvious mechanic when thinking about data design, as cards can contain a lot of different data sets in a simple mechanism.

The game is a turn based card duelling game, where each player has three separate life bars, cards either damage or heal one or more of these life bars. There are two types of cards; attack cards or defence cards. Attack cards deal damage and defence cards heal. I chose three different life bars as it would allow for more opportunities to practice data handling, as it introduces 3 separate data groups. It also seemed an interesting mechanic that I have not seen used often in a card game(but then again I need to explore more). The design of the game also let me practice using different types of data handling systems.

When designing the game, I focused on data handling as well as incorporating the turn based combat system as required. The Data sets that I have in my game are;

1. the number of attack cards
2. the number of defence cards
3. the total number of cards
4. the maximum value of each one of the health bars
5. attack and defence value modifiers
6. all cards individual stats

These data sets are the ones I have full and easy access to. To add and remove cards is a simple task, for either attack or defence card. The total health of the player and enemy can be modified on the player/enemy prefab. The attack and defence value modifiers change the applied value on the cards to the player. This helps with balancing as I can adjust how each attack or defence stat effects the player/enemy. These 'balancing numbers' are floats. This reduces the granularity between changes giving me fine and accurate control over the changes.

I have used 2 data design systems which are used to control certain aspects of the game. There is a player controller system and a combat system. The player controller system gives access to all relevant player/enemy data, such as the maximum value of each life bar as well as current values(In the current build, the enemy starts with 50% life in each life bar, this was done as I noticed it took long to complete a game. In future iterations the cards should be balanced). The combat system controls all combat interactions and how the card data interacts with the player or enemy data. This is the core data handler of the game. It is the central collection of data handling, that I can use to manipulate the game data.

A technique I used to help with data management was object orientated coding. This type of programming allows elements of the same nature to be linked and made easily. For example, each card object stores the same type of information, therefore it only needs to be scripted once for all cards. Adding extra functions and parameters to this script allows easy data controlling amongst all cards. It is now possible to access each cards data as well as create them simply.

As this is a base prototype I feel I managed the data available to me well. There is data in every aspect of the game, and I could spend countless hours trying to collect and organise it all. I can already see ways to improve the data handling if this game was to be developed further. Logging all stats that each card holds to get a mean, median, range as well as frequency of occurrence could be valuable information when trying to balance a complex card system. Tracking information, such as how many moves and time it takes to complete a game, could tell us designers how players interact with our game. This information could be used to manipulate the game play and players in a direction that the designer wants the game to be played.

Looking back at one of my goals of data designing; ‘to design a game with optimal architecture giving the developer the maximum amount of control’. I’m not sure I achieve this goal. Although my game does work and my data management is above sufficient, my code is messy and not as optimal as it could be. A part of this problem is due to lack of programming knowledge, however, I think a larger part is due to not having a clear idea of what my core mechanics were and how they would interact with each other.

I did not follow the MDA framework, I started with the aesthetics, as that is where the excitement lies for me, therefore I did not have a clear picture of the core mechanics. A resolution to this problem would be to use the MDA framework, and to spend more time with pen and paper understanding what needs to be done. This would force me to have a strong idea of how the core mechanics of the game work, giving me a clear picture of what I need to achieve with my code. This will lead to an improved more optimal core architecture of which the game is built.