Productivity Analytics Part B

P.A.T.H Finder

[Dylan Cathapermal](mailto:dcathap@g.clemson.edu)

**Introduction**:

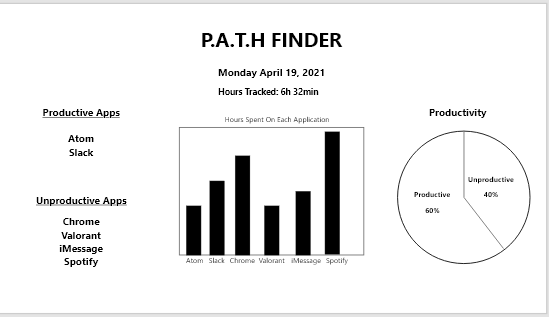
Productivity Analytics Part B was an implementation of the desktop/webapp for the Productivity Analytics Part A project. There were many things that went well, and I learned a lot in this project. However, there were many things that did not go as planned and I would still like to implement.

The final product, P.A.T.H Finder is a productivity analytics tool that helps users see how many hours they spend on their apps, and how many hours they spend on productive/unproductive apps.

**Design:**

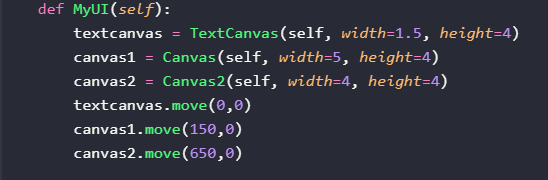
The design of this product was first determined during the wireframing stages. Since I began implementing color in this iteration of the project, I went with very sharp colors, blue and gold, and made sure they were contrasting. That way it would be very easy for users to determine whether the majority of their time was spent on productive apps or unproductive apps.

I used tips from Josef Muller-Brockmann’s book on Grid systems in graphic design. That book is what led me to design a “grid” system. I wanted to split up the different data into sections. That is why I put the app names on the left, and the graphs in the middle and right. Blocking the data makes it a lot easier for users to know which section they want to look at.



**Build:**

The build of this project was using python’s desktop app PyQt5 and plotting data with Matplotlib. The PyQt5 library has variables that you can use with Python’s window class that build desktop applications. In this class window I set up some default values for the windows size and defined the app title. From there, I created a buildUI method that would initialize the pyqt figures and let me make changes to their canvases.



This is how I defined the different canvases being used. As you can see, they are split up into 3 different canvases, just how I wanted the app to be split into three different blocks (grid view-layout).

An example of how the canvases were defined is shown below.

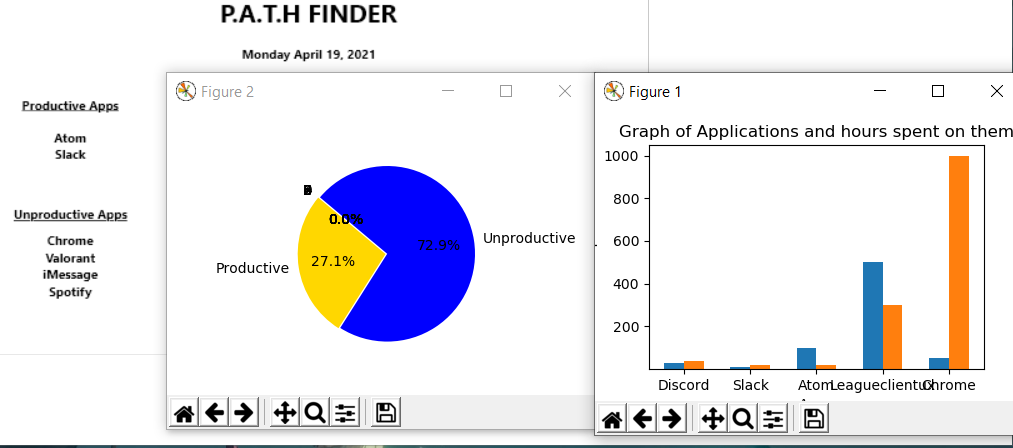


This is the most important canvas because it setups up the time signal. This signal is defined in the canvas, and then calls signalExample. signalExample is how I am able to grab every process that is currently running on my PC (using win32). The way signalExample is able to grab the running process every second is because in the workerthread run method there is a time.sleep method which causes the GetWindowThreadProcess to repeat every second.

