DUSIC IFY Programming Techniques

Python application for Projectile Motion Simulation

Jia Xiu Sai

Date: 2019-04-30

1 Python application for projectile motion simulation

1.1 Introduction

Simulation programs are used in engineering and science to investigate how complex systems will react to changing variables without doing the experiment in real-life. This saves time, cost and most importantly, it provides a safe environment as the engineers and scientist do not need to carry out the experiments in real life. In this project, the simulation will focus on a sphere object that is fired out of an air cannon at different speeds. It will show the changing speed and distance with both a spreadsheet and animated graphics from the variable parameters input by the user. This simulation will show how the initial angle, initial speed and weight of the projectile object affect the responding variables using physics equations taking gravitational force and air resistance into consideration.

1.1.1 Specifications

The minimum specification for this program was as follows:

The application must ...

- be written using the Python programming language
- use at least three relevant physics equations
- use the import function
- use a range of data types, including lists
- have a basic menu system with at least three subroutines to allow the user to:
  - have parsed input
  - enter system parameters and simulation properties
  - calculate and display some simulation data in tabular format
  - plot at least one graph of the data
  - optionally save to file in CSV format
  - optionally show a real-time animation of the simulation data using basic 2D graphics
  - exit the program in a controlled fashion

1.1.2 Projectile Motion Theory

In this program, the cannonball is calculated as a 2-dimensional projectile where the total velocity can be split into two individual forces, X and Y. X velocity only acts horizontally while the Y velocity only acts vertically. The object is supposed to continue moving in a straight line because of inertia but there are forces opposing it namely, air resistance and gravity. Air resistance that opposes the velocity is also a 2-dimensional force which acts in opposite directions to the horizontal and vertical velocity moving forward regardless of which direction the velocity is moving.[1] Furthermore, the gravity of Earth is also considered in this simulation which always pulls the cannonball downwards.[2] Figure 1 below shows the forces acting on the cannonball when it is moving in a parabolic path.

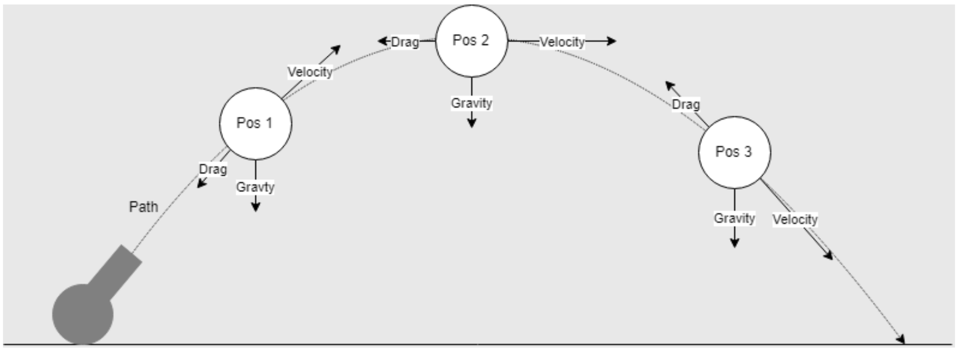

Figure 1: Illustration of a cannon and the forces acting on the cannonball

1.2 Design

The program is designed so that the program is both user-friendly and efficient. The following section will explain some of the design decision made in this project.

1.2.1 System Parameters

[682–691] The program uses the following system parameters and their units:

Variable	Parameter	Default Value	Unit
`mass`	Mass of the projectile	5.5	(Kilograms)
`height`	Height of initial projection	0.0	(Meters)
`speed`	The initial speed of projectile	100	(Meter per second)
`area`	The surface area of projectile	0.0154	(Meters^2).
`step`	Step size	0.01	(Seconds)
`ang_in_deg`	Initial angle	45	(Degrees)
`ang_in_rad`	Initial angle	0.79	(Radians)

The default values are determined by real-life examples. A cannonball with a mass of 5.5kg has a radius of 7.7cm. Thus, the frontal area is calculated to be 0.0154 meters^2.[3] Furthermore, The initial speed is the average speed of real-life cannon launch speed.[4] The step size is a value that can be adjusted to be more accurate but slower simulation by having a smaller step size or a faster but more inaccurate simulation with a larger step size. The step size greatly affects the smoothness of the simulation. There are 2 system variables to hold data for the angle because angle in degrees are used for user related display and input while the angle in radians is used for the program to run its calculations. The angle in radians is calculated when the user first inputs the value for angle in degrees.

1.2.2 Equations

The simulation is modelled using the following equations:

$$F_g = -mass \times gravity\ on\ Earth \tag{1}$$

$$F_d = 0.5 \times v \mid v \mid k_d \times area \times \rho \tag{2}$$

$$a = \frac{F_g - F_d}{m} \tag{3}$$

$$\frac{dv}{dt} = a \tag{4}$$

$$\frac{dx}{dt} = v \tag{5}$$

$$Z^2 = X^2 + Y^2 \tag{6}$$

The simulation algorithm uses the method of finite differences, or Euler's method, to determine approximate solutions to the differential equations of the system. After setting initial conditions, the algorithm iterates, calculating the forces acting on the cannonball and therefore the acceleration of the cannonball over small steps in time. At each point, the next values of velocity, distance and time are determined from current values. Errors are reduced and accuracy improved by using very small increments in time. However, a small increment may dramatically increase the time required for the calculation to be completed.

$$\frac{dv}{dt} \approx \frac{v_{i+1} - v_i}{h} \tag{7}$$

$$\frac{dx}{dt} \approx \frac{x_{i+1} - x_i}{h} \tag{8}$$

$$v_{i+1} = v_i - (a_i \times h) \tag{9}$$

$$x_{i+1} = x_i + (v_i \times h) \tag{10}$$

$$t_{i+1} = t_i + h \tag{11}$$

The basic steps of the calculate function:

1. The initial time set to zero.
2. The initial X displacement and Y displacement set to zero.
3. The initial X velocity and Y velocity are calculated using trigonometry.
4. The total velocity is set to the value input by the user.
5. Initial values are stored in their respective lists.
6. This loop runs until Y displacement reaches 0 again (Cannonball reaches ground level):
   (a) Force of gravity is calculated using equation (1).
   (b) Current X air resistance is calculated using equation (2).
   (c) Current X acceleration is calculates using equation (3).
   (d) Next X velocity is calculated using equation (9).
   (e) Next X distance is calculated using equation (10).
   (f) Current Y air resistance is calculated using equation (2).
   (g) Current Y acceleration is calculates using equation (3).
   (h) Next Y velocity is calculated using equation (9).
   (i) Next Y distance is calculated using equation (10).
   (j) Total velocity is calculated using the Pythagoras theorem (6).
   (k) Next time value is calculated using equation (11).
   (l) Results are added to their respective lists.

The values for X and Y velocity are calculated separately because the air resistance and gravity acting on them are different. The X velocity is only opposed by air resistance but the Y velocity is opposed by both air resistance and gravity (when ascending). The gravity increases of the acceleration when the cannonball is descending in altitude thus, increasing its speed. The air resistance is always the opposing the Y velocity regardless when the cannonball is increasing or decreasing in altitude.The equation (2) drag uses an absolute value for the current velocity in its equation because it will ensure that when used in the equation (3), the value will be added to the force of gravity when the cannonball is ascending in altitude and cancel out by the gravity when descending.

1.2.3 Problem Decomposition

The program is designed around the main menu which allows the user to run functions by inputting commands. Figure 2 below shows the structure diagram of the program with the main components.

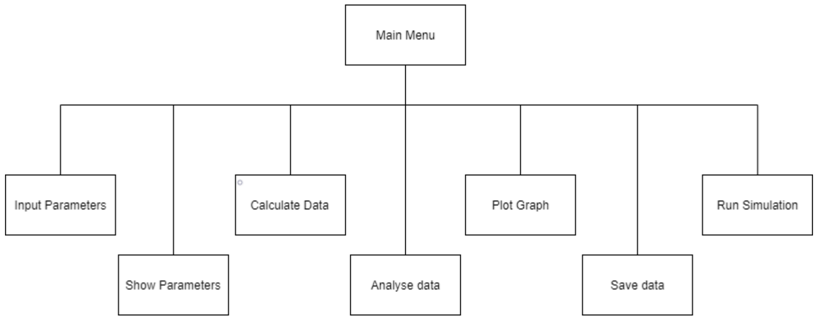

Figure 2: Program Structure

1.2.4 Test Plan

The following test plans templates in Figures 3, 4 and 5 are to be filled by doing manual testing with running the code. The test plans aim to find any errors in the program.

1.3 Implementation

The program was implemented by creating a menu system with functions defined for each of the commands listed on the menu. The menu allows the user to input commands to start different functions. This program uses Python 3 and referred to [5] where required.

1.3.1 Main program loop

Figure 6 shows the general flowchart of the main program. The code listing can be found in Appendix A.

[682–691] The program begins by defining and initialising the system parameters:

`mass`	Mass of the projectile	(Kilograms)
`height`	Height of initial projection	(Meters)
`speed`	The initial speed of the projectile	(Meter per second)
`area`	The surface area of projectile	(Meters^2).
`step`	Step size	(Seconds)
`ang_in_deg`	Initial angle	(Degrees)
`ang_in_rad`	Initial angle	(Radians)

[697–703] Data lists are then created for holding the simulation data:

`t[]`	time	(Seconds)
`x[]`	X displacement	(Meters)
`y[]`	Y displacement	(Meters)
`xv[]`	X velocity	(Meter per second)
`yv[]`	Y velocity	(Meter per second)
`v[]`	Velocity	(Meter per second)

[760–842] The main program loop is a `while` loop. This loop repeats until the user enters the required input to quit the program, which in this case is the character `quit`.

[760–725] First, the menu is displayed on the console with the commands and functions.

[728–841] A basic `if-elif` structure is used to invoke a specific function depending on the menu choice entered by the user.

1.3.2 Function: input parameters

[59–84] Figure 7 shows the flowchart for the function `input_parameters` that allows the user to input parameters to run calculations for the projectile. The function checks every value input by the user by running the value through function `get_float`. The value input is then recorded if it passes all requirements.

1.3.3 Function: get float

[21–53] Figure 8 shows the flowchart for function `get_float`. Although the user is given the ability to input any values, this function checks the value input for three requirements:

1. must be smaller than maximum value.
2. must be larger than the minimum value.
3. must be a float.

If all three requirement are meet, the value input is recorded. However, if one of the requirements are not met, the pre-set default value will be recorded instead.

1.3.4 Function: show parameters

[90–104] Figure 9 shows the flowchart for this simple function that displays the parameters input by the user that have been checked by the function `get_float`. A basic `if-elif` structure checks if parameters are entered to prevent the function from crashing.

1.3.5 Function: calculate data

Figure 10 shows the general flowchart of the function `calculate_data` used to calculate the projectile motion simulation data. A basic `if-elif` structure checks if parameters are entered to prevent the function from crashing. The system parameters are passed to the function when it is called.

[120–144] Firstly, variables for both current and next are declared for holding the calculations. Furthermore, the fixed parameters are declared:

`g`	Gravity on Earth	9.807	(Meters per second^2)) [6]
`a_d`	Air density at sea-level	1.225	(Kilogram per meter^3) [7]
`drag`	Drag coefficient	0.47	

[147–152] Next, lists are declared for holding the calculation results, and initial values are appended to the lists to create the first set of results i.e. `t[0]`, `x[0]`, `y[0]`, `xv[0]`, `yv[0]` and `v[0]`.

[177–231] A `while` loop is used with an incremental index `i`. During each execution of the loop, the next output value is calculated using Equation (11) and appended to the list of output values. The time value is incremented by the step size and appended to the list of time values.

[236] Once the time value reaches the total simulation time required, the loop is exited and the lists of data `i,t,xv,yv,x,y,v` are returned to the main program.

1.3.6 Function: analyse data

[242–295] Figure 11 shows the flowchart for the function that displays additional information (Furthest distance and highest altitude) from the data set calculated using the calculate data function. A basic `if-elif` structure to check if the user entered the parameters and if the data is calculated to prevent the function from having errors.

[249–254] The console output present the user with a option to exit the function to the main menu or proceed to display the analysis for the data.

1.3.7 Function: plot graph

[301–404] Figure 12 shows the flowchart for the function that presents the option to display one of two graphs using the data set calculated. A basic `if-elif` structure to check if the user entered the parameters and if the data is calculated to prevent the function from having errors. A second menu with commands for 2 different graphs with appear and it allows the user to choose which graph to display first.

1.3.8 Function: save data

[414–461] Figure 13 shows the flowchart for the function that saves the data set in an external CSV file. A basic `if-elif` structure to check if the user entered the parameters and if the data is calculated to prevent the function from having errors.

1.3.9 Function: run simulation

[467–668] Figure 13 shows the flowchart for the function that runs an animation using the current data set to provide the user with a better visualisation of the projectile motion in real time. A basic `if-elif` structure to check if the user entered the parameters and if the data is calculated to prevent the function from running and encountering errors.

[487–496] The function determines the scale for the visuals to ensure the cannonball to always be within the screen. The scale adjusts the change in movement of the cannonball on the screen.

1.4 Testing

Code testing is an important part of programming as by testing code, the developers are able:

1. To identify errors
2. To ensure code meets the specification (fit for purpose)
3. To ensure code is efficient
4. To ensure code is maintainable

By testing a program, companies are able to prevent financial losses and delayed deadlines due to faulty programs. It ensures the code to be properly functioning and is able to process the tasks properly.

