Eng ek 127 fall 2012 project # 3

ENG EK 127 Fall 2012 Project # 3 Project development by Josh Koerpel Instructions: You may work alone, or in groups of 2 or 3. You are encouraged to work with classmates in your lab section. This project is due on Wednesday, December 12th, by 7: 00PM. Please note that there are only 44 PCs in the lab and over 240 students in the class. Not everyone can be in the lab at the last minute! You will be graded on the correctness of your solution, and on your programming style. This includes using loops correctly, writing useful comments, etc. Follow the Programming Style Guidelines at the end of the chapters! You will submit your solution both electronically and in person. Hand in a document consisting of a typed cover page, your solution, and the output. The cover page must include your name(s), lab section, the name of the submitted folder, the name of the main program file, the time it took for your code to run (see Tasks section), and a signed statement from each group member stating that the program is the original work of the group. Make sure that you also submit the folder containing every file needed to run the program electronically in the “ Project Hand In" folder, one per group, with your name(s) on the folder. Background: First of all, congratulations! You have made it through the physical training and have been chosen as a member of the first-ever team of privatized astronauts. That is, you will be one of the first astronauts to fly into space upon a space shuttle produced in the private sector. As you sit there during the countdown to launch, try not to think about how the giant controlled explosion you’re strapped to was built by the lowest bidder. Before you actually get to launch into space however, you have to gain a familiarity with the zero gravity environment. To do this, you get to participate in BUSAT’s (Boston University’s Satellite Applications and Training) micro-gravity testing aboard a modified 727

aircraft, nicknamed the ‘ vomit comet’. Here’s an example of how this testing looks: http://www. youtube. com/watch? v= _crSIg72Wu0. Zero gravity is produced (simulated, to be more correct) in the following way - the plane itself flies in a parabolic trajectory, meaning the flight path of the airplane follows a sinusoidal curve. The portions of the flight path that are along the peaks of the sine curve are where zero gravity is experienced (see Figure 2). This is when the pilot throttles back after climbing at such a steep angle, the plane begins to free-fall, and your body momentum along with the passing over the Figure 1: The 45 degree pull-up of the vomit comet. For comparison, commercial airlines during take-off pull up at approximately 9 degrees. Figure 2: The flight path parabola showing the steep angles required to simulate zero G. Notice that while not in zero G, as a passenger, you are experiencing almost 2 G. ‘ crest’ of the sine peak causes you to experience roughly 20 seconds of a weightless environment. A typical flight consists of 40 parabolas in a row, allowing you to run a zero gravity experiment 40 times. As an engineer and part of this BUSAT and astronaut +Z +X team, you have been charged with the task of examining how a piece of scientific equipment will behave in the zero G environment. You need to ask yourself questions like these: ï,· What kind of external forces will this equipment be subjected to as it floats? ï,· The equipment is dynamic, meaning it has moving parts, so how will this dynamic behavior affect its +Y orientation in 3D space? Figure 3: The 6 degrees of freedom of an ï,· How will I measure these forces? aircraft. Linear movement in x, y, and z, The answer to the third problem is to measure the as well as rotational movement in roll, equipment’s accelerations using sensors called pitch, and yaw. accelerometers (these measure linear accelerations) and gyroscopes (these

measure angular accelerations). The accelerations are useful because, thanks to Newton, we know that F = ma. Since we'll know the mass of the equipment, the accelerations provide all that's necessary to calculate force. Thus, you have decided to measure accelerations (both in the LINEAR dimensions of X, Y, and Z, and in the angular dimensions of roll Î¸, pitch Ï†, and yaw Ïˆ. See figure 3) using the accelerometer and gyroscope built into the iPhone. To do this you have strapped the iPhone to the equipment during the zero gravity testing and an app reads the accelerometer/gyroscope data, saves it into a text file, and emails it to you. The data file also has some information that is not relevant to your needs, such as magnetometer and attitude data, so these can be ignored within the data file. Here is what the beginning of a sample text file would look like:

