INDIVIDUAL ASSIGNMENT

TECHNOLOGY PARK MALAYSIA

EE038-2-2ESA

ENGINEERING SOFTWARE AND APPLICATIONS

MATLAB

Name : **Su Xin Hong**

TP No. : **TP061159**

Intake Code : **APD1F2106CE**

Hand Out Date : **6th August 2022**

Hand In Date : **6th September 2022**

Lecturer's Name : **Ir. Eur. Ing. Ts. Dr. Lau Chee Yong**

Table of Contents

List of Figures

List of Tables

Introduction

This assignment is the report about creating a Graphical-User-Interface (GUI) using MatLab to simulate projectile motion. The program created provides the choice to the user to simulate projectile motion also on other environment or location which includes Earth, Moon, Mars and Sun. In addition, the program created also has a sub program which is a projectile motion mini game to entertain the user. The game is about throwing a ball to hit a random target created by the program. The gravitational acceleration utilized in the game is the Earth's gravitational acceleration.

Projectile is where an object is thrown into the space and the only force acting on it is the gravity (Byju's, n.d.). While other forces such as air, friction and so on may have effect on the projectile in real life. However, in this assignment, only gravity is assumed in the calculations for the simulation and the results. Accordingly, projectile motion is the path of the object moves or travelled when it is projected to the space or air. When the projected object is moving it will have a constant acceleration acting towards the Earth commonly known as the gravitational acceleration force. Hence the object will end up stop moving forward and fall to the ground after some time after the projection.

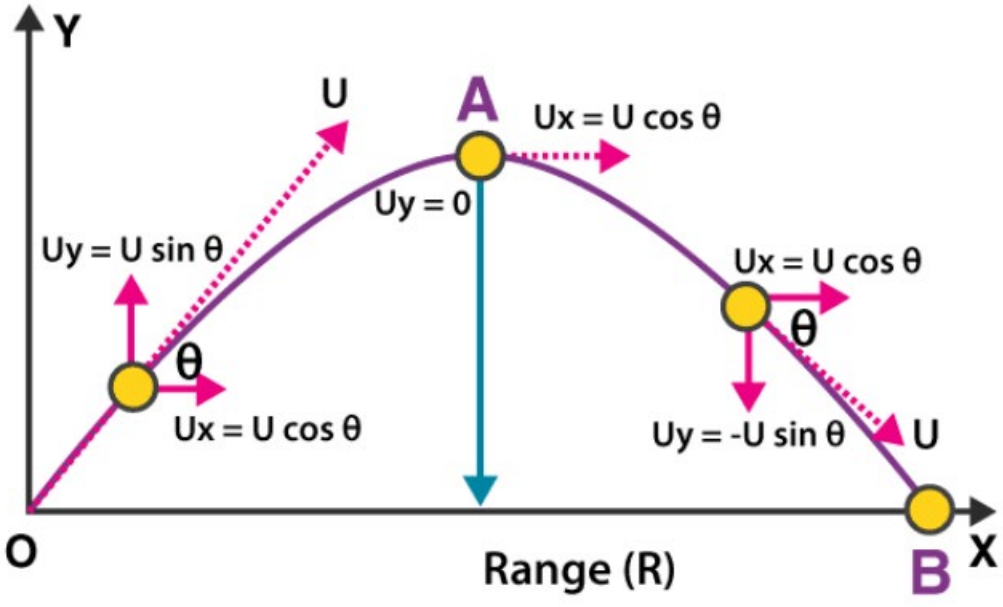


Figure 1: Projectile Motion (Byju's, n.d.)

Parameters and Equations

Many parameters and results can be calculated in projectile motion such as flight time, distance travelled, maximum height and so on. Different initial projection parameters such as projection angle and initial speed will affect the results of projectile motion. All of them will be calculated using the program created in this assignment. The formulas that are utilized to simulate and analyze the projectile motion in the program are listed as below.

Calculated Value	Formula
Horizontal Velocity	$Vx = V_0 * \cos(\theta)$
Vertical Velocity	$Vy = V_0 * sin(\theta)$
Flight Time	$t = 2 * \frac{Vy}{g}$
Range of Projectile / X Displacement	$X = X_0 + 2 * Vx * \frac{Vy}{g}$
Maximum Height / Max Y Displacement	$Y = Y_0 + \frac{(Vy^2)}{2 * g}$

Table 1: List of Formulas

The table below lists what does the variables means in the formula above.

Variable Name	Content
V_0	Initial Velocity
g	Gravitational Acceleration (Earth, Moon, Sun, Mars)
Y_0	Initial Height
X_0	Initial Displacement

Table 2: List of Variables

The table below lists the different gravitational acceleration at different environment or planet.

Location	Gravitational Acceleration
Earth	$9.81\, m/s^2$
Moon	$1.62\, m/s^2$
Sun	$274\, m/s^2$
Mars	$3.72\, m/s^2$

Table 3: Gravitational Acceleration at Different Location

Design and Flow of Routines

The program created in this assignment can be divided into 2 parts which are the projectile motion simulation part and the projectile motion simulation part.

In the projectile motion simulation part. The user is provided choices of different environment at different planet with different gravitational acceleration which includes Earth, Moon, Sun and Mars. The program will automatically assign the correct gravitational acceleration value to be applied in the calculation when the user chooses the planet that they want at the dropdown menu. The value of the gravitational acceleration will also be displayed to the user. Then the program will need the input from user for the Initial Velocity, Initial Height, Initial Displacement and Initial Angle. All the inputs are assumed as 0 if the user did not provide any input. And also the maximum angle that the user can input is 90 degrees and the minimum is 0 degrees. If the user input any angle more than 90 degrees, the program will automatically assumed it as 90 degree while if lesser than 0 degree the program will assume it as 0 degree. Then the user can press the green go button to start the simulation and the results of Horizontal Displacement, Flight Time and Max Height.

On the other hand, for the projectile motion mini game part the user will be given a random target generated by the program at the x axis after pressing the generate target button. Then, the user will need to give inputs of firing angle and initial velocity to the program. After giving the inputs, the user will have to press the fire button then the projectile motion will be displayed to the user to see if it hits the target. If the user successfully hit the target, the program will display congratulation, add 1 point to the user and generate a new target to the user. If the user failed to hit the target, the user can continue to give new inputs to the program and press the fire button again until he hits the target. Or else the user can also press the generate target button to generate a new target. To reset the points and the game the user can press the reset button.

The flow chart of this program will be divided into 2 parts which is the Projectile Motion Simulation and the Projectile Motion Mini Game.