1.4.1 Completed Test Tables

The following completed test tables in Figures 15, 16 and 17 are filled in after doing testing with the templates provided.

1.4.2 Testing evidence for Test table 1

When first running the program, the system menu and system information can be seen in the following console output:

```
 DUISC
 Projectile Motion Simulator
 Jia Xiu Sai-2427165
 22:27 21/5/2019
 Main menu
------------------------------
 Commands| Functions
------------------------------
    1    | Change parameters
    2    | Show current parameters
    3    | Calculate and show current data set
    4    | Analyse data
    5    | Plot graph
```

```
    6    | Save data
    7    | Run visuals
   quit  | Quit program

Enter command:
```

The following console output was displayed when Test 1 in Test table 1 (15) was carried out:

```
 DUISC
 Projectile Motion Simulator
 Jia Xiu Sai-2427165
 12:32 24/5/2019
 Function: Change parameters
------------------------------
 Enter the mass of the cannonball in kg (1-20): 10
 Value recorded.
 Enter the initial height (0-10): 5
 Value recorded.
 Enter the initial speed of the cannonball in m/s (10-200): 100
 Value recorded.
 Enter the frontal area of cannonball in meters2 (0.005-1): 0.02
 Value recorded.
 Enter the step size in seconds (0.001-1): 0.005
 Value recorded.
 Enter the initial angle in degrees (10-90): 30
 Value recorded.
```

The following console output was displayed when Test 2 in Test table 1 (15) was carried out:

```
 DUISC
 Projectile Motion Simulator
 Jia Xiu Sai-2427165
 12: 6 24/5/2019
 Function: Change parameters
------------------------------
 Enter the mass of the cannonball in kg (1-20): 0
 Oops!  The value of the number is too small. Using default value.
 Enter the initial height (0-10): -10
 Oops!  The value of the number is too small. Using default value.
 Enter the initial speed of the cannonball in m/s (10-200): -5
 Oops!  The value of the number is too small. Using default value.
 Enter the frontal area of cannonball in meters2 (0.005-1): 0.00001
 Oops!  The value of the number is too small. Using default value.
 Enter the step size in seconds (0.001-1): 0.0001
 Oops!  The value of the number is too small. Using default value.
 Enter the initial angle in degrees (10-90): 0
 Oops!  The value of the number is too small. Using default value.
```

The following console output was displayed when Test 3 in Test table 1 (15) was carried out:

```
 DUISC
 Projectile Motion Simulator
 Jia Xiu Sai-2427165
 12:12 24/5/2019
 Function: Change parameters
------------------------------
 Enter the mass of the cannonball in kg (1-20): 50
 Oops!  The value of the number is too big. Using default value.
 Enter the initial height (0-10): 50
 Oops!  The value of the number is too big. Using default value.
 Enter the initial speed of the cannonball in m/s (10-200): 300
 Oops!  The value of the number is too big. Using default value.
 Enter the frontal area of cannonball in meters2 (0.005-1): 5
 Oops!  The value of the number is too big. Using default value.
 Enter the step size in seconds (0.001-1): 10
 Oops!  The value of the number is too big. Using default value.
 Enter the initial angle in degrees (10-90): 180
 Oops!  The value of the number is too big. Using default value
```

The following console output was displayed when Test 4 in Test table 1 (15) was carried out:

```
DUISC
Projectile Motion Simulator
Jia Xiu Sai-2427165
12:19 24/5/2019
Function: Change parameters
------------------------------
Enter the mass of the cannonball in kg (1-20): heavy
Oops! That was not a valid number. Using default value.
Enter the initial height (0-10): high
Oops! That was not a valid number. Using default value.
Enter the initial speed of the cannonball in m/s (10-200): fast
Oops! That was not a valid number. Using default value.
Enter the frontal area of cannonball in meters2 (0.005-1): average
Oops! That was not a valid number. Using default value.
Enter the step size in seconds (0.001-1): daily
Oops! That was not a valid number. Using default value.
Enter the initial angle in degrees (10-90): fish
Oops! That was not a valid number. Using default value.
```

The following console output was displayed when Test 5 in Test table 1 (15) was carried out:

```
DUISC
Projectile Motion Simulator
Jia Xiu Sai-2427165
12:21 24/5/2019
Function: Change parameters
------------------------------
Enter the mass of the cannonball in kg (1-20): !
Oops! That was not a valid number. Using default value.
Enter the initial height (0-10): @
Oops! That was not a valid number. Using default value.
Enter the initial speed of the cannonball in m/s (10-200): #
Oops! That was not a valid number. Using default value.
Enter the frontal area of cannonball in meters2 (0.005-1): $
Oops! That was not a valid number. Using default value.
Enter the step size in seconds (0.001-1): %
Oops! That was not a valid number. Using default value.
Enter the initial angle in degrees (10-90): ^
Oops! That was not a valid number. Using default value.
```

The following console output was displayed when Test 6 in Test table 1 (15) was carried out:

```
DUISC
Projectile Motion Simulator
Jia Xiu Sai-2427165
11:56 24/5/2019
Function: Change parameters
------------------------------
Enter the mass of the cannonball in kg (1-20):
Oops! That was not a valid number. Using default value.
Enter the initial height (0-10):
Oops! That was not a valid number. Using default value.
Enter the initial speed of the cannonball in m/s (10-200):
Oops! That was not a valid number. Using default value.
Enter the frontal area of cannonball in meters2 (0.005-1):
Oops! That was not a valid number. Using default value.
Enter the step size in seconds (0.001-1):
Oops! That was not a valid number. Using default value.
Enter the initial angle in degrees (10-90):
Oops! That was not a valid number. Using default value.
```

1.4.3 Testing evidence for Test table 2

The following console output was displayed when Test 1 in Test table 2 (16) was carried out:

```
DUISC
Projectile Motion Simulator
Jia Xiu Sai-2427165
22:27 21/5/2019
Function: Show current parameters
------------------------------
Current parameters of projectile object:
Mass = 5.50 kg
Height = 0.0 meters
Speed = 100.00 meters per seconds
Surface area = 0.1540 meters2
Step size = 0.01 seconds
Initial angle = 45.00 degrees (0.79 radians)

Would you like to run calculations?

1: Yes
Press any key to go back to main menu.

Enter command:
```

The following console output was displayed when Test 2 in Test table 2 (16) was carried out:

```
DUISC
Projectile Motion Simulator
Jia Xiu Sai-2427165
22:27 21/5/2019
Function: Calculate and show current data set
------------------------------

Data table:

Step|  Time  |  X_Velocity |  Y_Velocity |  Velocity |  X_Distance  |  Y_Distance
 0  |  0.00  |    70.670   |   70.572    |  100.000  |    0.000     |    0.000
 1  |  0.01  |    70.630   |   70.434    |  99.874   |    0.707     |    0.707
 2  |  0.02  |    70.590   |   70.296    |  99.748   |    1.414     |    1.413
 3  |  0.03  |    70.550   |   70.158    |  99.622   |    2.120     |    2.117
 4  |  0.04  |    70.510   |   70.020    |  99.496   |    2.826     |    2.820
 5  |  0.05  |    70.470   |   69.883    |  99.370   |    3.532     |    3.522
 6  |  0.06  |    70.430   |   69.745    |  99.245   |    4.237     |    4.222
 7  |  0.07  |    70.390   |   69.608    |  99.120   |    4.941     |    4.921
 8  |  0.08  |    70.350   |   69.471    |  98.995   |    5.646     |    5.618
 9  |  0.09  |    70.310   |   69.334    |  98.870   |    6.349     |    6.314
 10 |  0.10  |    70.270   |   69.197    |  98.745   |    7.053     |    7.009
.................................................................................
1312|  13.12 |    40.437   |   -59.051   |  71.519   |    692.778   |    5.289
1313|  13.13 |    40.423   |   -59.121   |  71.569   |    693.182   |    4.700
1314|  13.14 |    40.410   |   -59.191   |  71.620   |    693.587   |    4.109
1315|  13.15 |    40.397   |   -59.261   |  71.670   |    693.991   |    3.518
1316|  13.16 |    40.384   |   -59.331   |  71.720   |    694.395   |    2.926
1317|  13.17 |    40.371   |   -59.401   |  71.771   |    694.799   |    2.333
1318|  13.18 |    40.358   |   -59.470   |  71.821   |    695.203   |    1.740
1319|  13.19 |    40.345   |   -59.540   |  71.871   |    695.606   |    1.146
1320|  13.20 |    40.331   |   -59.609   |  71.921   |    696.010   |    0.551
1321|  13.21 |    40.318   |   -59.679   |  71.971   |    696.413   |    -0.044
------------------------------
Number of data points for:
Time - 1322
Velocity - 1322
X - 1322
Y - 1322
------------------------------

Would you like to display analysis?

1: Yes
Press any key to go back to main menu.

Enter command:
```

As shown above, the last line of calculation showed that the Y displacement to be a negative value which is not supposed to happen as the `while` loop is supposed to stop at when Y displacement reaches 0. The value for Y distance 0 and jumps from 0.551 to -0.044. This is because the simulation uses Euler's method to determine approximate solutions to the differential equations of the system and inaccuracy can be reduced by using a smaller step value but the calculation will take longer as more steps need to be calculated for the same amount of distance.

The following console output was displayed when Test 3 in Test table 2 (16) was carried out:

```
DUISC
Projectile Motion Simulator
Jia Xiu Sai-2427165
22:27 21/5/2019
Function: Analyse data
-----------------------------
Data analysis:

Furthest distance achieved at:
X-coordinate - 226.571 meters
Y-coordinate - -0.607 meters
Speed - 33.337 meter per seconds
Time - 9.15 seconds

Highest distance achieved at:
X-coordinate - 146.289 meters
Y-coordinate - 101.214 meters
Speed - 21.703 meter per seconds
Time - 3.95 seconds

Press any key to go back to main menu.
```

The following console output was displayed when Test 4 and 5 in Test table 2 (16) was carried out:

```
DUISC
Projectile Motion Simulator
Jia Xiu Sai-2427165
22:27 21/5/2019
Function: Plot graph
-----------------------------

Commands| Graph:
   1    | Distance Y against Distance X
   2    | Velocity against Time

Input any key to go back to main menu.

Enter command:
```

When on of the command is selected, the console output becomes:

```
DUISC
Projectile Motion Simulator
Jia Xiu Sai-2427165
22:27 21/5/2019
Function: Plot graph
-----------------------------
Plotting graph......

Press any key to go back.
```

Test 4 will create Graph 1: Displacement Y against displacement X as shown in Figure 18. The highest height achieved by the cannonball is marked [8] and labelled [9]. This extra feature was later added to the program.

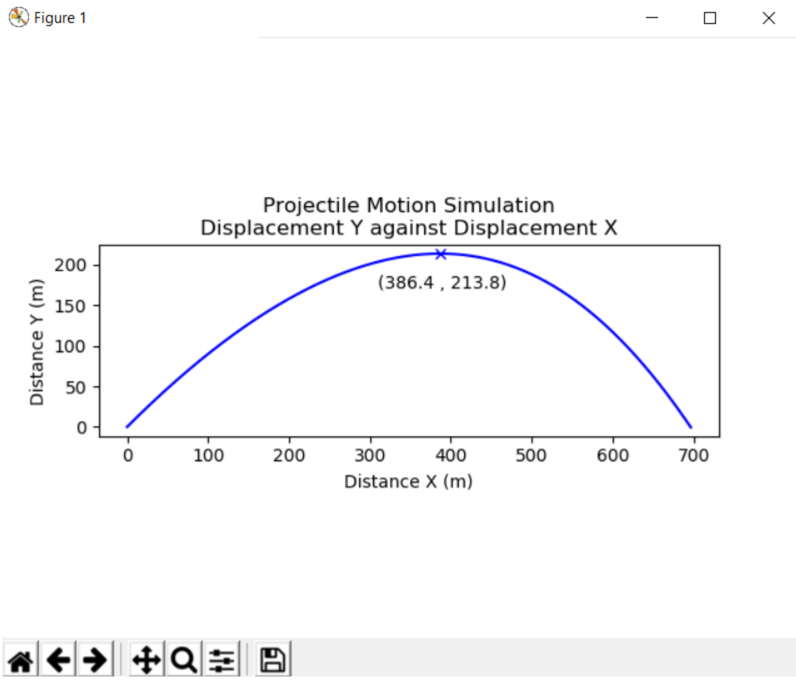

Figure 18: Graph 1

Test 5 will create Graph 2: Velocity against time as shown in Figure 19

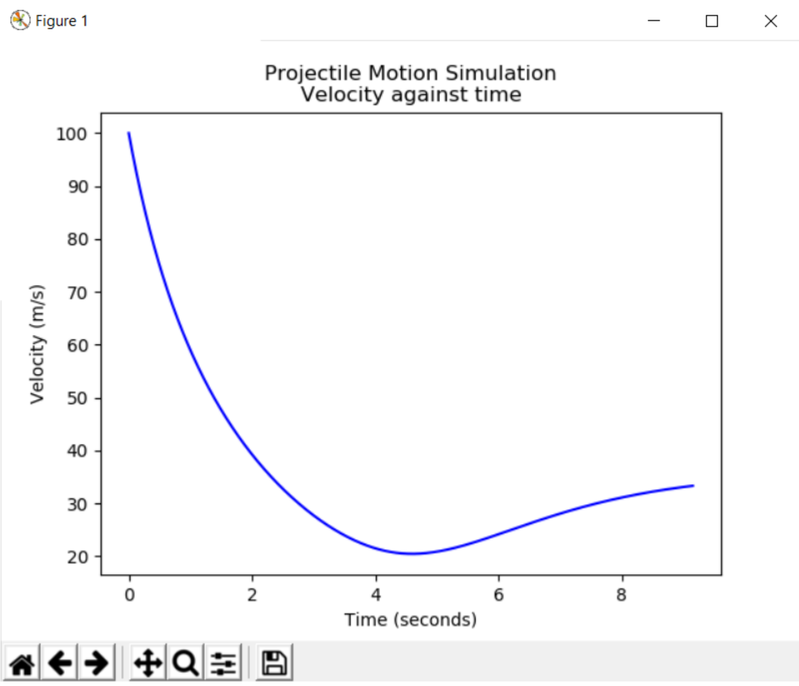

Figure 19: Graph 2

As shown above, the velocity of the cannonball rapidly decreases but at no point it reaches 0 because it is still moving forward. However, the Y velocity at the lowest point is 0 as the cannonball is starting it's descend towards the ground.

