**Project Proposal**

**Project Description [2.5 pts]:**

Name: GraphIt

Description: An interactive graphing calculator that graphs functions and derivatives. It will also contain a 3D graphing feature and a drawing function feature where the user will draw the function and the program will plot the function is (post MVP).

**Competitive Analysis [2.5 pts]:**

I have only seen one similar project so far and that is the Graphing calculator project from the 112 TP gallery (Fall 2020 Semester). This project has a feature that I like which is a keyboard for typing in mathematical operations in the functions (such as square root, square, trigonometric functions and fractions). I will be using a similar keyboard in my project, my layout and design of the keyboard will be different. Also, this project is strictly a 3D graphing calculator, while mine will also include 2D graphing and open CV features. Next, this project uses dots to graph functions (so for every point in the function it draws a dot). My project uses lines to graph the functions (so it draws lines connecting the plotted points in the function). Finally, the overall design of my project will also look very different visually.

I am basing my project design off Desmos. For my 2D calculator, the layout and the functionality (zooming in and out and typing in numbers) look very similar visually to that of the website.

**Structural Plan [2.5 pts]:**

Main Folder: Graph Calc

* Folder: Images (includes all images used for design)
* File: Cmu\_112\_graphics.py
* File: main.py (main file)
  + AppStarted
    - – initiating variables
  + drawFirstScreen
    - draws initial screen (front page)
  + drawGrid
    - draws grid used in 2D feature
  + drawScale
    - draws initial 2D grid scale
  + drawButtons
    - draws every button in the button dictionary in pushed or not pushed position
  + plotFunction
    - stores x and y values of the function points
    - graphs function by drawing lines though those points
  + defineButtons
    - stores all of the buttons and their positions in a dictionary
  + is PointInButton
    - determines if click is in a button
  + mousePressed
  + mouseReleased
  + zoomIn
  + zoomOut
  + keyPressed
  + inititateScale
    - generates changeable scale for the graph
    - draws scale
  + changeScale
    - changes scale based on zooming in and out
  + redrawAll
* equations.py
  + predefines the equations and what they should return (ex:. x^2 return x\*\*x)
* buttons.py
  + button class that determines if buttons are pushed or not
* grid.py
  + grid class that determines all the operations with respecting to drawing the grid and the function plot
* open CV (post MVP)
  + will determine when the person is drawing a function with finger

**Algorithmic Plan [2.5 pts]: A detailed algorithmic plan for how you will approach the trickiest part of the project. Be sure to clearly highlight which part(s) of your project are algorithmically most complex, and include details of the algorithm(s) you are using in those cases.**

User Input:

Getting user input and interpreting it is going to be one of the harder parts of my MVP. It will be challenging design wise as the python user input box is not visually appealing, so making it like the one in Desmos will take a lot of drawing. Interpreting the user input will also be hard. I will approach this by using string parsing and split up the strings accordingly so that each function is calculated separately and then combined later.

Drawing Feature (MVP):