Projectile Motion Simulation Flow Chart

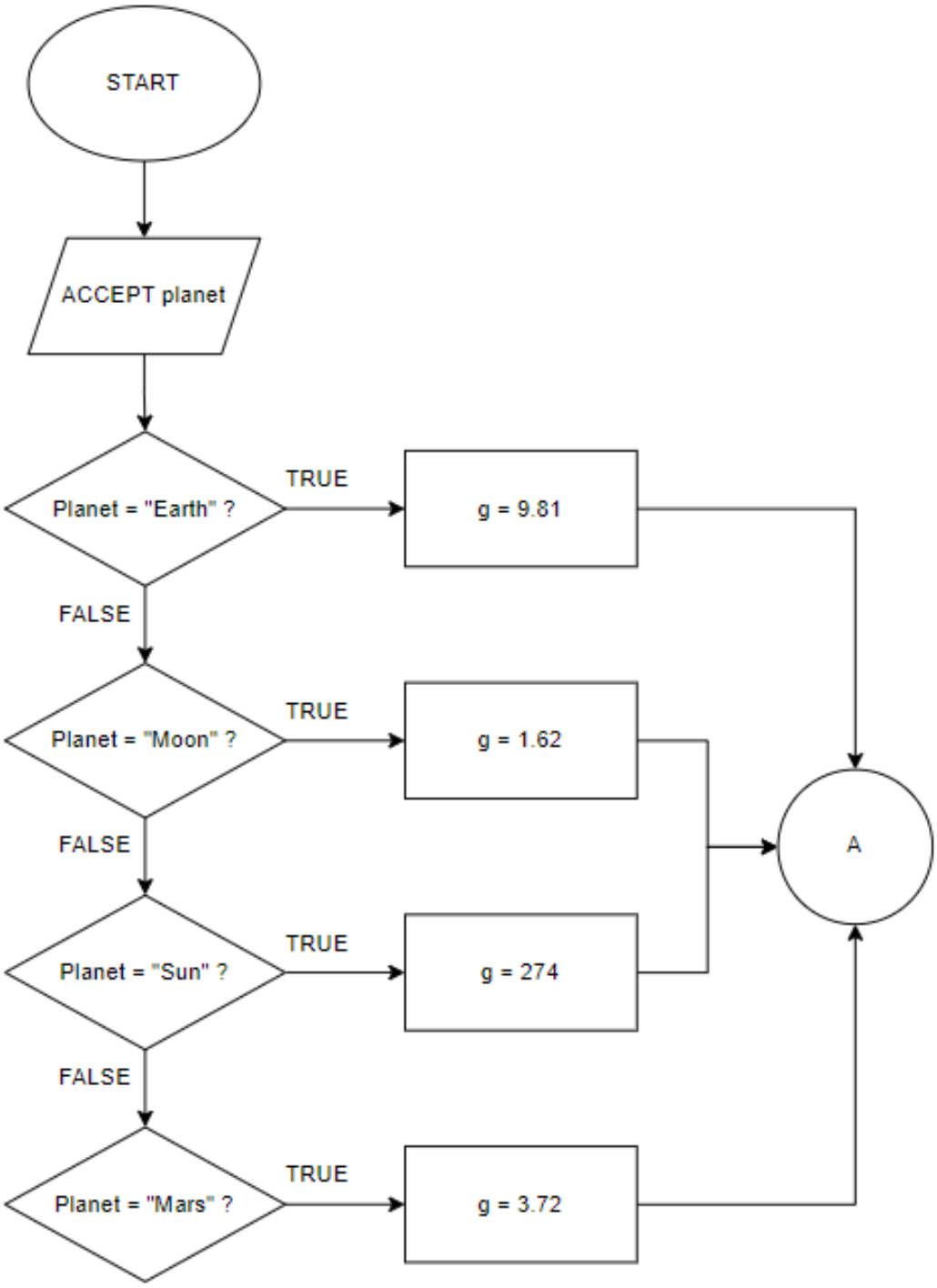


Figure 2: Simulation Flow Chart 1

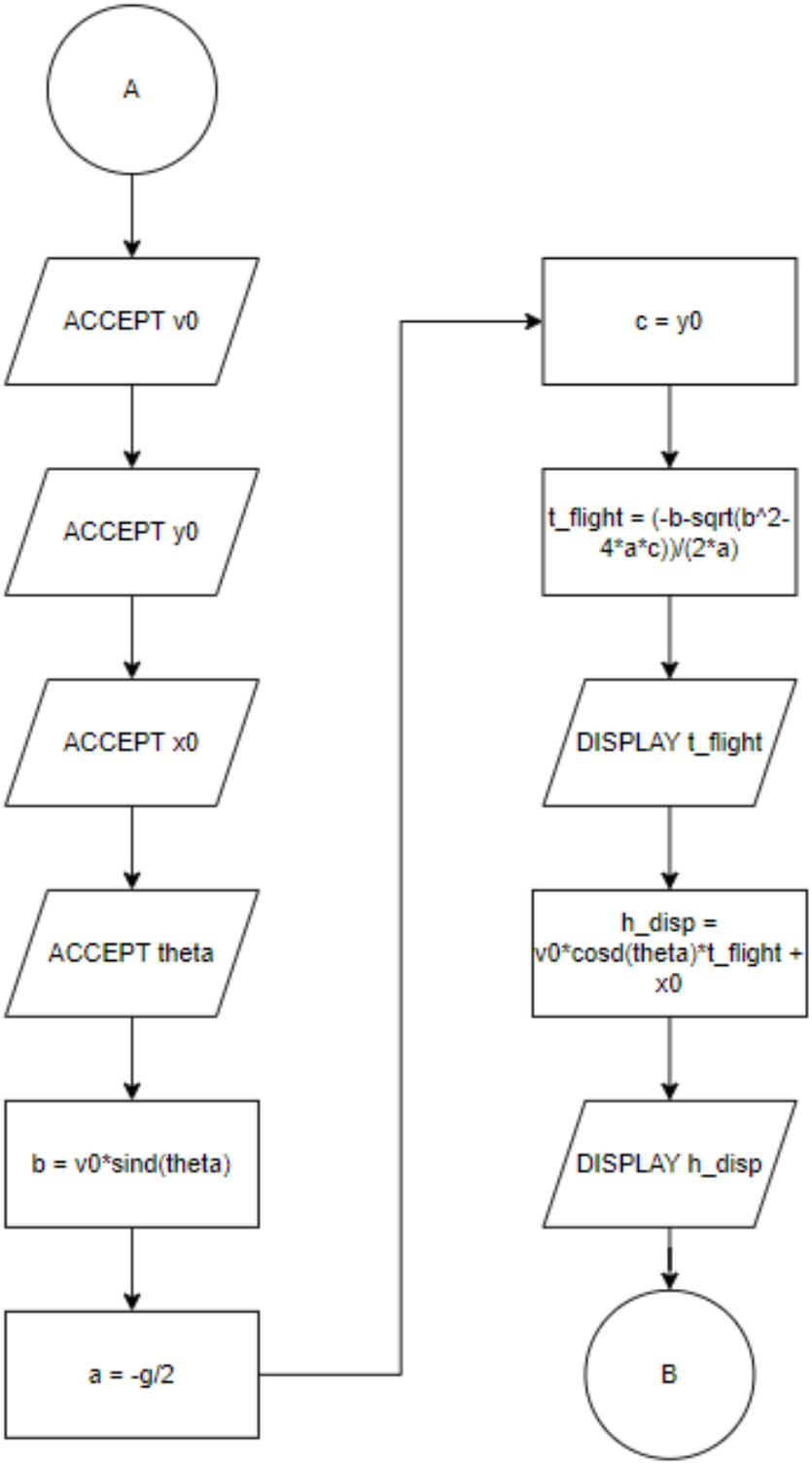


Figure 3: Simulation Flow Chart 2

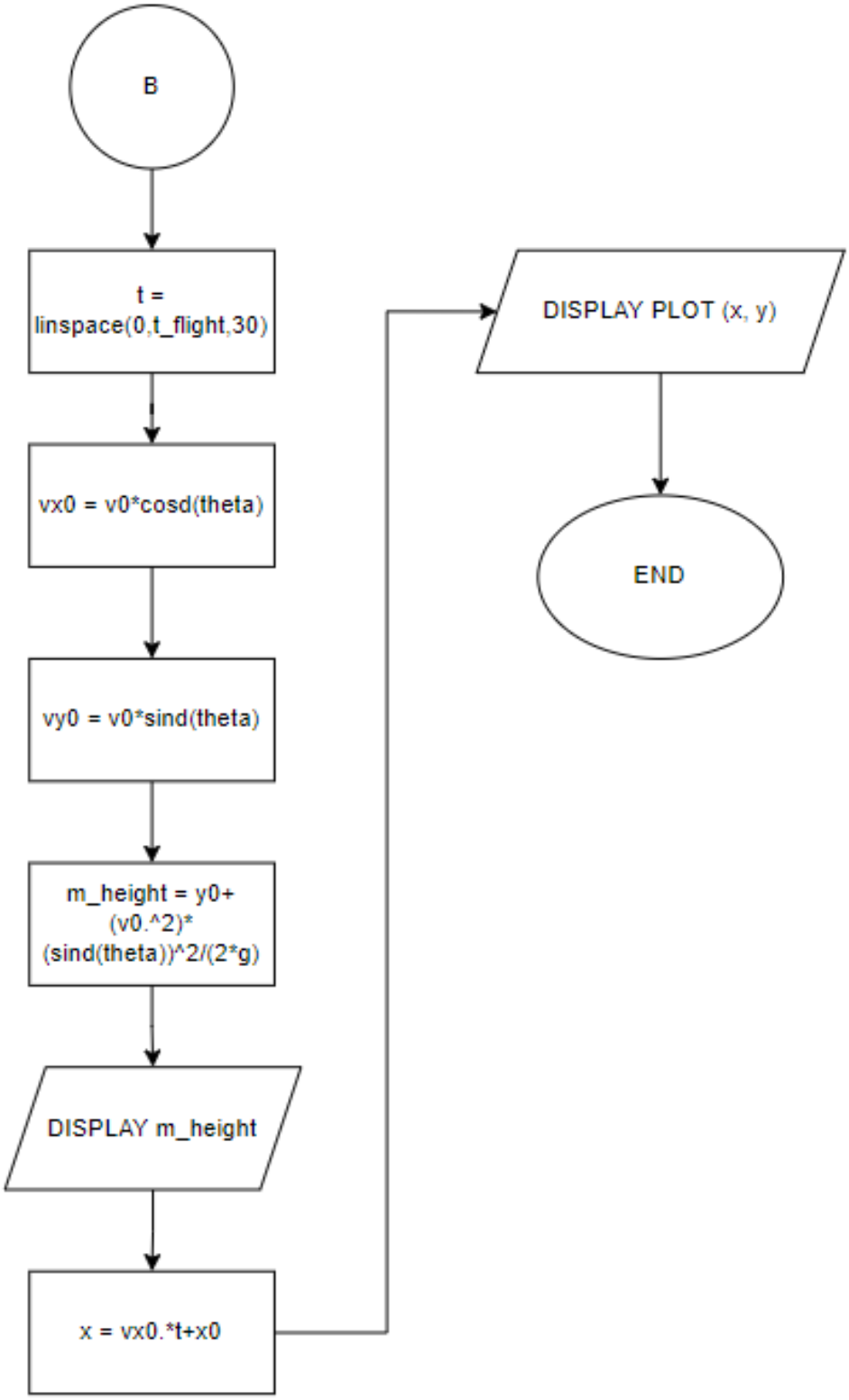


Figure 4: Simulation Flow Chart 3

Projectile Motion Mini Game Flow Chart

- Create Target Button Function

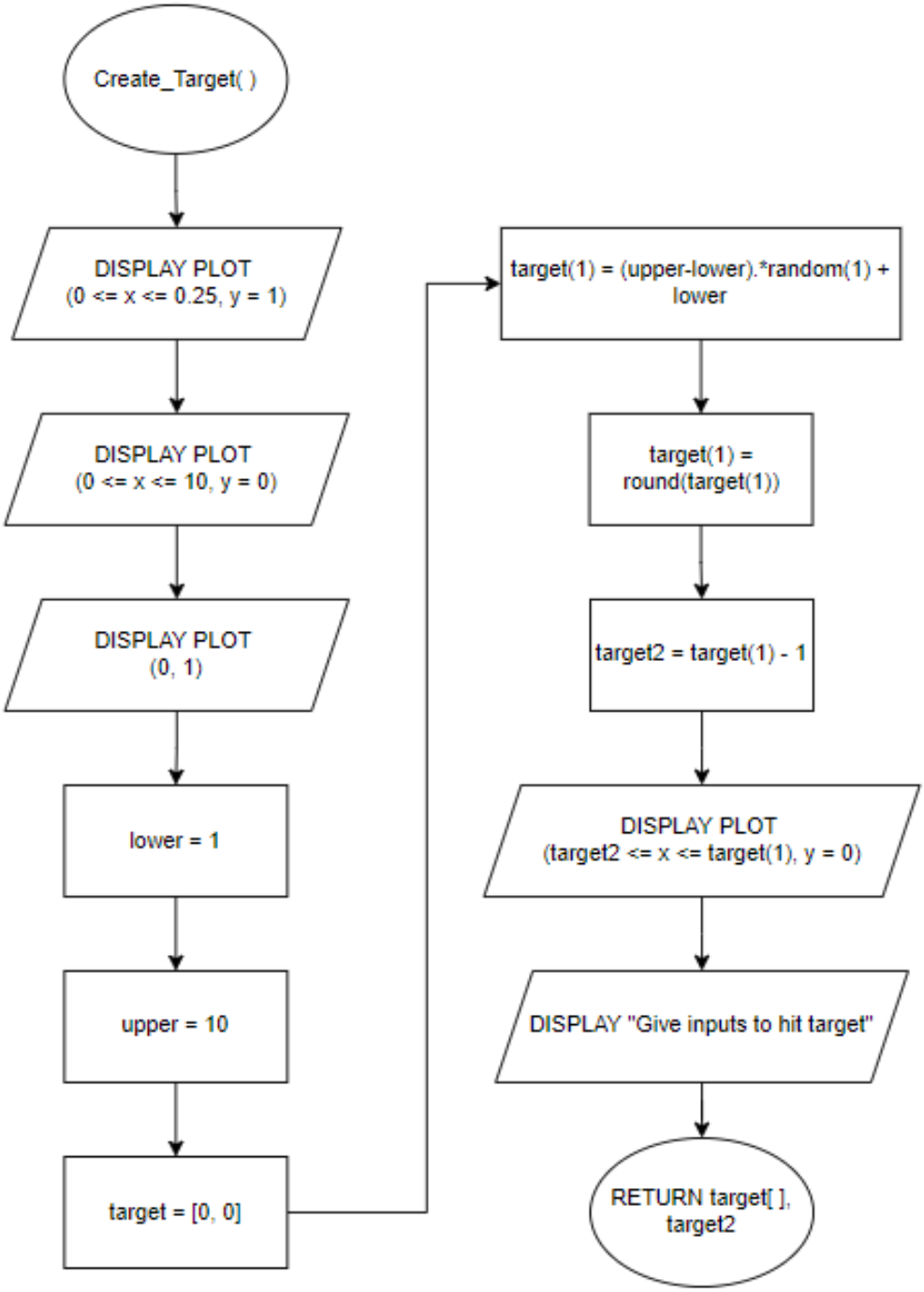


Figure 5: Mini Game Flow Chart (Create Target)

- Fire Button Function

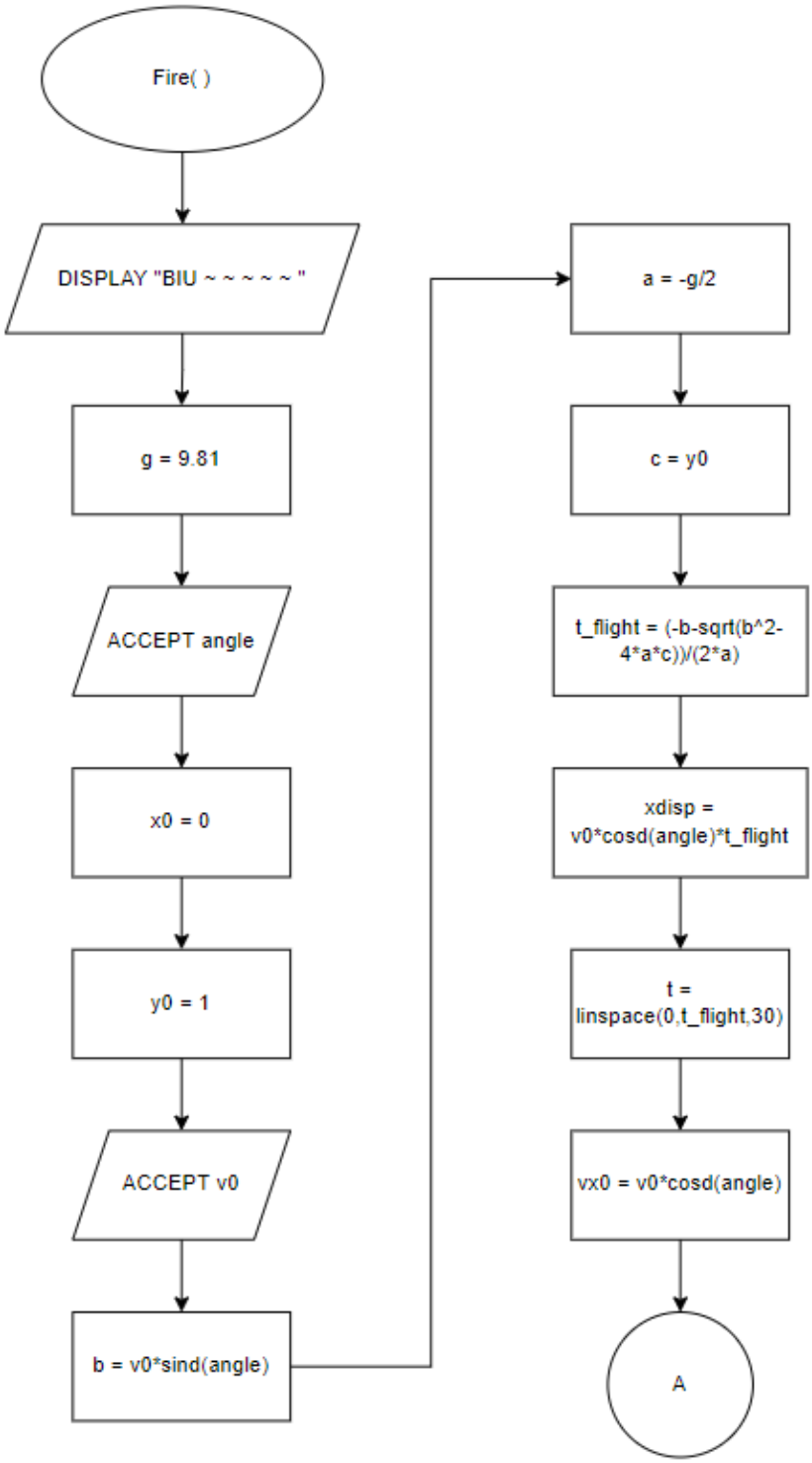


Figure 6: Mini Game Flow Chart (Fire 1)

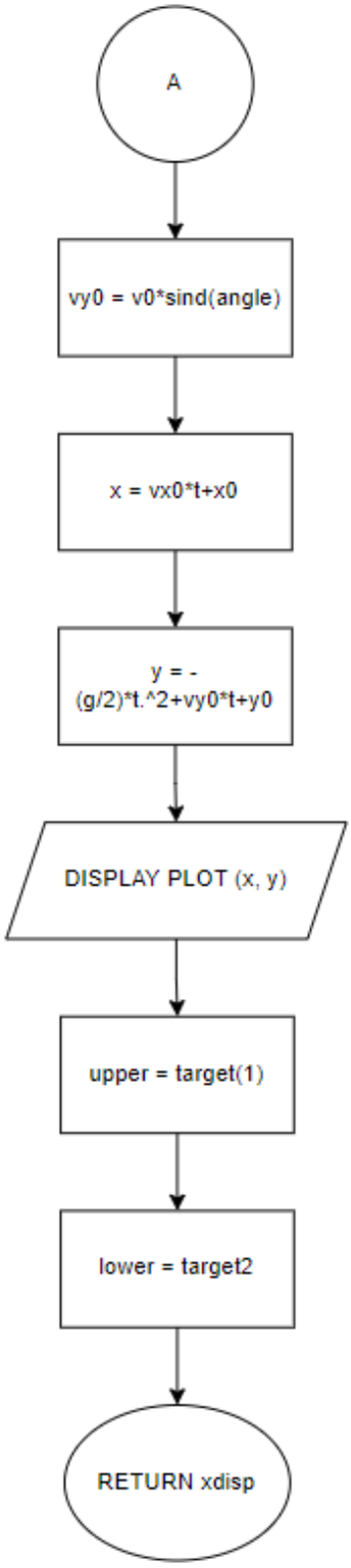


Figure 7: Mini Game Flow Chart (Fire 2)