The following console output and Figure 20 was displayed when Test 6 in Test table 2 (16) was carried out:

```
DUISC
Projectile Motion Simulator
Jia Xiu Sai-2427165
22:27 21/5/2019
Function: Save data
-----------------------------
Data is now saved in file name: 'Projectile_Motion.csv'.

Press any key to go back to main menu.
```

Step	Time(Seconds)	X_Velocity(m/s)	Y_Velocity(m/s)	Velocity(m/s)	X_Distance(m)	Y_Distance(m)
0	0.00	7.071	7.071	10.000	0.000	0.000
1	0.00	6.985	6.975	9.871	0.007	0.007
2	0.00	6.900	6.881	9.745	0.014	0.014
3	0.00	6.818	6.789	9.622	0.021	0.021
4	0.00	6.738	6.700	9.502	0.028	0.028
5	0.01	6.659	6.613	9.385	0.035	0.034
6	0.01	6.583	6.527	9.270	0.041	0.041
7	0.01	6.508	6.444	9.159	0.048	0.048
8	0.01	6.435	6.362	9.049	0.054	0.054
9	0.01	6.363	6.283	8.942	0.061	0.060
10	0.01	6.293	6.205	8.838	0.067	0.067

Figure 20: Data is now save in csv.file "Projectile Motion"

The following console output and Figure 21 and 22 was displayed when Test 7 in Test table 2 (16) was carried out:

```
DUISC
Projectile Motion Simulator
Jia Xiu Sai-2427165
22:41 21/5/2019
Function: Run simulation
------------------------------
Running simulation......
```

In Figure 21, the simulation has just started and is waiting for the user to press any key to start the visualisation. The animation provide real-time information like iteration, velocity and time. A real-time indicator is added to inform the user if the program in real-time.

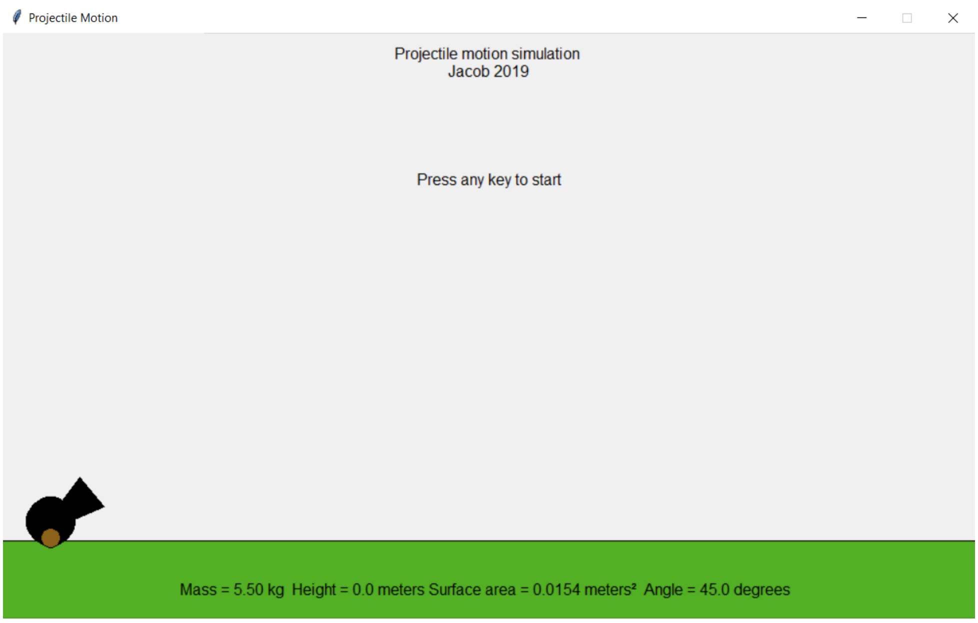

Figure 21: Simulation before starting

In Figure 22, the simulation has been started and stopped by the user by pressing any key again. The additional data showing the highest and furthest distance achieved are added to provide the user with more information.

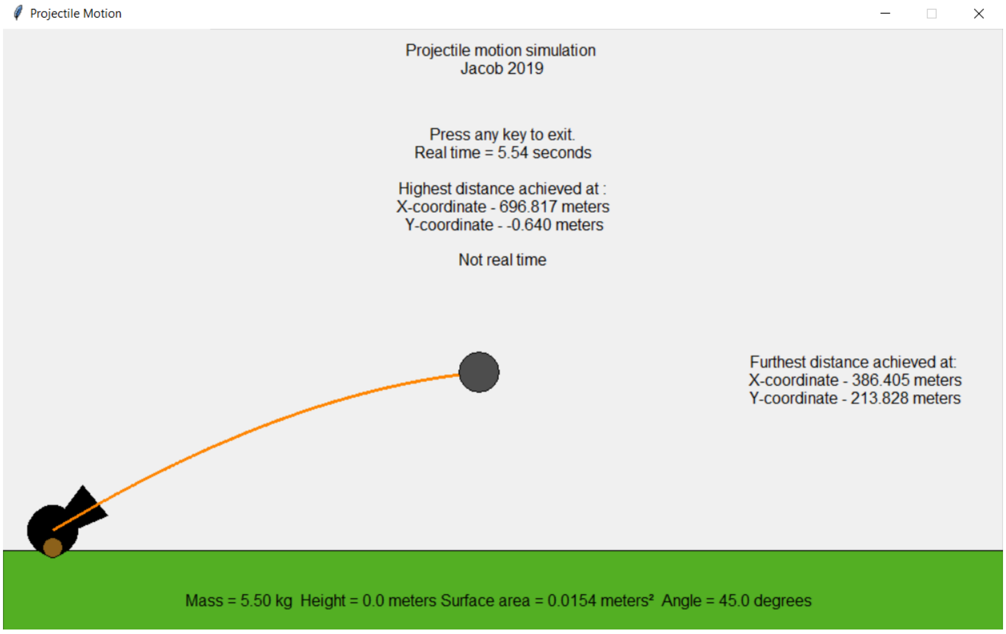

Figure 22: When the simulation is running

1.4.4 Testing evidence for Test table 3

The following console output was displayed when Test 1,2,3,4,5 and 6 in Test table 3 (17) was carried out:

```
DUISC
Projectile Motion Simulator
Jia Xiu Sai-2427165
22:47 21/5/2019
Function: Analyse data
------------------------------
No parameters enter yet.

Press any key to go back to main menu.
```

This function is to prevent the user from crashing the program by not inputting any parameters.

The following console output was displayed when Test 7,8,9 and 10 in Test table 3 (17) was carried out:

```
DUISC
Projectile Motion Simulator
Jia Xiu Sai-2427165
22:47 21/5/2019
Function: Run simulation
------------------------------
No data calculated yet.

Press any key to go back to main menu.
```

This function is to prevent the user from crashing the program by not calculating the data set.

The following console output was displayed when Test 11 in Test table 3 (17) was carried out:

```
 Quitting...
 Program has ended
 Bye


------------------
(program exited with code: 0)

Press any key to continue . . .
```

1.4.5 Dry run

Figure 23 shows the dry run of function `calculate_data` by manually inputting functions into an excel document and calculating the values for the first 10 iteration using default parameters by going through all the steps as shown in Section 1.2.2.

The results from the dry run matches with the calculation from the program, this shows that the program has no calculation error.

1.5 Conclusion

1.5.1 Key findings

The program has been able to use loops and functions to create a menu to simulate the projectile motion of a cannonball with the parameters input from a user. Furthermore, it give an idea of how a cannonball will react in a similar experiment in real-life but it is slightly inaccurate as other factors like wind are not considered in this simulation. This simulation should be used as a reference and should not replace real-life data collected from experiments. The data calculated is an estimation using the limited information used in the calculations.

1.5.2 Future improvements

Future improvement that can be implemented include:

- Calculation
  - Take into consideration other forces like wind.
    - This feature has been tested but the results were not always consistent as the vector can only either be more with or against the velocity in the calculations thus, causing the wind to be only working in on direction. Furthermore, if the wind is moving the projectile further or faster, the calculation will have to continue for a longer period of time. It has been decided that it will be more realistic to exclude this feature until a more optimal solution can be implemented.
  - Ability to change the projectile to other objects like a Frisbee.
    - This has not been tested as wind and lift will play a major part in the simulation as shown in Figure 24 and the reason stated in the previous point, it is more realistic to exclude this feature until a more optimal solution can be implemented.

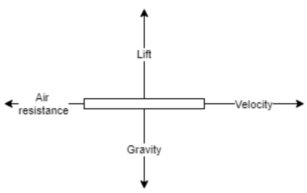

Figure 24: Force diagram of a Frisbee

- Animation
  - A changing scale is calculated using the furthest distance divided by the margin size.
    * This has been tested and the object will always land at the same spot in the animation thus, fully utilising the margin. However, if the angle is changed to a border value (5 or 85 degrees), the scale will be too small/big making the projectile move out of the margin.
  - A ruler that shows how far or high the projectile travelled that changes according to the maximum distance or height.
  - A vector with a label or colour code which changes with the magnitude and direction of the force acting on the projectile object.
  - A pause and play option.
- Graphs
  - An option to change the axis of the graph to other data calculated like acceleration or air resistance.

1.6 Evaluation

1.6.1 Appraisal

All the requirements are met for this project include:

- The code to be written in Python language. The code is shown in Appendix A.
- Using 12 Physic and Math equation. The equations used are shown in Section 1.2.1.
- Using the import function
- Using lists to store data. The function of this feature is shown in Section 1.3.1.
- User input required to run the program. The function of this feature is shown in Test 1 in Test Table 2 shown in Section 1.4.3.
- Allow user to change system parameters. The function of this feature is shown in all the Tests done in Test table 1 shown in Section 1.4.2.
- Allow user to plot 2 different graphs. The function of this feature is shown in Test 4 and 5 in Test table 2 shown in Section 1.4.3.
- Allow user to save data in a CSV file. The function of this feature is shown in Test 6 in Test table 2 shown in Section 1.4.3.
- Allow user to run a real-time animation. The function of this feature is shown in Test 7 in Test table 2 shown in Section 1.4.3.
- Allow user to exit program properly as shown in Test 11 in Test table 3 shown in Section 1.4.4.

1.6.2 Limitations

Although all specifications are met, there are some limitations in this program. The limitations include:

- Unable to calculate data at a specific point in the program. Without calculating the whole simulation. This is because of the lack of time to research a more advanced way to calculate the data required in the data set instead of using the Euler's method where it is an estimation.
- Unable to reliably compare the results in simulation as there is no recourse or time available to fire a cannon in real life and record the real-time statistics. Furthermore, cannons are no longer used in modern times as more advanced alternatives are available.
- Unable to create a better representative of a cannon in the animation because the module `graphics` doesn't allow shapes to be placed diagonally.