DataCollection_2012_08_25_163542. txt (Preferred) sampling frequency set to 120 Hz. Your device may not support this sampling frequency, so always check your timestamps! ATTITUDE 20824. 418432 5. 760622 -0. 051823 71. 854103 ACCEL 20824. 419997 0. 014572 -0. 096497 -1. 016251 GYRO 20824. 420479 0. 019078 -0. 002094 0. 002610 MAGNETO 20824. 421015 -20. 378906 -59. 524216 -146. 901550 ATTITUDE 20824. 452569 5. 763187 0. 051214 71. 963783 ACCEL 20824. 452911 0. 019516 -0. 098434 -1. 002335 GYRO 20824. 455456 0. 042706 0. 013091 0. 014109 MAGNETO 20824. 455917 -19. 277344 -58. 432022 -149. 737488 ATTITUDE 20824. 484231 5. 778090 0. 099470 71. 903496 ACCEL 20824. 485857 -0. 016739 -0. 123932 -0. 975555 GYRO 20824. 486191 0. 044957 0. 024662 0. 011630 . . . Figure 4: The coordinate system of the iPhone during the experiment and the emailed data file format. The only data being analyzed from the file is the accelerometer and gyroscope data. You can assume that each file is in

this format. The iPhone axes are pictured above in Figure (4), and for the actual experiments the iPhone was secured flat on its back with the plane's forward velocity in the Y+ direction. Thus pitch is about the X-axis, roll about the Y-axis and yaw about the Zaxis (same as in figure 3). We want to extract the ACCEL and GYRO data, so let's take a closer look at one of these lines. ACCEL 20824. 419997 0. 014572 -0. 096497 -1. 016251 *** You can also assume each line will be in this format. Sensor type Sample time Ax Ay Az The sample time is a real number in seconds, but we will not be concerned with the sample time insomuch as the sample number (where both the first ACCEL and GYRO samples in the file are 1, etc.), thus the sample time value is meaningless to us. Also the Ax, Ay, and Az are linear accelerations measured in G's where 1 G is 32. 2 ft/s^2. For the purposes of this project, all linear acceleration values can be left in G's, while all angular accelerations (measured by GYRO) have units of radians/s^2. X axis units should be the sample number. You will need to read in this data, extract all relevant data from the file and plot each of the 6 accelerations. The easiest way to do this is to plot all the linear accelerations on one graph, while plotting all the angular accelerations on a separate graph. The final plots should have a similar format (though the data will be different): Sample # Acceleration in rads/s^2 Sample # Figure 5: Sample plots of accelerometer and gyroscope data with noise. You will be removing the noise of the data with a particular algorithm. One unfortunate setback to using the iPhone, however, is the noise level of the sensor. To correct and ' smooth out' some of this noise, a moving averages algorithm can be applied to the data, similar to the moving averages algorithm discussed by Prof. Attaway in class. For example, with a sample size, say 3 elements, the average value is

calculated across those 3 elements and substitutes in for the original value. Visually, this looks like the following: 4 2. 66 4. 66 rawData = [2 5 1 8 5 4 3 1 9 9 …] 4. 66 5. 66 2. 66 [2. 66 4. 66 4. 66 5. 66 4 2. 66 …+ ** This adjusted data is now the vector of acceleration values to plot. Figure 6: Pictorial representation of the moving mean algorithm. Tasks: In general, you will be responsible for the following. Larry, a previous BUSAT team member and abacus aficionado, wrote a working copy. However, his copy is very inefficient, and he only left behind an executable file and some scraps of code. Your job is to recreate a similar GUI, but make it more efficient. You can use the scraps of his code as a guideline. See the next section SPECIFICS for more detailed information regarding these tasks. NOTE: All of the functions in italic bold are SEPARATE FUNCTIONS from your ZEROG_GUI function. Thus, if used either within your ZEROG_GUI function or the callback functions for your GUI objects, THESE FUNCTIONS HAVE TO BE CALLED by passing the appropriate inputs, etc. ï,· Larry wrote a function, called LarrysFileRead, to read in the data from the text files. The problem is that he underestimated the large file sizes, thus, his code is very, very slow. He didn’t even comment his code. Your first task will be to interpret Larry’s code and write a faster, clearer way to accomplish the same thing. Make sure you pay attention to Larry’s output. We don’t want to change the way the data is outputted, only improve how the data is read in and thus, the speed of our GUI. Use tic/toc on Larry’s code and then on your adjusted code to verify the speed difference, then write your time vs. Larry’s time on your cover sheet. Reading in the data from an iPhone app . txt file (of which there are multiple, and the user will choose which file to load) and organizing this information into 6 vectors. 3 of these vectors will contain the linear acceleration data,

