Hello! This is a guide for Ryan’s 2016-17 SRD project.

Note: Terminal commands are shown in brackets []. Don’t type the brackets, just the text inside ☺

1. Setup/contents

So, you have been left in charge of this mess of code, eh? Even though it’s mostly commented, there’s room for communication and guidance, and that’s where this readme comes in! When you open up the folder for the first time, the contents should include:

* This document (ReadMe.docx)
* AnalyzeButterfly.py
* butterfly.mp4
* ModelCoordinatesTwo.txt
* CoordinateProcess.java, Vector.java, and SplitFiles.java
* PositionGraphing.py, VelocityGraphing.py, AccelerationGraphing.py, DragForceGraphing.py, and AppliedForceGraphing.py

If any other files are present, some of the runtime steps have been taken already; you should check the data present in any .txt files and consult with either the original caretaker of the files or your past self (if you did do something with this program in the past).

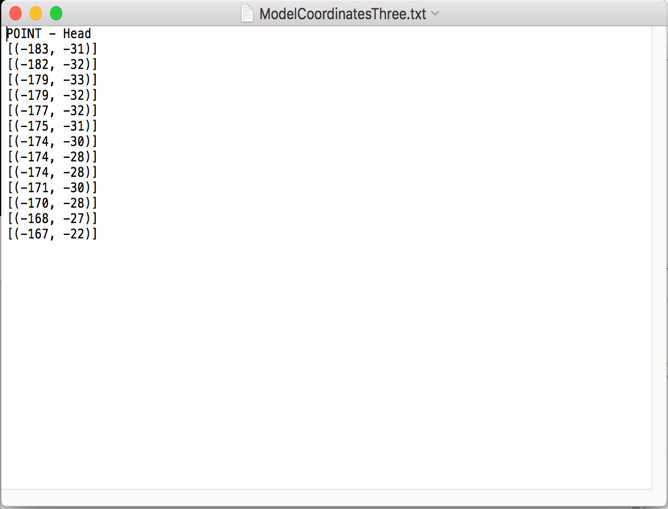
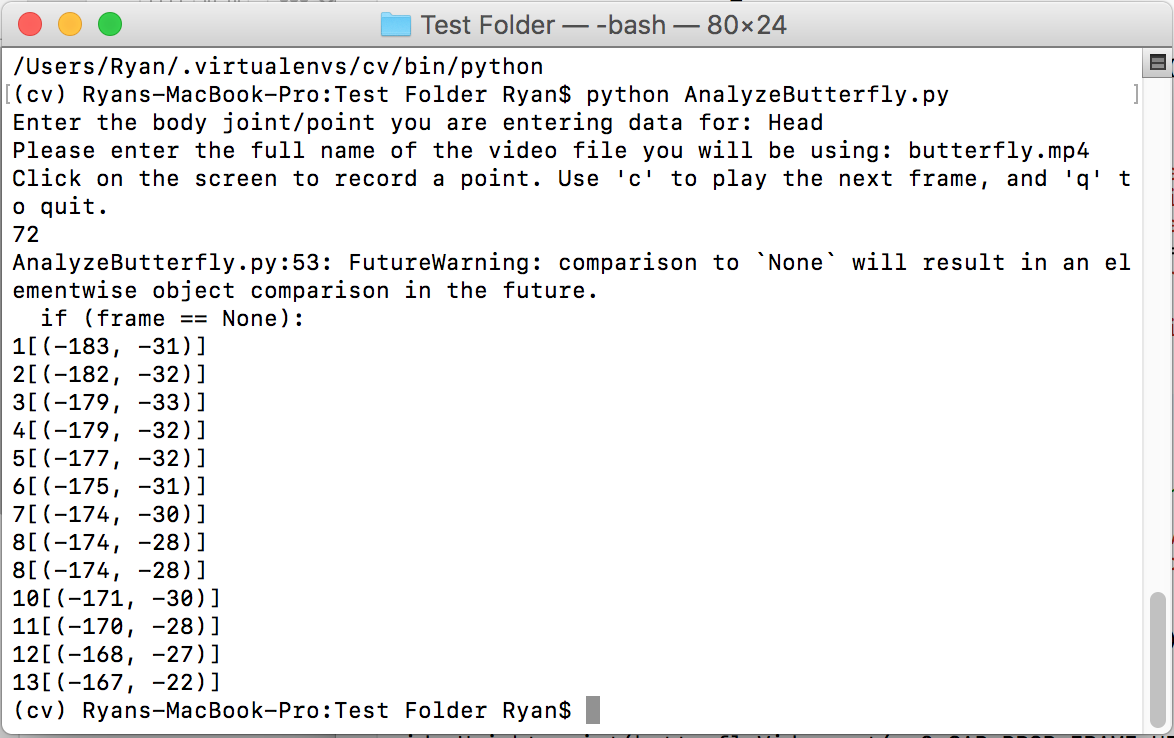
Please note that to run the programs, your computer will need Java 7 or higher (I think; if not, try Java 8) and Python 2.7 or higher. Additionally, while you will be able to do most of this from Terminal/Command Line, there are two Python modules and one editor you will have to install to make use of the libraries used in the Python programs.

