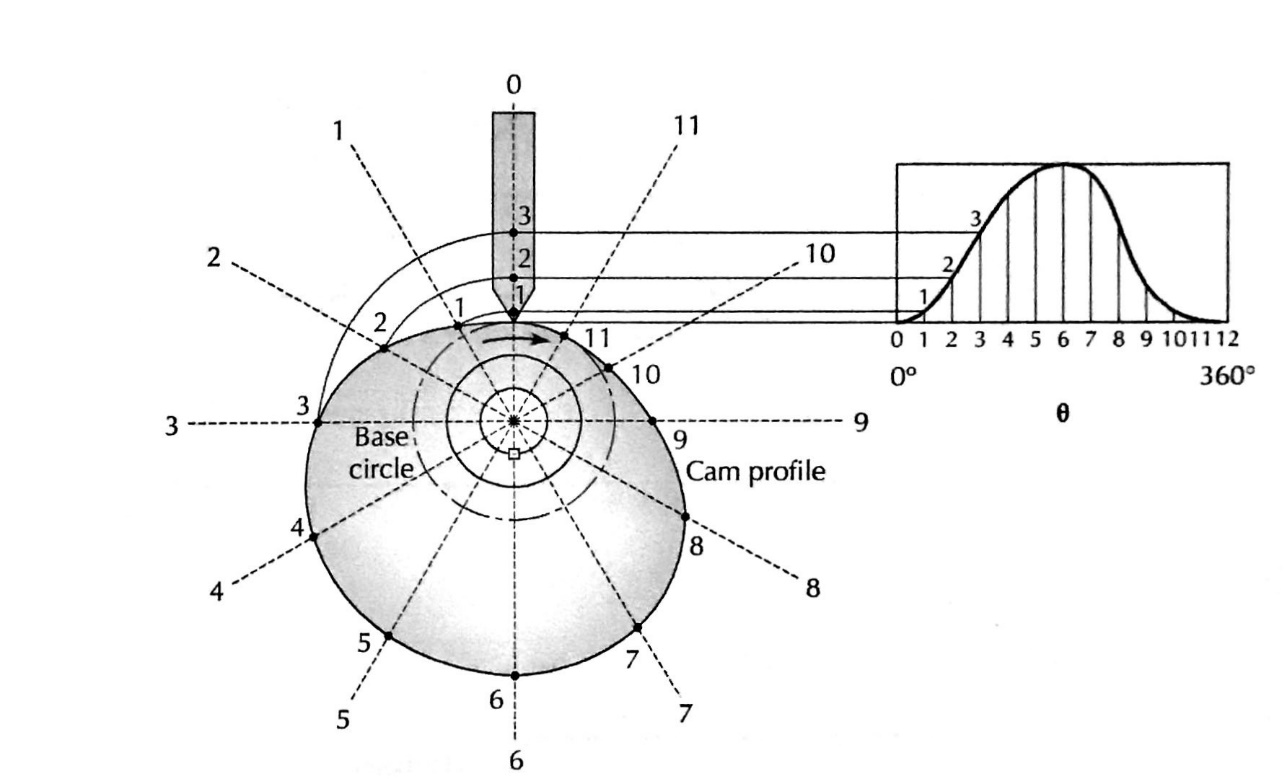
**Final Project: Cam Design and Analysis**

**Brief Project Overview**

The main functionality of this program is to allow for the user to generate cam profiles for knife edge followers given a set of motion types and displacements over a collection of motion intervals. After this cam profile is generate or imported, this program then allows the user to preform a multitude of tasks such as creating a .csv file for the kinematics of the cam follower, changing the units or rotational speed of the cam, creating a .txt cam profile for later use, and visually displaying the cam geometry as a static image or animation.

**Intended Use Case**

The main intended user of this program is for mechanical engineers or machine designers that need to have some type of repetitive, simple or complex translational motion while maintaining a simple input method. In these scenarios, a knife edge cam can be designed so that the input motion can be produced by a simple one speed motor with the output motion being a predefined translational motion. An example of a cam with a knife edge follower can be found below along with the follower’s displacement diagram. Additionally, an example design problem similar to what a mechanical engineer may encounter can be found in Appendix B along with documentation of how this program aids in the design process.



