Personal Project Portfolio

Software Design and Development

Edward Webb

Contents

[Proposal 2](#_Toc483898836)

[Define the Problem 2](#_Toc483898837)

[Identification of the Problem 2](#_Toc483898838)

[Ideas 2](#_Toc483898839)

[Requirements 2](#_Toc483898840)

[Gantt Chart 4](#_Toc483898841)

[Understanding the Problem 5](#_Toc483898842)

[Storyboard 5](#_Toc483898843)

[Context Diagram 5](#_Toc483898844)

[Feasibility Study 6](#_Toc483898845)

[Social and Ethical Issues 6](#_Toc483898846)

[Malware 6](#_Toc483898847)

[Intellectual Property 6](#_Toc483898848)

[Quality 6](#_Toc483898849)

[Issues 7](#_Toc483898850)

[Ergonomics 7](#_Toc483898851)

[Inclusivity 7](#_Toc483898852)

[Privacy 7](#_Toc483898853)

[Plan and Design 7](#_Toc483898854)

[Data Flow Diagram 7](#_Toc483898855)

[System Flowchart 8](#_Toc483898856)

[Specifications 8](#_Toc483898857)

[Data Dictionary 8](#_Toc483898858)

[Algorithms 9](#_Toc483898859)

[Test Data 12](#_Toc483898860)

[IPO Chart 13](#_Toc483898861)

[Window Design 13](#_Toc483898862)

[Feedback for Projectile Motion Analyser 14](#_Toc483898863)

[Data Structures 14](#_Toc483898864)

[Files that are used by the program 14](#_Toc483898865)

[Implementation 14](#_Toc483898866)

[Tools Used 14](#_Toc483898867)

[User Documentation 14](#_Toc483898868)

[How to use Projectile Motion Analyser: 14](#_Toc483898869)

[Feedback 18](#_Toc483898870)

[Evaluation 19](#_Toc483898871)

[Testing 19](#_Toc483898872)

[Requirements 19](#_Toc483898873)

[Testing on school computers and other computers 20](#_Toc483898874)

[Maintenance 20](#_Toc483898875)

[Social and Ethical Issues Testing 21](#_Toc483898876)

[Authorship/Copyright 21](#_Toc483898877)

[Reliability 21](#_Toc483898878)

[Response to Problems 21](#_Toc483898879)

[Malware 21](#_Toc483898880)

[Plagiarism 21](#_Toc483898881)

[End User Agreement 22](#_Toc483898882)

# Proposal

This program will take several variables relating to projectile motion eg. Initial velocity, maximum height the projectile reaches, the range of the projectile, etc. and will determine the rest of the variables and show the path of the projectile on a graph. Developing the graph will involve measuring what height the projectile is at a point of the projectile’s motion and placing a dot on the graph. The graph could have multiple dots to create a curve. Excel could also be used as an alternative in creating the graph.

Advantages: Small project, is possible in Visual Basic 6

Disadvantages: Possibly too hard to do in Visual Basic 6 in creating the graph, and will require many functions to determine the rest of the variables.

# Define the Problem

## Identification of the Problem

The problem presented is that physics students require a program that can easily solve and visualise projectile motion problems. This would include calculating the initial velocity, its maximum height, its range, its horizontal and vertical velocity at any point of time, and its time. It must be able to input different variables and output the rest of the variables. A graph which visualises the projectile’s motion is also required to allow physics students to easily solve any projectile motion problem.

## Ideas

* The program could calculate the position of the ball at a certain point in its motion and pinpoint it on the graph. After creating many points, these points can be connected, creating a curve.
* Several variables will be listed with a box allowing the user to input a number into it
* If there is not sufficient information, a prompt can appear after the user enters the information saying that there is not sufficient information.
* After some information has been inputted by the user, the rest of the boxes can be filled by calculating from the user’s information and the graph can be shown.
* Excel can be used to create the curve for the graph, as Excel has a function which allows the line of a graph to be curved.
* The data required could be entered when the program is first opened, then when all necessary data has been entered, the program can then load the graph and enter all the variables into their boxes.
* A function to calculate the velocity at any point during the projectile’s motion can be added, which can be used after the graph has been loaded.

## Requirements

1. The program is to run on Microsoft Windows machines and must fit a 1920x1080 resolution as this is the resolution that school computer displays have.
2. The program must not crash when the program is put in use
3. The program must run on a computer with 4GB RAM.
4. The program must be able to visualize the projectile motion specified by the user and output variables that can be calculated from the information inputted into the program.
5. If the program cannot calculate any information from the data given, the program will output an error displaying that the information given is not sufficient.
6. The user should be able to use the program again after the user has inputted information for a projectile motion problem without having to reopen the program.
7. The program must give an error when non-numeric characters have been entered.

