Computer Science Department

Capstone Project Proposal

Kerbal Engine Optimizer

12/13/2020

Matt Jones

CSC 520

CSC 521 Project Advisor

Dr. Bo Hatfield

Contents

Student Objectives

This project is intended to achieve a few goals beyond the basic creation of a useful application. By the end of the process I would like to say I have learned several things that will be able to help me in the future, whether that is in the job market or in creation of other apps down the road.

The most basic objectives of developing this project are to increase my experience with Java as well as learning how to set up and implement GUIs. Also, I would like to learn how to create databases such as Amazon Web Services using DynamoDB, which can be a crucial skill to have in the job market. After the completion of this project, I will have experience with using AWS and it would be beneficial to be able to back that claim up with a well-done database.

Another goal of doing this project is to gain experience by going through a complete life cycle of project development, from initial idea, problem specification, creation and then verification. This will be extremely useful given that every application will go through the same concept, albeit different in its own way.

Problem Specification

The overall objective for this project is to increase user's(and my own) enjoyment of a game called Kerbal Space Program(KSP) by streamlining rocket design. KSP is a fun game to play but getting the correct engine and fuel amount can make things difficult at times. The point of KSP is very similar to that of NASA or SpaceX(just much simpler): you make rockets from in game parts and explore the solar system.

Kerbal Engine Optimizer is a program to support playing of KSP where the main function of the app is to assist the user in the planning and building of a rocket. The largest issue when building a rocket is knowing what you need in terms of fuel and engine thrust to get from point a to point b. Too large of an engine means more fuel and more structural parts to keep the rocket stable during flight. Too small of an engine means you can't generate the thrust required to move your rocket. Then comes fuel, too much fuel means you need a bigger engine to be able move that much weight. But if you don't add enough it means you fall short and get stuck without fuel in the middle of space, which can be just a little problematic. Therefore, you need to find a happy middle ground, with the right engine for the right job and just enough fuel to get to your destination and back.

In order to pick the best engine, there are many calculations that need to be done; even for a simple game which ignores many things that real-world rocketry would need to take in to account, such as wind speed and weather conditions.

For example, if you want to put a satellite in orbit, you build your satellite with all the gadgets and gizmos you want, its total mass will be your payload mass. In order to put that satellite in orbit at 100k meters around Kerbin(the main planet) for example. It will take at least 3400 Delta-V(DV) plus 10-20% to be safe and a minimum of 1 Thrust to Weight Ratio (TWR) when at sea level to lift off. Now you can just pick an engine, add fuel and hope for the best or do the math yourself. To start the process of choosing the best engine, you need to figure out how fuel efficient an engine is at different atmospheric pressures using its specific impulse(Isp). The higher the Isp the more fuel efficient the engine will be in that environment. At sea level the pressure is at its highest and lowers the closer it gets to space; your calculations need to account for this by find the "atmospheric curve". An engine might have a good sea level Isp, but the curve is too drastic to allow it to reach space before running out of fuel. Most don't operate to that extreme, rather more efficient in either-or, but not both. After finding the atmospheric curve for the engine, you would need to find out how much fuel it's going to take. The heavier your rocket is, the more fuel it will need. If you add too little you will fall short of your destination. If you add too much your rocket might be too heavy to achieve liftoff, let alone make orbit. So, you must select a careful balance of enough fuel to still give you the required DV while keeping your TWR high enough to liftoff. If you're lucky your first engine pick and fuel amount will work. If not, either your fuel amount is incorrect, or you need to select a better engine for the job and redo everything from the beginning. The latter is the more likely outcome. If you are extremely unlucky and no engine that you have tested works, you now need to see if using several of the same engines would fit your requirements. It isn't an uncommon occurrence to have several engines working together to lift off, especially on larger rockets.

This is where my app comes in: you find your payload mass, your desired DV and TWR range, where this maneuver takes place, (each planet has its own gravity and atmospheric pressure to take into account) and maximum number of engines you would be willing to use. The app will do the thousands of tedious calculations, graph the results and allow you to select the engine that gives you exactly what fits your needs. Users will be able to select which engines they want tested in case they want to compare a few. Users will also be able to create custom engines if they downloaded a separate mod that had more engines or changed engines stats in the game's files. The app will also allow users to create accounts that will enable them to see their recent engine tests, create and save custom engines that will be stored on a database. Lastly, it will help the user create a stage planner for longer, more complex journeys. The app however will not require an account and will be fully usable without an internet connection.