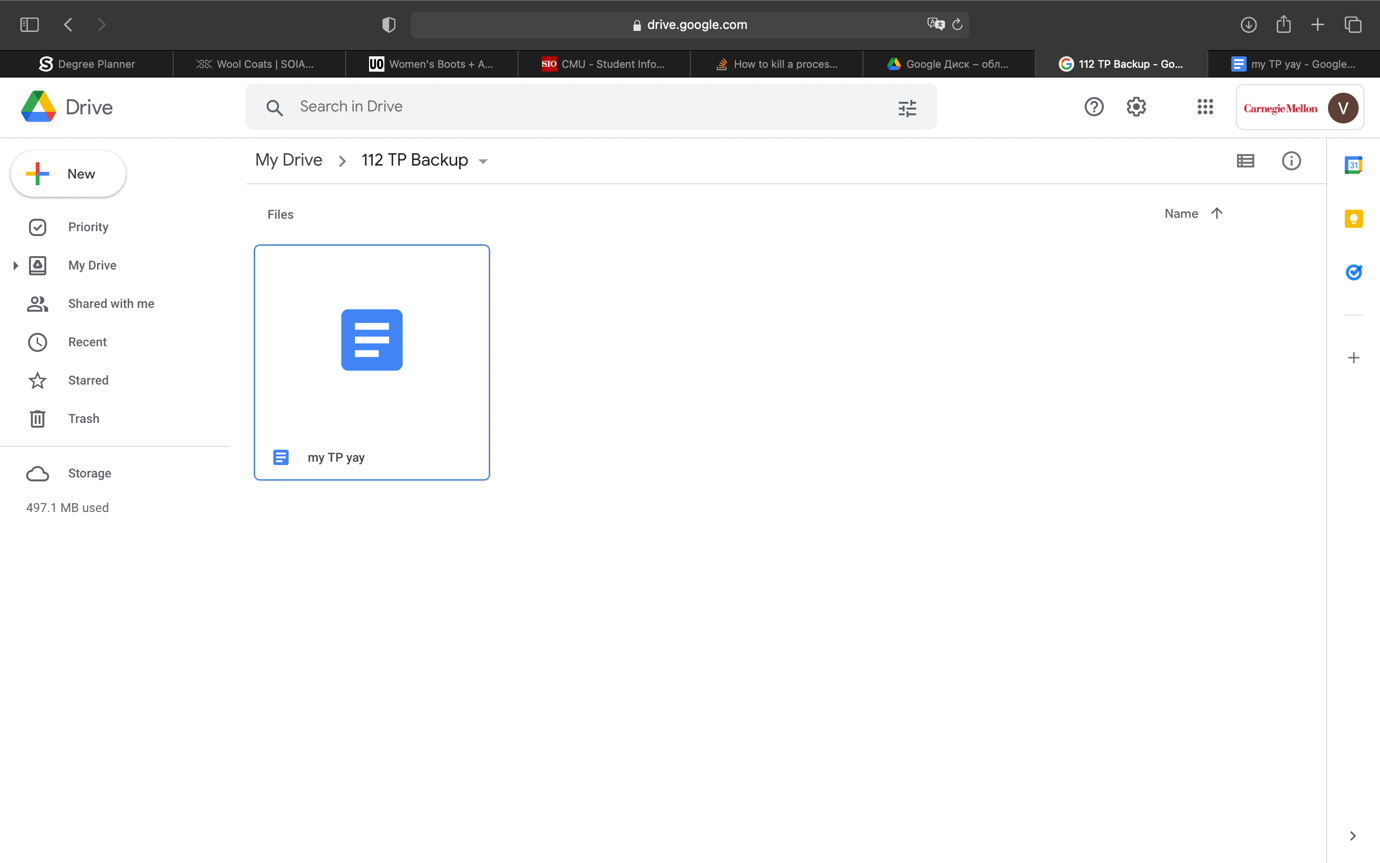
The drawing feature using open CV is the hardest part of my project. I am not sure how I am going to approach this algorithmically, however; I think I will be using some form of image parsing and hard coding in function ranges and things like that. I have not spent much time thinking about this portion, but I will once I reach MVP.

**Timeline Plan [2.5 pts]:**

I expect to complete the 2D feature of my project by MVP, however it is likely that I will be done sooner. Once I am finished with my MVP, I will next work on the 3D feature of my graph. I am currently looking at how to use 3D graphics so I will likely be able to complete it either by MVP or 1 or 2 days after MVP. After this is completed, I will then move on to working on the open CV feature. Hopefully I am able to compete this by December 1st.

**Version Control Plan [1.5 pts]:**

I am using version control by saving my code in Google Drive. After every session, I copy and paste the code from my editor into my backup file. I also save my code as a zip file and upload it into the same folder.

****

**Изображение выглядит как текст

Автоматически созданное описание**

**Module List [1 pts]:**

Now, I am not using any modules. After MVP, I plan on using 3D graphics and open CV.

**TP2 Update:**

My design remains the same for the 2D graphing calculator. Now, brackets need to be used around the x for functions like sin(x), or else the code encounters an error. I am also currently using eval to render function points for the plot, however, I will implicate string parsing for TP3. Additionally, functions that have asymptotes (such as 1/x or tan(x)) currently have a large gap at the point where division by 0 occurs or the two dots plotted before and after the “error” are being connected with a line. I will be fixing this by increasing the number of points plotted and possibly coding in limits, and by adding a “checker” in my plotting function to see if the line connecting two points exceeds a certain length so that the dots do not get connected. I will also be adding a screen that shows the plotted function and an “add function” feature that will allow the user to plot more than one function at once.

I have also added most of the 3D feature to my calculator. The keyboard issue is also present in this feature. I am currently plotting only dots in this function; however, I will be adding lines to connect those dots and possibly color in the planes to make the graph easier to see and visually nicer. I will also be adding a z axis to make the graph clearer and possibly a scale.

I have also started thinking about the openCV-function-identifying feature of my project. I have not started coding it yet, but I believe that I will be splitting the 2D grid into little squares and looping through them to see which squares were drawn in and identifying a pattern (maybe numbering the squares and storing the ones that were drawn in). I will also have a data base for common functions and what pattern they follow. I will then loop through the data base and see which function “most closely” resembles the one that was drawn and then identify how they are different. This will also need to include a list of graph transformations and other possible changes that could have been done to a graph. Once my 3D and 2D features are fully functional (hopefully by Wednesday the 24th), I will begin to look at this more algorithmically.

**TP3 Update:**

My 2D and 3D features of my project are fully functional and complete. I attempted to do the OpenCV feature that I planned, and I got as far as identifying which squares were “drawn in” however it did not work out since it needed a lot of machine learning based algorithms which I did not know, and which were too complicated to learn in only 3 days. I removed this feature from my TP. Other than that, the program now has derivatives and a sped up version of the matrix.py file which makes rotation of the 3D graph faster and smoother.