Cover page:

Good morning everyone, my names Matt Jones and my capstone proposal will be for an app called Kerbal Engine Optimizer. Its an app that will support Kerbal space program or KSP for short, where you are kind of like NASA or SpaceX these days, your job is to build a rocket and visit other planets. While the game is simplified compared to real life standards it follows many real-world rocket and space travel physics, and the underlying math behind them.

Overview: just read

Objectives: just read

Problem specifications:

Building rockets is easy, anyone of you here could slap together some parts and call it a rocket but whether it can make orbit, let alone take off is an entirely different matter.

I don’t want to go into a lot of details about all of this, but it is important to understand a few things when it comes to rockets and space travel in general. two of the bigger contributors to your stage planning is change in velocity or delta-v and thrust to weight ratio, TWR for short. TWR is more or less your acceleration the higher it is the faster you can change your velocity and delta-v is essentially how much fuel you have. Unlike on earth where we have air resistance and friction when driving, in space there isn’t any of that. So, once you let off the gas in space, however fast you’re going is how fast you’re going to go for a very very long time, orbit pending of course.

Which is why understanding delta v is so important your relative speed to the body you’re orbiting will determine your orbit around that planet. Speeding up is known as prograde, this will increase your distance around the planet or body you are orbiting and decreasing your speed or retrograde is going to reduce your orbit. So, when it comes to maneuvers say you want to increase your orbit or even break orbit to transfer to another planet you will need to burn prograde until you increase your velocity by lets say 1000 m/s. that means its going to cost you 1000 delta v in order to execute that maneuver. If you only have 600 delta v left that means you wont be able to execute the full maneuver which very well could leave you stranded in space somewhere.

It would be pretty awkward to be on your return trip from being the first person to land on mars, only to realize you don’t have enough fuel or *delta v* to finish your capture burn around earth, instead you sling shot past never to be heard from again. That’s a bad day if you ask me, so to avoid that, when trip planning you need to ensure you have enough delta-v to get to your destination and back if that’s the plan. People have made delta v maps as it’s called to help with that. While that part sounds easy enough the hardest part of executing that plan is finding the right engine and fuel amount to get the job done.

Every engine has a differing maximum thrust and specific impulse or ISP which is your fuel efficiency in a given atmosphere, some engines work better in atmospheres, while others work better in the vacuum of space.

That being said it is important to pick the right engine for the right job, Much like how you wouldn’t go furniture shopping with a motorcycle or do your weekly grocery trip using a semi, the same concept is applied when choosing the right engine for your rocket.

But in-order to make an informed decision you have to do some math, then some more math and once that’s done, do even more. Then assuming you actually selected the right engine the first time you might be good to start building your rocket. Just as a caveat because it works doesn’t mean it’s the best engine for the job though.

In the more likely scenario though you selected the wrong engine and you must scrap everything you did and start over again. Doing all that math is tedious at best, border lining insanity if you have to test every engine only to find that none of them will work, which is a very real possibility.

And ALL of what we just did, all the planning and calculations and such was for one segment, one stage of your journey. Depending on what you’re doing and where you’re going you could have a half dozen or more stages that need to have the same delta-v planning, TWR and engine selection done.

This is where my app comes in, instead of having the user painstakingly do the math themselves or just guess at which engine is best, they will be able to select what engine or engines they want to test for. They will have an input form allowing them to request a range of delta v and twr values, what planet this will take place on or around and then have the results calculated for them in a matter of seconds. The results will be output to a graph so the user can pick a specific delta v and twr point to see detailed information, such as the best and second best engine, how much fuel is needed and what the total mass of your rocket will be after completion for example.

The user will also be able to create an account that will allow them to create and save custom engines. This will allow users to test engines that aren’t part of the base game if they installed mods or in case the user has directly edited the values of the engines that are in the game.

Sequence diagram:

The sequence diagram is straight forward-you have the user, admin and database. The user will have an input form for their trip planning, select a specific point on the graph, create custom engines, log in and stage planning. The admin is a special role that allows them to update the database directly for the eventuality of game updates for engines and planetary values. Everything that the user and admin do will be reflected and stored on the database.

UML/Database:

Here is the uml for the engine creation and calculations to find the best engine. Starting things off every engine will be turned into an engine object that stores the values that will be needed to perform the necessary calculations, such as max thrust, mass, ISP points, and current ISP. which will be found out by setting the atmospheric curve, which I briefly mentioned earlier about how fuel efficient an engine is in a given atmosphere. Once that’s done it takes each engine and calculates everything needed to plot the graph. After it’s plotted the user selects the delta v and Thrust to weight point on the graph, which gets passed as parameters to find the best engine at that given delta v and twr.

The database is simple, it’s a non-relation database simply storing the user’s username, password, email and custom engines. The planets and engines will have their own respective sections that store the needed values.

Benchmarks:

Benchmark one is going to be the core functionality, it will handle user input, do the calculations, output the results on the graph and allow the user to select the delta v/twr point.

Two takes care of the database side of things, one for users’ info and custom engines, another for in game engines and the last for planetary values.

Three is all about the user functions, input form to create an account that stores their basic info and input form for the creation of custom engines.

Four is a bonus benchmark that if time allows will be laying the groundwork for Kerbal space program 2 which isn’t scheduled to be released until sometime in 2022 now. But the more I do now, the less time I will have to spend building it when the game comes out and the more time I can spend actually playing the game.

Tools: read from slide

Time schedule:

Benchmark one will be the first through fifth week, due to it being the larger more difficult one, it gets the most time. But after it gets completed the app should be in a workable state.

Two is the sixth through eighth week, when done will allow the app to directly pull information from the database as well as have a database for users and their custom engines ready to go.

Three is ninth through twelfth week, afterwards the input forms for user account creation and custom engines will be done, only needing to attach it to the database to save and retrieve information. After this the app should be fully functional.

The final two weeks will be for bug fixing, report generation, and presentation.

Benchmark four will be split up amongst respective time slots or as time allows.

Grading: read slides

Grading revised:

Benchmark one will be 45 percent due to its importance, after its completion the app should be in a usable state, if its not, it makes things much more difficult down the road and could even make the rest of the project not even worth doing.

Two will be worth 25 percent, 15 going to planet and engine database and 10 to user and custom engine database

Three is 20 percent, 10 for user account work, 5 for custom engine input and 5 for a stage planner

The final 10 percent will be for the presentation

Deliverables: read from slide

Prototype: just go over it