Requirement Collection and Analysis

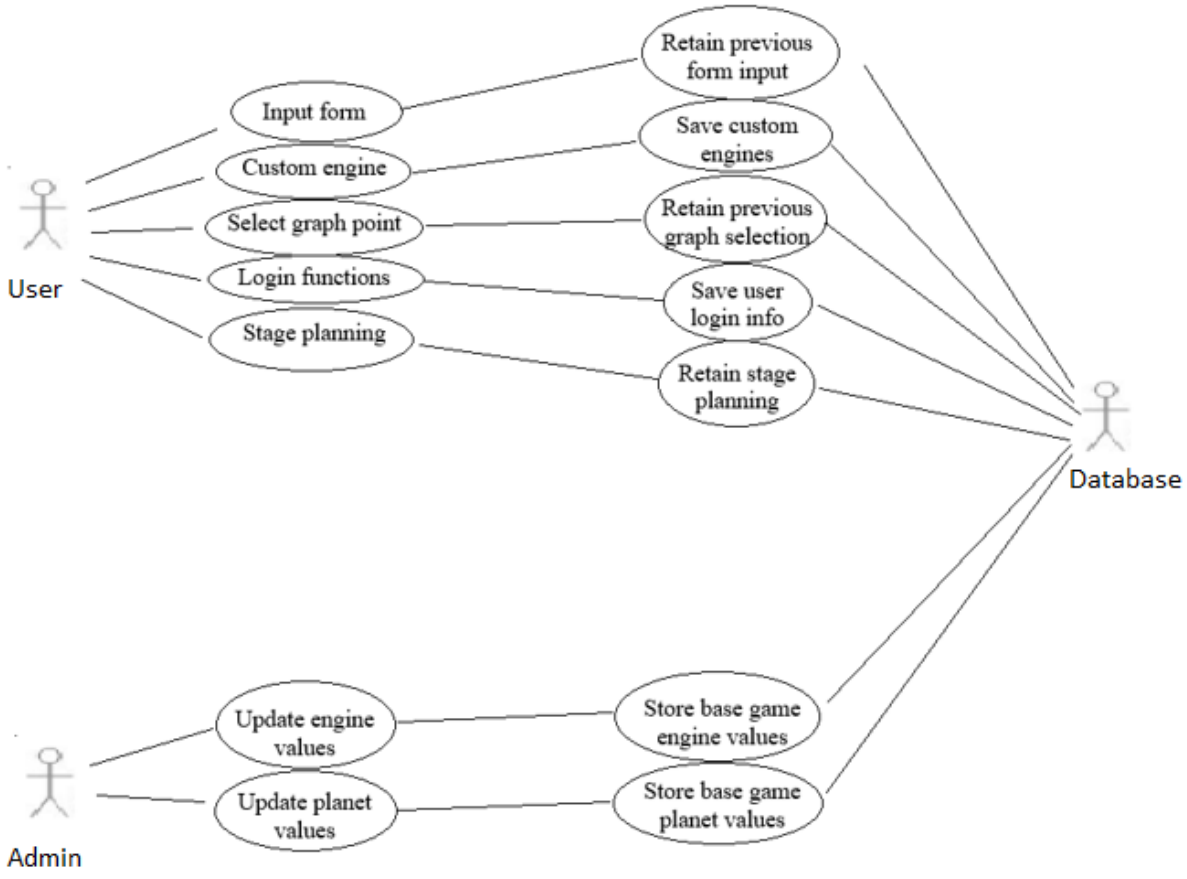
- Stakeholder Identification
    - Standard user – myself and friends, but available for anyone who plays Kerbal Space Program
    - Admin – role that monitors KSP game updates to ensure engine/planet values match the in-app values and updating the database if there are changes.

- Requirement Lists for Requirement Specifications
    - Users will be able to compare engines
        - By selecting their desired engines and fill out the form that captures payload, max/min TWR & DV, gravity, atmosphere, and max number of engines.

    - Graph the results
        - Results of the calculations will be graphed and allow the user to select a point on the graph to find the best engine for a specific DV/TWR point value.
    - Create and save custom engines
        - Users can create custom engines that will be saved to a database allowing them to test for engines that aren't in the base game and/or if they changed the game files themselves.
    - Login and general user functionalities
        - Users will be able to create accounts that allows them to log into the app, which will save recent comparisons, custom engines and stage planner to a database.
    - Admin
        - Admin will be able to log on with privileged functionality allowing them to update the database in the eventuality that the Kerbal base game updates engines and/or planets values.