- Start / Reset Button Function

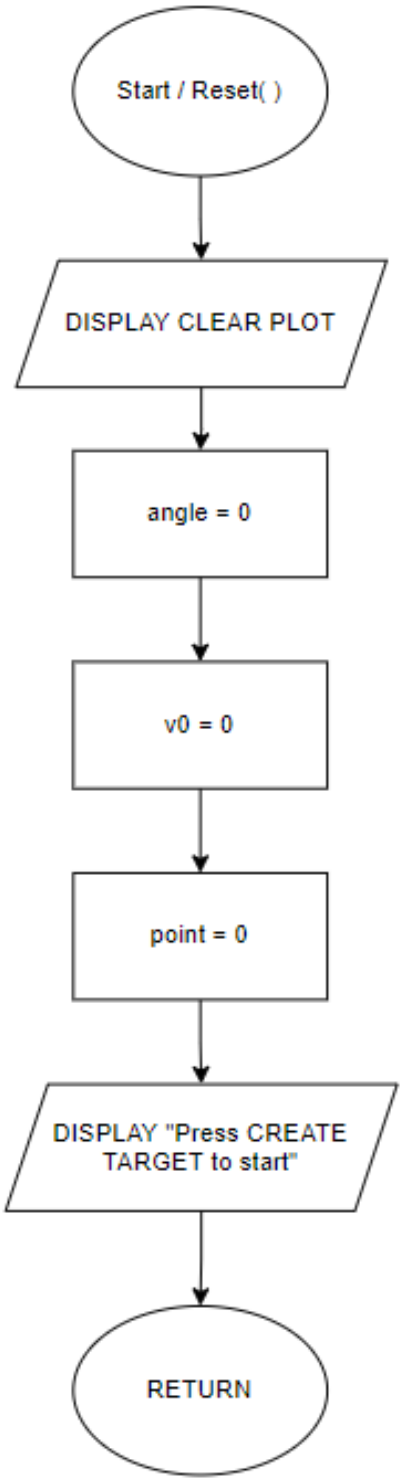


Figure 8: Mini Game Flow Chart (Start / Reset)

- Stop / Quit Button Function

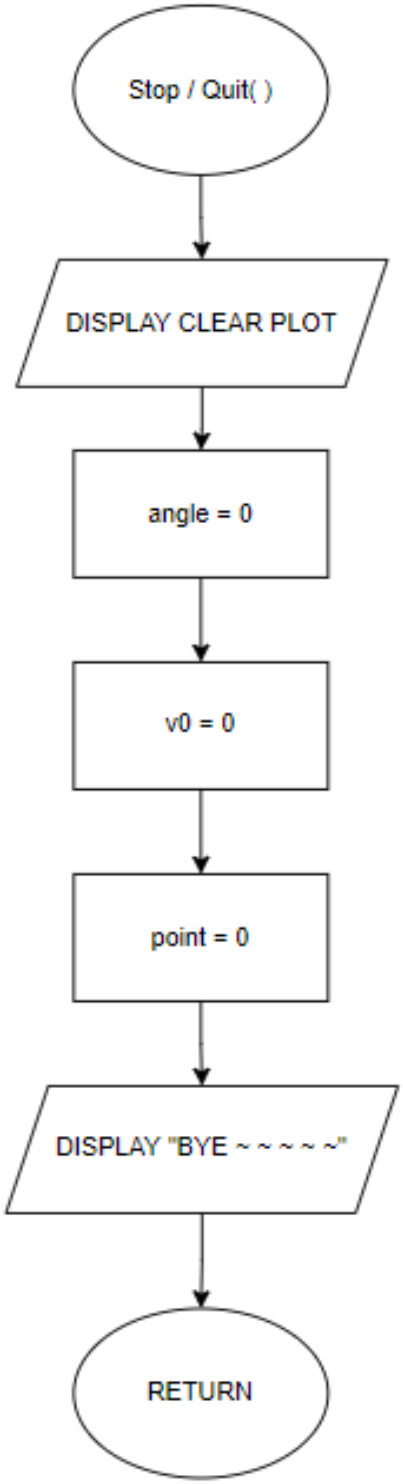


Figure 9: Mini Game Flow Chart (Stop / Quit)

- Main Code

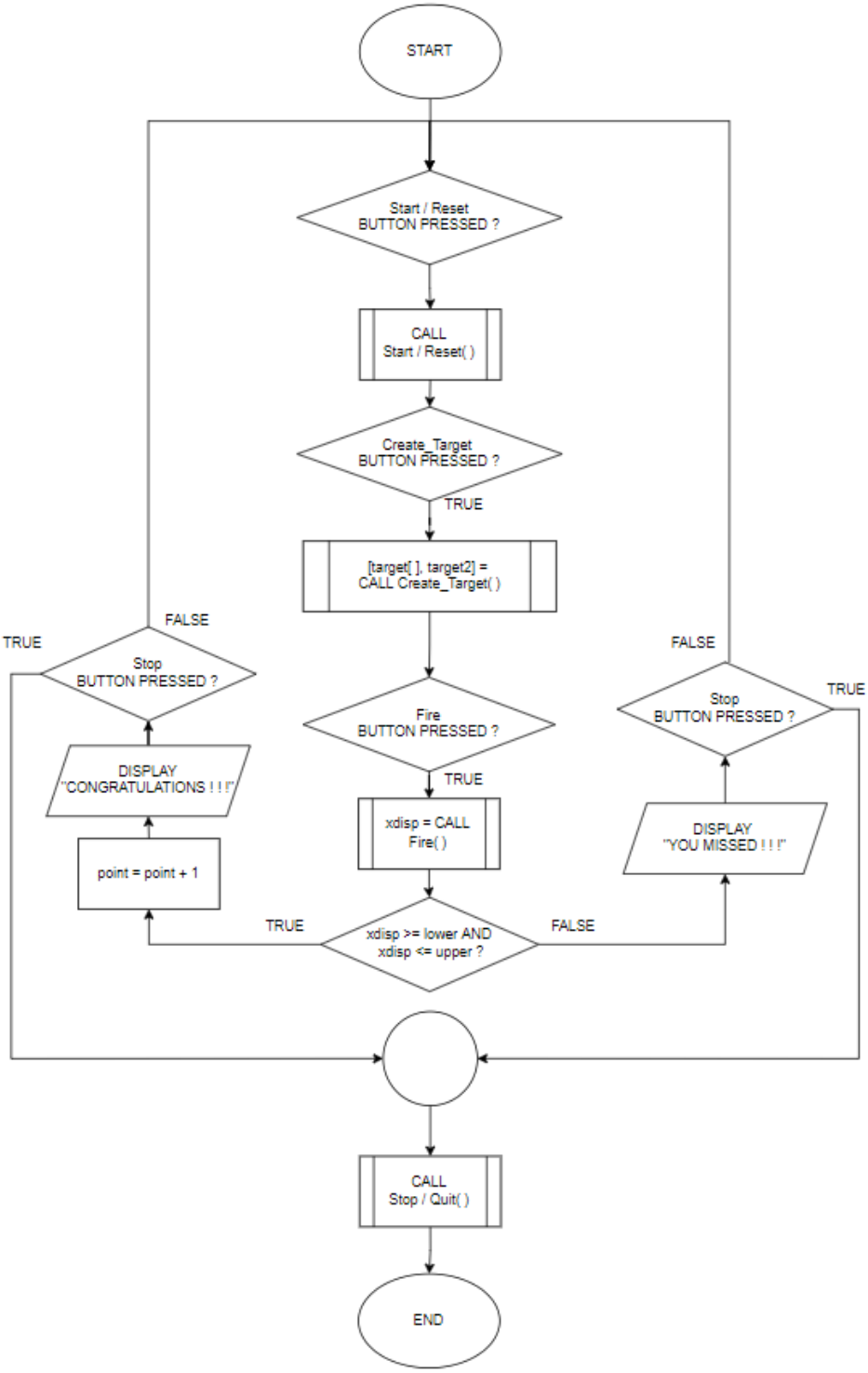


Figure 10: Mini Game Flow Chart (Main Code)

Format and Style of Routines

```
properties (Access = public)
    %To be used across the whole program
    target
    target2
end

methods (Access = public)

    function reset(app)
        %To be called by all button callback functions
        app.FIREANGLEEditField.Value = 0;
        app.INITIALVELOCITYEditField.Value = 0;
        app.PointEditField.Value = 0;
        cla(app.UIAxes2)
    end
end
```

Figure 11: Codes 1

First and foremost, the variables and function that is going to be used throughout the program are declared as public function and public variable. The function and variable above will be used in the mini game.

Projectile Motion Simulation

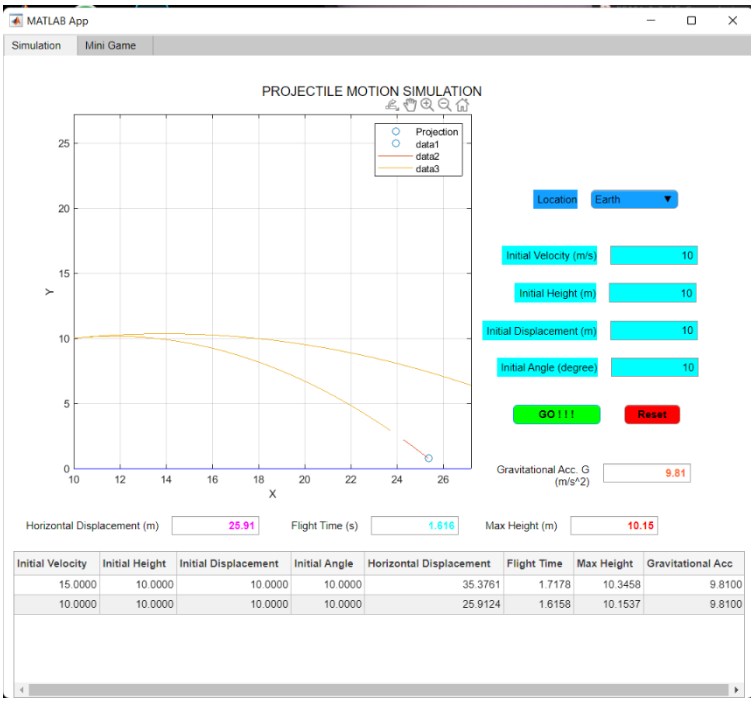


Figure 12: Projectile Motion Simulation UI

Figure above displays the user interface of the simulation. The user is required to choose the location that they want using the drop-down menu and then insert the parameters to start the simulation. To start the simulation, user has to press the green go button.

```
% Button pushed function: GOButton
function GOButtonPushed(app, event)
    %Assign the correct gravitational acceleration to g according
    %to drop down menu
    if strcmp('Earth', app.LocationDropDown.Value)
        g = 9.81;
    elseif strcmp('Moon',app.LocationDropDown.Value)
        g = 1.62;
    elseif strcmp('Sun',app.LocationDropDown.Value)
        g = 274;
    elseif strcmp('Mars',app.LocationDropDown.Value)
        g = 3.72;
    end
    %Display the gravitational acceleration of the planet chosen
    app.GravitationalAccGms2EditField.Value = g;
    %Accepting inputs from user
    v0 = app.InitialVelocitymsEditField.Value;
    y0 = app.InitialHeightmEditField.Value;
    x0 = app.InitialDisplacementmEditField.Value;
    theta = app.InitialAngledegreeEditField.Value;
    %Ensure no angle over 90 degree
    if theta > 90
        theta = 90;
    elseif theta < 0
        theta = 0;
    end
```

Figure 13: Projectile Motion Simulation Code 1

Firstly, the program will read what location is chosen by the user and assign the correct gravitational acceleration into “g” using if else function. Then the corresponding gravitational acceleration will be displayed to the user.

Then the program will check if the user input angle is within 0 to 90 or not. If it is more than 90 the program will set it as 90 while if it is less than 0 the program will set it as 0. This will prevent the program from crashing.

Then the program will read all the parameters that is given by the user and store them into variables to be calculated.

```
%Assigning the inputs into a variable to make the calculation
%looks less complex
b = v0*sind(theta);
a = -g/2;
c = y0;
%Calculation For Flight Time
t_flight = (-b-sqrt(b^2-4*a*c))/(2*a);
%Display Flight Time
app.FlightTimesEditField.Value = t_flight;
%Calculation For Horizontal Displacement
h_disp = v0*cosd(theta)*t_flight + x0;
%Display Horizontal Displacement
app.HorizontalDisplacementmEditField.Value = h_disp;
%Component's Velocity
t = linspace(0,t_flight,30); %Create 30 points to be plotted
vx0 = v0*cosd(theta);
vy0 = v0*sind(theta);

%Calculation For Maximum Height
m_height = y0+(v0.^2)*(sind(theta))^2/(2*g);
app.MaxHeightmEditField.Value = m_height;

x = vx0.*t+x0;
y = -(g/2).*t.^2+vy0*t+y0;
```

Figure 14: Projectile Motion Simulation Code 2

After assigning all the inputs, they are further complied into another variable to reduce the complexity of the codes. What all the codes is calculating is also labeled in the code using comment to let the code easily understandable. After the calculations, flying time, horizontal displacement and maximum height which are stored in the variables h_disp, t_flight and m_height are all displayed to the user as results.