2 References

[1] N. Hall. Drag of a Sphere. [Online]. Available: https://www.grc.nasa.gov/www/k-12/airplane/dragsphere.html

[2] ——. Ballistic Flight Equation. [Online]. Available: https://www.grc.nasa.gov/www/k-12/airplane/ballflght.html

[3] D. A. Collins. British Cannonball Sizes. [Online]. Available: https://www.arc.id.au/Cannonballs.html

[4] R. Allain. CANNONBALLS: SIZE MATTERS. [Online]. Available: https://www.wired.com/2011/12/cannon-balls-size-matters/

[5] P. S. Foundation. Python 3.7.3rc1 documentation. [Online]. Available: https://docs.python.org/3/

[6] M. Williams. How strong is the force of gravity on Earth? [Online]. Available: https://phys.org/news/2016-12-strong-gravity-earth.html

[7] A. M. Helmenstine. What Is the Density of Air at STP? [Online]. Available: https://www.thoughtco.com/density-of-air-at-stp-607546

[8] M. development team. Markevery Demo. [Online]. Available: https://matplotlib.org/gallery/lines_bars_and_markers/markevery_demo.html

[9] ——. matplotlib.pyplot.annotate. [Online]. Available: https://matplotlib.org/api/_as_gen/matplotlib.pyplot.annotate.html

Test Plan 1

No.	Test Description	Test Data	Expected outcome	Actual outcome	Improvements
1	Test if parameters within range are recorded properly.	Mass – 10kg Speed – 100 m/s Step – 0.005 s Height – 5 meters Area – 0.02 m^2 Angle – 30 degrees	All values should be recorded and displayed in show current parameters console display.		
2	Test if parameters below range are rejected and default value is used.	Mass – 0kg Speed – 5 m/s Step – 0.0001 s Height – -10 meters Area – 0.00001 m^2 Angle – 0 degrees	All values should be rejected and the default values are displayed in show current parameters console display.		
3	Test if parameters above range are rejected and default value is used.	Mass – 50 kg Speed – 300 m/s Step – 10 s Height – 50 meters Area – 5 m^2 Angle – 180 degrees	All values should be rejected and the default values are displayed in show current parameters console display.		
4	Test if letters are input for parameters.	Mass – heavy Speed – fast Step – daily Height – high Area – average Angle – fish	All values should be rejected and the default values are displayed in show current parameters console display.		
5	Test if symbols are input for parameters.	Mass – ! Speed – # Step – % Height – @ Area – $ Angle – ^	All values should be rejected and the default values are displayed in show current parameters console display.		
6	Test if no values are input for parameters.	Mass – Speed – Step – Height – Area – Angle –	All values should be rejected and the default values are displayed in show current parameters console display.		

Figure 3: Test-1 template

Test Plan 2

No.	Test Description	Test Data	Expected outcome	Actual outcome	Improvements
1	Try command 2 (show parameters) with default parameter.	Command 2	Console output will proceed to display parameters when command 2 is selected.		
2	Try command 3 (calculate data) with default parameter.	Command 3	Console output will proceed to calculate data when command 3 is selected.		
3	Try command 4 (analyse data) with default parameter and data set calculated.	Command 4	Console output will proceed to display analysis of the data calculated when command 4 is selected then		
4	Try command 5 (plot graph) to display graph 1 with default parameter and data set calculated.	Command 5, Command 1	Console output will proceed to display another menu for a selection of graphs and graph 1 will be displayed when selected.		
5	Try command 5 (plot graph) to display graph 2 with default parameter and data set calculated.	Command 5, Command 2	Console output will proceed to display another menu for a selection of graphs and graph 2 will be displayed when selected.		
6	Try command 6 (save data) with default parameter and data set calculated.	Command 6	The data set will be save in a csv file named "Projectile_Motion".		
7	Try command 7 (run simulation) with default parameter and data set calculated.	Command 7	Simulation will run when the user press any key and stop either when the user presses any key again or when the cannonball reaches the ground again.		

Figure 4: Test-2 template

Test Plan 3

No.	Test Description	Test Data	Expected outcome	Actual outcome	Improvements
1	Try command 2 (show parameters) without input parameters	Command 2	Console output will inform user that no parameters have been input.		
2	Try command 3 (calculate data) without input parameters	Command 3	Console output will inform user that no parameters have been input.		
3	Try command 4 (analyse data) without input parameters	Command 4	Console output will inform user that no parameters have been input.		
4	Try command 5 (plot graph) without input parameters	Command 5	Console output will inform user that no parameters have been input.		
5	Try command 6 (save data) without input parameters	Command 6	Console output will inform user that no parameters have been input.		
6	Try command 7 (run simulation) without parameters	Command 7	Console output will inform user that no parameters have been input.		
7	Try command 4 (analyse data) with parameters but without calculating data set	Command 4	Console output will inform user that data has not been calculated.		
8	Try command 5 (plot graph) with parameters but without calculating data set	Command 5	Console output will inform user that data has not been calculated.		
9	Try command 6 (save data) with parameters but without calculating data set	Command 6	Console output will inform user that data has not been calculated.		
10	Try command 7 (run simulation) with parameters but without calculating data set	Command 7	Console output will inform user that data has not been calculated.		
11	Try command quit (quit program)	Command quit	Console output will inform user that program is quitting		

Figure 5: Test-3 template

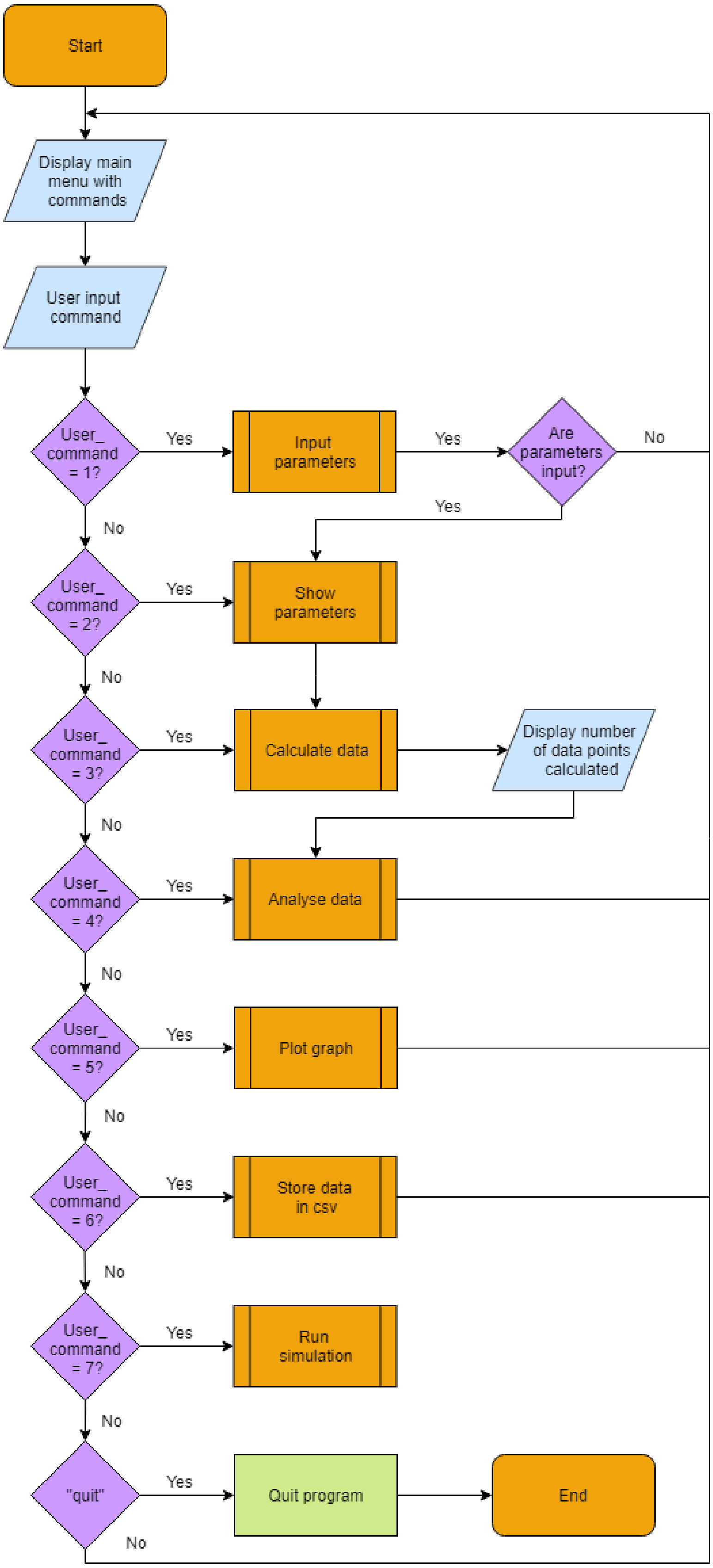

Figure 6: Flowchart: main program

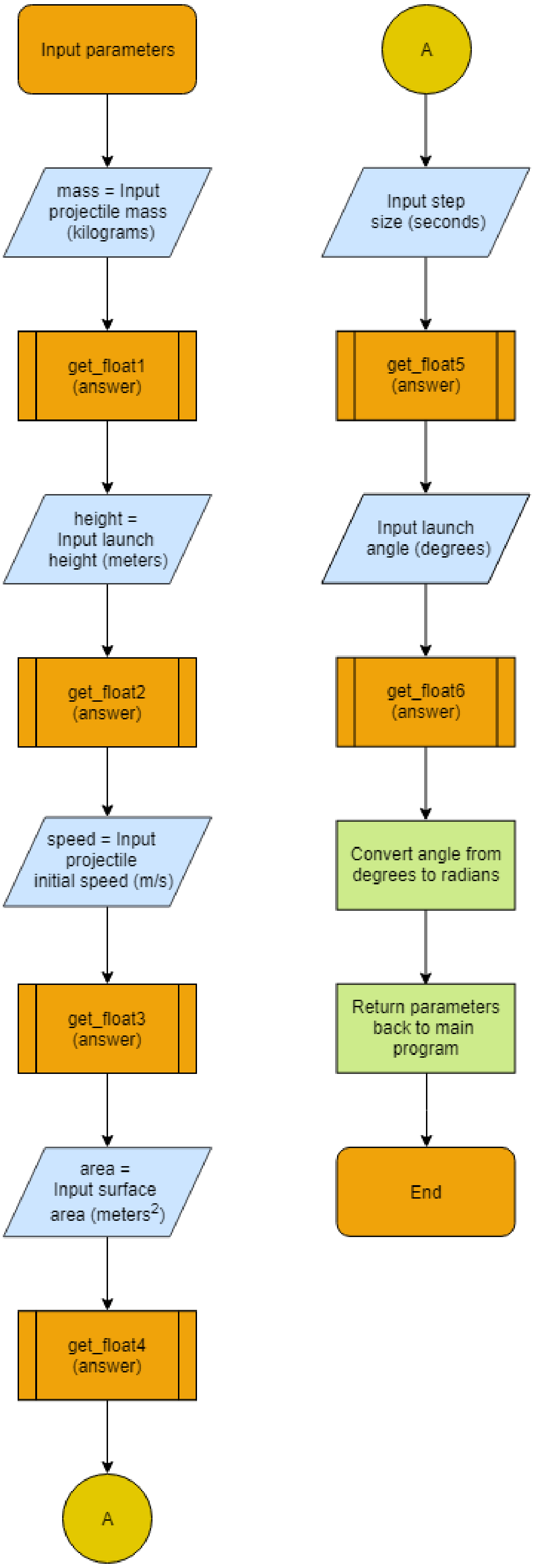

Figure 7: Flowchart: function to input parameters projectile motion simulation data