**Project Significance**

The process of designing an accurate cam can be difficult without the aid of a computer. The individual designing the cam must use a graphic methodology that gives a very rough estimate of what profile may be shaped like and must use a multitude of reference points in order to design a more accurate cam as specified by the design requirements. Unfortunately, this process can be extremely time consuming and inaccurate even when several reference points are used. This program streamlines this process so that it can be done accurately and quickly in comparison to the graphical approach.

Additionally, since this program uses the analytical approach to calculate its positional values, important additional data such as follower velocity, acceleration, and jerk can be calculated for. This additional information provides significant value to the engineer designing the cam since the graphical approach doesn’t have an accurate methodology to find this data. Finally, this data can be used in later calculations or Finite Element Analysis (FEA) programs to further analyze the cam structurally and dynamically.

**Important Technical Details and Hardware Limitations**

In order for this program to have full functionality, the following requirements must be met:

* The machine running this program must support an IDE that allows for tkinter GUI to display (EI: Chromebooks using PythonAnywhere are not fully supported)
* The machine must be running a version of Python that supports the functionality found within tkinter library (EI: Python 2 is not officially supported)
* The machine must have the Pillow (PIL) and Ghostscript 3rd party libraries installed
* The machine must have the ability to install the Ghostscript 9.50 Application (Supported Platforms include only Windows and Linux Machines)
* The binary directory of Ghostscript 9.50 must be added to the machine’s PATH/Path Folder found in the System Environment Variables

It should be noted that the last two requirements are only needed for the feature that allows the user to export the static cam profile as a .png file. This program has been coded to cater to edge cases so that it will not crash if these libraries are not installed. In situations where the user wants to export the static cam profile as a .png file, the following steps can be followed.

1. Install the Pillow and Ghostscript Libraries via the PIP Installer
   * Below are examples of pip install commands for Python 3.7
     + py -3.7 -m pip install Pillow
     + py -3.7 -m pip install ghostscript
2. Install the Ghostscript 9.50 Application (AGPL Release Version)
   * This can be downloaded from the following link:
     + <https://www.ghostscript.com/download/gsdnld.html>
3. Add the Binary Files Found in Ghostscript’s Install Folder to the PATH/Path Folder
   * The following is an example absolute path of where the binary directory can be located for a Windows machine
     + C:\Program Files\gs\gs9.50\bin
   * The following is a guide from Help Desk Geek explaining how to add to the Windows PATH/Path Environment Variables:
     + <https://helpdeskgeek.com/windows-10/add-windows-path-environment-variable/>

**Kinematic Analysis Formulas**

The following is the collection of the formulas utilized in the program to calculate the position, velocity, acceleration, and jerk for the knife edge follower.

**Dwell:**

**Uniform:**

**Parabolic:**

**Harmonic:**

**Cycloidal:**

**Visual Design Implications**

Due to the nature of this program allowing for the user to preform a multitude of tasks on a cam profile that they either create or import, this program was designed with a tier system of command line menus. Furthermore, the tasks to create a static cam image and to show the animation of the cam use a tkinter GUI before returning to the analysis command line menu. The finalized sequence of events can be seen later in the visual diagram or in a step by step, detailed guide found within Appendix B.

**Data Plan**

Overall, this program uses the following data structures in a multitude of ways: single value variables, 1D arrays (tuples or lists), 2D arrays (Combinations of tuples and lists), IO files, classes, and imported libraries. Below is a list of prominent examples where each of these data types are used

* Single Value Variables
  + Temporary Input Holders
  + Loop counters
  + File Strings
  + Long Text Body Strings
* 1D arrays
  + Single rows of information for a selected angle
* 2D arrays
  + Lists of the single row data to create comprehensive data sets
  + Read lines from input files that are iterated through by item of the line
* IO files
  + .txt file of cam profiles
  + .csv file of kinematic data for each angle
* Classes
  + Class for the analysis menu
  + Class for producing tkinter GUI’s
* Imported Libraries
  + A private utilities file of menu functions was imported opposed to being written in the main file
  + The cam analysis class inherited the cam visual class that was stored in a separate Python file
  + 3rd Party Libraries such as Pillow and Ghostscript were imported so that classes and functions within those libraries could be utilized