Then the array “t” is created starting from 0 to the flying time with 30 elements. This means that no matter how long the projectile motion is only 30 points will be created using linspace( ) function to be plotted on the graph. This will reduce the time of plotting when the user gives inputs that provides a larger motion.

After all the calculations, the precise value of the horizontal displacement, flying time, maximum height and gravitational acceleration is all displayed to the user in the program.

```
    %Recording in Table
    tablerecord = app.UITable.Data;
    [n,m] = size(tablerecord);
    tablerecord(n+1, 1:8) = [v0 y0 x0 theta h_disp t_flight m_height g];
    app.UITable.Data = tablerecord;

    %Plotting the Graph
    hold (app.UIAxes, "on")
    rectangle(app.UIAxes, 'Position', [0 0 9999 0], 'EdgeColor', 'b')
    if m_height >= h_disp
        axis(app.UIAxes, [x0, (m_height*1.05), 0, (m_height*1.05)])
    else
        axis(app.UIAxes, [x0, (h_disp*1.05), 0, (h_disp*1.05)])
    end
    comet(app.UIAxes,x,y)
    legend(app.UIAxes,'Projection')
end
```

Figure 15: Projectile Motion Simulation Code 3

After that all the values are also recorded in a table located at the bottom of the program to make the user able to see all the values of the simulation that is made and also easier to let the user to record all the simulations.

Then the motion is plotted using the comet( ) function to let the user has a clearer view on the process of the projectile motion. The line x = 0 is also plotted for user to know where the ground is. The scale of the axis is also fixed as the same with the highest value between maximum height or horizontal displacement so that the user can visualize actually how the projectile motion looks or else MatLab will automatically set the axis to fit the values. The reason of multiplying them by 1.05 to be set as axis is to let the graph to stretch out a bit so that the graph is not too fixed to the values.

```
% Button pushed function: ResetButton
function ResetButtonPushed(app, event)
    app.InitialVelocitymsEditField.Value = 0;
    app.InitialHeightmEditField.Value = 0;
    app.InitialDisplacementmEditField.Value = 0;
    app.InitialAngledegreeEditField.Value = 0;
    app.HorizontalDisplacementmEditField.Value = 0;
    app.MaxHeightmEditField.Value = 0;
    app.FlightTimesEditField.Value = 0;
    app.UITable.Data = [];
    cla(app.UIAxes)
end
```

Figure 16: Projectile Motion Simulation Code 4

Lastly there is a reset button to clear the graph, table and reset the input values.

Projectile Motion Mini Game

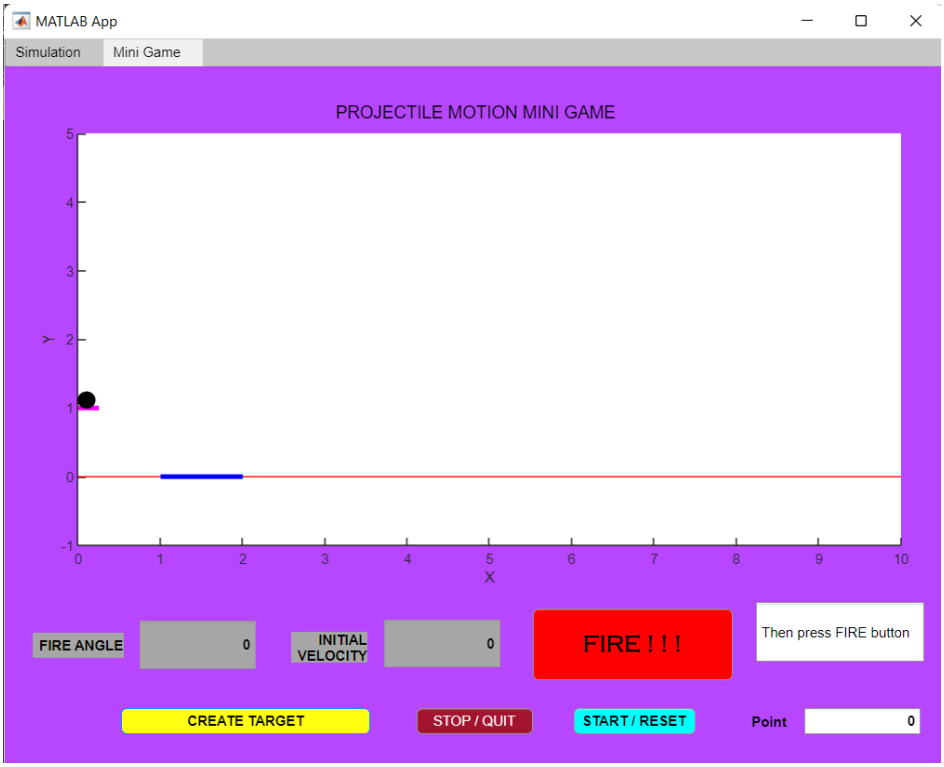


Figure 17: Projectile Motion Mini Game UI

The figure above shows the user interface of the mini game. User can switch between simulation and mini game tabs by clicking on the tabs at the top left.

```
% Button pushed function: STARTRESETButton
function STARTRESETButtonPushed(app, event)
    %To provide instructions reset the parameters, points and target
    reset(app)
    app.EditField.Value = 'Press CREATE TARGET \nto start';
end
```

Figure 18: Projectile Motion Mini Game Code 1

Firstly, the user will press the START / RESET button. Then the program will call the public function reset which is declared earlier shown in figure 11. All the input parameters and the plot on the graph will be removed and instruction will be given to the user.

```
% Button pushed function: CREATETARGETButton
function CREATETARGETButtonPushed(app, event)
    cla(app.UIAxes2)
    %Drawing line for user to understand the game easier
    rectangle(app.UIAxes2, 'Position', [0 1 0.25 0], 'EdgeColor', 'm', 'LineWidth', 3)
    rectangle(app.UIAxes2, 'Position', [0 0 10 0], 'EdgeColor', 'r')
    axis(app.UIAxes2, [0, 10, -1, 5]);
    %Drawing ball for user to understand the game easier
    rectangle(app.UIAxes2, 'Position', [0 1 0.2 0.23], 'Curvature', [1 1], 'FaceColor', 'k', 'EdgeColor', 'k', 'LineWidth', 1);
    axis(app.UIAxes2, [0, 10, -1, 5]);
    %Starting Point is constant (0 ,1)
    %Create Random Target
    lower = 1;
    upper = 10;
    global target
    global target2
    target = [0, 0];
    target(1) = (upper-lower).*rand(1) + lower;
    target(1) = round(target(1));
    target2 = target(1) - 1;
    rectangle(app.UIAxes2, 'Position', [target2 0 1 0], 'EdgeColor', 'b', 'LineWidth', 3);
    app.EditField.Value = "Give inputs to hit target";
    pause(3)
    app.EditField.Value = "Then press FIRE button";
end
```

Figure 19: Projectile Motion Mini Game Code 2

The main purpose of the create target function is to produce some graphics to the plot for the user as shown in figure 16 to let user understand how the game works. The graphics are drawn using the rectangle( ) function. The initial height and horizontal displacement of the game is constant which is at (0, 1). The axis of the graph in the game is also fix to 5 for X and 10 for Y. Hence the random target generated at the X axis must be within the range of 1 to 10. Thus, the codes are specially designed to generate random number within the range using the rand( ) function.

Since that the target is also a range so the upper bound of the target is the one generated while the lower bound will be the upper bound minus by 1. The variable that contains the upper and lower value of the target is also declared as global variable to be used by other callbacks. Then the location of the target is also plotted with blue color on the x axis of the graph to notify the user as shown in figure 17. Then the program will indicate the user to give input to fire the ball to hit the target and then press the fire button. There is a 3 second pause between 2 messages so that the user has sufficient time to read them. The pause( ) function is utilized.

```
% Button pushed function: FIREButton
function FIREButtonPushed(app, event)
    %Calling global variables to be used
    global target
    global target2
    %Let the user know that the program is calculating
    app.EditField.Value = "BIU~ ~ ~ ~ ~";
    %Accepting and assigning firing parameters
    g = 9.81;
    angle = app.FIREANGLEEditField.Value;
    v0 = app.INITIALVELOCITYEditField.Value;
    x0 = 0;
    y0 = 1;
    %Ensure no angle over 90 degree
    if angle > 90
        angle = 90;
    elseif angle < 0
        angle = 0;
    end
    %Reduce complexity
    b = v0*sind(angle);
    a = -g/2;
    c = y0;
    %Calculations to obtain the x displacement
    t_flight = (-b-sqrt(b^2-4*a*c))/(2*a);
    xdisp = v0*cosd(angle)*t_flight;
    t = linspace(0,t_flight,30); %Create 30 points to be plotted
    vx0 = v0*cosd(angle);
    vy0 = v0*sind(angle);
    x = vx0*t+x0;
    y = -(g/2)*t.^2+vy0*t+y0;
    hold (app.UIAxes2 ,"on");
    comet(app.UIAxes2, x, y)
```

Figure 20: Projectile Motion Mini Game Code 3

After giving inputs the user will press the fire button and the fire button function will be called. The purpose of this function is like what is done in the simulation part. Hence the calculation process is same with what is done in figure 14. The main thing is that the game only simulates the gravitational acceleration on Earth. Hence “g” is fixed to 9.81.

```
    %Check if hit target or not
    upper = target(1);
    lower = target2;
    if (xdisp >= lower && xdisp <= upper)
        app.EditField.Value = "CONGRATULATIONS ! ! !";
        %Adding point for user
        app.PointEditField.Value = app.PointEditField.Value + 1;
        pause(3)
        CREATETARGETButtonPushed(app, event)
    else
        app.EditField.Value = "YOU MISSED ! ! !";
    end
end
```

Figure 21: Projectile Motion Mini Game Code 4

The main difference of the code in mini game part is that the most important result is the x or horizontal displacement. Because this indicates if the user hit the target or not. The global

variable that contains the range of the target is obtained and store in the “upper” and “lower” variable is used to check if the x displacement is within the range or not. Then the if else function is used to check the condition. If the user hit the target the system will congratulate the user, add 1 point and automatically call the generate target function to generate new random target. Or else the program will tell the user that he missed and the user can give new inputs until the target is hit. Or he can also press the generate new target straightaway to generate new target.

```
    % Button pushed function: STOPQUITButton
    function STOPQUITButtonPushed(app, event)
        %To end the game and wish goodbye to user
        reset(app);
        app.EditField.Value = 'BYE~ ~ ~ ~ ~';
    end
end
```

Figure 22: Projectile Motion Mini Game Code 5

Then the last callback function is the STOP / QUIT button function. The user can stop the game whenever he wanted. When this button is pressed the program will also call the public reset function to clear all the inputs and the plots on the graph. Then the program will also display “BYE ~ ~ ~ ~ ~” to the user.

User Manual

Simulation

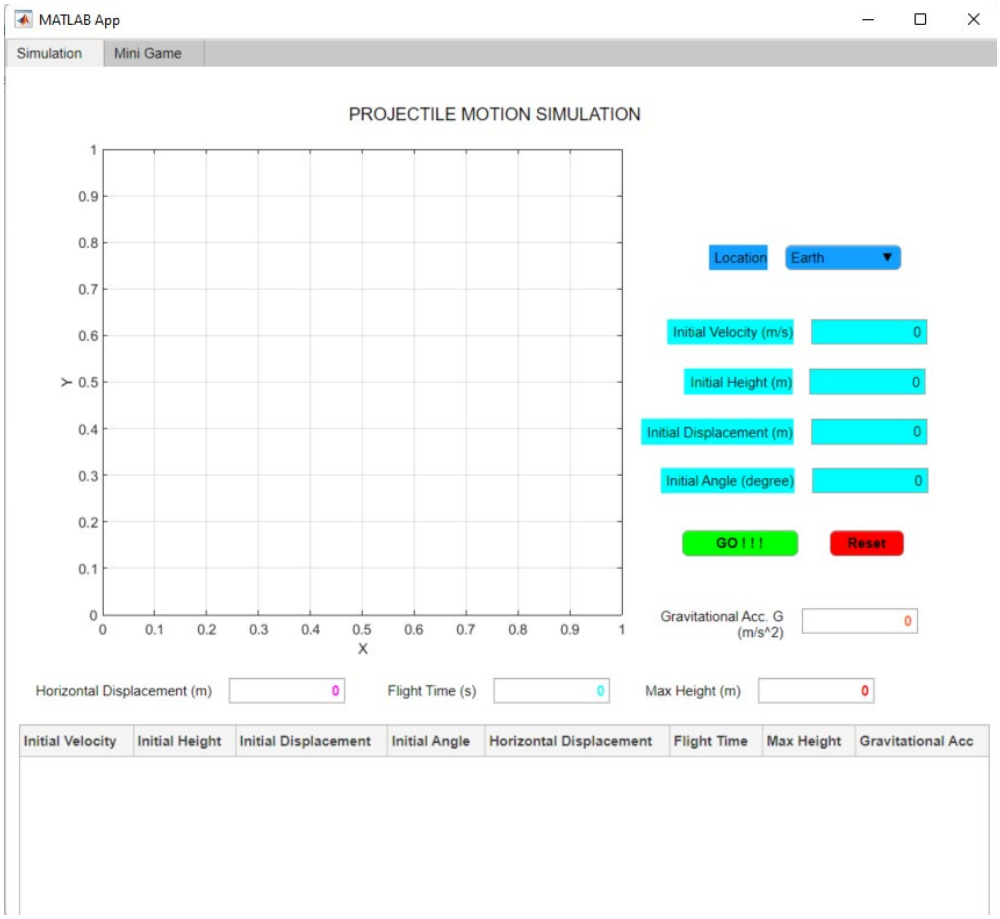


Figure 23: User Manual 1

This is what the user will see once the program is started. The user can choose the location that he wants to do the simulation on by pressing on the darker blue drop-down menu. The selection available are Earth, Moon, Sun and Mars. Then the user will have to input the parameters for the projectile motion into the boxes that are colored in blue which are initial velocity, initial height, initial displacement and initial angle. After giving all the parameters, the user will have to press the green “GO ! ! !” button to start the simulation.