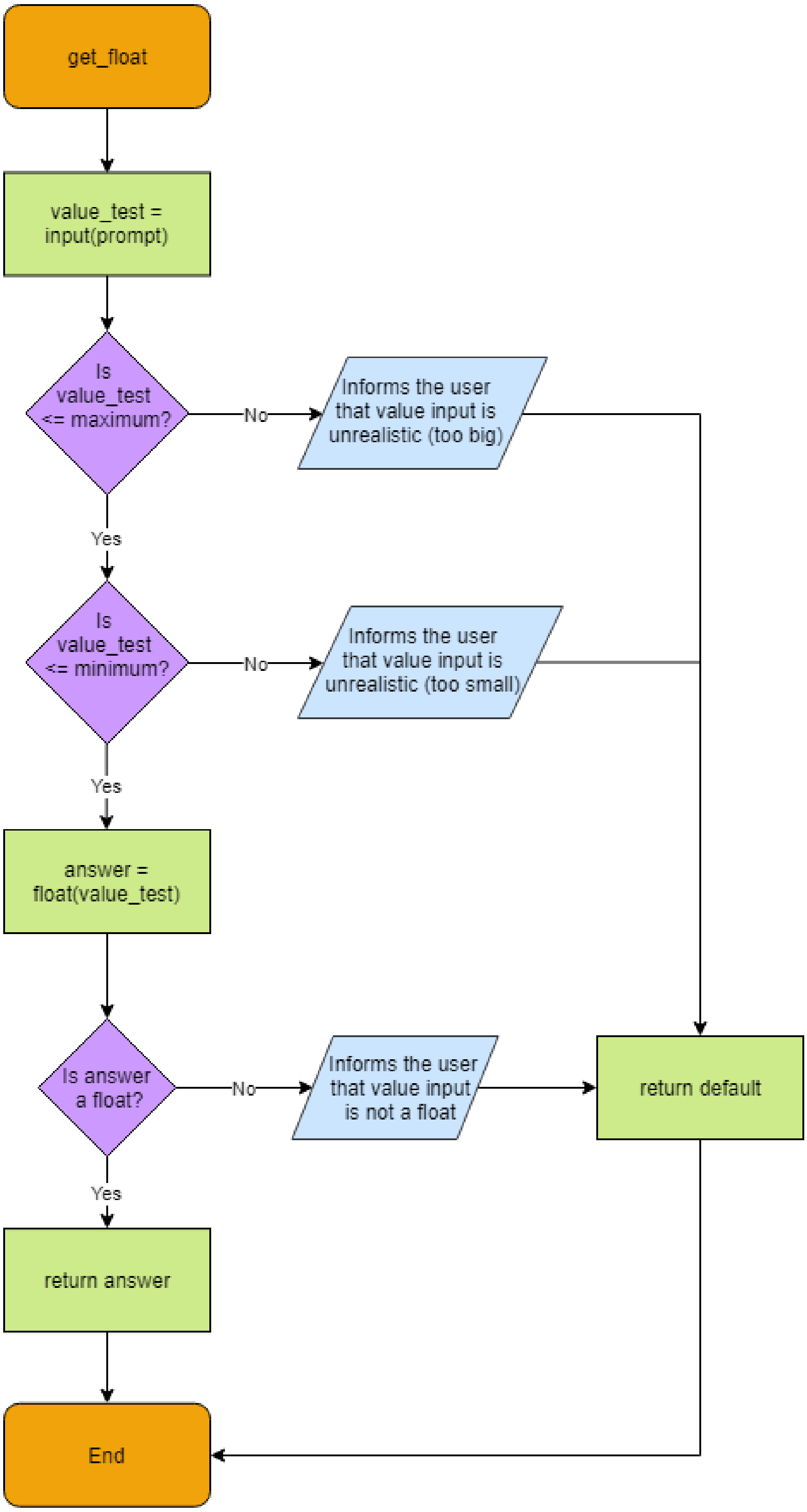

Figure 8: Flowchart: function to ensure user input realistic parameters

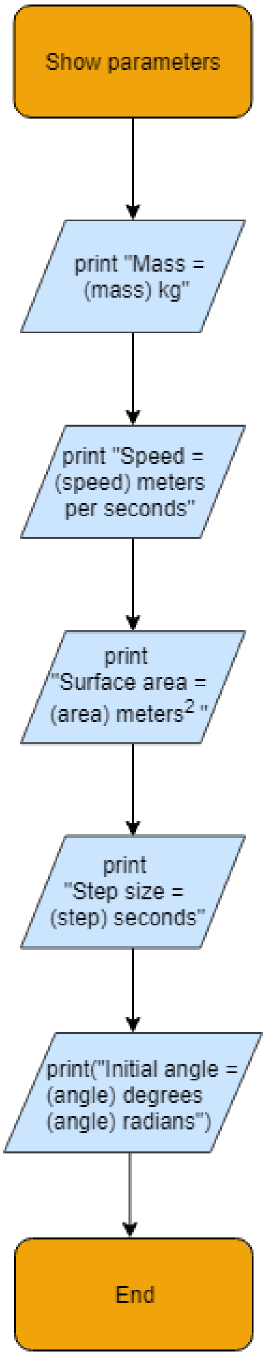

Figure 9: Flowchart: function to show current parameters

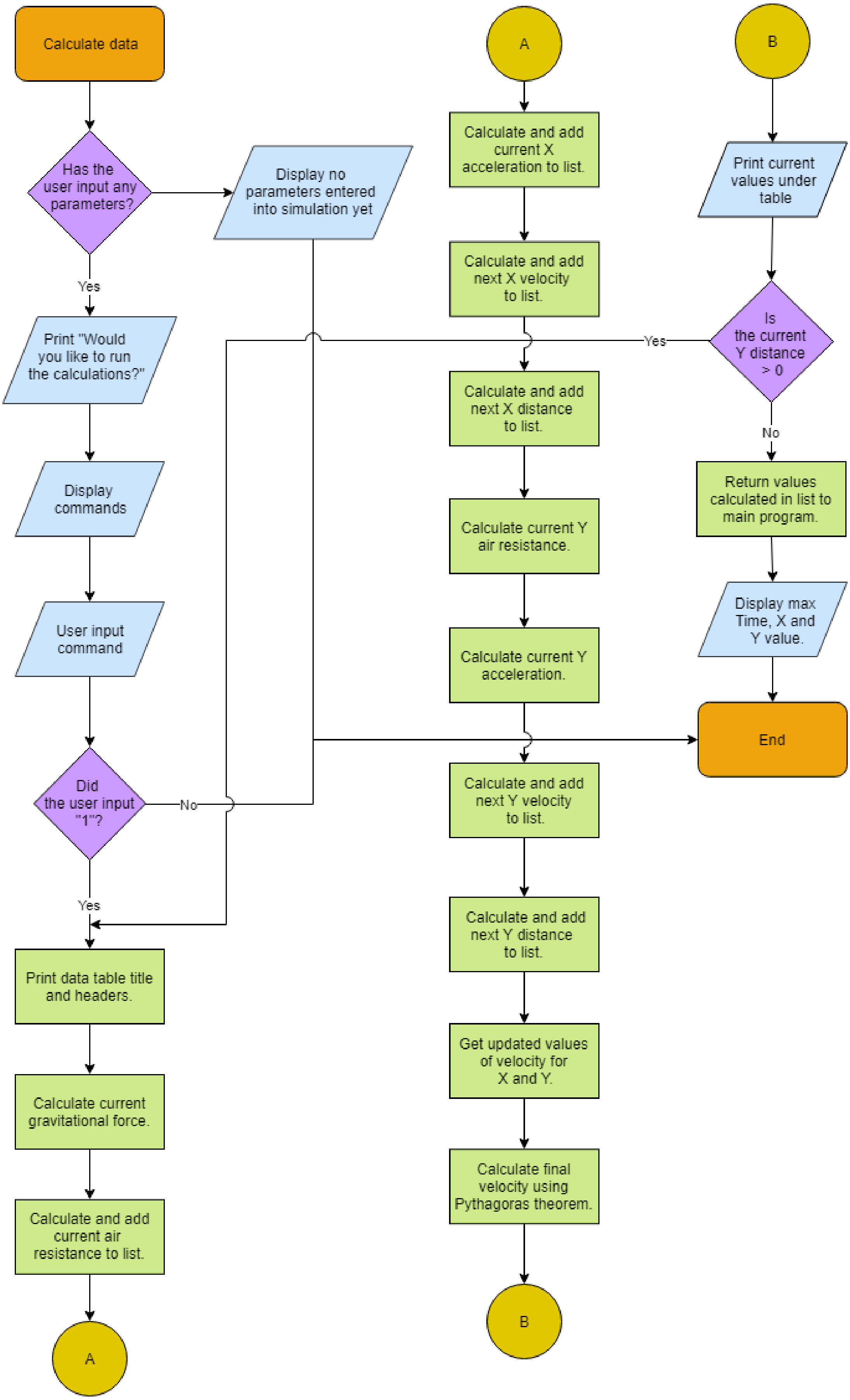

Figure 10: Flowchart: function to calculate projectile motion simulation data

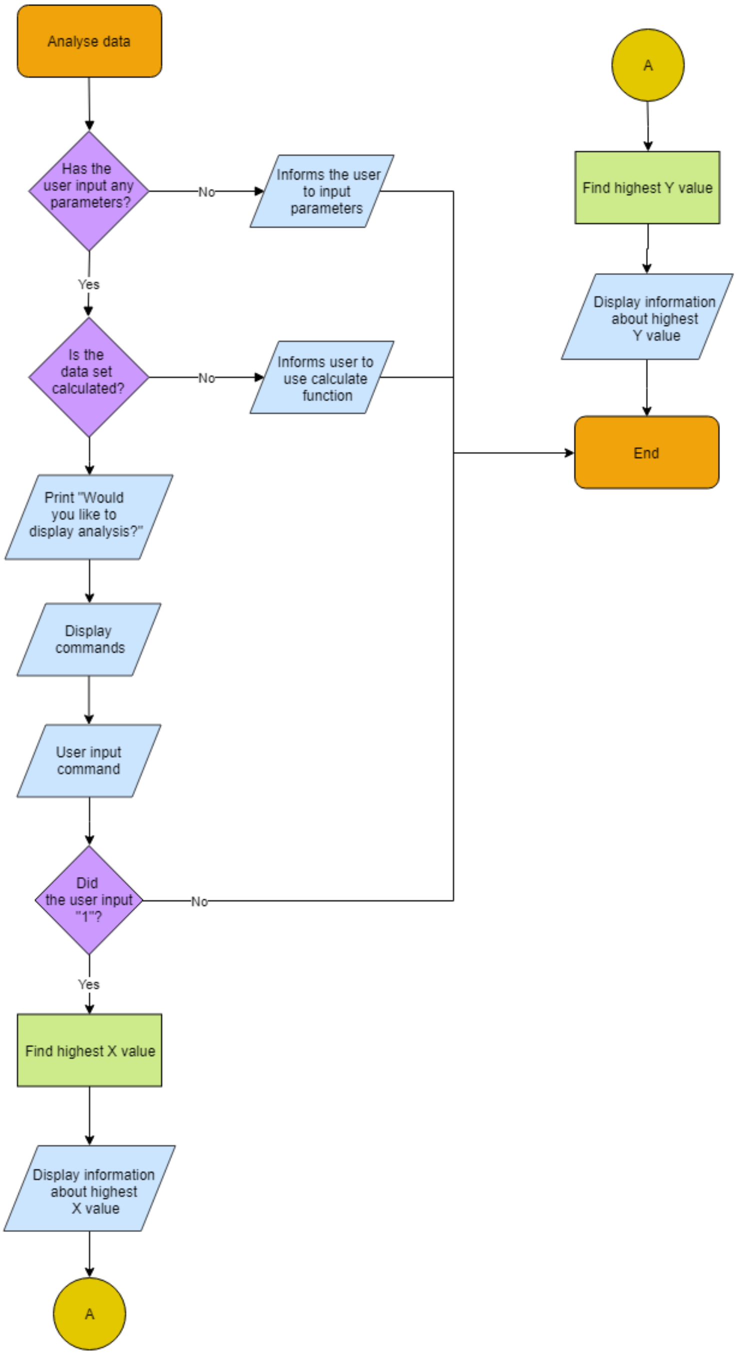

Figure 11: Flowchart: function to display analysis for the data calculated

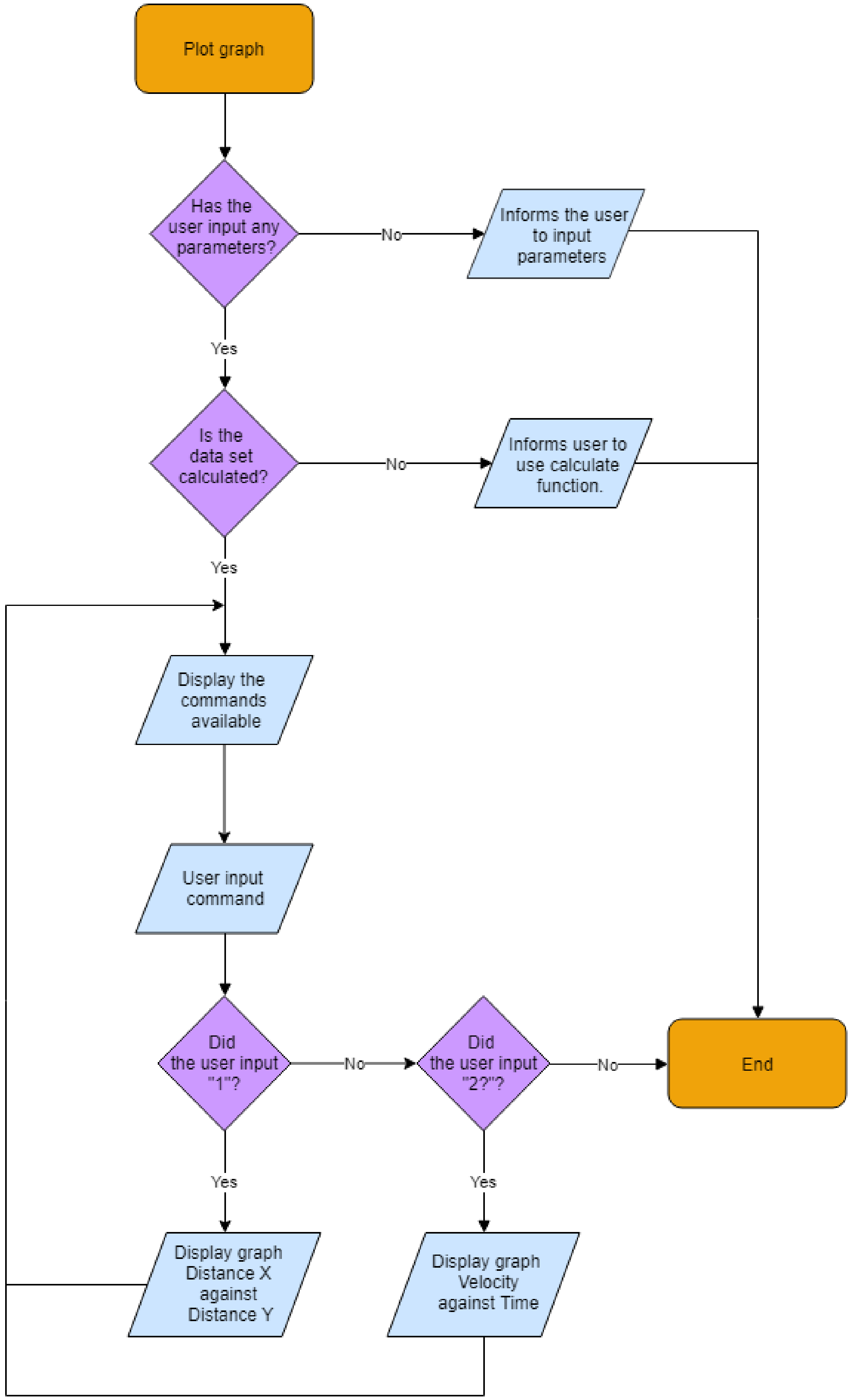

Figure 12: Flowchart: function to display graphs

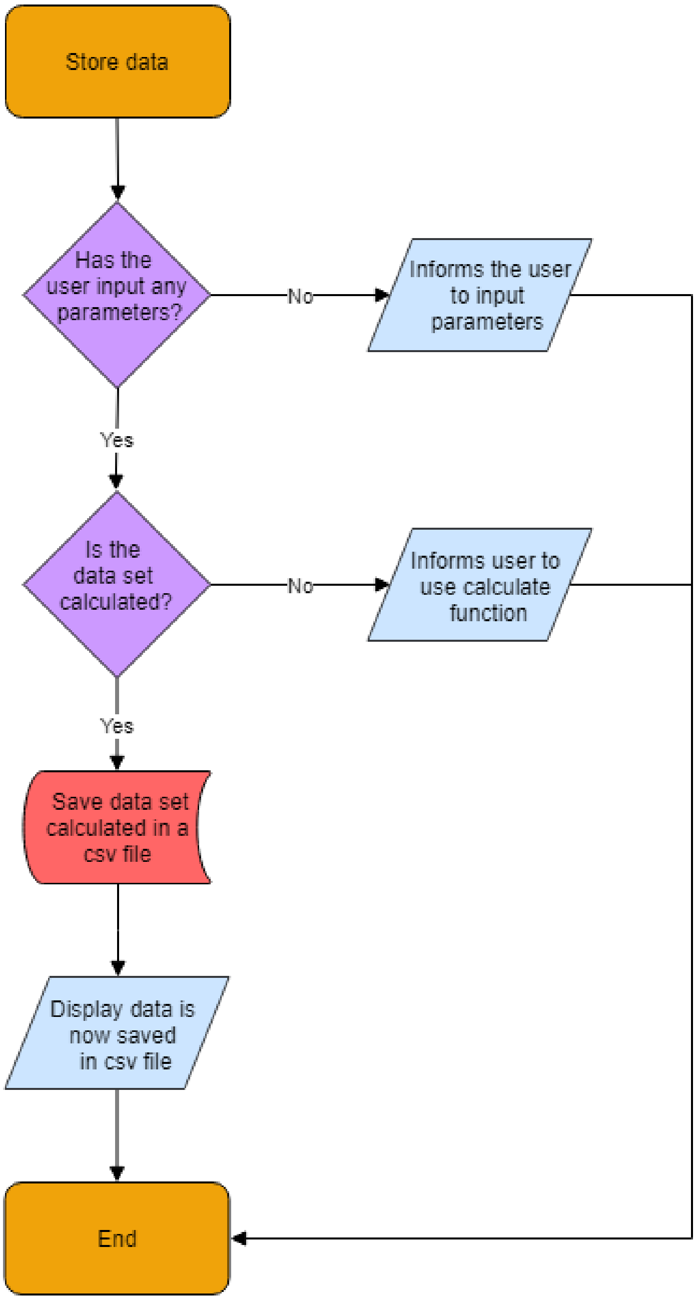

Figure 13: Flowchart: function to display graphs

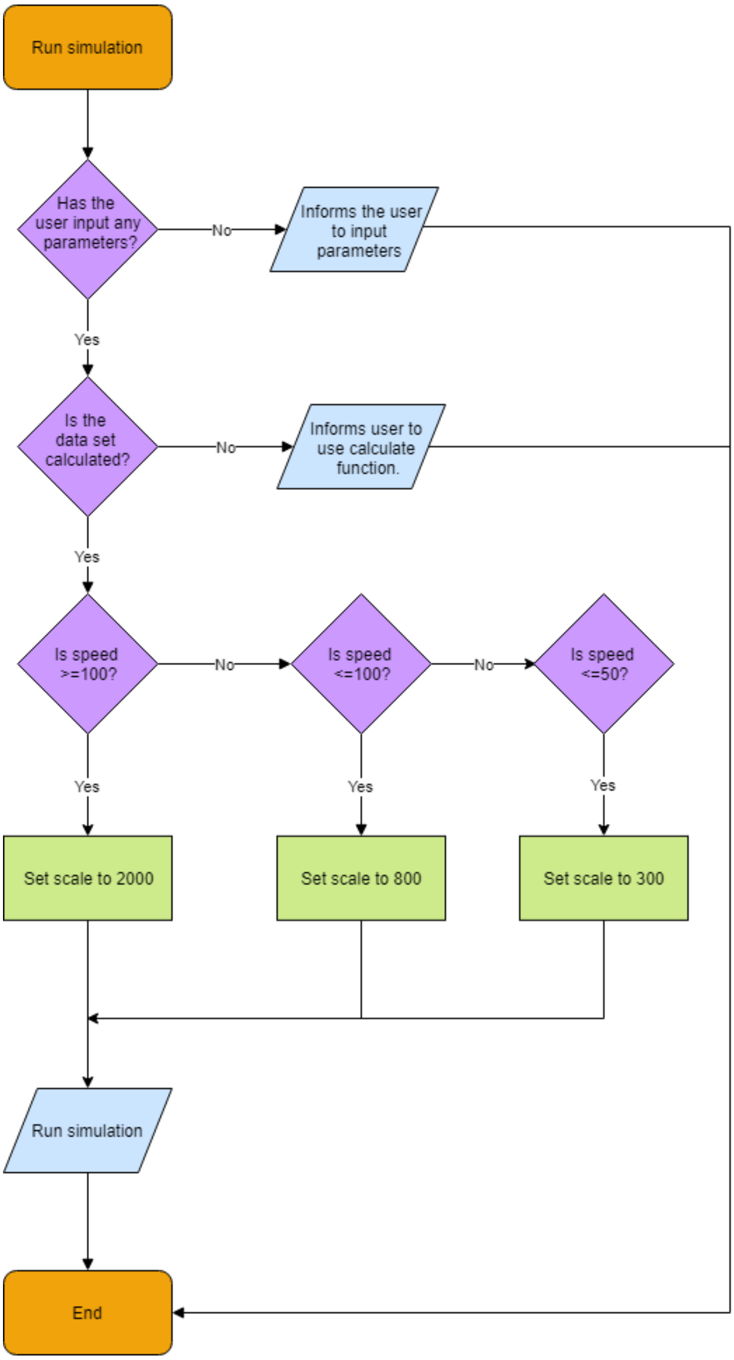

Figure 14: Flowchart: function to run simulation

Test Plan 1

No.	Test Description	Test Data	Expected outcome	Actual outcome	Improvements
1	Test if parameters within range are recorded properly.	Mass – 10kg Speed – 100 m/s Step – 0.005 s Height – 5 meters Area – 0.02 m^2 Angle – 30 degrees	All values should be recorded and displayed in show current parameters console display.	The actual outcome is as expected.	N/A
2	Test if parameters below range are rejected and default value is used.	Mass – 0kg Speed – 5 m/s Step – 0.0001 s Height – -10 meters Area – 0.00001 m^2 Angle – 0 degrees	All values should be rejected and the default values are displayed in show current parameters console display.	The actual outcome is as expected.	More details (default value) can be stated when user input is rejected.
3	Test if parameters above range are rejected and default value is used.	Mass – 50 kg Speed – 300 m/s Step – 10 s Height – 50 meters Area – 5 m^2 Angle – 180 degrees	All values should be rejected and the default values are displayed in show current parameters console display.	The actual outcome is as expected.	More details (default value) can be stated when user input is rejected.
4	Test if letters are input for parameters.	Mass – heavy Speed – fast Step – daily Height – high Area – average Angle – fish	All values should be rejected and the default values are displayed in show current parameters console display.	The actual outcome is as expected.	More details (default value) can be stated when user input is rejected.
5	Test if symbols are input for parameters.	Mass – ! Speed – # Step – % Height – @ Area – $ Angle – ^	All values should be rejected and the default values are displayed in show current parameters console display.	The actual outcome is as expected.	More details (default value) can be stated when user input is rejected.
6	Test if no values are input for parameters.	Mass – Speed – Step – Height – Area – Angle –	All values should be rejected and the default values are displayed in show current parameters console display.	The actual outcome is as expected.	More details (default value) can be stated when user input is rejected.

Figure 15: Test-1 Completed

Test Plan 2

No.	Test Description	Test Data	Expected outcome	Actual outcome	Improvements
1	Try command 2 (show parameters) with default parameter.	Command 2	Console output will proceed to display parameters when command 2 is selected.	The actual outcome is as expected.	N/A
2	Try command 3 (calculate data) with default parameter.	Command 3	Console output will proceed to calculate data when command 3 is selected.	The actual outcome is as expected.	N/A
3	Try command 4 (analyse data) with default parameter and data set calculated.	Command 4	Console output will proceed to display analysis of the data calculated when command 4 is selected then	The actual outcome is as expected.	N/A
4	Try command 5 (plot graph) to display graph 1 with default parameter and data set calculated.	Command 5, Command 1	Console output will proceed to display another menu for a selection of graphs and graph 1 will be displayed when selected.	The actual outcome is as expected.	N/A
5	Try command 5 (plot graph) to display graph 2 with default parameter and data set calculated.	Command 5, Command 2	Console output will proceed to display another menu for a selection of graphs and graph 2 will be displayed when selected.	The actual outcome is as expected.	N/A
6	Try command 6 (save data) with default parameter and data set calculated.	Command 6	The data set will be save in a csv file named “Projectile_Motion”.	The actual outcome is as expected.	Allow the csv to open when the data has been saved successfully.