## Gantt Chart

# Understanding the Problem

### Storyboard

## Context Diagram



## Feasibility Study

This project is designed for physicists and physics students, and its purpose is to solve projectile motion problems. Many physics students and physicists have to solve projectile motion problems, and this program can help visualise and aid with solving these problems.

The boundaries of this project include not being able to pinpoint a specific point of the projectile’s motion on the graph, as Visual Basic 6 does not have the capability to create this. The graph will also not be a smooth curve, as Visual Basic 6 does not have a proper graphing feature. However, dots can be created on the graph to attempt to create a curve.

The benefits of creating this project include being able to solve projectile motions much more easily, and allowing physics students to check their answers easily if they need to. The program can also provide more accurate answers for projectile motion problems than what physicists can calculate by hand, and can create a visual aid for students by showing what the projectile motion should look like on a graph.

This project is possible in Visual Basic 6 as points can be placed on a drawn graph to create a curve on the graph, which visualises the projectile’s motion. There is also a limited amount of possibilities for projectile problems, and so all projectile motion problems can be solved using a program. Although Visual Basic 6 does not have a graphing function, it is still possible using Excel to create a graph.

One person is needed to actually create and maintain the program, and it will most likely take around 1-2 months to create. Visual Basic 6 must be used to create the program as the programmer who will develop it can program efficiently in Visual Basic 6. A computer with Windows is needed as Visual Basic 6 can only run on Windows.

If the project is not completed, the graph can be left out to allow most of the functionality of the program to operate. As the graph is the most complex part of the program and only helps achieve the purpose of the program, it can be left out if it has to. This will allow students to still easily use the program, and the project to be completed in less time.

## Social and Ethical Issues

### Malware

Malware has been considered as a possible issue for the software, and so the software after completion will be scanned by antivirus software and uploaded to virustotal.com, a Google-developed website that uses many antivirus software to ensure that the software is free of malware and viruses.

### Intellectual Property

The program is owned by the Department of Education and Communities of New South Wales, as it is developed due to educational purposes. Therefore, the program’s intellectual property is protected by copyright laws as no license has been given to it.

### Quality

The program will be developed to the highest quality possible in the time frame provided. The program will be tested with school computers to ensure that it meets hardware requirements, which are that it must run on Windows with a 1920x1080 resolution. Error checking will be built into the code, and all errors that are found during the development process will be fixed and solved.

### Issues

After development, users who use the program may report any bugs and issues with the developer, and they will be fixed.

### Ergonomics

User interface elements will be used to ensure consistency with other software, which makes the program easy to use. As the program is targeted towards physicists and physics students, the program will be developed with the demographics of this target market in mind to ensure that the target market find the program easy to use. Industry standards will be used to do this. The program will also be tested by physics students to make sure the software meets the needs of physics students and physicists.

### Inclusivity

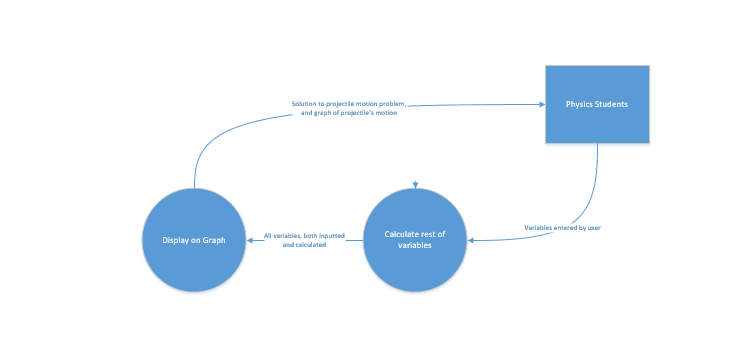
The graph in the program will follow normal science standards, with the dependent variable on the y axis and the independent variable on the x axis. All units used for variables entered and outputted will follow SI units as well to give physics students units they expect. The program will have no cost, therefore economic issues will not be considered. No offensive language will be used in the program, and a variety of testers will be used including people from both genders to address the possible inclusivity issue of gender. To address disability issues, shortcuts like TAB and ENTER can be used to operate the program as well.

