User Manual: Vector Calc

A Vector Force Visualizer and Calculator

Last Update Apr 22, 2022

Peyton Roseberry

Check https://github.com/IntelX86Assembly/VectorCalcStable For Most Recent Version of Manual to ensure you are up to date with your version.

Index

This Page intentionally Left Blank

Dependencies/Install:

OS Requirements:

The following operating systems have been tested and work

- linux (Debian derivatives)
- MacOS
- Windows 10

Other operating systems may work, but their functionality and stability has not been tested.

Languages/Dependencies:

(For most Windows users this can be ignored because the dependencies are in the .exe file)

Requires a modern operating system with python3 installed:

In addition to the python3 interpreter the python core library needs to be installed. By default most systems already have these libraries since they are a part of the standard python library. But **the following will need to be downloaded if they are not already installed.**

- **Matplotlib (Helps with drawing Graphs)**
- **Tkinter (Allows Gui to be drawn on system)**
- **Math (For Sin and Cos calculations)**

To Get Most Recent Source Code for Developers:

https://github.com/IntelX86Assembly/VectorCalcStable

Note*** download the stable version for the most recent working version.

Interface Structure/Functionality:

How to Install and Run for Windows:

Download VectorCalc.zip From
https://github.com/IntelX86Assembly/VectorCalcStable/releases/tag/release_Windows
Right+Click → Extract All →Open folder
Double click on VectorCalc.exe

How to Install and Run for linux/Mac:

- git clone https://github.com/IntelX86Assembly/VectorCalcStable.git
- cd VectorCalcStable
- $ python3 GuiRoot.py

Menu Options:

File:Exit

- Exits program

File:Clear

- Completely clears workspace

Plot:Single vector

- Takes an X and Y component and graphs vector

Plot:Multiple Vectors

- Takes multiple vectors and graphs them all

Calc:Resultant

- Adds vectors together and displays their resultant

Calc:Equilibrium

- Finds the equilibrium vector and displays it showing how to counter-balance forces

Help:

- Displays a help pop-up dialog for helping the user if they are stuck.

Note On Input:

The following data are supported in this program.

Integers (A whole number without a decimal)

- Ex. 2, 10, 15, -4, -25

Floats (A number with a decimal value)

- Ex. 2.2, 4.7, -8.0

Scientific notation

- 10e4, -4e3, 2e-2

Note we can mix and match these. So you can type “2” in the first field and “3e10” in the second field and the program will run. This program will ignore garbage data so that it can still provide results even in the case of a mistake. So if you add a vector and it's not showing up make sure you are using one of these numerical types.

Getting Started:(Example Plotting Multiple Vectors)