1. AnalyzeButterfly.py

So you have your video that you want to analyze. Great! First, however, you will need to install OpenCV (Open Computing Vision) for Python. Here is the link for OpenCV’s website: <http://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_setup/py_setup_in_windows/py_setup_in_windows.html>

I didn’t use this; instead, I followed the instructions on this website, and installed it through Terminal for my MacBook: <http://www.pyimagesearch.com/2016/12/19/install-opencv-3-on-macos-with-homebrew-the-easy-way/> After installing OpenCV this way, I continued through with the optional step detailed on the website and setup an OpenCV virtual environment, so I can run OpenCV programs directly from Terminal.

In any case, once you have OpenCV for Python installed, run the AnalyzeButterfly.py program using [python AnalyzeButterfly.py] for Terminal or a Python IDE. The instructions are printed to the console, but the program is very simple; your video file is broken down frame by frame, and you are asked to click on a body joint for every frame, one that you specify at the beginning. Controls include ‘c’ to move on to the next frame and ‘q’ to quit the program (although it will automatically quit when you reach the end of the video). Your clicked coordinates will be translated such that (0, 0) is in the center of the screen (you can change the source code for a different translation, under the *click* function) and +x is to the right and +y is up, and will be both output to the console, along with the frame number, and to a document called ModelCoordinatesThree.txt (you can also change this name in the program).