7	Try command 7 (run simulation) with default parameter and data set calculated.	Command 7	Simulation will run when the user press any key and stop either when the user presses any key again or when the cannonball reaches the ground again.	The actual outcome is as expected.	Better visualisation of the cannon in graphics module.

Figure 16: Test-2 Completed

Test Plan 3

1	Try command 2 (show parameters) without input parameters	Command 2	Console output will inform user that no parameters have been input.	The actual outcome is as expected.	Allow use to skip to input parameters after displaying no parameters have been input
2	Try command 3 (calculate data) without input parameters	Command 3	Console output will inform user that no parameters have been input.	The actual outcome is as expected.	Allow use to skip to input parameters after displaying no parameters have been input
3	Try command 4 (analyse data) without input parameters	Command 4	Console output will inform user that no parameters have been input.	The actual outcome is as expected.	Allow use to skip to input parameters after displaying no parameters have been input
4	Try command 5 (plot graph) without input parameters	Command 5	Console output will inform user that no parameters have been input.	The actual outcome is as expected.	Allow use to skip to input parameters after displaying no parameters have been input
5	Try command 6 (save data) without input parameters	Command 6	Console output will inform user that no parameters have been input.	The actual outcome is as expected.	Allow use to skip to input parameters after displaying no parameters have been input
6	Try command 7 (run simulation) without parameters	Command 7	Console output will inform user that no parameters have been input.	The actual outcome is as expected.	Allow use to skip to input parameters after displaying no parameters have been input
7	Try command 4 (analyse data) with parameters but without calculating data set	Command 4	Console output will inform user that data has not been calculated.	The actual outcome is as expected.	Allow use to skip to calculate data after data has not been calculated.
8	Try command 5 (plot graph) with parameters but without calculating data set	Command 5	Console output will inform user that data has not been calculated.	The actual outcome is as expected.	Allow use to skip to calculate data after data has not been calculated.
9	Try command 6 (save data) with parameters but without calculating data set	Command 6	Console output will inform user that data has not been calculated.	The actual outcome is as expected.	Allow use to skip to calculate data after data has not been calculated.
10	Try command 7 (run simulation) with parameters but without calculating data set	Command 7	Console output will inform user that data has not been calculated.	The actual outcome is as expected.	Allow use to skip to calculate data after data has not been calculated.
11	Try command quit (quit program)	Command quit	Console output will inform user that program is quitting	The actual outcome is as expected.	N/A

Figure 17: Test-3 Completed

Equations
1. Initial X Velocity
1. Initial Y Velocity
2. Gravitational force
3. X Air resistance
4. X Acceleration
5 X Velocity
6. X Distance
7. Y Air resistance
8. Y Acceleration
9. Y Velocity
10. Y Distance
10. Velocity from X and Y

		Velocity(m/s)				X Current			
Step	Time(Seconds)	Total	X	Y	Gravitational force	Air resistance (m/s**2)	Acceleration (m/s**2)	Velocity (m/s)	Displacement
1	0	100	70.69640468	70.72494868	53.911	22.15742703	4.028623096	70.65611844	0.706561184
2	0.01	99.87370311	70.65611844	70.58660991	53.911	22.13218148	4.024032996	70.61587811	1.412719966
3	0.02	99.74759822	70.61587811	70.44842872	53.911	22.10697905	4.019450737	70.57568361	2.118476802
4	0.03	99.62168495	70.57568361	70.31040461	53.911	22.08181966	4.014876302	70.53553484	2.82383215
5	0.04	99.49596293	70.53553484	70.1725371	53.911	22.05670321	4.010309674	70.49543175	3.528786468
6	0.05	99.37043176	70.49543175	70.03482571	53.911	22.03162958	4.005750833	70.45537424	4.23334021
7	0.06	99.24509109	70.45537424	69.89726995	53.911	22.0065987	4.001199764	70.41536224	4.937493832
8	0.07	99.11994053	70.41536224	69.75986934	53.911	21.98161046	3.996656447	70.37539568	5.641247789
9	0.08	98.9949797	70.37539568	69.6226234	53.911	21.95666476	3.992120866	70.33547447	6.344602534
10	0.09	98.87020825	70.33547447	69.48553166	53.911	21.93176151	3.987593003	70.29559854	7.047558519