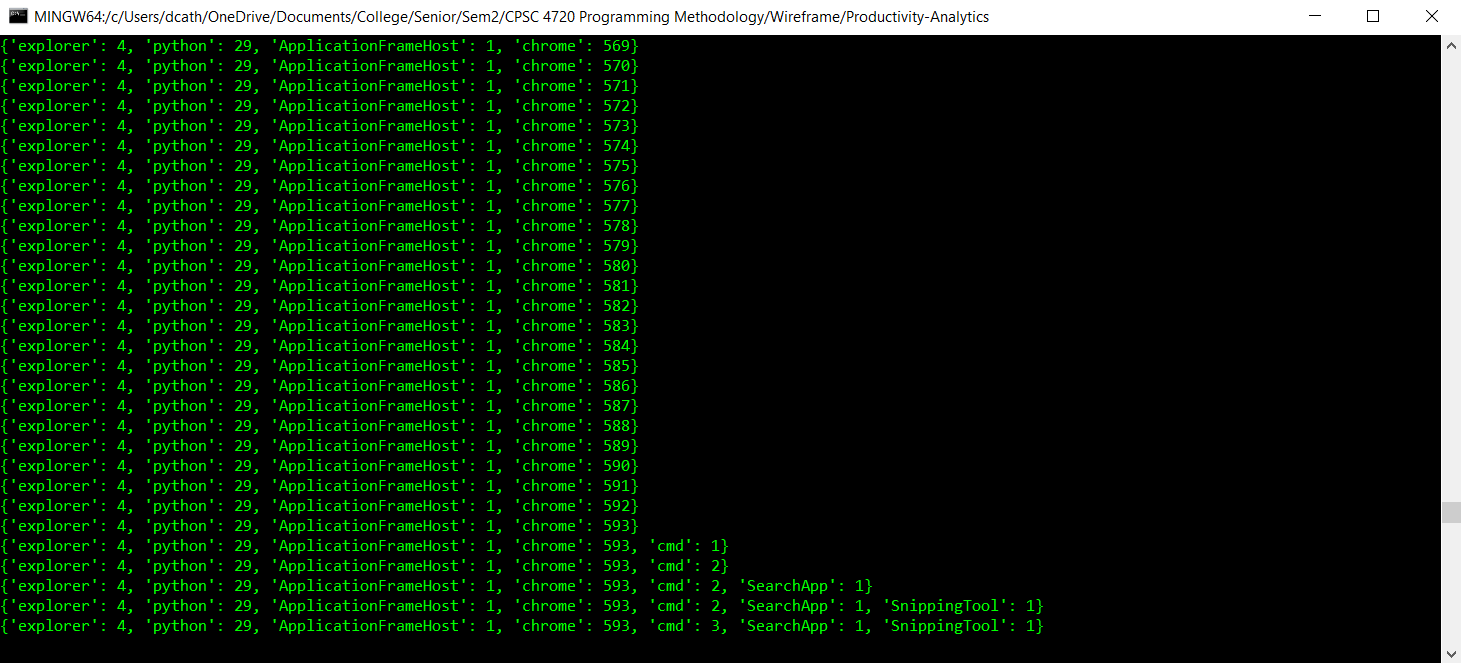
**The Application:**

In the demo video, you will see that the project was working in some ways, but broken in others. I had set up a window that would be composed of three canvases, one with the text that the user inputs in productive.txt and unproductive.txt. Then there were two graphs. The only issue was that the dynamic graph that I wanted to put into the desktop app was not loading into the window, along with the bar graph.

My solution to have a working app was to make three separate canvases that would each display the information they were meant to display. Below is a screenshot of the P.A.T.H Finder’s components. As you can see, each section is in its own figure.



This next screenshot shows the python script running and a row of each process that has been running, and how long they have been running for.



**Launching P.A.T.H Finder:**

1. **The P.A.T.H Finder project can be found on Github at** <https://github.com/dcathapermal/Productivity-Analytics>.
2. **Users can clone the repository by doing the following command:**

Git clone <https://github.com/dcathapermal/Productivity-Analytics.git>

1. **After downloading the repository the user can download the necessary packages as below (using pip):**

Pip install pyqt5

Pip install matlibplot

Pip install pywin32

1. **Once the project is installed, you can create a productive.txt and unproductive.txt file. These files are used to tell P.A.T.H Finder which applications you say are productive, and which are not.**
2. **From here, you can run the following commands to launch the desktop app:**

Python app.py

**Project Demo:**

The project demo can be found in the github, labeled as demo.mp4. In the video you can see that the app.py executable creates two charts that show a real-time productive/non-productive pie-chart and a bar graph with the number of minutes for each app that is running. The demo shows a work-day, as I am recording it while writing this report. So, you will see me open apps like Chrome, atom, command-line, and others.