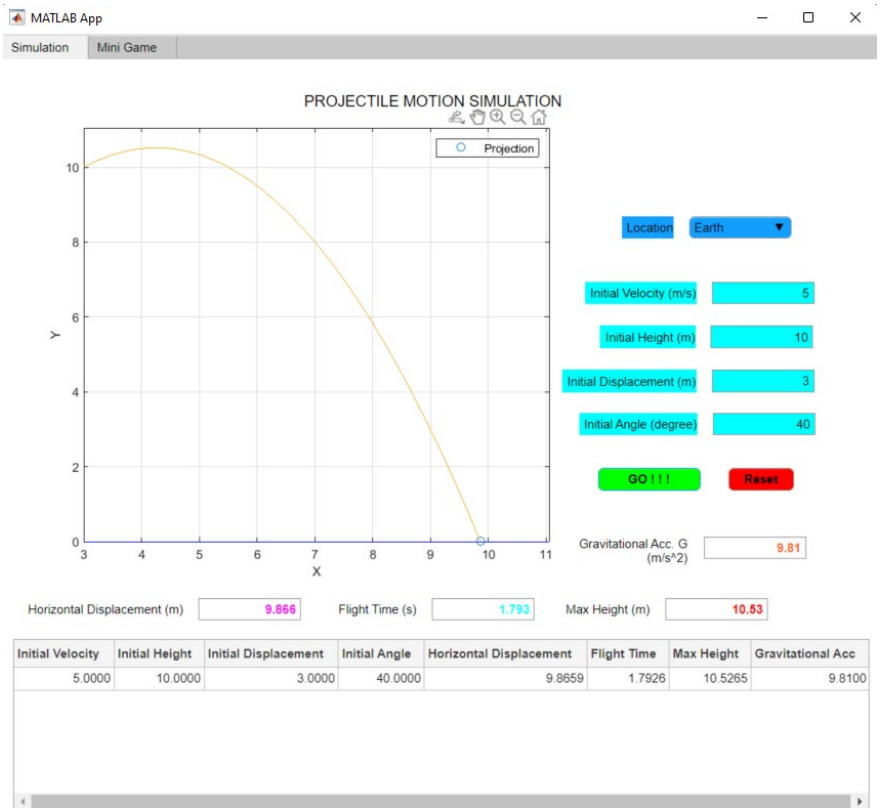


Figure 24: User Manual 2

After the user pressed the go button, all the results will be displayed at the white boxes which are horizontal displacement, flight time, maximum height and the gravitational acceleration of the location that is choose. Besides that, all the results and parameters will also be tabulated and recorded at the table below as a record.

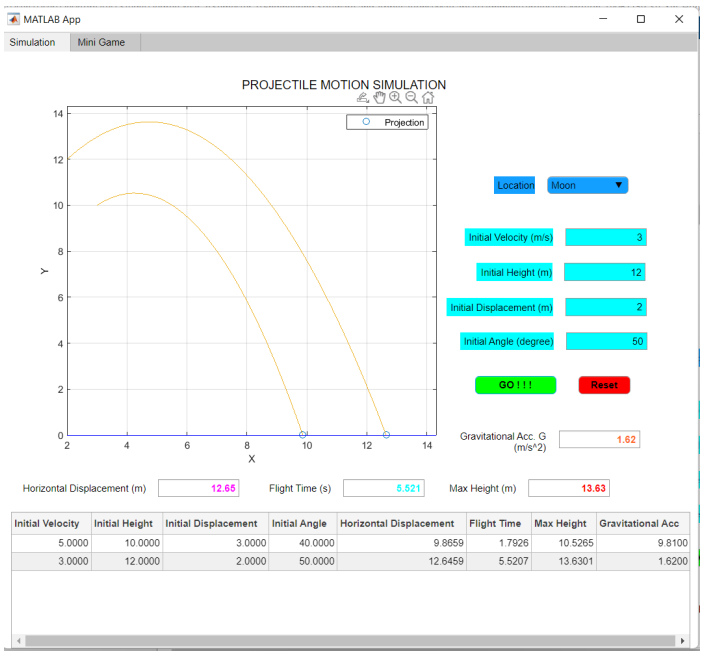
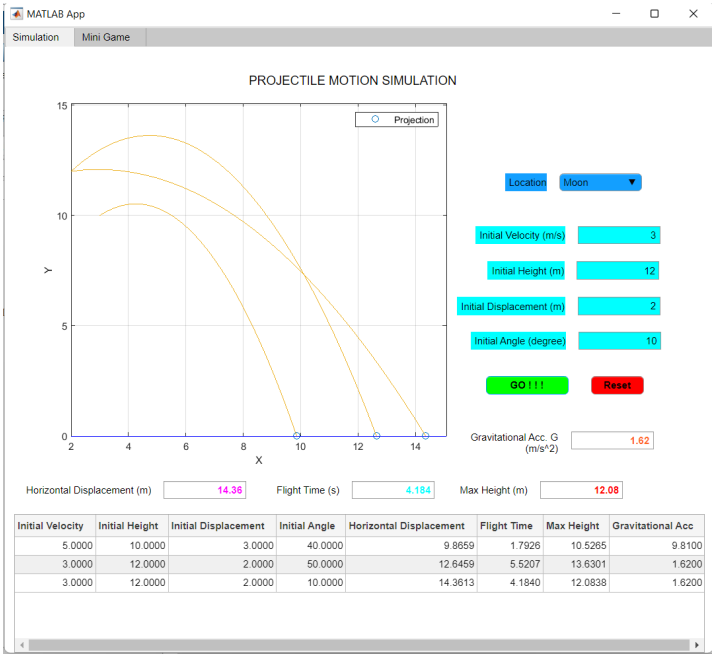


Figure 25: User Manual 3

Then if the user chooses new location and input new parameters for another projectile motion, the program will also display the new results of projectile motion in the boxes and plot on the graph. Then one new line of results will also be tabulated and added into the table below.



Initial Velocity	Initial Height	Initial Displacement	Initial Angle	Horizontal Displacement	Flight Time	Max Height	Gravitational Acc
5.0000	10.0000	3.0000	40.0000	9.8659	1.7926	10.5265	9.8100
3.0000	12.0000	2.0000	50.0000	12.6459	5.5207	13.6301	1.6200
3.0000	12.0000	2.0000	10.0000	14.3613	4.1840	12.0838	1.6200

Figure 26: User Manual 4

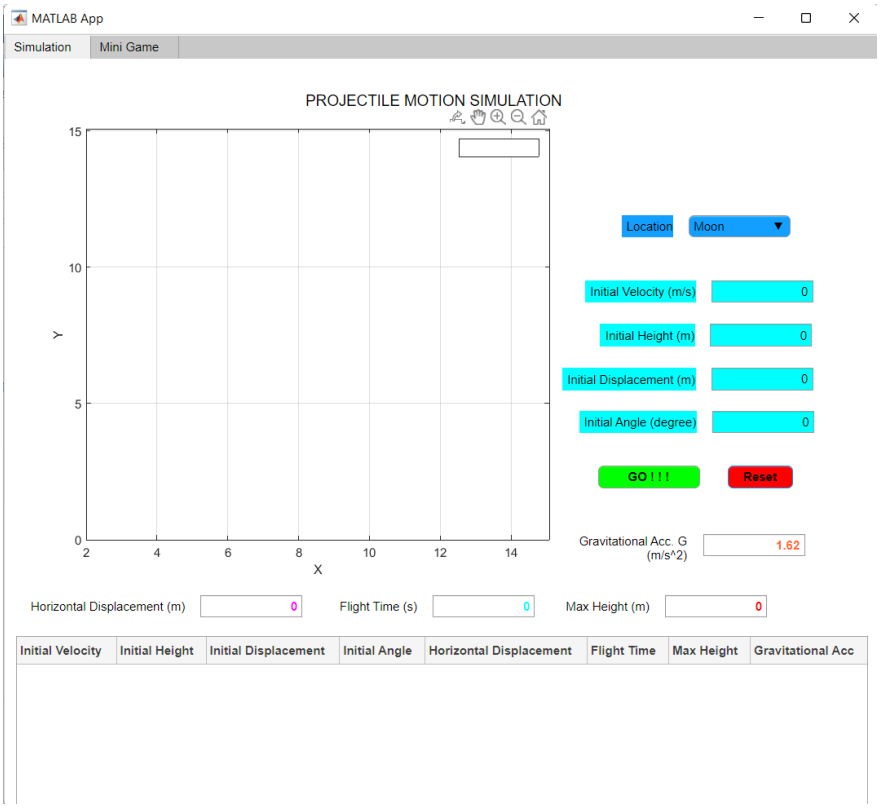


Figure 27: User Manual 5

The user can also choose to reset all the results, tables and the graph by pressing on the red "RESET" button.

Mini Game

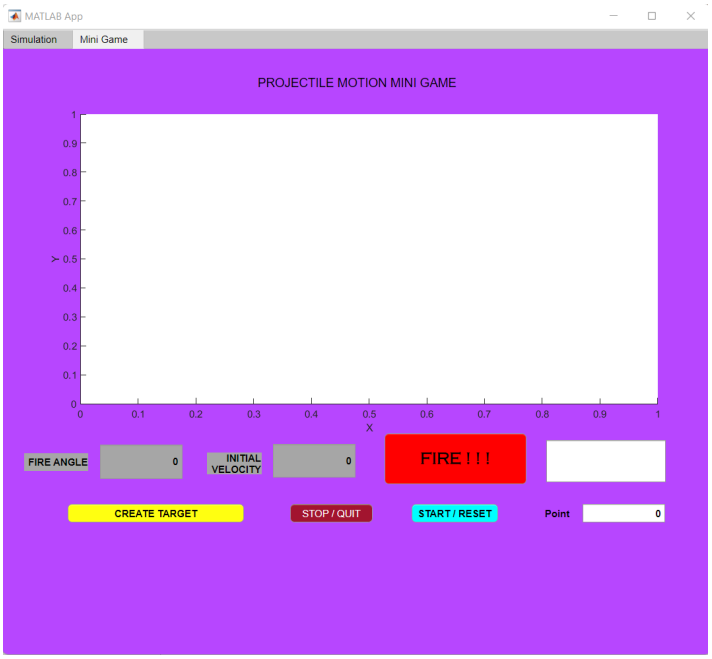


Figure 28: User Manual 6

To go to the mini game section, the user has to press the mini game tab that is located at the top left corner. To start the game, the user has to press the "START / RESET" button.

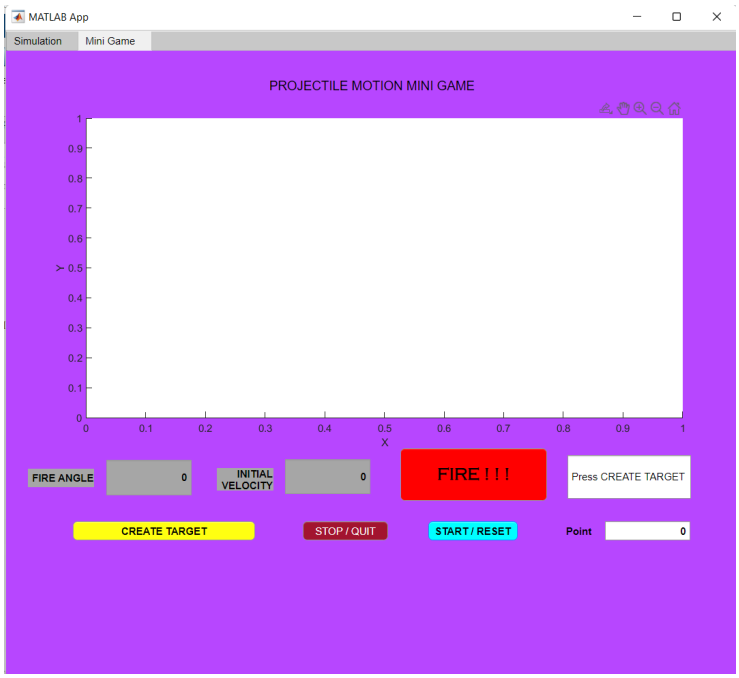


Figure 29: User Manual 7

After the button is pressed, the bigger white box at the right side will show instruction to the user which tells them to press the yellow "CREATE TARGET" button.