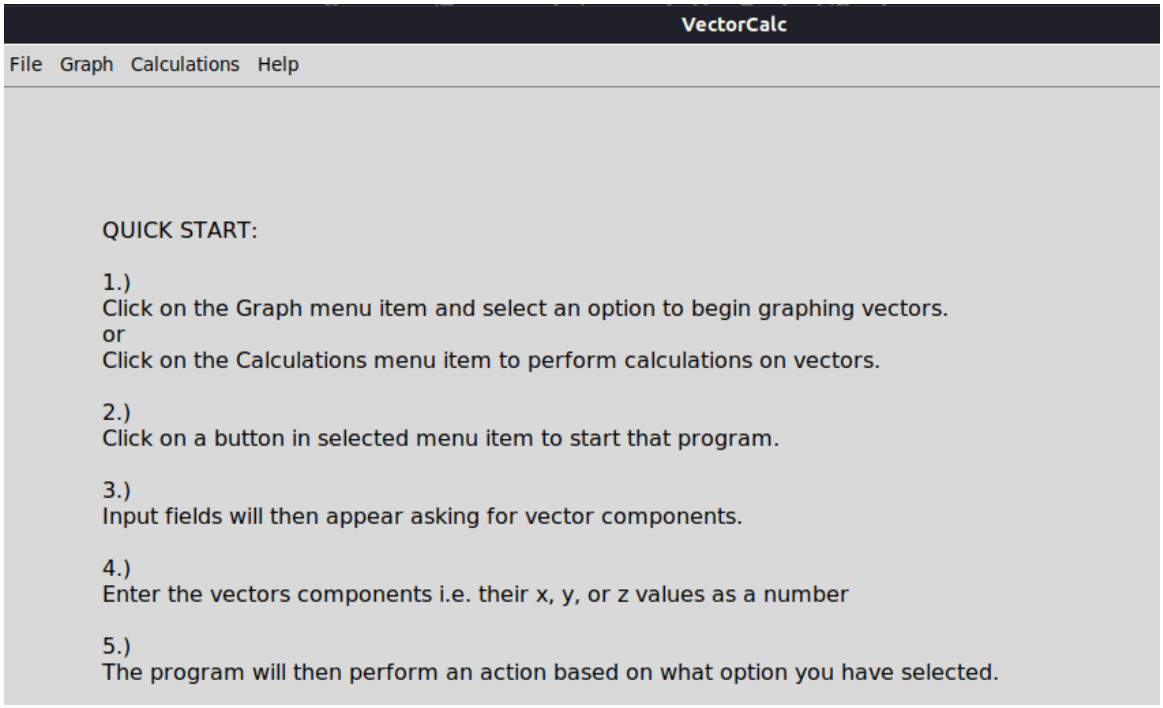

Figure 1.) Start Menu

This is the menu shown at start up. To start, click on one of the menu items. In this example we click Graph → Graph: Plot Multiple Vectors

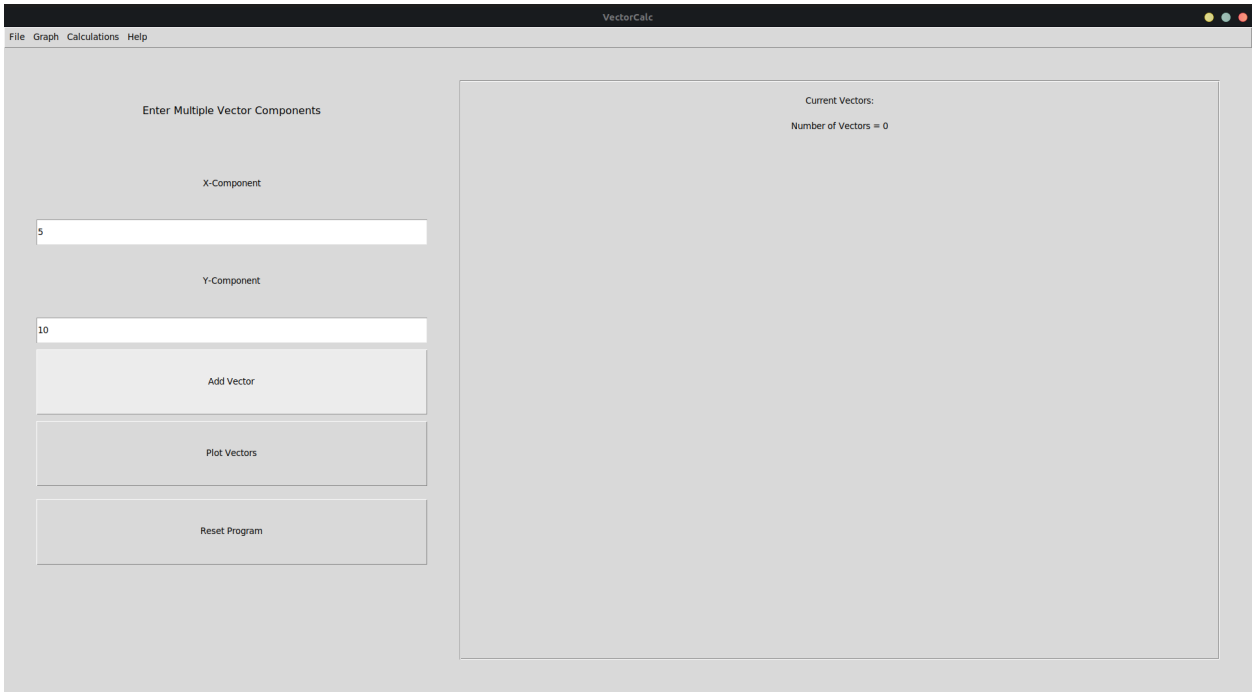

Figure 2.) Adding Vectors

Next we enter some text as either an integer, float, or in scientific notation. We click the add button to add the vector to the program and it will show up on the right side under “Current Vectors”.

