BINUS UNIVERSITY
BINUS INTERNATIONAL

Assignment Cover Letter

(Individual Work)

Student Information:	**Surname**	**Given Names**	**Student ID Number**
1.	**Wotulo**	**Asoka**	**2101693611**

Course Code	**: COMP650**	**Course Name**	**: Introduction to Programming**
Class	**: L1BC**	**Name of Lecturer(s)**	: 1. Jude Martinez 2. Minaldi Loeis
Major	**: CS**		
Title of Assignment (if any)	: Double Pendulum Simulation		
Type of Assignment	**: Final Project**		
Submission Pattern			
Due Date	**: 6-11-2017**	**Submission Date**	**: 6-11-2017**

The assignment should meet the below requirements.

1. Assignment (hard copy) is required to be submitted on clean paper, and (soft copy) as per lecturer's instructions.
2. Soft copy assignment also requires the signed (hardcopy) submission of this form, which automatically validates the softcopy submission.
3. The above information is complete and legible.
4. Compiled pages are firmly stapled.
5. Assignment has been copied (soft copy and hard copy) for each student ahead of the submission.

Plagiarism/Cheating

BiNus International seriously regards all forms of plagiarism, cheating and collusion as academic offenses which may result in severe penalties, including loss/drop of marks, course/class discontinuity and other possible penalties executed by the university. Please refer to the related course syllabus for further information.

Declaration of Originality

By signing this assignment, I understand, accept and consent to BiNus International terms and policy on plagiarism. Herewith I declare that the work contained in this assignment is my own work and has not been submitted for the use of assessment in another course or class, except where this has been notified and accepted in advance.

Signature of Student: (Name of Student)

1. Asoka Wotulo

Double Pendulum Simulation

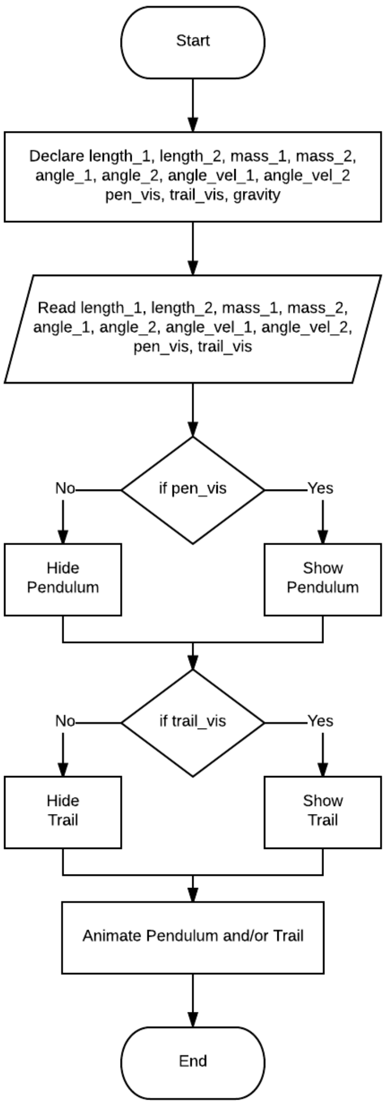
Name : Asoka Wotulo
ID : 2101693611

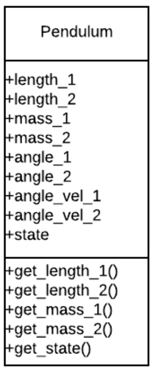
1. Description

Concept:

This program is meant to graphically simulate the double pendulum experiment. The double pendulum experiment is a simple system where a pendulum is attached on the end of another pendulum; however, what makes this system unique is that according to mathematical theory this system is chaotic, which means that it is a system that is very sensitive towards its initial conditions thus any minor changes will cause the system to change almost entirely.

2. Design


Start
Declare length_1, length_2, mass_1, mass_2, angle_1, angle_2, angle_vel_1, angle_vel_2 pen_vis, trail_vis, gravity
Read length_1, length_2, mass_1, mass_2, angle_1, angle_2, angle_vel_1, angle_vel_2, pen_vis, trail_vis
if pen_vis
No
Yes
Hide Pendulum
Show Pendulum
if trail_vis
No
Yes
Hide Trail
Show Trail
Animate Pendulum and/or Trail
End



3. Discussion

Implementation:

The things that were included as well as implemented within the program include: MatPlotLib, NumPy, SciPy, and real life Physics.