while the other 3 vectors will contain the angular acceleration data. For instance, if we use the example file on page 2 (lines in bold) the first elements in each of the 6 vectors (C_x, C_y, C_z, G_x, G_y, G_z) would be: C_x(1) = 0. 014572 C_y(1) = -0. 096497 C_z(1) = -1. 016251 G_x(1) = 0. 019078 G_y(1) = -0. 002094 G_z(1) = 0. 002610 ï,· ï,· Apply a moving averages algorithm to the data with a user-specified sample size. You are given part of this code in the mvgAverage function provided. You must determine what this code is doing, how it relates to taking a moving average, and write the remaining vectorized code that will return an averaged vector. Plotting the data (from a data set of the user's choice) on two separate axes, one for the accelerometer and one for the gyroscope. Combining all of these into an easy-to-use GUI for the rest of your space team to read and interpret. Having a GUI button that takes a snapshot of the current figure window and overlays a BUSAT watermark (to make it official), allowing you to save the image to a file. ï,· ï,· ï,· Specifics: You are given a number of files to get you started. First off, you will be given the function ZERO_GUI (must be all caps). It the start of a GUI, but it has not been completed. For it to work, a number of functions must be written, and then the rest of the GUI completed. The function's comments contain helpful information, and some directions about what needs to be done. Part A: Adjust Larry's function LarrysFileRead that loads the data from a user-chosen file and stores them into 6 vectors. It is fairly slow, and should be more efficient. Write a new function loadData that is more efficient. - The function should have one input and 6 outputs. - The input will be a string containing the name of the file to be read, including the extension '. txt'. - The output will be the 6 vectors that contain the Accel and Gyro data from the file. - The . txt files will be in the format shown above.

You will need to open the file, read in the data, and then organize the information. The first two lines of the files contain nothing needed. All of the lines starting with ATTITUDE and MAGNETO can be deleted. The timestamps are not important, instead, treat the first sample time as sample number one and use the sample number as the index into the vector. - Write a short separate script efficiencyTest that uses tic and toc to compare your function with Larry's. The script does not need any user input, it should just run LarrysFileRead and loadData on the same file and print the time it takes for each to run. Include both times on your cover sheet. Part B: Complete the function mvgAverage that calculates the running average of a vector. - The function should have two inputs and one output. - The first input should be the vector to be averaged, and the second input should be the user-specified sample size. You can assume the sample size will be anywhere from 2 to 10. - The output should be a vector that is the running average of the input vector, using the given sample size. - A portion of the code is given to you that contains a for loop using the circshift function. Determine what this loop is doing and write the rest of the function using vectorized code. (ie. no more loops!) Part C: Prepare the BUlogo. png to be a watermark. You DON'T have to write a new function for this. - Exchange the yellow background with a white background (hint: the yellow background color is the most abundant color in the logo). - Use the image property ' AlphaData' to make the logo more translucent. Part D: Plot your 6 un-adjusted vectors in the GUI. The comments in ZERO_GUI should help guide you in what needs to be written. - The GUI should have two axes similar to those shown on page 3. The top plot should display the linear acceleration data (all 3) and the bottom plot should display the angular acceleration data (all 3). Each vector within one of the

plots should be displayed in a different color. - Write code for radio buttons on the side of the GUI that allow the user to show which data to display. There should be a button for each axis of acceleration. When the GUI is first loaded, all the radio buttons should be selected. - Write code for a static text box that displays the date the current file was created and the size of the current data file. The box should have a title ‘ File General Information’. Type help dir for more information about the file size. The file creation date is within the file name. The file name, size and creation date should be saved as strings in a cell array, which is passed to the ‘ String’ property of the static text box. - Write code for a drop down menu to select which data file should be loaded, with a pushbutton to trigger the load. Read the GUI comments for more information on both this menu and the pushbutton. - There should be a toggle button to turn filtering on and off, and this code is already written for you (courtesy of our man Larry). Notice that if filtering is turned on, the callback function should call your mvgAverage function that will take in the current data vectors and adjust them based on a default sample size of 3. If filtering is turned off, the original data vectors are used in the plotting. - The sample size can also be determined by a slider bar value. Write code to that when the filtering is turned on, the slider bar appears and a user can choose a sample size. When off, the slider disappears. Under the slider there should also be a ‘ recalculate filtering’ push button whose callback will also call the mvgAverage function and readjust the data. Filter size should be from 2 to 10, - Finally, create a pushbutton that captures a screenshot of the figure. Part E: Write callback functions for each GUI object. Here, the names of your callbacks don’t matter as much because they call the separate functions you have written. - Create a slider callback function