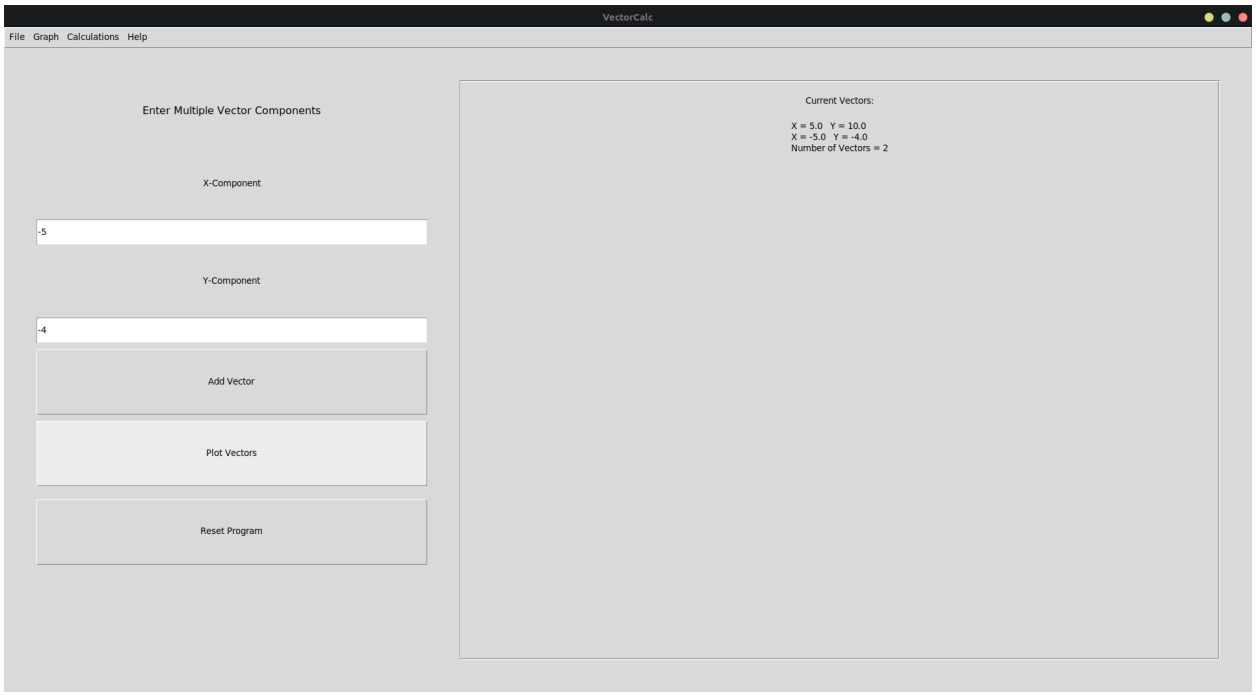

Figure 3.) Plotting Vectors
Click on the “Plot Vectors” button to draw a graph. A window will appear with the results.