Y Current			
Air resistance (m/s**2)	Acceleration (m/s**2)	Velocity (m/s)	Displacement
22.17532297	13.8338769	70.58660991	0.705866099
22.08865747	13.81811954	70.44842872	1.410350386
22.00226007	13.80241092	70.31040461	2.113454432
21.91612991	13.78675089	70.1725371	2.815179803
21.83026609	13.77113929	70.03482571	3.51552806
21.74466775	13.75557595	69.89726995	4.21450076
21.65933401	13.74006073	69.75986934	4.912099453
21.574264	13.72459345	69.6226234	5.608325687
21.48945686	13.70917398	69.48553166	6.303181004
21.40491174	13.69380214	69.34859364	6.99666694

Figure 23: Dry run done in excel manually

Appendix

A Code: Projectile Motion Simulator

Listing 1: Python Code

```
# Jia Xiu Sai
# 2427165
# 07/04/2019
# Project - Projectile motion simulation
#-----------------------------------------------------------------------------

# Import modules.

import os
import datetime
import math
import matplotlib.pyplot as plt
import csv
import time
from graphics import *
#-----------------------------------------------------------------------------

# Functions

# Checks if value input by user is resonable/float.
def get_float(prompt,default,minimum,maximum):

  value_test = input(prompt)

  # Test if the value input is a float.
  try:

    # Test if the value input is reasonable.
    if(float(value_test)<=maximum):

      if(float(value_test)>=minimum):

        # Test if the value input is a float.
        answer = float(value_test)

        print(" Value recorded.")
        # If the value input is a float, the value is returned back to the main program.
        return answer

      # If the value input is unrealistic, the default value that was set beforehand is returned to the main program.
      else:
        print(" Oops! The value of the number is too small. Using default value.")
        return default

    # If the value input is unrealistic, the default value that was set beforehand is returned to the main program.
    else:
      print(" Oops! The value of the number is too big. Using default value.")
      return default

  # If the value input is not a float, a ValueError will appear and the default value that was set beforehand is returned to the main program.
  except ValueError:
    print(" Oops! That was not a valid number. Using default value.")
    return default

# End of function get_float
#-----------------------------------------------------------------------------

# Lets the user change the current parameters.
def change_para():

  os.system('cls')
  print (program_info)
```

```
    print(" Function: Change parameters")
    print (line*30)

    # The parameters are input by user:
    #
    # Mass of projectile object (kilograms).
    # Height of initial projection (meters).
    # Initial speed of projectile (meter per second).
    # Frontal area of object (meters square).
    # The step size (seconds).
    # Initial angle (degrees).
    mass = get_float(" Enter the mass of the cannonball in kg (1-20): ",5.5,1,20)
    height = get_float(" Enter the initial height (0-10): ",0,0,10)
    speed = get_float(" Enter the initial speed of the cannonball in m/s (10-200): "
      ,100,10,200)
    area = get_float(" Enter the frontal area of cannonball in meters\u00b2 (0.005-1): "
      ,0.0154,0.005,1)
    step = get_float(" Enter the step size in seconds (0.001-1): ",0.01,0.001,1)
    ang_in_deg = get_float(" Enter the initial angle in degrees (10-90): ",45,10,90)
    # The angle is converted from degrees into radians.
    ang_in_rad = math.radians(ang_in_deg)

    # Parameters are returned back to the main program.
    return mass,height,speed,area,step,ang_in_deg,ang_in_rad

# End of function change_para
#-------------------------------------------------------------------------

# Show current parameters input by user.
def show_current_para(mass,height,speed,area,step,ang_in_deg,ang_in_rad):

    os.system('cls')
    print (program_info)
    print(" Function: Show current parameters")
    print (line*30)
    print(" Current parameters of projectile object:")

    # Prints the parameters input by user.
    print(" Mass = %.2f kg" %(mass))
    print(" Height = %.1f meters" %(height))
    print(" Speed = %.2f meters per seconds" %(speed))
    print(" Surface area = %.4f meters\u00b2" %(area))
    print(" Step size = %s seconds" %(step))
    print(" Initial angle = %.2f degrees (%0.2f radians)" %(ang_in_deg,ang_in_rad))

# End of function show_current_para
#-------------------------------------------------------------------------

# Calculates the data set using the parameters input by user.
def calculate_data(mass,height,speed,area,step,ang_in_deg,ang_in_rad):

    # Ask if user wants to calculate data.
    print("\n Would you like to run calculations?")
    print("\n 1: Yes\n Press any key to go back to main menu.")
    user_command = input("\n Enter command: ")

    # When '1' is input, continue to run function calculate_data.
    if (user_command =='1'):

        i = int(0)          # Iteration (i.e. step)
        t_current = float(0)  # Current time value (seconds)
        t_next = float(0)    # Next time value (seconds)
        fd_current = float(0) # Current air resistance (newtons)
        fd_next = float(0)    # Next air resistance (newtons)

        # Values for X-coordinates:
        x_current = float(0)  # Current distance (meters)
        x_next = float(0)    # Next distance (meters)
        xv_current = float(0) # Current velocity (meters/seconds)
        xv_next = float(0)    # Next velocity (meters/seconds)

        # Values for Y-coordinates:
        y_current = float(0)  # Current distance (meters)
```

```
    y_next = float(0)   # Next distance (meters)
    yv_current = float(0) # Current velocity (meters/seconds)
    yv_next = float(0)    # Next velocity (meters/seconds)

    v_current = float(0)  # Current velocity (meters/seconds)
    v_next = float(0)   # Next velocity (meters/seconds)

    # Fixed parameters:
    g = 9.807   # Gravity on Earth (meters/second square)
    drag = 0.47   # Drag coefficient of a ciruclar smooth surface
    a_d = 1.225   # Air density at sea-level (kg/m**3)

    # Lists:
    t = []    # Holds time data
    x = []    # Holds distance data for X axis
    y = []    # Holds distance data for Y axis
    xv = []   # Holds velocity data for X axis
    yv = []   # Holds velocity data for Y axis
    v = []    # Holds total velocity data

    t_sim_run = float(0)    # Current simulation run time (seconds)

    # Set the initial values and add to data lists.
    t.append(0.0)
    x.append(0.0)
    y.append(height)
    xv.append(speed*math.cos(ang_in_rad))
    yv.append(speed*math.sin(ang_in_rad))
    v.append(speed)

    os.system('cls')
    print (program_info)
    print(" Function: Calculate and show current data set")
    print (line*30)

    # Title and header of the table.
    print("\n Data table:\n")
    print(" Step|  Time  |  X_Velocity  |  Y_Velocity  |  Velocity  |  X_Distance  |
    Y_Distance")

    # Loop to create data points.
    i = 0;

    # Makes this function run until object lands on the ground.
    while (y_current >= 0):

      # Get current values
      t_current = t[i]
      x_current = x[i]
      y_current = y[i]
      xv_current = xv[i]
      yv_current = yv[i]
      v_current = v[i]

      # Calculate current gravitational force.
      fg_current = (mass*g)

      # Calculate and add current X air resistance to list.
      xfd_current = 0.5*xv_current*abs(xv_current)*drag*area*a_d

      # Calculate and add current X acceleration to list.
      xa_current = (xfd_current)/mass

      # Calculate and add next X velocity to list.
      xv.append(xv_current - (xa_current*step))

      # Calculate and add next X distance to list.
      x.append(x_current + (xv_current*step))

      # Calculate current Y air resistance.
      yfd_current = 0.5*yv_current*abs(yv_current)*drag*area*a_d

      # Calculate current Y acceleration.
```

```
      ya_current = (fg_current + yfd_current)/mass

      # Calculate and add next Y velocity to list.
      yv.append(yv_current - (ya_current*step))

      # Calculate and add next Y distance to list.
      y.append(y_current + (yv_current*step))

      # Get updated values of X velocity and Y velocity.
      xv_current = xv[i+1]
      yv_current = yv[i+1]

      # Calculate final velocity using pythagoras theorem.
      v.append(math.sqrt((xv_current*xv_current)+(yv_current*yv_current)))

      # Calculate and next time to list.
      t.append(t_current + step)

      # Update the simulation run time.
      t_sim_run = t_sim_run + step

      # Data table
      print("  %d  |  %.2f  |    %.3f    |   %.3f     |   %.3f  |    %.3f     |    %.3f" %(i,t_current,xv_current,yv_current,v_current,x_current,y_current))

      # Update the iteration / step.
      i = i + 1

    print (line*30)

    # Values in data lists are returned back to main program.
    return i,t,xv,yv,x,y,v

# End of function calculate_data
#----------------------------------------------------------------------------

# Analyse the data calculated.
def analyse_data(mass,i,t,xv,yv,x,y,v):

  # Checks if any parameters is input by user.
  if (0 < (mass)):

    # Checks if data set is calculated.
    num_x=len(x)
    if (0 < num_x):

      # Ask if the user wants to analyses data.
      print("\n Would you like to display analysis?")
      print("\n 1: Yes\n Press any key to go back to main menu.")
      user_command = input("\n Enter command: ")

      # When '1' is input, continue to run function calculate_data.
      if (user_command =='1'):

        os.system('cls')
        print (program_info)
        print(" Function: Analyse data")
        print (line*30)

        print(" Data analysis:\n")

        # Highest height achieved.
        mx = max(x)
        max_x = x.index(mx)
        print(" Furthest distance achieved at:\n X-coordinate - %0.3f meters\n Y-coordinate - %0.3f meters\n Speed - %0.3f meter per seconds \n Time - %0.2f seconds \n" %(mx,y[max_x],v[max_x],t[max_x]))

        # Furthest distance achieved.
        my = max(y)
        max_y = y.index(my)
        print(" Highest distance achieved at:\n X-coordinate - %0.3f meters\n Y-coordinate - %0.3f meters\n Speed - %0.3f meter per seconds \n Time - %0.2f seconds " %(x[max_y],my,
```

```
    v[max_y],t[max_y]))

        input ("\n Press any key to go back to main menu.")

      return i,t,xv,yv,x,y,v

    else:
      os.system('cls')
      print (program_info)
      print(" Function: Analyse data")
      print (line*30)
      print(" No data calculated yet.")
      input ("\nPress any key to go back to main menu.")

  # Informs the user to input parameters.
  else:
    os.system('cls')
    print (program_info)
    print(" Function: Analyse data")
    print (line*30)
    print(" No parameters enter yet.")
    input ("\n Press any key to go back to main menu.")

# End of function analysis_data
#--------------------------------------------------------------------------

# Plots a graph using data set calculated.
def plot_graph(mass,i,t,xv,yv,x,y,v):

  # Checks if any parameters is input by user.
  if (0 < (mass)):

    # Checks if data set is calculated.
    num_x=len(x)
    if (0 < num_x):

      os.system('cls')
      print (program_info)
      print(" Function: Plot graph")
      print (line*30)

      # Show commands for different graphs.
      print("\n Commands| Graph:")
      print("    1    | Displacement Y against Displacement X")
      print("    2    | Velocity against Time")
      print("\n Input any key to go back to main menu.")

      user_command = input("\n Enter command: ")
      os.system('cls')
      print (program_info)
      print(" Function: Plot graph")
      print (line*30)
      print (" Plotting graph......")

      # When '1' is input, display graph: Displacement Y against Displacement X.
      if (user_command =='1'):

        my = max(y)
        max_y = y.index(my)
        maxpoint = [max_y]

        # Plot graph with (x-axis,y-axis and line type).
        plt.plot(x,y,'b-x', ms=6, markevery=maxpoint)

        # Annotate the highest point on graph.
        my = max(y)
        max_y = y.index(my)
        plt.annotate("(%0.1f , %0.1f)" %(x[max_y],my) ,xy = (0,0), xytext = (x[max_y]*0.8,my
    *0.8))

        # Scale graph.
        plt.axis('scaled')

```

```
        # X-axis label.
        plt.xlabel('Distance X (m)')

        # Y-axis label.
        plt.ylabel('Distance Y (m)')

        # Graph title.
        plt.title('Projectile Motion Simulation\nDisplacement Y against Displacement X')
        plt.ion()

        # Show graph created.
        plt.show()
        input ("\n Press any key to go back.")
        plt.close()

        # Return to function plot_graph.
        plot_graph(mass,i,t,xv,yv,x,y,v)

      # When '2' is input, display graph: Velocity against Time.
      if (user_command =='2'):

        # Plot graph with (x-axis,y-axis and line type).
        plt.plot(t,v,'b-')

        # X-axis label
        plt.xlabel('Time (seconds)')

        # Y-axis label
        plt.ylabel('Velocity (m/s)')

        # Graph title
        plt.title('Projectile Motion Simulation\nVelocity against time')
        plt.ion()

        # Show graph created
        plt.show()
        input ("\n Press any key to go back.")
        plt.close()

        # Return to function plot_graph.
        plot_graph(mass,i,t,xv,yv,x,y,v)

    # Informs the user to use calculate function.
    else:
      os.system('cls')
      print (program_info)
      print(" Function: Plot graph")
      print (line*30)
      print(" No data calculated yet.")
      input ("\n Press any key to go back to main menu.")

  # Informs the user to input parameters.
  else:
    os.system('cls')
    print (program_info)
    print(" Function: Analyse data")
    print (line*30)
    print(" No parameters enter yet.")
    input ("\n Press any key to go back to main menu.")

# End of function plot_graph
#---------------------------------------------------------------------------

# Save data set calculated in a csv file.
def save_data(mass,i,t,xv,yv,x,y,v):

  # Checks if any parameters is input by user.
  if (0 < (mass)):

    # Checks if data set is calculated.
    num_x=len(x)
    if (0 < num_x):