### Privacy

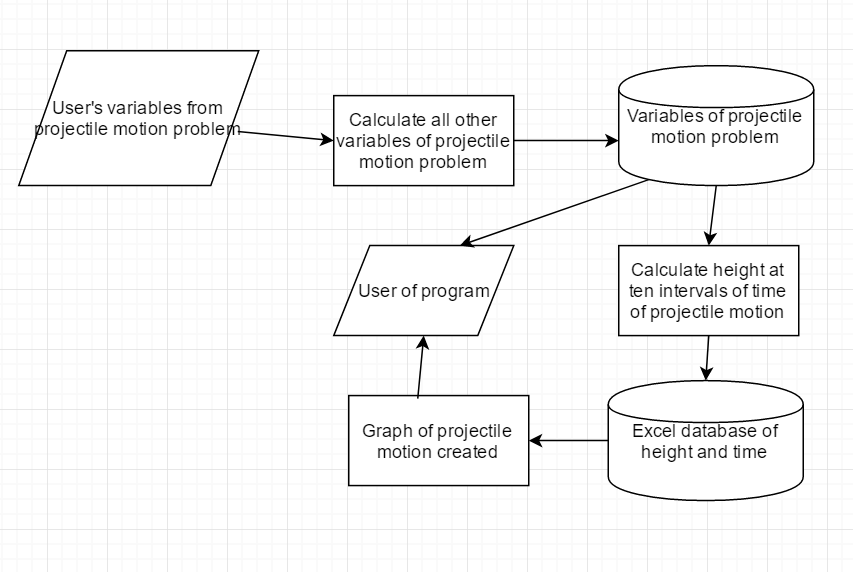
To address a possible privacy issue, no information will be kept in a database, and the program will not save any information entered by the user.

# Plan and Design

## Data Flow Diagram



## System Flowchart



## Specifications

* Windows XP or later
* 1GB RAM or more
* 1920X1080 resolution
* Any processor will run

## Data Dictionary

|  |  |
| --- | --- |
| Variable Name | Description |
| time | The amount of time the projectile travelled before landing. |
| initVelo | The projectile’s initial velocity. |
| angle | The angle the projectile was launched from the horizontal. |
| xVelocity | The x (horizontal) component of the projectile’s initial velocity. |
| yVelocity | The y (vertical) component of the projectile’s initial velocity. |
| timeSpecific | The time the projectile took from launch to reach maximum height. |
| timeSpecific2 | The time the projectile took to land from maximum height. |
| maxHeight | The projectile’s maximum height it reached. |
| range | The horizontal distance the projectile travelled from its launch to its landing point. |
| divisor | A temporary variable used to find the angle of the projectile for the 2nd Algorithm. |
| height | The projectile’s initial height from the ground. (Always given by the user) |
| heightEnd | The projectile’s landing height from the ground. (Always given by the user) |
| heightDiff | The difference in height between the landing height and the initial height (heightEnd – height = heightDiff) |
| angleR | The angle from the horizontal the projectile was launched at converted to radians. |
| isNumeric | Boolean used to tell an if statement whether the input is numeric or not (for error checking) |
| xlApp | Variable used to contain the Excel application instance |
| xlWkb | Variable used to contain the Excel workbook instance |
| xlSht | Variable used to contain a specific sheet instance within the Excel workbook |
| timeInterval | Total time of projectile divided by 10. Used to plot points on the graph. |
| Times | Array used to contain the time part of each plot point on the graph |
| Heights | Array used to contain the height part of each plot point on the graph |
| I | Variable used for incrementing in for loops |

## Algorithms

Algorithm 1: Given Initial Velocity and Angle (Basic algorithm of the 5 algorithms that will be used in the program. All algorithms that calculate the variables are based off this one)

BEGIN

angleRadians = (angle / 180) \* 3.14 // Convert angle into radians

xInitialVelocity = initialVelocity \* Math.Cos(angleRadians)

yInitialVelocity = initialVelocity \* Math.Sin(angleRadians)

maxHeight = (yInitialVelocity^2 / (2 \* 9.8)) + height // v^2 = u^2 + 2as (projectile motion equation)

timeTillMaxHeight = yInitialVelocity / accel // v = u + at

timeAfterMaxHeight = sqrt(maximumHeight / (0.5 \* 9.8)) // s = ut + 0.5at^2

range = xInitialVelocity \* time

Output(time, range, initialVelocity, xInitialVelocity, yInitialVelocity, timeTillMaxHeight, angle, maxHeight) // Output these variables to the user

excelGraph(time, yInitialVelocity, height) // Graph function needs variables

END

Algorithm 2: Given initial velocity and time (2nd algorithm of the 5 algorithms that will be used in the program. This algorithm is based off Algorithm 1, but some equations have been rearranged and modified)

BEGIN

yInitialVelocity = (heightDiff – (0.5 \* accel \* time^2))/time // Rearrange the projectile motion equation s = ut + 0.5at^2 to (s – 0.5at^2)/t. Height is negative as height is below the initial velocity.