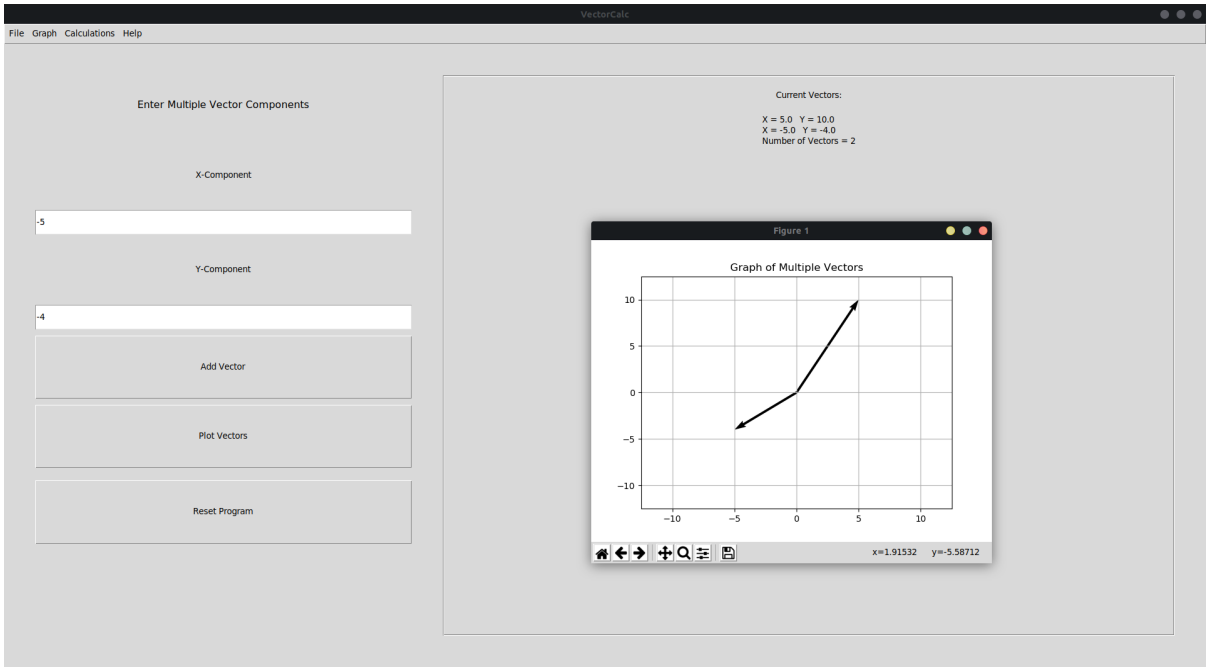

Figure 4.) Graph Drawn

Example of graph drawn appears on the right side of window.