**Required Data for Analysis**

The required data for this analysis to properly function includes the following: angular velocity of the cam, the units used for the length values, the initial and final angle for each interval, the type of motion that occurs during this interval, and the initial and final position of the cam for each interval. Once this data is produced either through the creation tool or by importing the cam profile .txt file, it can be manipulated by using the formulas outlined earlier to produce the angular position, time, follower position, follower velocity, follower acceleration, and follower jerk. Finally, once the angular position and follower position are found, these can be then be manipulated to create the geometry of the cam profile that is used for the static image and animation.

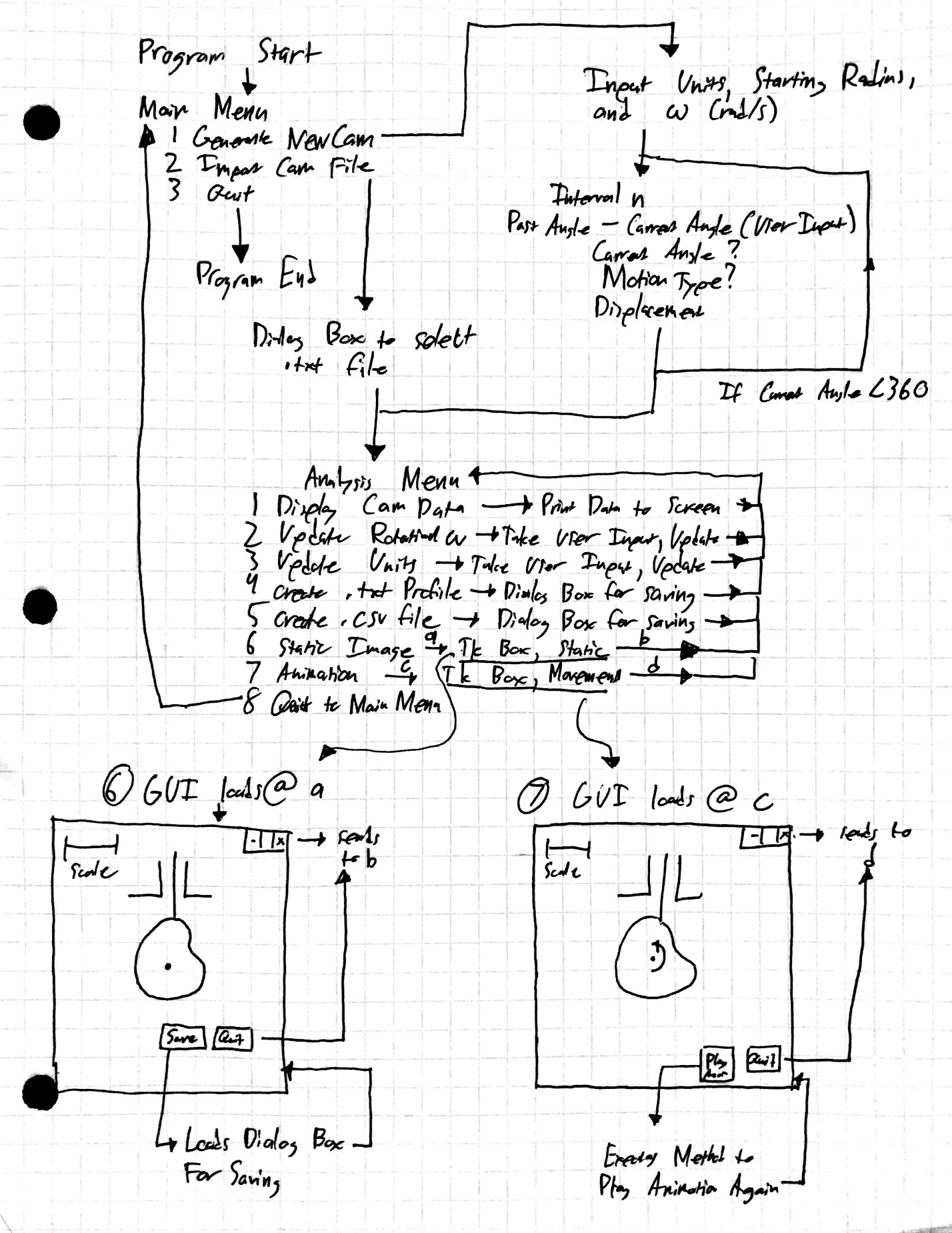
**Use of Classes and Files**

This program consists of 4 files: The main file (evanpark\_Cam\_Main.py), a file containing utility functions (evanpark\_Utilities.py), a file containing the class used to manipulate the required data inputted (evanpark\_Cam\_Properties.py), and a file containing the visual class (evanpark\_Cam\_Visual.py). Once the required data has been produced or extracted from the main file, this data is passed on to the cam analysis class (CamProperties). From here, all of the methods can manipulate the data passed in to accomplish the goal of the user selected task. It should noted that for the static image and animation, the kinematic data for each angular position along with the cam’s units are passed on to an inherited class (CamVisual) that then uses the units to create a reference scale and the follower position corresponding with each angular position to create a visual representation of the cam.

**Visual Overview**

This program follows the structure of using a main menu that has two diverging paths that eventually converge to the analysis menu after the cam file is generated or imported. Once this occurs, the analysis menu has 8 options (3 command line inputs, 2 dialog boxes for saving, 2 tkinter boxes, and 1 for quitting) that the user can select from. A visual breakdown of the how the program operates along with example screenshots of what type of section looks like can be in the following section. Additionally, a 14-page, comprehensive and in-depth example problem that visually displays every functionality in full can be found in Appendix B. Finally, the technical formatting details for the visual elements of the tkinter GUI’s can be found in the algorithm for evanpark\_Cam\_Visual.py found in Appendix A or in the stand-alone algorithm file found in the zipped folder of Python files and algorithms.