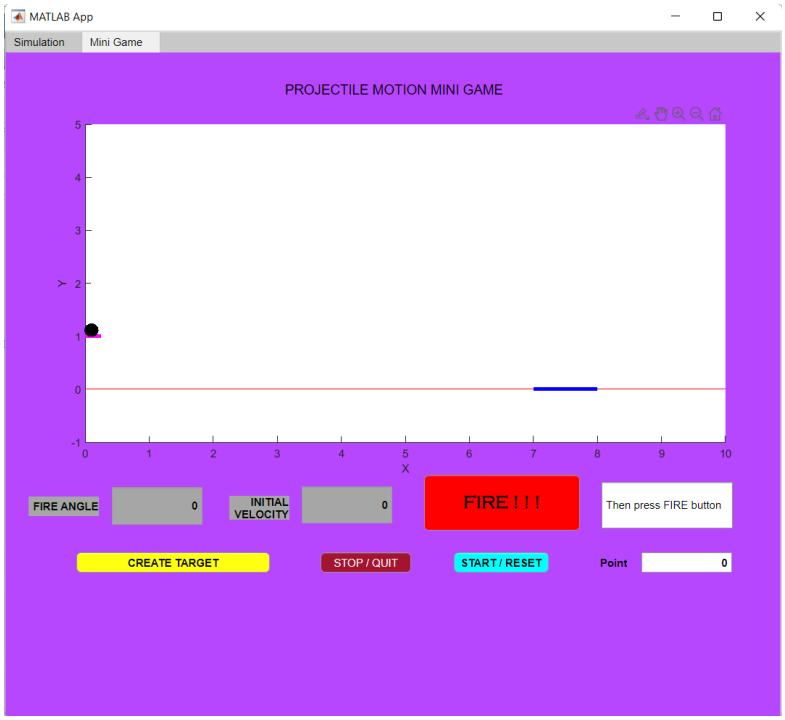


Figure 30: User Manual 8

Then the program will plot the starting point and put a random target which is colored in blue at line Y=0. The user is required to give input of firing angle and the velocity into the gray boxes and then press the red “FIRE ! ! !” button to shoot the ball.

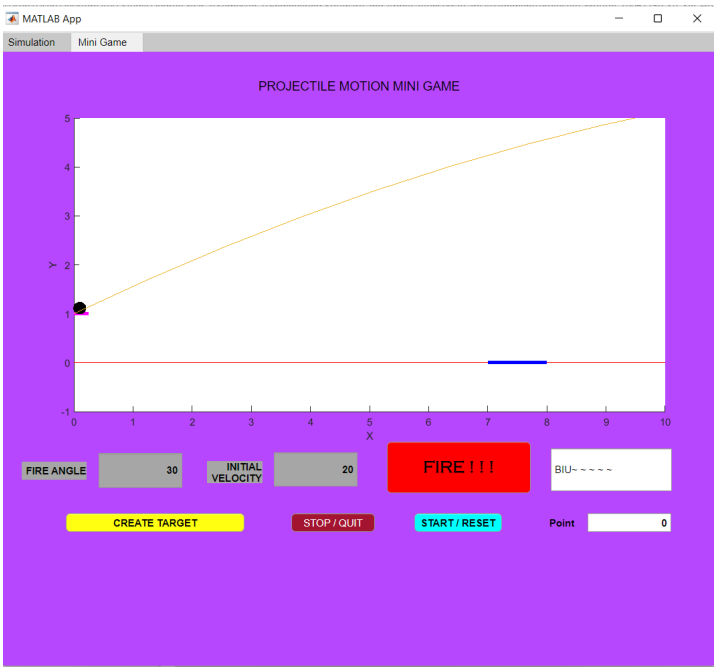


Figure 31: User Manual 9

After the “FIRE ! ! !” button is pressed, the program will show the motion of the ball and display “BIU ~ ~ ~ ~ ~” at the large white box to indicate that the motion is running.

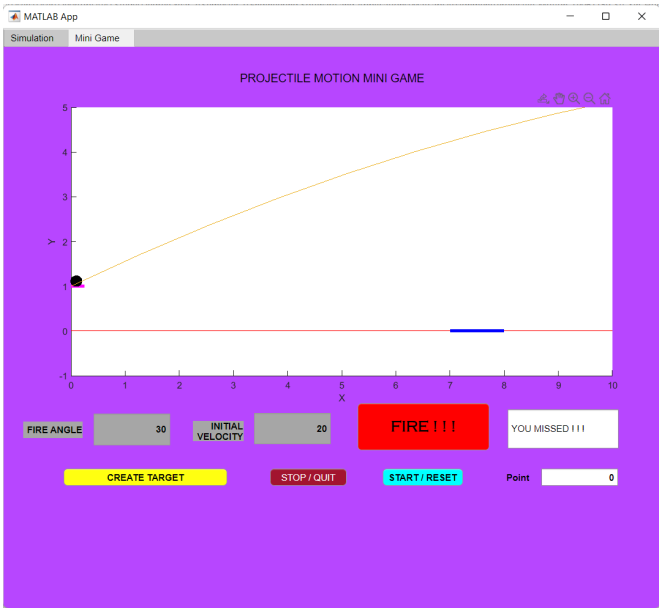


Figure 32: User Manual 10

Then if the user missed the target the program will tell the user that he had missed, then the user can input new parameters into the gray boxes to shoot again.

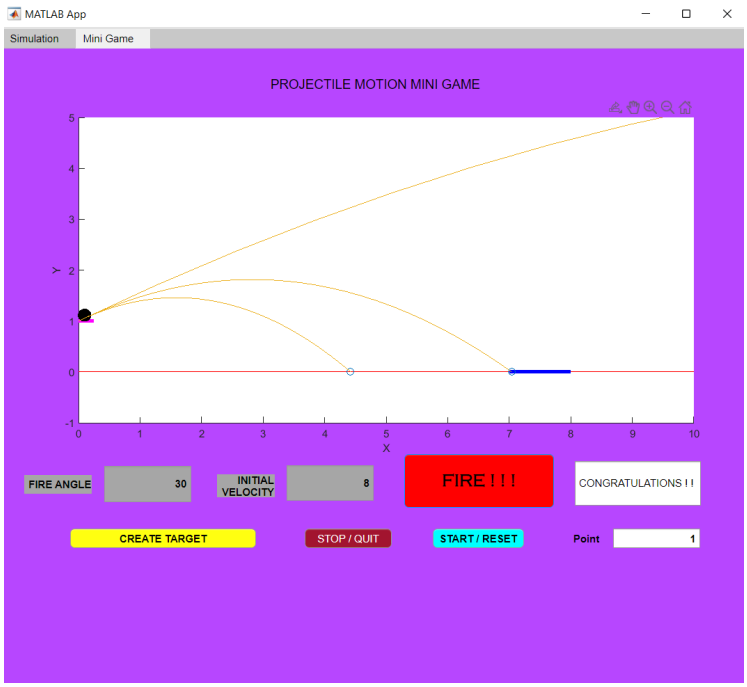


Figure 33: User Manual 11

When the user hit the target, the program will also tell the user and also add 1 point for the user at the small white box at the right side.

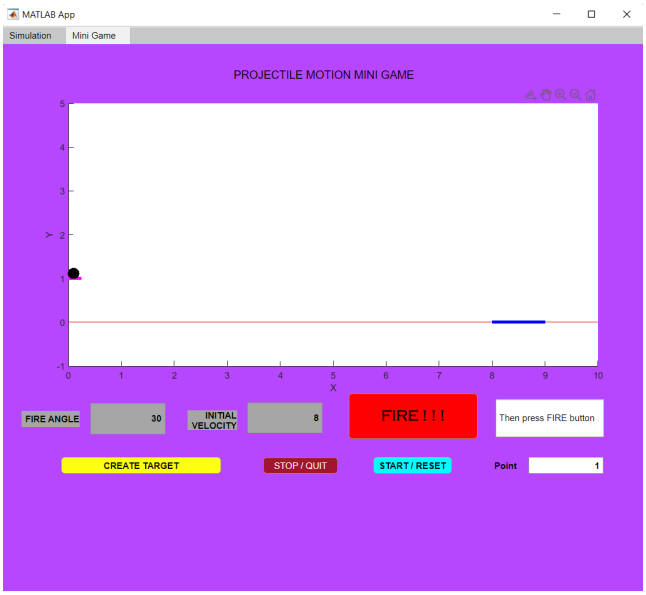


Figure 34: User Manual 12

After that the program will automatically generate the next random target to the user.

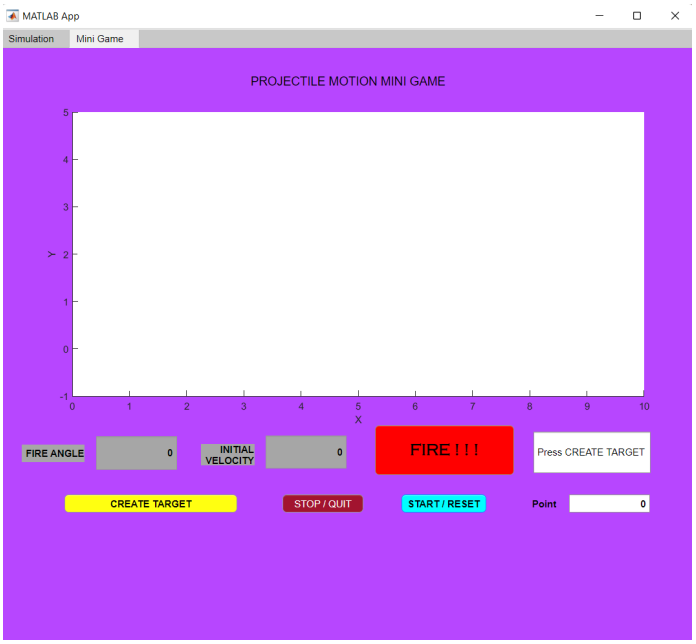


Figure 35: User Manual 13

To reset the point, the user can press again the “START / RESET” button and then repeat the process of creating target.

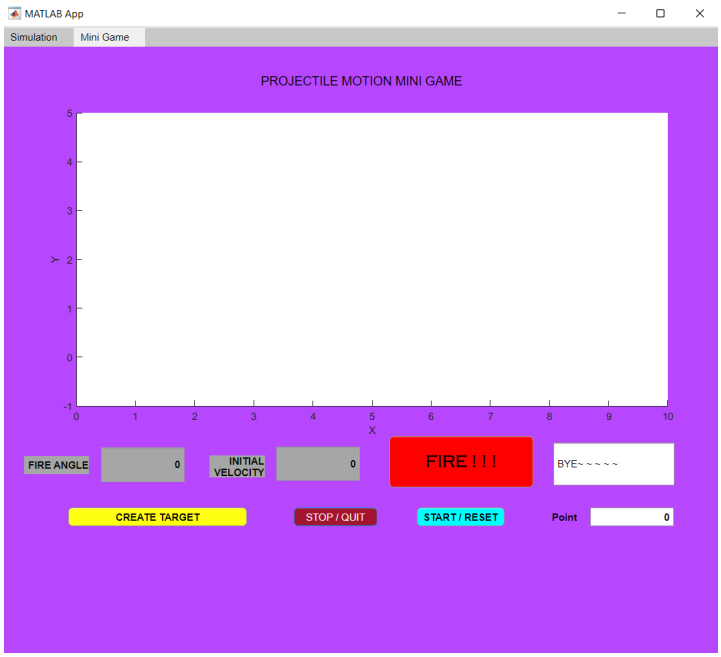


Figure 36: User Manual 14

To quit the game, the user can press the “STOP / QUIT” button and the program will wish bye to the user.

Result and Analysis

This section will be recording the analysis of the projectile motion which I will divide into 3 major parts. The first part is with the initial angle as constant and the initial velocity as variable. While the next part is vice versa. The range of angle being tested will be from 0 to 90. The last part is with same initial velocity and angle but with different location or environment which means different gravitational acceleration. All conditions will have the similar initial starting point of (0, 0). All the conditions only have 1 variable which is changing while others kept as constant. The results that will be recorded will be the horizontal displacement, maximum height and the flying time for all conditions.