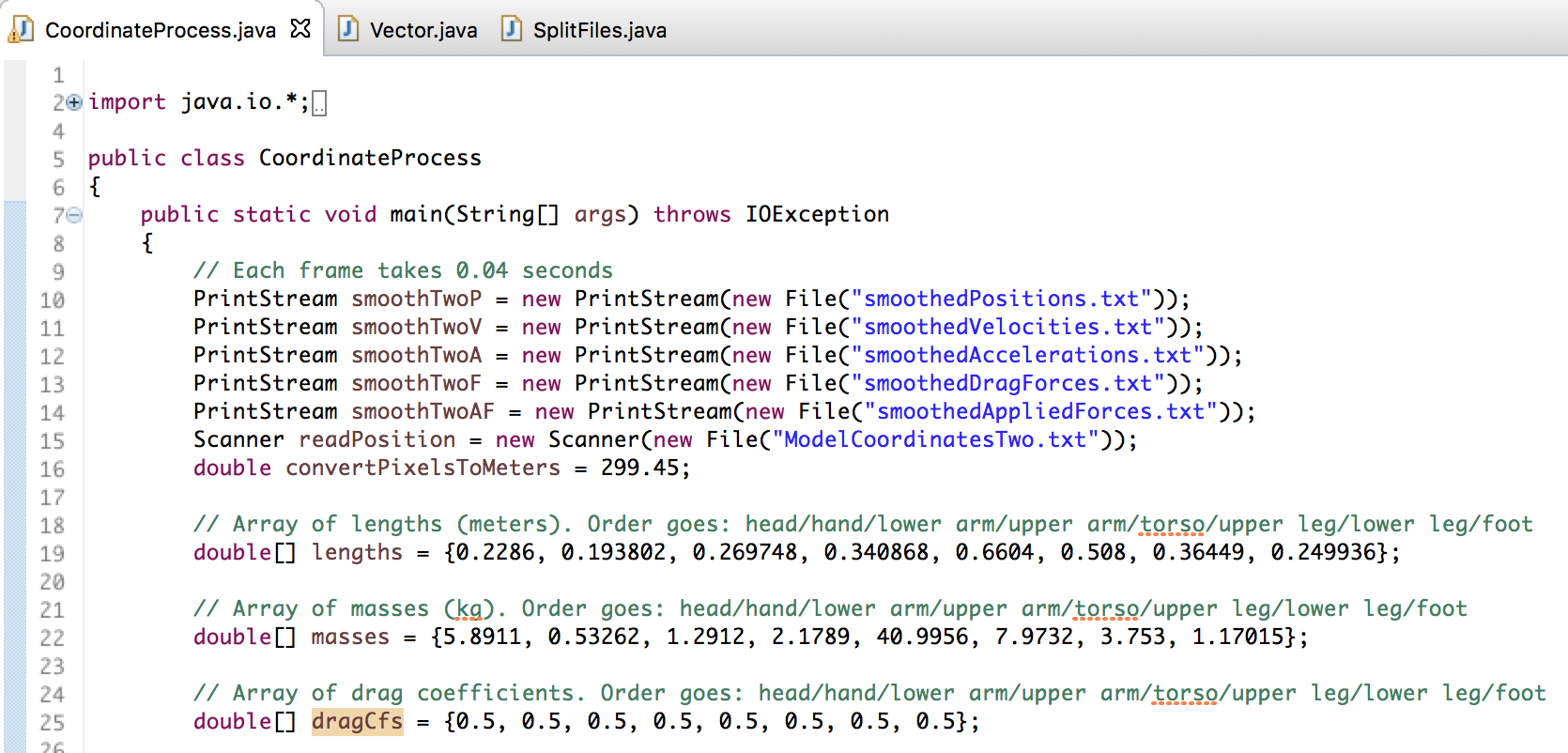


Sample text output – frame numbers not shown here; corrections can be made manually to the raw data.

Sample console output – notice how the 8th frame was clicked twice and the 9th frame skipped entirely.

Try and complete the whole video for one body part in one sitting/run of the program; know that duplicates (e.g. two sets of coordinates both under the “head” title) will simply have the later sets appended to the earlier ones; additionally, make sure that you have the same number of coordinates selected for each body part, or the Java program for analyzing will run into problems later.