**Visual Diagram**

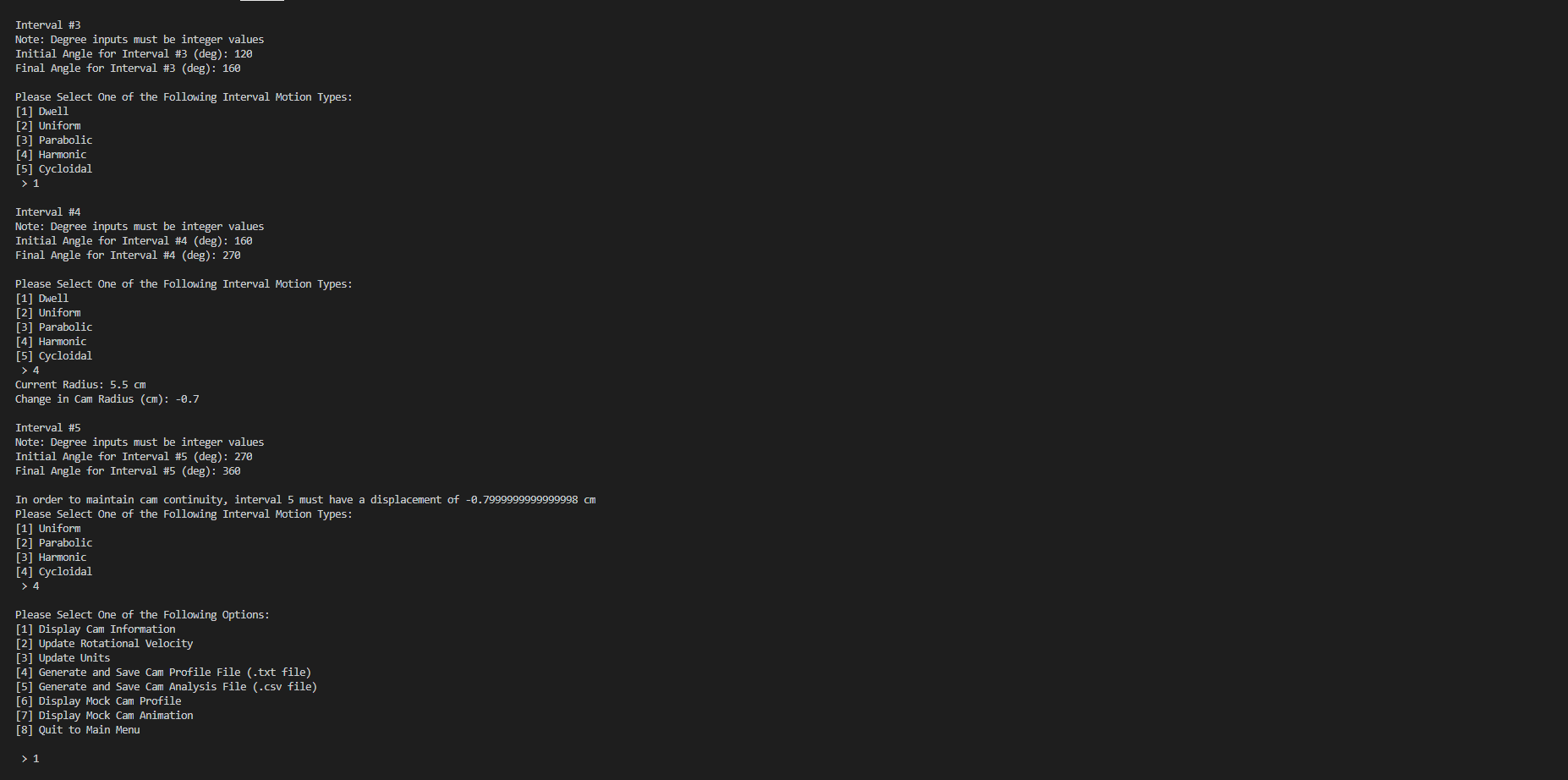


**Main Menu:**

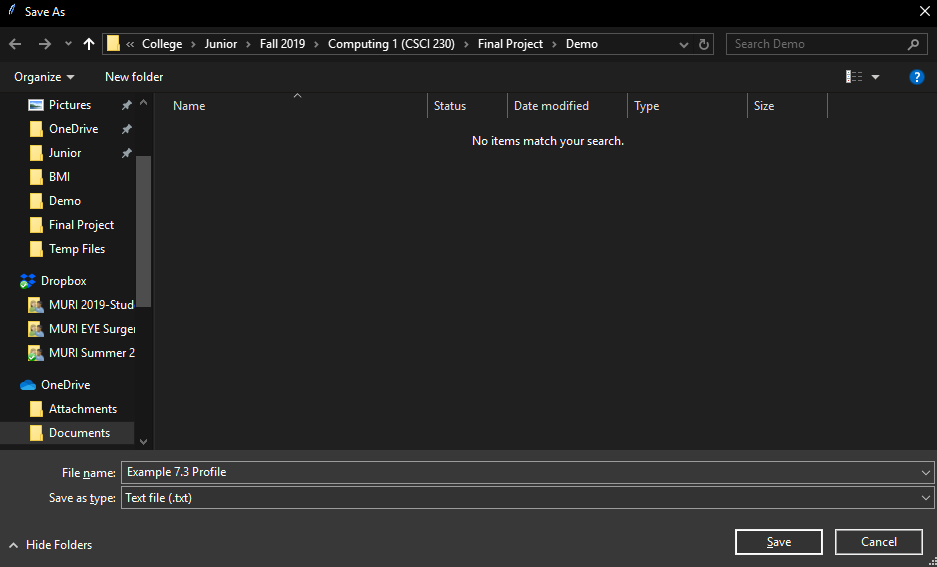


**Cam Generation:**

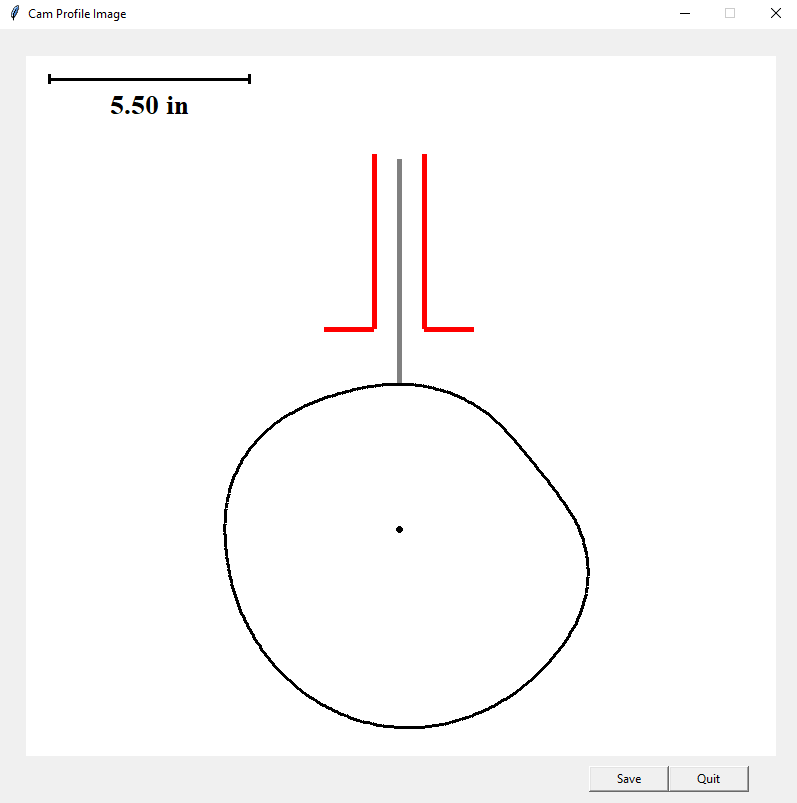


**Analysis Menu:**

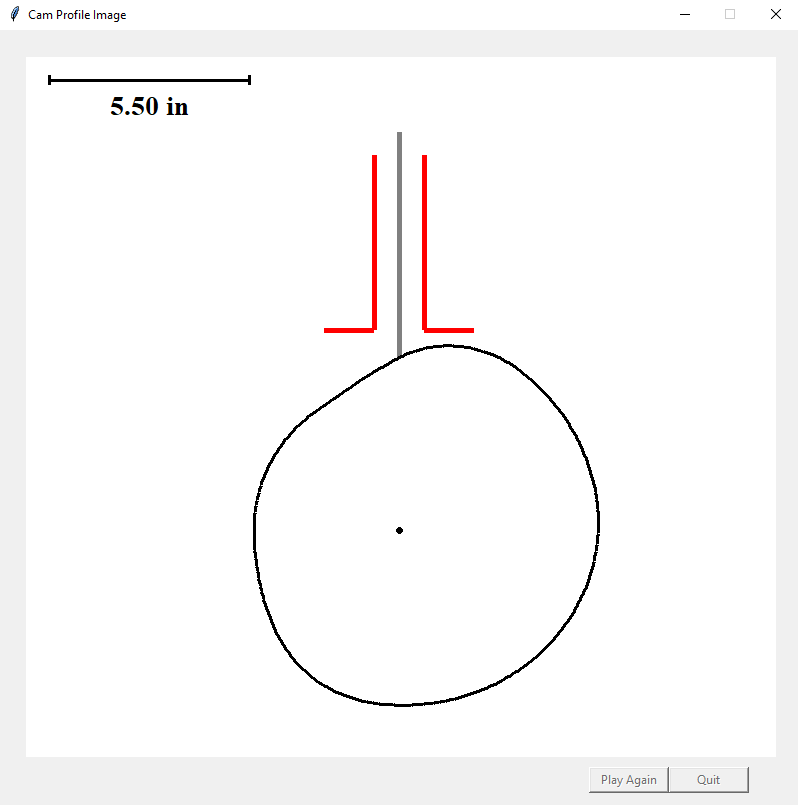
**Example Dialog Box:**



**Tkinter Static Image GUI:**



**Tkinter Animation GUI:**



**References**

Cleghorn, W.L. & Dechev, N. (2015) Cams. In *Mechanics of Machines: Second Edition* (pp. 310-354). New York, NY: Oxford University Press.

Ghostscript Downloads (n.d.). Retrieved from <https://www.ghostscript.com/download/gsdnld.html>

Kishore, A. (2017, September 26). *How to Add to Windows PATH Environment Variable.* Retrieved from <https://helpdeskgeek.com/windows-10/add-windows-path-environment-variable/>