Angel Variable

Gravitational Acceleration (Earth), ms^{-2}	Initial Angle, θ	Initial Velocity, ms^{-1}	Maximum Height, m	Horizontal (X) Displacement, m	Flying Time, s
9.81	0	10	0	0	0
9.81	15	10	0.34	5.10	0.53
9.81	30	10	1.27	8.83	1.02
9.81	45	10	2.55	10.19	1.44
9.81	60	10	3.82	8.83	1.77
9.81	75	10	4.76	5.10	1.97
9.81	90	10	5.10	0	2.04

Table 4: Projectile Motion Results (Angle Variable)

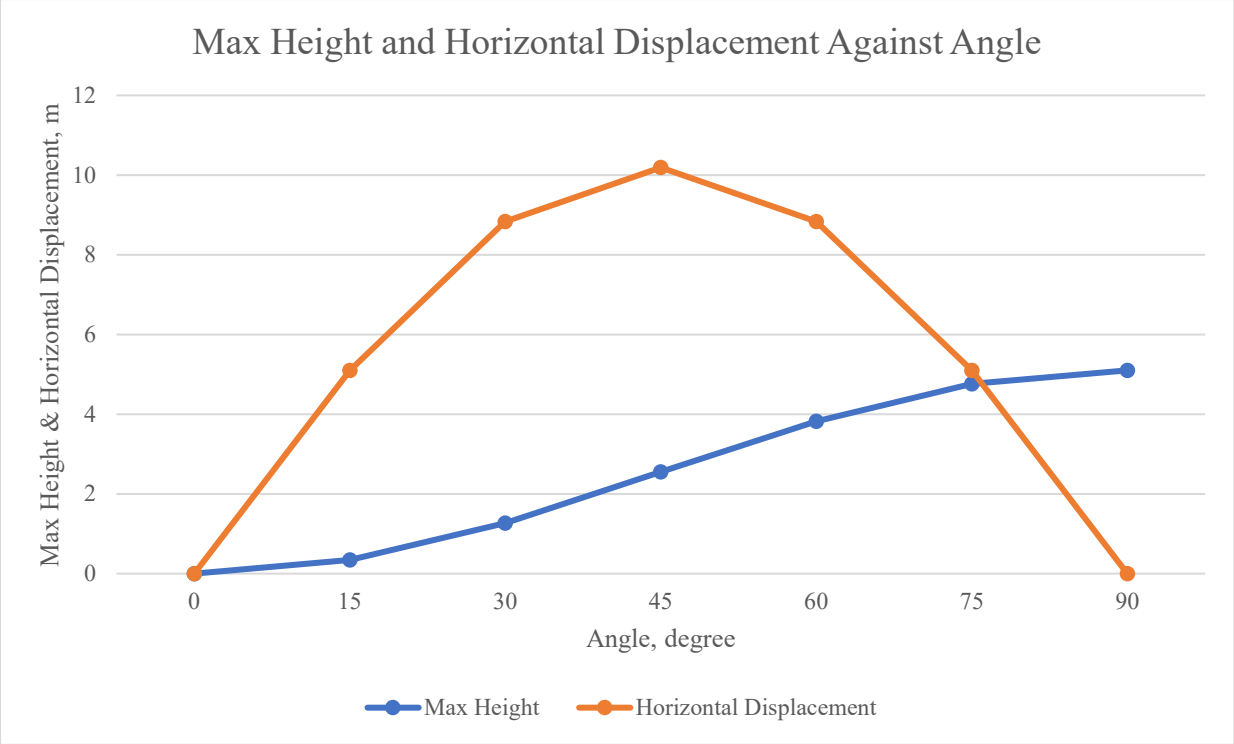


Figure 37: Max Height & Horizontal Displacement Against Angle Graph

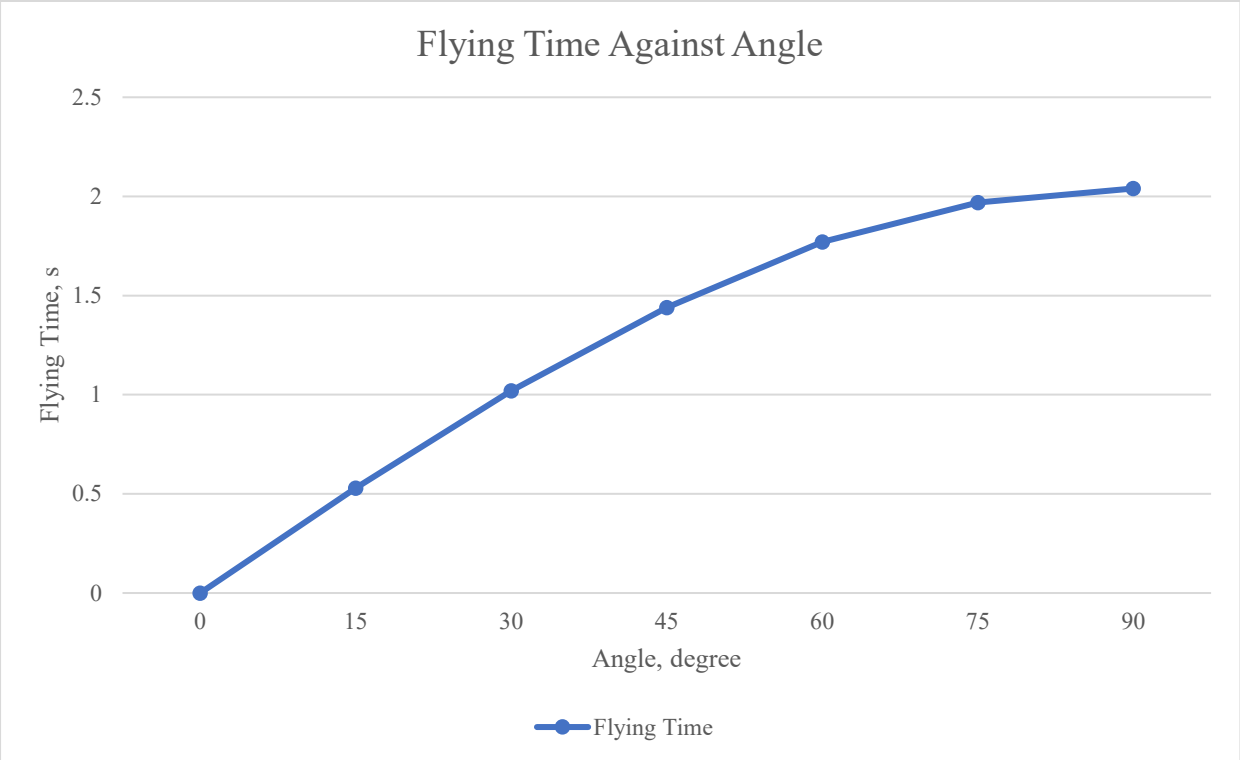


Figure 38: Flying Time Against Angle Graph

To have a clearer understanding about what is the relation between the results and the angle, figure 37 and 38 can be referred. According to the result, the maximum height of projectile motion reaches its peak when the angle is 90 degrees. This is because when the angle

is lesser than 90 degrees, force provided by the initial velocity will be divided to the horizontal component. But when it is 90 degrees, there will be no force given to the horizontal component and all force is applied at the vertical component, hence you can see there is no horizontal displacement when the angle is 90 degrees. The closer the angle to 90 degrees, the more force is applied to the vertical component, the higher the maximum height. This is also the same case for the flying time which also reaches its peak at 90 degrees. This is because when more force is acting on the vertical component, the higher the object travel, then the longer the time needed for the object to fall to ground and stop.

On the other hand, the horizontal displacement reaches its peak when the angle is 45 degrees because 45 degree is the most optimal angle where the distribution of force between vertical and horizontal component are equal. This ended up with the peak of the horizontal displacement because this is where the combination of flying time and the horizontal velocity is at its best. The motion has sufficient flying time for the object to move horizontally at the optimum speed. Hence the horizontal displacement reached its maximum.

Initial Velocity Variable

Gravitational Acceleration (Earth), ms^{-2}	Initial Angle, θ	Initial Velocity, ms^{-1}	Maximum Height, m	Horizontal (X) Displacement, m	Flying Time, s
9.81	45	0	0	0	0
9.81	45	2	0.1	0.41	0.29
9.81	45	4	0.41	1.63	0.58
9.81	45	6	0.92	3.67	0.87
9.81	45	8	1.63	6.52	1.15
9.81	45	10	2.55	10.19	1.44
9.81	45	12	3.67	14.68	1.73

Table 5: Projectile Motion Results (Initial Velocity Variable)

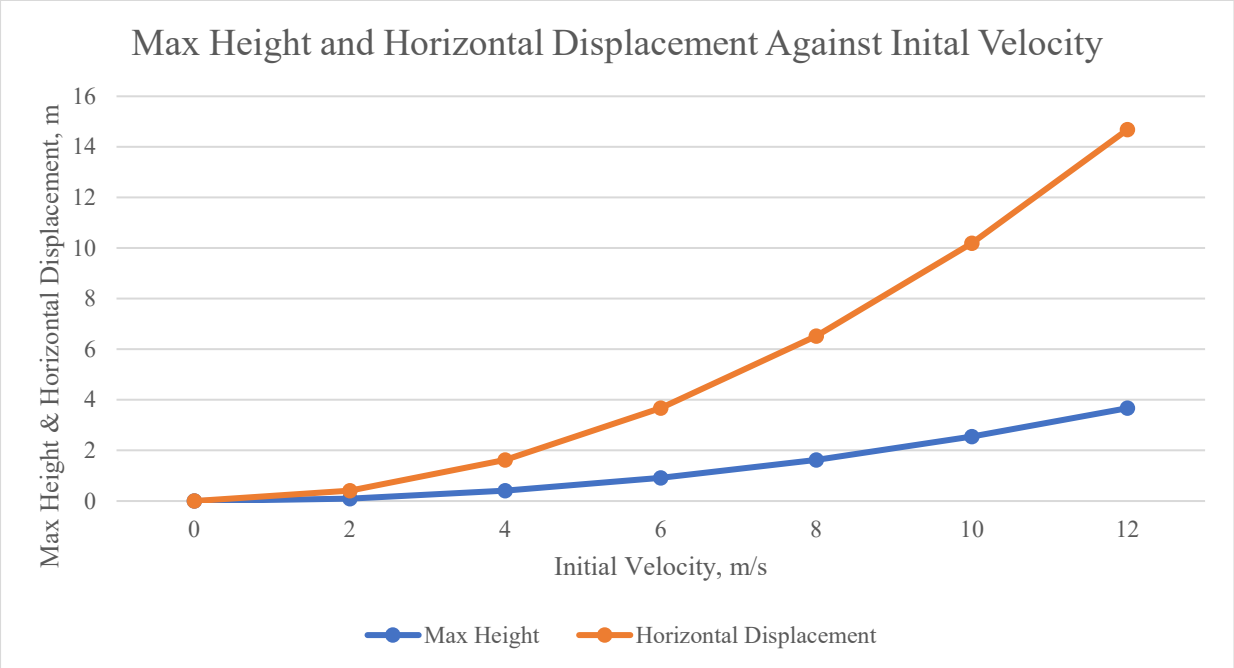


Figure 39: Max Height & Horizontal Displacement Against Initial Velocity Graph

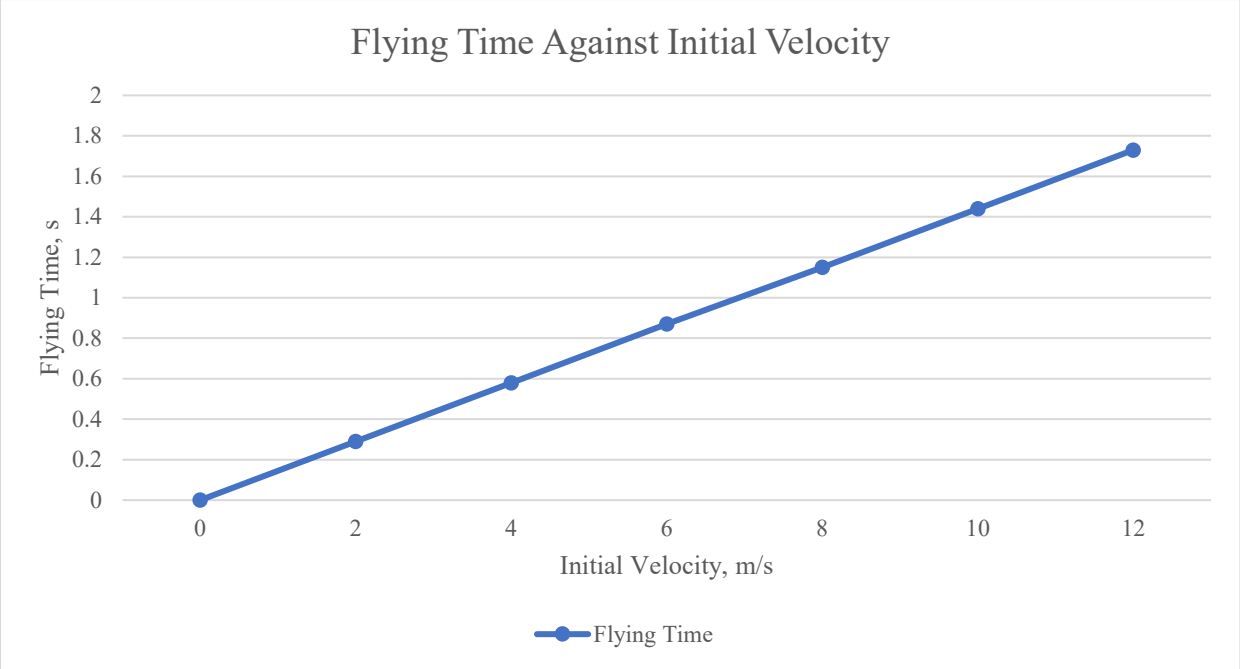


Figure 40: Flying Time Against Initial Velocity Graph

According to graph in figure 39, 40 and the table above, maximum height, horizontal displacement and flying time all increases when initial velocity increases. This is because when initial velocity increases, the force that is applied to the motion also increases. The vertical velocity of the object will increase which causes the object to project higher and leads to longer flying time and higher maximum height. In this case the object will also have more time to travel horizontally alongside also the increase in the horizontal velocity which results in increase of horizontal displacement.