Solution Process and Design

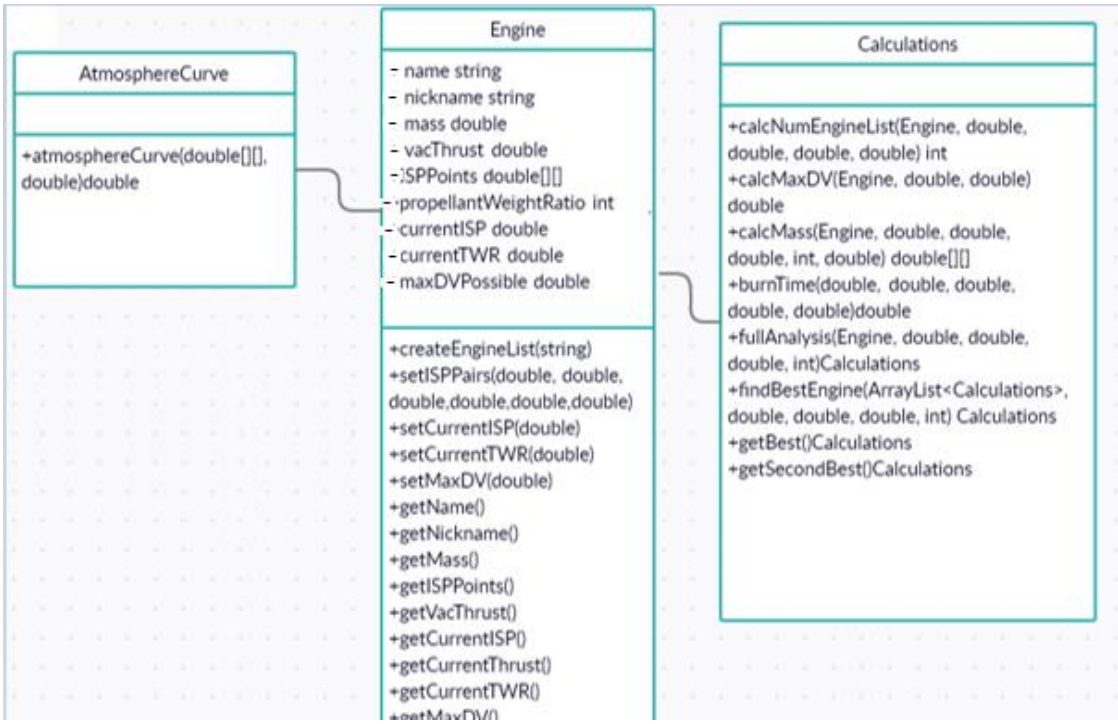
Sequence Diagram



The sequence diagram is broken up into three parts: user, admin and database. The user will have several functions available to them: an input form to allow them to specify their requests, select a specific point on the graph, create an account to use login functionalities, create

custom engines, and a stage planner. The admin will have special access in the app that will allow them to update the database when/if the in-game values for engines and planets change. The database will be responsible for storing the user's general info, their custom engines, previous form input and stage planning, as well as in-game engines and planet values.

UML Diagram



This UML is for the base calculations that every engine will go through. At app start up every engine will create its own Engine object that will store the various information needed. Once the user has filled out the form and selected their desired engines to test, each engine will be sent to have its atmospheric curve found for their requested planet. Then the users selected engine(s) are sent to calculations to determine the needed number of engines, the fuel requirements etc. When that is done the engines are ready to be graphed in the UI. Finally, when the user goes to select their DV/TWR point on the graph it will return the best and second-best engine for that selection.

Database Design

The database will be non-relational and straightforward, storing a small amount of user info, just enough to log in and a way to verify an account if a username and/or password is forgotten. Each planet and engine will be stored for easy updating when/if the in-game values are changed. Special admin privileges will be created to allow specified users to update the database without having to access the database itself.

Benchmark Specifications

Three benchmarks for my project can be identified as: core functionality, specific user content and database set up. Breaking the project down into these groups will help to keep the project moving forward at a measurable pace. Each benchmark will contain its own testing and bug fixing as needed, with benchmark three and the final weeks containing the last overall testing and bug fixes.

Benchmark one

First part of the project will be to set the core functions of the app, after this the app should be able to work at a basic level. The core components would be to handle the engine and planet values, these values will be stored in a document. Later being able to take updates directly from the database. This will allow the user to be able to use the app with or without an internet connection. Next is the handling of the users input for their staging requirements and engines to test, then doing the calculations on those provided values. Finally, the results should be stored locally and set to be graphable. Create the KSP home GUI that allows user to select engine(s), provide requirement input form and graph the results. After this benchmark is completed and tested, the app should be in a limited but usable state.

Benchmark Two

Benchmark two will focus on the database construction, it should be expected to store the base games engines and planet values, general user info and user specific custom engines limited to 10(number may change during creation). Admin account(s) will be created allowing certain users access to the database from the app. Allowing them to update the database in the eventuality of game updates that change in game engines and planet values.