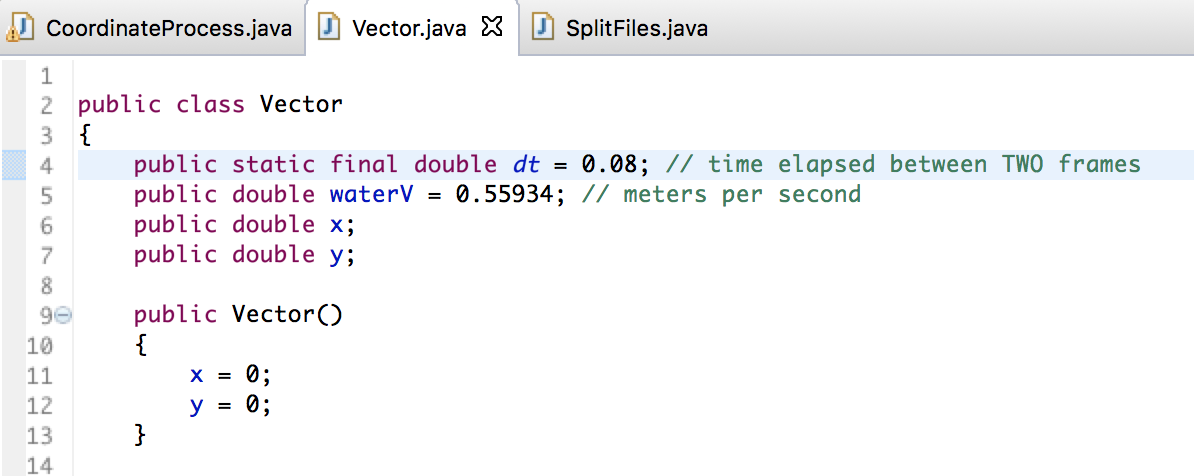
1. CoordinateProcess.java

This is the big one! CoordinateProcess.java is the Java program doing all the heavy lifting and analysis. An overview of the source code will reveal calculations, file handling, and even data smoothing! To use CoordinateProcess.java, simply run it in an IDE (make sure it is linked to Vector.java, a helper class), or on Terminal, simply use [javac CoordinateProcess.java] to compile, and then [java CoordinateProcess] to run the program! You should see “Got here!” printed to your console when the program is done running. The result will be 5 .txt files, each of which contains the smoothed data for each body part in head-to-toe order, labeled within the document. There are some things to be aware of in CoordinateProcess.java; firstly, I was unable to find the drag coefficient of each body part in water, and just have a universal 0.5 for everything. Obviously, this is inaccurate and should be changed; if you find more accurate drag coefficients, you can change that in the list at the beginning of the code. 

Beginning of CoordinateProcess.java; notice the list of drag coefficients at the bottom is filled with 0.5’s. Additionally, you will find the names of the output files here. Don’t change these!

Additionally, notice the ‘double convertPixelsToMeters = 299.45’ statement in the beginning; this was my conversion factor from pixels to meters (299.45 pixels = 1 m) in the video that I used. This will be different for your video, most likely, so make sure to measure it and change the constant, since all the calculations are done in metric/SI units. (One way I used to measure this was to run AnalyzeButterfly.py and click on two points on the first frame, pre-determined to be about one meter apart, and find the distance between the recorded pixel coordinates.)

Another two things to look out for, in the Vector class, are the dt constant and waterV constant; the former represents the time elapsed during 2 frames (velocity calculation for a frame requires using the frame before and after it), and the latter represents the speed of the water relative to the swimmer, i.e. how fast the swimmer’s body is moving relative to a stationary observer, in m/s. These values both can and should be changed in the source file.