```

```
      os.system('cls')
      print (program_info)
      print(" Function: Save data")
      print (line*30)

      # Open/create csv filed named "Projectile_Motion.csv' and enable write mode.
      Projectile_Motion_csv = open("Projectile_Motion.csv", "w")

      # Write column titles
      Projectile_Motion_csv.write("Step, Time(Seconds), X_Velocity(m/s), Y_Velocity(m/s),
    Velocity(m/s), X_Distance(m), Y_Distance(m)")

      num_t=len(t)

      # Runs this function for all data calculated stored in list/arrays.
      for i in range(0,num_t):

        # Record data calculated in different coloumns.
        Projectile_Motion_csv.write("\n%d,%.2f,%.3f,%.3f,%.3f,%.3f,%.3f" %(i,t[i],xv[i],yv[i
    ],v[i],x[i],y[i]))

      # Close csv file.
      Projectile_Motion_csv.close()

      # Informs user data is saved in file.
      print(" Data is now saved in file name: 'Projectile_Motion.csv'.")
      input ("\n Press any key to go back to main menu.")

    # Informs the user to use calculate function.
    else:
      os.system('cls')
      print (program_info)
      print(" Function: Save data")
      print (line*30)
      print(" No data calculated yet.")
      input ("\n Press any key to go back to main menu.")

  # Informs the user to input parameters.
  else:
    os.system('cls')
    print (program_info)
    print(" Function: Save data")
    print (line*30)
    print(" No parameters enter yet.")
    input ("\n Press any key to go back to main menu.")

# End of function save data
#---------------------------------------------------------------------------

# Runs simulation with parameters input by user and the data calculated.
def run_simu(mass,height,speed,area,step,ang_in_deg,ang_in_rad,i,t,xv,yv,x,y,v):

  # Checks if any parameters is input by user.
  if (0 < (mass)):

    # Checks if data set is calculated.
    num_x=len(t)
    if (0 < num_x):

      # Determine whether values are possible to display.
      if (0 >= (height)):

        try:
          os.system('cls')
          print (program_info)
          print(" Function: Run simulation")
          print (line*30)
          print (" Running simulation......")

          # Run this simulation for speed above speed of 100.
          if (100 < (speed)):
            scale = 2000

```

```
        # Run this simulation for speed under speed of 100.
        if (100 >= (speed)):
          scale = 800

        # Run this simulation for speed under speed of 50.
        if (50 >= (speed)):
          scale = 300

        # Set margin size.
        width_x = 1000
        length_y = 600
        centre_x = width_x/2
        centre_y = length_y/2
        start_x = height+50
        start_y = 500

        # Set projectile size.
        radius = 20
        radius_dot = 1

        # Print blank margin.
        win = GraphWin("Projectile Motion", width_x, length_y)

        # Path of projectile object.
        path = Circle(Point(start_x,start_y), radius_dot)
        path.setFill("#FF8500")
        path.setOutline("#FF8500")
        path.draw(win)

        # Projectile object.
        ball = Circle(Point(start_x,start_y), radius)
        ball.setFill("#4D4D4D")
        ball.draw(win)

        # Ground.
        ground= Rectangle(Point(0,start_y+radius), Point(width_x,length_y))
        ground.setFill("#53AF23")
        ground.draw(win)

        # Cannon 1.
        cannnon_1= Circle(Point(start_x,start_y), radius+5)
        cannnon_1.setFill("#000000")
        cannnon_1.draw(win)

        # Cannon 2.
        cannnon_2= Polygon(Point(start_x-15,start_y+15), Point(start_x+30,start_y-45),
  Point(start_x+55,start_y-15))
        cannnon_2.setFill("#000000")
        path.setOutline("#000000")
        cannnon_2.draw(win)

        # Cannon 3.
        cannnon_3= Circle(Point(start_x,start_y+17),10)
        cannnon_3.setFill("#8C611A")
        path.setOutline("#8C611A")
        cannnon_3.draw(win)

        # Create text area.
        message = Text(Point(centre_x,centre_y-150), 'Press any key to start')
        message.draw(win)

        # Create text area 2.
        message_2 = Text(Point(centre_x,centre_y-70), '')
        message_2.draw(win)

        # Create text area 3.
        message_3 = Text(Point(centre_x,570),"Mass = %.2f kg  Height = %.1f meters Surface
   area = %.4f meters\u00b2  Angle = %.1f degrees  " %(mass,height,area,ang_in_deg,))
        message_3.draw(win)

        # Create text area 4.
        message_4 = Text(Point(centre_x,30),"Projectile motion simulation \nJacob 2019")
        message_4.draw(win)
```

```

        # Create text area 5.
        message_5 = Text(Point(centre_x+350,centre_y+50)," ")
        message_5.draw(win)

        win.getKey()

        num_t = len(t)

        for i in range(1,num_t):

          # Start timing current time.
          start_time = time.time()

          # Change text area 1 to show changing parameters of running simulation.
          message.setText("Step=%d  Time=%0.2f seconds  Velocity=%0.2f m/s\nPress any key to stop" %(i,t[i],v[i]))

          # Calculate next position for object.
          next_delta_x = x[i]*(width_x/scale)
          next_delta_y = y[i]*(length_y/scale)
          next_x = start_x + next_delta_x
          next_y = start_y - next_delta_y

          # Draw path of projectile object.
          new_path = Circle(Point(next_x,next_y), radius_dot)
          new_path.setFill("#FF8500")
          new_path.setOutline("#FF8500")
          new_path.draw(win)

          # Draw projectile object.
          new_ball = Circle(Point(next_x,next_y), radius)
          new_ball.setFill("#4D4D4D")

          # Print new object position.
          new_ball.draw(win)

          # Calculate running time for simulation.
          elapsed_time = time.time() - start_time

          h = t[i]-t[i-1]

          # Test if simulation is running same speed/ faster than real time.
          if (elapsed_time < h):

            # Slow down simulation if simulation is running too fast.
            time.sleep(h - elapsed_time)
            message_2.setText("Real-time")

          # If simulation is too slow, continue running the simulation.
          else:

            # Inform the user that simulation is not in real time.
            message_2.setText("Not real time")

          # Undraw starting/current object.
          ball.undraw()

          # Make new object the current object.
          ball = new_ball
          path = new_path

          # End simulation when user press any key.
          keypress = win.checkKey()
          if (keypress !=""):
            break

        # Show how long the simulation ran and the highest height achieved.
        mx = max(x)
        max_x = x.index(mx)
        message.setText("Press any key to exit.\nReal time = %0.2f seconds\n\nHighest distance achieved at :\n X-coordinate - %0.3f meters \n Y-coordinate - %0.3f meters" %(t[i],mx,y[max_x],))
```

```

          # Show the furthest distance achieved.
          my = max(y)
          max_y = y.index(my)
          message_5.setText("Furthest distance achieved at:\n X-coordinate - %0.3f meters\n Y-coordinate - %0.3f meters" %(x[max_y],my))
          win.getKey()
          win.close()
          return()

        except:
          print()

      # Informs that values are not able to simulate due to height.
      else:
        os.system('cls')
        print (program_info)
        print(" Function: Run simulation")
        print (line*30)
        print(" Unable to run simulation")
        input ("\n Press any key to go back to main menu.")

    # Informs the user to use calculate function.
    else:
      os.system('cls')
      print (program_info)
      print(" Function: Run simulation")
      print (line*30)
      print(" No data calculated yet.")
      input ("\n Press any key to go back to main menu.")

  # Informs the user to input parameters.
  else:
    os.system('cls')
    print (program_info)
    print(" Function: Run simulation")
    print (line*30)
    print(" No parameters enter yet.")
    input ("\n Press any key to go back to main menu.")

# End of function run simu.
#------------------------------------------------------------------------

# MAIN PROGRAM
now = datetime.datetime.now()
program_info = (" DUISC\n Projectile Motion Simulator\n Jia Xiu Sai-2427165\n %s:%2.f %s/%s/%s" %(now.hour, now.minute, now.day, now.month, now.year))
menu_options = {}
user_command = "nothing"

# Line used in program.
line ="-"

# Projectile parameters.
mass = 0.0
height = 0.0
speed = 0.0
area = 0.0

# Sim parameters.
step = 0.0
ang_in_rad = 0.0
ang_in_deg = 0.0

# Other variables.
t_0 = 0.0
i = 0.0

# Data list.
t = []
x = []
y = []
xv = []
```

```
yv = []
v = []

# Main menu function.
while (user_command != 'quit'):

  os.system('cls')
  print (program_info)
  print(" Main menu")
  print (line*30)

  # Print main menu and commands available.
  print (" Commands| Functions")
  print (line*30)
  print("    1    | Change parameters")
  print("    2    | Show current parameters")
  print("    3    | Calculate and show current data set")
  print("    4    | Analyse data")
  print("    5    | Plot graph")
  print("    6    | Save data")
  print("    7    | Run visuals")
  print("   quit  | Quit program")

  user_command = input("\nEnter command: ")

  # When '1' is input, run function calculate_data.
  if (user_command =='1'):
    mass,height,speed,area,step,ang_in_deg,ang_in_rad = change_para()

    # Checks if any parameters is input by user.
    if (0 < (mass)):

      try:
        show_current_para(mass,height,speed,area,step,ang_in_deg,ang_in_rad)
        i,t,xv,yv,x,y,v = calculate_data(mass,height,speed,area,step,ang_in_deg,ang_in_rad)
        print(" Number of data points for:")
        print(" Time - %d" %(len(t)-1))
        print(" Velocity - %d" %(len(v)-1))
        print(" X - %d" %(len(x)-1))
        print(" Y - %d" %(len(y)-1))
        print (line*30)
        analyse_data(mass,i,t,xv,yv,x,y,v)

      except:
        print()

    # If no parameters are input by user yet, no parameters can be displayed.
    else:
      os.system('cls')
      print (program_info)
      print(" Function: Change parameters")
      print (line*30)

      # Informs the user to input parameters.
      print("No parameters enter yet.")

      # Stop the function to let the user choose when to exit to main menu.
      input ("\nPress any key to go back to main menu.")

  # When '2' is input, run function show current parameters.
  if (user_command =='2'):
    show_current_para(mass,height,speed,area,step,ang_in_deg,ang_in_rad)

    # Checks if any parameters is input by user.
    if (0 < (mass)):

      try:
        i,t,xv,yv,x,y,v = calculate_data(mass,height,speed,area,step,ang_in_deg,ang_in_rad)
        print(" Number of data points for:")
        print(" Time - %d" %(len(t)-2))
        print(" Velocity - %d" %(len(v)-2))
        print(" X - %d" %(len(x)-2))
        print(" Y - %d" %(len(y)-2))
```

```
        print (line*30)
        analyse_data(mass,i,t,xv,yv,x,y,v)

      except:
        print()

    # If no parameters are input by user yet, no parameters can be displayed.
    else:
      os.system('cls')
      print (program_info)
      print(" Function: Show current parameters")
      print (line*30)

      # Informs the user to input parameters.
      print(" No parameters enter yet.")

      # Stop the function to let the user choose when to exit to main menu.
      input ("\n Press any key to go back to main menu.")

  # When '3' is input, run function calculate data.
  if (user_command =='3'):

    # Checks if any parameters is input by user.
    if (0 < (mass)):

      try:
        i,t,xv,yv,x,y,v = calculate_data(mass,height,speed,area,step,ang_in_deg,ang_in_rad)
        print(" Number of data points for:")
        print(" Time - %d" %(len(t)-2))
        print(" Velocity - %d" %(len(v)-2))
        print(" X - %d" %(len(x)-2))
        print(" Y - %d" %(len(y)-2))
        print (line*30)
        analyse_data(mass,i,t,xv,yv,x,y,v)

      except:
        print()

    # If no parameters are input by user yet, no parameters can be displayed.
    else:
      os.system('cls')
      print (program_info)
      print(" Function: Calculate and show current data set")
      print (line*30)

      # Informs the user to input parameters.
      print(" No parameters enter yet.")
      input ("\n Press any key to go back to main menu")

  # When '4' is input, run function plot graph.
  if (user_command =='4'):
    analyse_data(mass,i,t,xv,yv,x,y,v)

  # When '5' is input, run function plot graph.
  if (user_command =='5'):
    plot_graph(mass,i,t,xv,yv,x,y,v)

  # When '6' is input, run function save data.
  if (user_command =='6'):
    save_data(mass,i,t,xv,yv,x,y,v)

  # When '7' is input, run function run simulation.
  if (user_command =='7'):
    run_simu(mass,height,speed,area,step,ang_in_deg,ang_in_rad,i,t,xv,yv,x,y,v)

  # When 'quit' is input, exit main menu function.
  if (user_command =='quit'):
    print ()

# End of main menu function.
print(" Quitting...") ;
print(" Program has ended") ;
print(" Bye") ;
```

```
# End of main program.
#-----------------------------------------------------------------------
```