MatPlotLib was the main framework that was used in the creation of this project as it served as the graphical representation of the system itself. Furthermore, the main use of MatPlotLib is not only to plot the points resulted by the system but is also used in order to animate the movement of the pendulum to give a more accurate representation.

NumPy was used within this project due to its diverse mathematical potential, as this project has complex mathematics in order to calculate the next position. In addition to that NumPy was also a pivotal aspect of this project as it has the ability to create lists that have iterations smaller than 1; e.g. 0.5, 0.1, 0.001.

SciPy was implemented within this project as the equations handled by the project include differential equations, which require SciPy's integrate method in order to return a value for the needed time.

Finally, real life Physics plays a vital part within this system as the simulation is meant to resemble the real world as closely as possible and thus create the closest to real life simulation.

How it works:

How the program works can be understood briefly by the flowchart within section two, however, I will proceed to explain the steps in a more comprehensive and detailed manner.

As soon as the program starts you will be prompted to inserting a few numbers that will be used as the variables that the program will use; e.g. pendulum length, pendulum mass, initial pendulum angle, initial pendulum velocity, total time. Furthermore, the program will then ask you what you would like to see on the screen, which include the pendulum as well as a trail or plots left behind by the pendulum.

Afterwards, the program will then generate a list of numbers that will serve as the intervals of time that will be plotted; the numbers in the list will generate the time starting from 0 to the total time that the user input and will have 0.01s (10ms) as the increment.

Thereafter, the program will generate the area where the the pendulum will subside and where the trails will exist. Moreover, the program will hide or show the pendulum according to the user input; this will also happen for the trail of the pendulum.

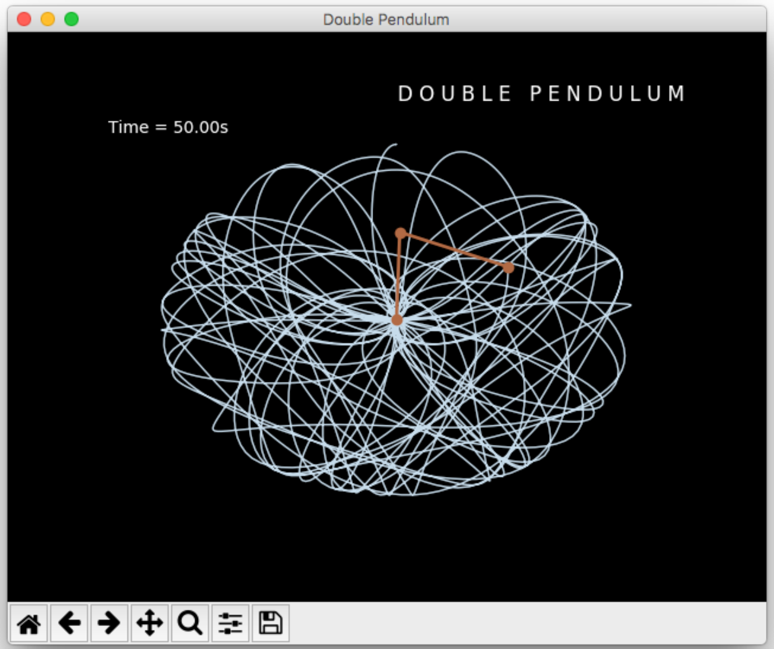
Subsequently, the program will go through every iteration based on the time array that was previously made and take the initial values for the angle and angular velocity then put them through the differential equation, which is the Euler-Lagrange equation, using SciPy's integration method then replace the initial states with the newly calculated values. For all of the resultant values that are created after the integration the program will use the newly calculated values and use those values in order to get the x and y positions for every iteration of time.

Finally, after the program has calculated the x and y positions for both pendulums it will then animate the generation of the positions based on the time iteration. Also the pendulum's center point will be at origin (0, 0).

Class Explanation:

The pendulum class within the program is what prompts the user into inputting the variables that will be used for the entirety of the program. Furthermore, because of the fact that the variables within the class are private then that will guarantee that the values of the variables will not change throughout the span of the program and can only be read.

4. Evidence



5. Resources

Altic, J. (2008, May 15). *Double Pendulum.* Retrieved Nov 5, 2017, from http://sophia.dtp.fmph.uniba.sk/~kovacik/doublePendulum.pdf