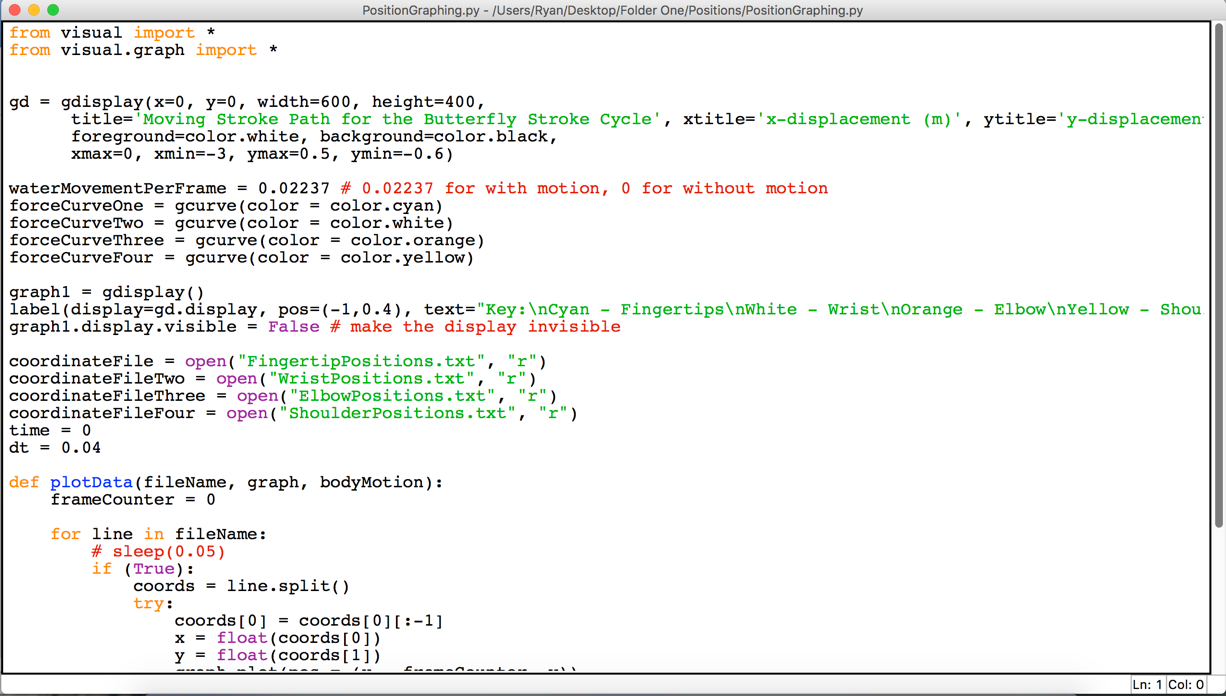
Beginning of Vector.java – note that you will have to change dt and waterV according to the characteristics of your own video!

Finally, try not to change the names of any of the output files in CoordinateProcess.java (shown above). These names are hard-coded and will be used later by SplitFiles.java to organize the data, so if you run into problems there, check the names of the output files!

1. SplitFiles.java

Just like CoordinateProcessing.java, SplitFiles.java is a run-of-the-mill Java program that divides the data output of CoordinateProcessing.java into separate folders. Simply [javac SplitFiles.java] and [java SplitFiles], and you should see five folders appear, one for each analysis component (i.e. position, velocity, acceleration, drag forces, and applied forces). Within these folders, you should see .txt files, named with the body part and analysis component accordingly, of the appropriate data. If SplitFiles runs into an error, again check the names of the output files from CoordinateProcessing.java. They should match the ones in the picture above!

1. Now you have all the data neatly sorted into folders. But you want to visualize it! Luckily for you, VPython is here to save the day! For this final step, you will have to download VPython online (link here: <http://vpython.org/>; I would recommend getting VIDLE, the Visual Python IDE/code editor as well. I think it comes with the install from the website, but I’m not 100% sure). Next, copy/paste the five files, PositionGraphing.py, VelocityGraphing.py, AccelerationGraphing.py, DragForceGraphing.py, and AppliedForceGraphing.py, into their respective folders. Finally, open up VIDLE (or whichever Python editor you are using; this part can’t be done from Terminal/Command Line, unfortunately), and open the program of choice, finally running it to plot the curves. Note that from the source code, the title of the plot, colors of the curves, graph scaling, x/y axes, and more can be changed (shown below).



Source code for PositionGraphing.py – note that various aspects of the graphs may be changed from the source code file, and that more data can be plotted on the same graph by opening more files and plotting them using the plotData() function. waterMovementPerFrame is the distance (in meters) travelled by the swimmer per frame. For motion relative to the swimmer’s torso, set this value to 0.

1. And there you have it! You should now be able to visualize the data and draw conclusions or perform further analysis on any of the .txt files generated. If you have any other questions, feel free to email me at [goldenviolin7@gmail.com](mailto:goldenviolin7@gmail.com) and I will get back to you. Thanks for reading this, and have a great day!