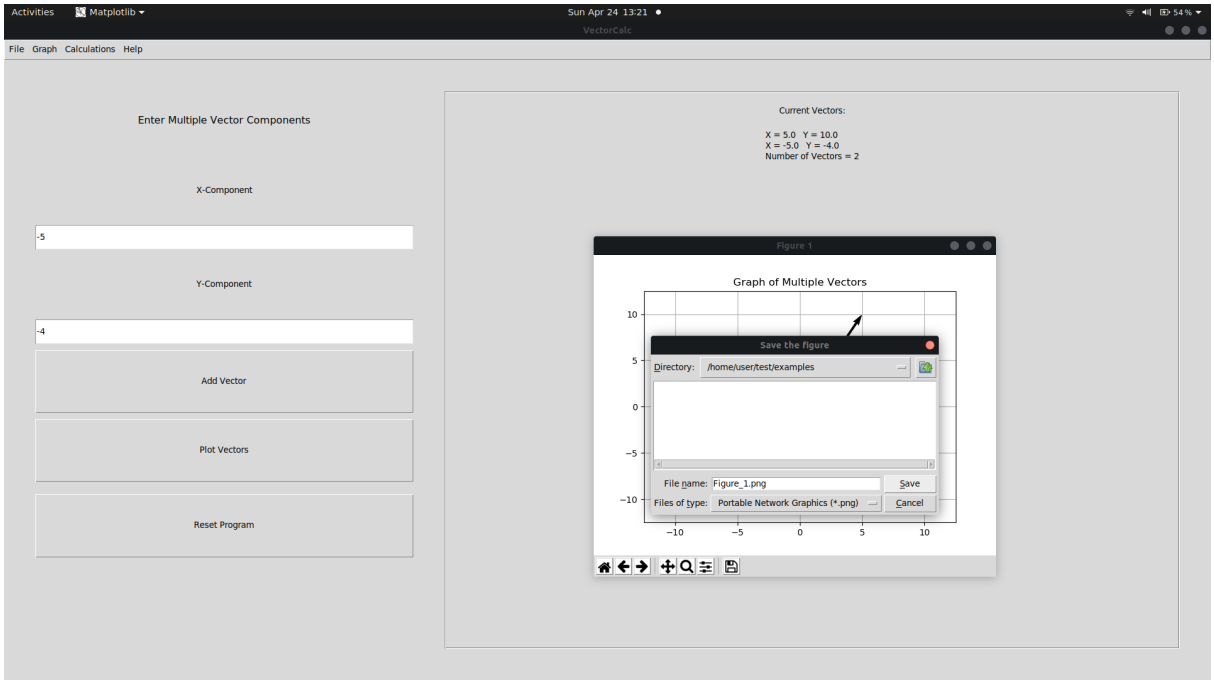

Figure 5.) Saving Image

Click on the save button in the graph menu to save your graph to your computer.

Accessibility:

- This program uses dynamic features based around user settings to make it more accessible.

For the Visually Impaired:

The program dynamically scales to the system font size and choice. This means if you have accessibility fonts turned on it will use this as the default.

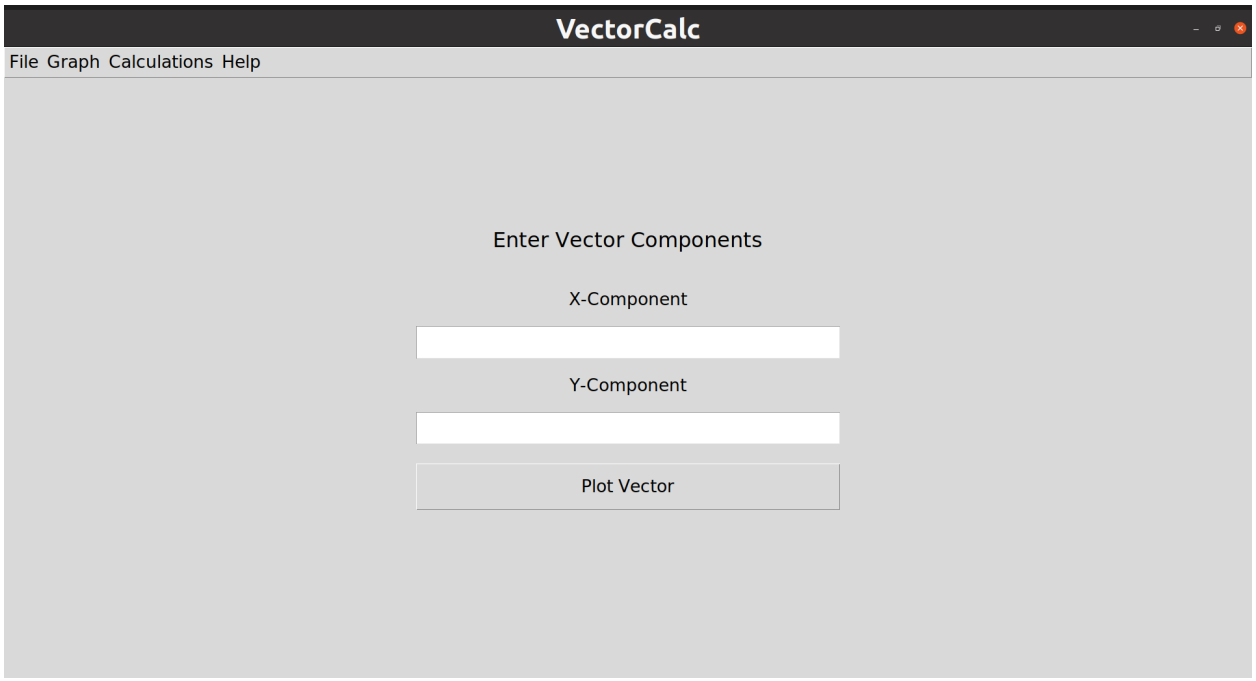

Figure 6.) Accessible Fonts and UI

For Motor Impared individuals.