that, when someone clicks on the slider buttons, the sample size is updated and displayed as a static text box above the slider. - Create a filter on/off callback function. As mentioned before, when the filter is turned on, it should calculate adjusted vectors by first calling mvgAverage on all them, using the value from the slider bar to determine the filter size. After the mvgAverage, the callback should call another function (a subfunction within - - ZERO_GUI, not separate) that determines whether the noise filter is on or off and plots the data set accordingly. Write another callback for the radio buttons on the side of the GUI. When a radio button for an axis is selected, it should make the corresponding plot visible, and when the radio button is deselected, it should make the corresponding plot invisible. Finally, when the dropdown menu is used, it should change which data is plotted. This will involve calling your separate function loadData and afterwards, mvgAverage. Part F: Add to the ZERO_GUI function a pushbutton that will save snapshot of the GUI and superimpose the BUlogo. png onto it. - The pushbutton will use the saveas function to save the snapshot as a . png file named GUIpic. png without the snapshot pushbutton on it (hint: pushbutton visibility turns off). - When the button is pressed, a camera snapshot sound should play using the camera. wav file and the wavread function. - Close the GUI after the pushbutton has been pressed. - A new figure window should appear that has the GUIpic. png displayed on it and is centered (hint: movegui) and the same size as the original ZERO_GUI figure window. - Give the user an option to choose where the logo will go on the GUIpic. png by using the ginput function. - Once the user chooses the spot for the logo to go, the watermark logo (from part C) should be superimposed onto the GUIpic. png image. Manually save this image using the figure window menus (file ïƒ saveas, etc.) as

BUSAT_SCREENSHOT. png. *There is an example of what this image should look like in the project 3 folder under BUSAT_SCREENSHOT Example. png. Deliverables: You must submit electronically a folder containing: ï,· All scripts and functions you wrote. ï,· A . mat file called ACCEL_GYRO_DATA. mat (also in all caps except the . mat) that holds the six vectors of data gathered from the file A_DataCollection_2012-08-25. txt. ï,· An image file, saved as a png, called GUIpic. png (all caps except the . png). This picture should be an image of your GUI with A_DataCollection_2012-08-25 loaded, with only the Z LINEAR ACCELERATION & ANGULAR ACCELERATIONS ABOUT THE X AXIS (PITCH) plotted. Do not worry about placing a watermark on this one. ï,· An image file, saved as a png, called BUSAT_SCREENSHOT. png (all caps except the . png). This picture should be the GUIpic. png superimposed with the watermark version of BUlogo. png. ï,· All the files necessary to run your program(all . dat files, and anything else) You must submit a hard copy of all your code, and include a print out of BUSAT_SCREENSHOT. png. Hints: - The file LarrysProj3Solution. exe is an executable file that can be run outside of windows. In order for it to work properly, it needs to be in a folder with a number of files, specifically BUlogo. png, camera. wav, LarrysFileRead, and at least one of the . txt data files. The best way for you to use it is to copy it onto your x drive and and run it from there. When you run it, a black command window should appear, and after a little while a GUI should appear. It is quite slow. If the command window disappears before a GUI appears, double check that all of the needed files are in the same folder. - Make sure to test run what you turn in with a clean start. Right before you turn your code in try running it in the folder you are turning in after using the clear command. This makes sure that some variables are not saved in your

workspace that the code cannot function without. -Use the debugger to find where problems are occurring. Especially useful since the GUI will be a function, and to see what values a variable has will be hard without stopping the code. Mention the inspect command and using the get command to see a list of properties while debugging. -Write clear code. Often when writing GUI’s, it is easier to see if you use the ellipsis to separate one long line of code into a series of shorter lines that are easier to read. This makes your code clearer, easier to edit, and simpler to debug. For example: myButton = uicontrol(fig1, ‘ Style’,’pushbutton’,... ‘ Units’,’Normalized’... ‘ Position’,*0. 1 0. 1 0. 2 0. 1+,... ‘ String’,’Push Me!!!’); Acknowledgements: I want to say thanks to Connor Cantelmo, Greg Maniatis, Alan Morse, Evan Ramos and Zac Sacher for all their hard work and contributions towards the project’s creation. An important shoutout to mother earth for providing the coffee beans. Most importantly I would like to thank Prof. Attaway not only for poetic license for this project idea, but for putting up with me for the last 5 semesters. I am sincerely grateful and honored to be a part of such a fantastic team of people.