Gravity Acceleration Variable

Environment / Location / Planet	Gravitational Acceleration, ms^{-2}	Initial Angle, θ	Initial Velocity, ms^{-1}	Maximum Height, m	Horizontal (X) Displacement, m	Flying Time, s
Earth	9.81	45	5	0.64	2.55	0.72
Moon	1.62	45	5	3.86	15.43	4.37
Sun	274	45	5	0.02	0.09	0.03
Mars	3.72	45	5	1.68	6.72	1.9

Table 6: Projectile Motion Results (Gravity Acceleration Variable)

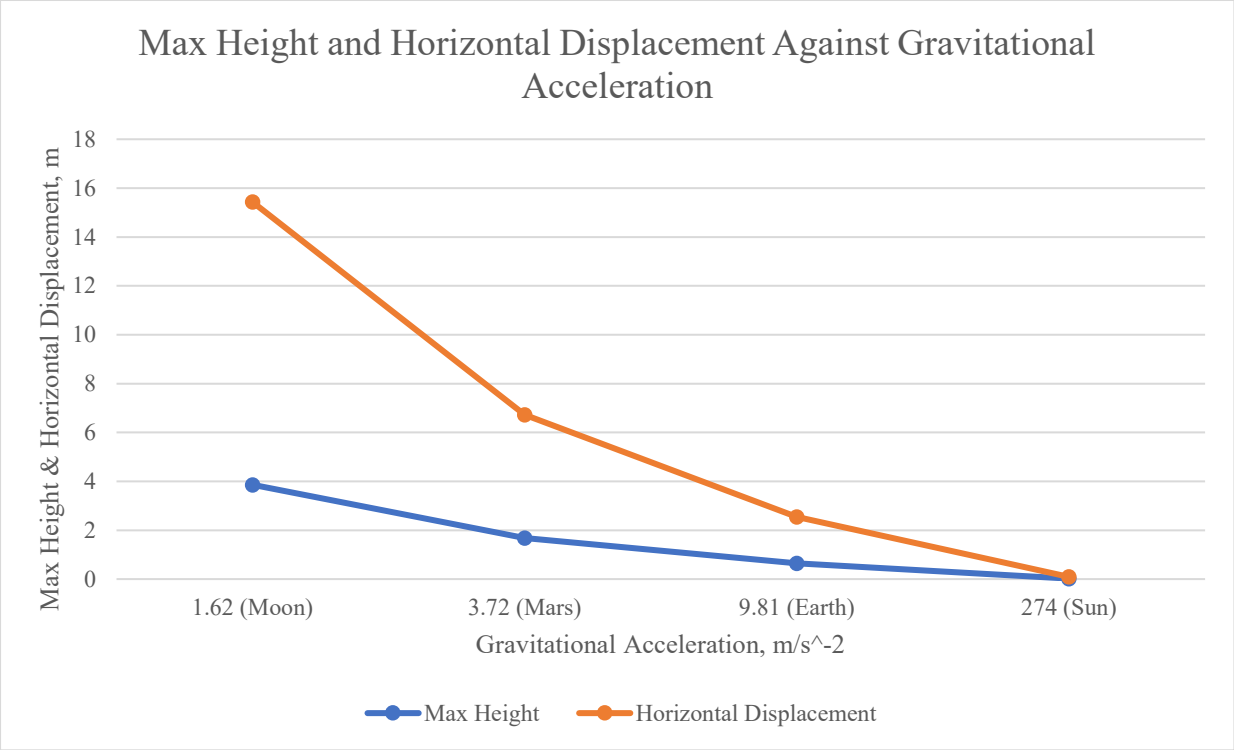


Figure 41: Max Height & Horizontal Displacement Against Gravitational Acceleration Graph

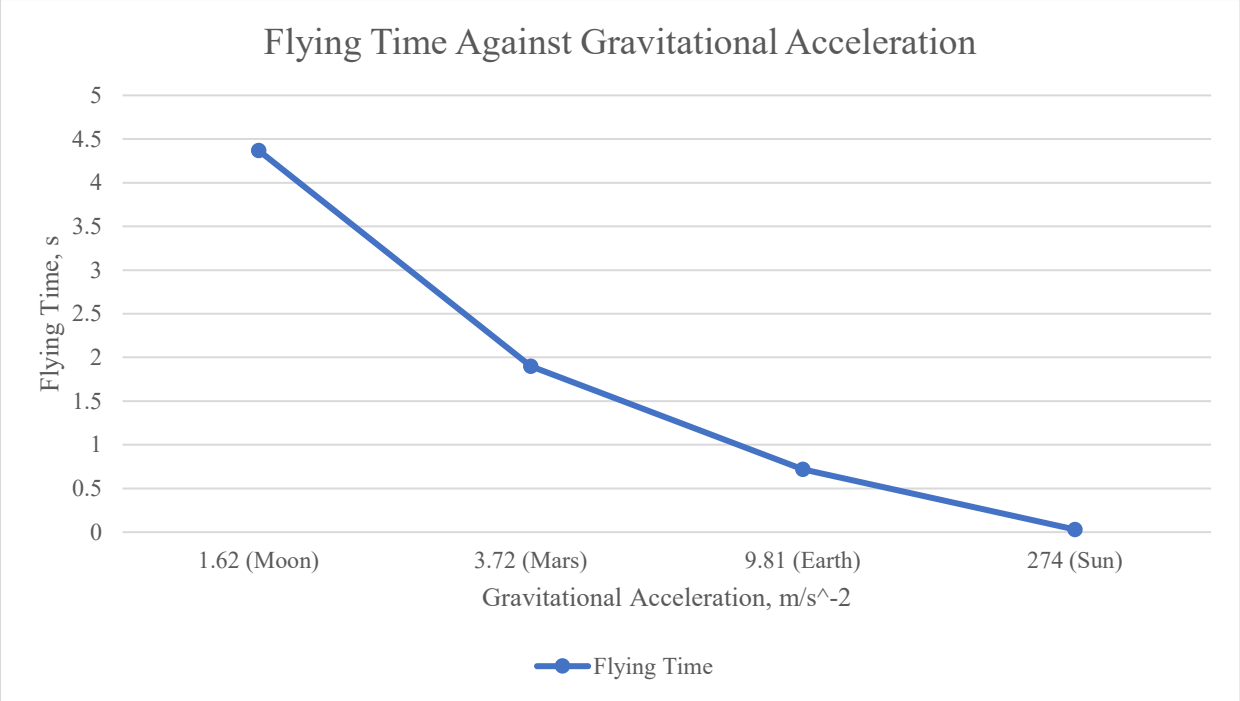


Figure 42: Flying Time Against Gravitational Acceleration Graph

Different planets or location have different gravitational acceleration. Hence, the projectile motion of an object will also be different if we throw the object at the particular planet. In this simulation program, there are 4 different environment or planet with different gravitational acceleration that is prepared for the user to run simulation. They include Earth, Moon, Sun and Mars. Sun has the highest gravity of $274ms^{-2}$ following by Earth $9.81ms^{-2}$, Mars $3.72ms^{-2}$ and lastly Moon $1.62ms^{-2}$. According to table 6, figure 41 and 42, the higher the gravitational acceleration, the lower the maximum height, horizontal displacement and flying time. This is because higher gravitational acceleration also means that higher force that is impeding the object to project higher from the ground. The force that is pulling the object back down to ground is higher. This directly reduces the maximum height and the flying time since that the object will reach ground faster if the gravitational acceleration is high. Then when the flying time is low the horizontal displacement will also be low because the object will stop moving faster.

Discussion

After going through simulation with different conditions and parameters the whole concept of projectile motion is understood. The initial velocity will be divided into 2 parts which is vertical and horizontal component. How much portion each of them takes mainly depends on the projection angle. The closer the angle to 90 degrees, the bigger the portion vertical component takes and vice versa for the horizontal component. The higher the velocity in the vertical component the higher it can travel.

However, this is not the case for horizontal component because the distance travelled horizontally mainly depends on the flying time. Even if most the initial velocity is given to the horizontal component at closer 0 degree, but the object still can't travel too far because it will hit the ground very soon. Alternately where flying time is higher when angle is closer to 90 degrees, but the horizontal velocity is too low. Hence at 45 degree is where the best combination of flying time and the horizontal velocity where the force is divided equally to horizontal and vertical component. This enables the object to travel the furthest given that initial velocity and gravitational acceleration is constant.

Secondly, initial velocity also affects the maximum height, horizontal displacement and flying time given that angle and gravitational acceleration is constant. This is easier to understand because initial velocity is the force given to the object. Hence the higher the force, the higher the maximum height, horizontal displacement and flying time.

Thirdly, gravitational acceleration is also one of the major things that is affecting the projectile motion. This is normally not used as a changing variable because we all stay on Earth and the gravitational acceleration will always be constant. However, this is not the case when we are at a different location environment and planet. Hence, this program also provides the choice for the user to simulate projectile motion on other planet by using their gravitational acceleration in the calculations. After going through the simulation and analysis, it is found that the higher the gravitational acceleration, the lower the maximum height, horizontal displacement and flying time. This is because gravitational acceleration is the force that is pulling the object back to the ground. Hence the higher the gravitational acceleration the lower the ability of the object to travel.

Nevertheless, in real life there are more factor that can affect a projectile motion such as air resistance, wind and so on which is assumed as 0 and ignored in the program and the simulation.

While for the mini game, the gravitational acceleration is fixed to 9.81m/s^2 and the range of the random target is also limited from 1 to 10 only. Further improvements can be made in the code in order to let the mini game more interesting such as provide a larger range of target and also at different location with different gravitational acceleration.

In this assignment there are a few innovative ideas that is implemented into the program. First and foremost, there is a series of codes to control the axis of the graph of the simulation. This is extremely important because originally, MatLab will adjust the axis to fit the max values on each axis which will make the user to hardly visualize their simulated projectile motion especially when they are trying out on different angle values. Hence in the program created, both X and Y axis are set to 5 percent more than the highest value between maximum height and horizontal displacement. In this case the user will be able to identify and visualize clearly on the projectile motion with different angles.

In addition, the next innovative idea is giving the user to choose the location that he wants to simulate the projectile motion. By providing this feature in the program, the user will be able to simulate projectile motion on different location which make the program more interesting.

Moreover, providing the mini game feature is also one of the innovating ideas that is implemented in the program. This feature is added by using the tab function to separate the simulation and mini game into different tab which prevents the program to be too complicated. This feature will make the program more interesting.

Furthermore, the table and keeping old simulation projectile motion plots feature is also one of the most user-friendly innovating ideas that is implemented into this program. This is because by providing a table that records all the results of simulations and keeping the plots that are done, the user can easily refer all the results and compare among the projectile motions. Besides that, it is also easier for the user to record or collect all the results. The user can easily clear all the old simulations by pressing the reset button.

During the process of creating the program, several challenges were encountered. Firstly, my personal understanding regarding the MatLab app designer is not too deep. Thus, extra research is done to increase my understandings about the syntax and so on. Fortunately, the way of creating the graphical user interface is through drag and drop hence it is easier to be understood and to be done. Moreover, I also having a hard time to choose between creating a

simulation or a mini game. Then I found out that I can make several tabs of user interfaces which enables me to make both simulation and mini game.

Besides that, I also encounter some difficulties on understanding all the equations that is used on calculating the results of the projectile motion. And it comes handier when I am trying to convert them into codes. At first when I type the formulas into the code, I have hard time to understand the code myself. Then I found out a way to simplify the formulas by assigning them part by part into a variable. This process helped me a lot.

On the other hand, I also faced some hard time in figuring out how to bring the variable that is created in a callback function to other callback function. At the end I found out that the best way to do so is to make the variables that I want to use it at different callback functions as global variable. This is also the same case for global function.

Lastly, I also faced some hard time while figuring out the algorithm and logic for creating the table feature. After going through many try and error, I finally managed to find the best logic for the feature.

Conclusion

In a nutshell, the program created in this assignment contains 2 parts which are the projectile motion simulation and the projectile motion mini game. Both the parts can run properly and carry out their tasks diligently without any error. The result of projectile motion is also obtained through the projectile motion simulator and all of them were accurate. The analysis on the results is also done and explained clearly in the report.

The 3 main factor that is affecting a projectile motion that is analyzed in this report are initial velocity, initial angle and gravitational acceleration. A projectile motion will have higher max height, horizontal displacement, and flying time when the initial velocity increases. While the projectile motion will have its maximum horizontal displacement when the initial angle is 45 degrees provided with constant initial velocity and gravitational acceleration. Horizontal displacement will decrease when the angle is further form it. But the time travel and the maximum height will getting closer to its peak when the angle is closer to 90 degrees. Lastly, higher gravitational acceleration will have negative effect on all max height, horizontal displacement and flying time.

Despite everything, the program created is not perfect and many improvements and features can be added into the program such as including air resistance in the calculations, adding more location choices, providing choice to do 3D projectile motion simulation with X, Y and Z axes, giving environment choices for the user in the mini game and provide a larger range of random targets.

In essence, during the process of completing this assignment, I had learned many new things about MatLab and realized that MatLab is a very powerful app and language which can perform a lot of functions and solve many problems. My logic in coding is also improved drastically. On top of that, I also have a clearer understanding on projectile motion.

MatLab OnRamp Certificate

Reference

Byju's. (n.d.). *Projectile Motion.* https://byjus.com/physics/projectile-motion/#what-is-projectile

toppr. (n.d.). *Projectile Motion.* *https://www.toppr.com/guides/physics/motion-in-a-plane/projectile-motion/*

MathWorks. (n.d.). *pause.* https://www.mathworks.com/help/matlab/ref/pause.html

MathWorks. (n.d.). *global.* https://www.mathworks.com/help/matlab/ref/global.html

MathWorks. (n.d.). *Property Attributes.* https://www.mathworks.com/help/matlab/matlab_oop/property-attributes.html