Benchmark Three

This benchmark will be all about creating the specific user functions, account creation/login/out which will allow a user to create and save custom engines. This will be saved on a database only, so the user must create an account and be able to log in to access this functionality. The app will also temporarily store the most recent 5(number may change during creation) user staging requests and their DV/TWR selection point if they chose one. Then finally the GUI creations, one for the custom engines that will allow the user to input the engine values. Another tab for account viewing, login, logout, etc. Bonus for this benchmark will be making a stage planner if it is feasible to do so accurately. This very well may not be feasible due to building differences that will compound errors in weight and DV planning the more stages you add to your flight plan. Once this section has been completed the app should be finished and any remaining time will be devoted to testing and bug fixes.

Benchmark Four

This is a bonus benchmark but will be kept in mind during the creation of the other three benchmarks. KSP 2 is currently in development but is not scheduled to be released until 2022. There will likely be differences between the games, but the app needs to be created in such a way to be easily updated when the next game comes out. Time permitting creating a GUI for KSP 2 and setting up preparations in the database to handle separate engines and planets values without interfering with KSP 1 values. This benchmark can be spread up between the previous ones or at the end depending on how well each benchmark goes.

Tool List

- Front-End Development tools

    - Java: will be the main language used
    - CSS: only small amount might be needed for extra GUI control
- IDEs
    - NetBeans: Will be the main IDE used to write the general backend code which will all be done in Java.
- Hardware Tools
    - Amazon Web Service: My database is non-relational and currently planned on using AWS. When it comes time to start/prepare to start building it if AWS doesn't fit my needs, I will find something else to use.
    - Hardware: This app is intended to be used primarily on a PC due to the majority of KSP players do so on a PC instead of a Mac. Though, as its written in Java, any PC or Mac should be able to run it, if Java is installed.
- Version Control
    - Git: This will be my primary version control due to it being only me working on this project.
    - GitHub: I will also be using GitHub in conjunction with Git in case I switch computers, serving as a backup storage for my project and just to get general practice. This will also allow my advisor to see my progress.
- Other Tools
    - Scene Builder: will be used to create the GUIs for my app
    - Trello: will be used to track my progress

Time Schedule

This project is expected to be completed in 14 weeks, so having a set expectation on how long each benchmark is expected/planned to take is important to the overall competition of the project. Each benchmark will be debugged during its allotted timeframe and during the last two weeks final testing and debugging will take place.

Benchmark One

Weeks 1-5: This will be the core of the program, it will be the hardest and most important section to complete, because without it the rest of the app is pointless. Therefore, this section has the most time allocated to it. After this section is completed the app will be in a usable state, albeit missing features.

Benchmark Two

Weeks 6-8: Will be the shortest section to build, due to it mainly focusing on just building the database. It will be used for storing user's info, base game engines and planets as well as users' custom engines.

Benchmark Three

Weeks 9-12: This section will focus on increasing the user's available functionality by giving them the ability to create accounts, engines and view recently done staging requests. These alone should be relatively straightforward. The more time-consuming portion would be the stage planner and this section may need stripped from the final product as whole. I would like to have extra time to devote to trying to find a way to make it work conveniently but more importantly accurately.

Week 13-14: Will be spent doing final testing and preparing the necessary items to turn the project in and the creation of the presentation as well.

Benchmark Four

Pending: If possible four will be split up into parts and done in their corresponding benchmarks. The more that gets done now, the less that will be needed to be done when KSP 2 comes out and the more time I would get to spend playing it instead.

Grading Scheme

The entirety of benchmark one should be worth 45%, because without a stable fully functioning base app, the rest of the benchmarks aren't worth doing. This will mainly focus on the retrieval of engine and planet values, the base calculations needed for comparisons, taking user input and displaying the results.

Benchmark two would be worth 25% because without a functioning database there is no reason to create benchmark three. Each portion of the database is equally important so base game engines and planets data base is 15%, users account and custom engines will be done together for a combined 10%.

Benchmark three in total should be worth the remaining 20%. User accounts would be worth 10%, custom engines 5%, followed by staging planner at 5%.

The final presentation should be worth 10% because at the end of the day the most important aspect of any application is the app itself.

Deliverables

The deliverables for my project will be as follows

- original proposal and presentation file(s) (from CSC 520)
- amendments to the proposal (approved by the project supervisor)
- system architecture diagram(s) (UML, DFD context, etc.), enhanced with details determined during implementation
- appropriately commented source code
- documentation of project functionality (test results, screenshots, video capture of project execution, etc.)
- sample output (screen shots and/or reports)
- executables and/or projects
- presentation documents (used to support the presentation of the completed CSC 521 project), including any presentation file(s)
- project journal: a narrative of the progress of the project, in clear, concise English, including any problems encountered and how said problems were addressed
- project post-mortem: a summary of what was learned from the project and (based on that experience) discussion of how various aspects of the project might have been approached differently
- a list of what areas of the proposal (if any) were not completed, including reasons why
- presentation of the completed project (PowerPoint format), including screenshots of the functioning project
- user's manual