The text and button fields are made large for easy accessibility. If this is still not to your liking, increasing the system font size makes these even larger. If you compare figure 6. with the other diagrams in this manual you will see that the buttons and menu options increase in size when this is done. Picking a value like 200% font size makes the buttons easier to click on.

Administration and Maintenance:

In the event of hosting site removal:

- In the event the hosting repository goes down this project can easily be moved to another site. This is because it's cross-platform and open source. The original at the time of this writing is hosted at Github.

- While a new site is being found to host this, options exist for distributing this. It will likely be moved to a P2P model where volunteers seed the software until an alternative is reached because this is not a large project. Links and information will be updated in the meantime. At the time of this manual there is no current backup site.

Future Porting:

- As long as the system has the ability to run python3 this project will remain portable. This was a large consideration in the language and frameworks used. While Windows, Mac, and Linux were tested this program should run on most systems so long as they have a python interpreter and the correct libraries for that system.

Extending the Project Further:

Adding More Menus:

- **To add more menu options you will need to modify the Frames.py class and add a new class to instantiate a new Frame object. Then you will need to modify the GuiRoot.py file and do two things. Add a Tkinter menu object and create a listener method to instantiate the new class in the Frames.py file when clicked. Look at the Class Description and UML in the Appendix for more information on these classes.**

Adding More Operations:

- **To add more operations you will need to modify the Vector2D.py, Vector2DList.py, and Vector2DGraph.py files. You will need to add methods to the Vector2D list if you want additional 2D vector operations. In addition, you will probably need to add methods for controlling what data you want to display and how you want to display it. Optionally you could create your own vector object to interface with the other classes. Note this program is made for 2D vector operations. It is extensible , but you will need to create a class for drawing 3D windows if you decide to add in 3D vector operations. Check the Class Description and UML in the Appendix for more information on these classes and the layout of the program.**

Further notes:

- **This project is open source. You are free to use, modify, and redistribute as you see fit. Because this project uses an interpreted language all the code is readable when you download it. There are comments throughout the code describing the layout, operations, and workings of the project. The appendix can be a great resource for understanding the relationships between classes when modifying the code.**

Appendix:

Classes and Description:

- **GuiRoot.py:** is the main class. I may create a driver class to instantiate and call the methods of this class but this is more of a code style and organization problem than a practical one so for now this is the primary class. It does a few things. It uses the tkinter framework to start up a gui window and present a basic menu with options such as getting help and selecting what the user wants to do. There are no buttons for calculations or graphing; this is mainly a container class for sub frames to be placed in the gui. There are menu options that when clicked instantiate objects from the frame class.

- **Frames.py:** These are the objects that can be instantiated from the GuiRoot.py file. These objects when instantiated draw a new window inside our application for performing a certain action. For example if the user wants to graph a vector the GuiRoot class will create a new frame dedicated to this purpose and load it into the main gui as a sub frame. This file has all the frames that can be loaded to achieve a specific action.

- **Vector2D.py:** This is a file describing a vector object. When the user types in the data and hits a button to take their data this object will be created describing the things a vector can do and its attributes.

- **VectorList2D.py:** This file describes a list of vectors. This is essentially a list of vectors that has methods and properties that are useful for calculations. For instance it can sum all the vectors in a list it creates or calculate the resultant given a list of vectors. It greatly increases the organization of calculations and allows code sharing. The Frame classes use the Vector2D and VectorList2D classes for working with vector mathematics.

- **Vector2DGraph.py:** This is a mixture of a wrapper and helper class for the matplotlib framework. This is because the framework itself is not tailored to work with vectors easily. This class handles all the rendering and graphing of vectors. For instance it can take a vector(s), scale it properly to fit the window, center the view about the vectors, and render it, and is easy to interface with.

Complete UML Diagram: