William Caldwell

1/22/2023

# Project Specification

This will expand upon the Strategy Game project idea. To get the obvious thing out of the way this project would be to make a game as per the purpose for the program. Otherwise, it would be proof of work as per the context of the class. To state the needed specifications for the project, include a use of a database, and API to access it and at least one application.

As for how the game will be played. It will be a tuned based strategy game where each match played on a square tile board that will be no smaller than 5 tiles by 5 tiles. Initial planning will be around the idea that all the units will be set up in an initialization phase for ‘turn 0’ where the game will play from there. Any specifics of units, ability’s, spells shall be determent at a later date. Any such as implementing a card game style twist or an additional strategic layer where multiple matches will be played in a single game will be stretch goals. The question right now is if multiplayer would fall under the stretch goals.

To look for reference text the first and most applicable one that comes to mind is one about good database structure. This good structure was learned by Blizzard while making the popular World of Warcraft. As per this instance of the developer speaking about it. In the “How Engineering has changed” section.

<https://worldofwarcraft.com/en-us/news/21881587/dev-watercooler-world-of-warcraft-classic>

The core of what they say is because of how they structured the database spells were limited to only having a max of three effects. On account of that being limiting and inefficient with data entries taken up by nothing this was eventually changed.

As for games more applicable to Turn base strategy games there are quite a few strategy series about that can fit the bill from Fire Emblem, Advance War, and X-com. To name some popular ones. However, to play to the concept for the project a dead game by the name of Duelyst whose source code has gone open source can be looked too. Not for the open-source nature of the code which is the least important thing but instead the fact it has two successors Duelyst.gg, and Duelyst II. As for why that is important is because they both fork from different points in the original Duelyst development. While Duelyst is a hybrid card game and strategy game seeing where its limits and how its mechanics work can be applied to the project.

Now then for some constants that will be required too regardless of anything else. The first is inputs, for one it would be purely Graphical User Interface (GUI) base with at most a search bar during the initialization phase. As such most of the outputs would be updating the GUI with what ever the Logic engine calculates. The Logic engine is certainly going to be the hardest part and can only get harder if a poor data base format is used. As for what the logic engine would need to hold during a match would be anything a strategy game would be expected to keep track of. Such as but not limited too Unit Health, Unit Attack, Unit movement range, Unit Ability’s, Spell effects, and Tile Effects.

As for the database it should keep in mind the one spoke about in the article above. With it being split up and made to be flexible. While such might cause some data entry errors it is probably for the best to keep with that style even if the project does not reach the size of when the advantages of such a set up would show.

As such that sums some of the more Specifics needed for the project. While more could be expanded upon what that would be are mostly subject to change or something to put forward later such as recommending to try and follow something like Linxu coding styles spoke about here.

https://github.com/torvalds/linux/blob/master/Documentation/process/coding-style.rst