Divisor = yInitialVelocity / initVelo // inverse sine of yInitialVelocity/initVelo will find angle

Angle = Math.Asin(divisor) // Uses inverse sine function to find angle

xInitialVelocity = initVelo \* Math.Cos(angle) // Uses cosine function to find xInitialVelocity

range = xInitialVelocity \* time // Standard projectile motion equation

maxHeight = (yInitialVelocity^2 / (2 \* accel)) + height // Same as Algorithm 1

timeTillMaxHeight = yInitialVelocity / accel

Output(time, range, initialVelocity, xInitialVelocity, yInitialVelocity, timeTillMaxHeight, angle, maxHeight) // Output variables to user

excelGraph(time, yInitialVelocity, height) // Graph function needs variables

END

Algorithm 3: Given range and time (3rd algorithm of the 5 algorithms that will be used in the program. This algorithm is similarly based off Algorithm 1, and similarly equations have been rearranged and modified)

BEGIN

yInitialVelocity = (heightDiff – (0.5 \* -accel \* time^2) / time // Same as Algorithm 2

xInitialVelocity = range / time // Standard projectile motion equation

angle = Math.Atan(yInitialVelocity / xInitialVelocity) // Using inverse tan function to find angle

initVelo = ((xInitialVelocity^2) + (yVelocity^2) ^ 0.5 // Using Pythagoras theorem to find initVelo

maxHeight = (yInitialVelocity^2 / (2\*accel)) + height // Same as Algorithm 1

timeTillMaxHeight = yInitialVelocity / accel // Same as Algorithm 2

Output(time, range, initialVelocity, xInitialVelocity, yInitialVelocity, timeTillMaxHeight, angle, maxHeight) // Output variables to user

excelGraph(time, yInitialVelocity, height) // Graph function needs variables

END

Algorithm 4: Given range and maxHeight (4th algorithm of the 5 algorithms that will be used in the program. This algorithm is also based off Algorithm 1, and some code has been copied or modified from Algorithm 1)

BEGIN

yInitialVelocity = Math.Sqr(2 \* accel \* maxHeight) // v^2 = u^2 + 2as rearranged to give ‘u’. v = 0 as v is 0 at maxHeight.

timeTillMaxHeight = yInitialVelocity / accel // Same as Algorithm 2

timeAfterMaxHeight = Math.Sqr((heightDiff + maxheight) / 0.5 \* accel) // s = ut + 0.5at^2 but u = 0 at maxHeight. heightDiff used to take into account heightEnd (most questions giving maxheight will set heightEnd to 0 though.)

time = timeTillMaxheight + timeAfterMaxHeight

xInitialVelocity = range / time // Same as Algorithm 2

angle = Math.Atan(yInitialVelocity / xInitialVelocity) // inverse tan function used to calculate angle

initVelo = (xInitialVelocity^2 + yInitialVelocity^2)^0.5 // Pythagoras Theorem

Output(time, range, initialVelocity, xInitialVelocity, yInitialVelocity, timeTillMaxheight, angle, maxHeight) // Output variables to user

excelGraph (time, yinitialVelocity, height) // Graph function needs variables

END

Algorithm 5: given range and angle (5th algorithm of the 5 algorithms that will be used in the program. This algorithm is the hardest to construct, and HSC questions rarely give problems that involve using this algorithm. The last time a HSC paper gave a question using this algorithm was in 2012.)

BEGIN

timeTemp = (heightDiff – ((range \* Math.Sin(angle)) / Math.Cos(angle))) / (0.5 \* -accel) // range = xInitialVelocity \* time substituted into s = ut + 0.5at^2. This is one of the most complex calculations in the program. timeTemp is a temporary variable that will be used to calculate time. It will be put in a separate variable as putting it in the same line would make the code harder to read.

time = Math.Sqr(timeTemp)

initVelo = range / (time \* Math.Cos(angle)) // time substituted into range = xInitialVelocity \* time

xInitialVelocity = initVelo \* Math.Cos(angle)

yInitialVelocity = initVelo \* Math.Sin(angle)

maxHeight = (yVelocity^2 / (2 \* accel)) + height

timeTillMaxHeight = yVelocity / accel

Output(time, range, initialVelocity, xInitialVelocity, yInitialVelocity, timeTillMaxheight, angle, maxHeight) // Output variables to user

excelGraph (time, yinitialVelocity, height) // Graph function needs variables

END

Algorithm 6: excelGraph Function used to display the graph and set plot points on it to construct a curved line on the graph.

timeInterval = time / 10 // Needed to create the graph accurately and to plot points on it

For i = 1 to 10:

Times[i – 1] = timeInterval \* i

Heights[i – 1] = (yInitialVelocity \* times[i – 1]) + (0.5 \* -9.8 \* times[i – 1]^2) // s = ut + 0.5at^2

End For

AddToExcelWorksheet(Times)

AddToExcelWorksheet(Heights)

excelChart.Type(lineChart)

excelChart.Smooth = True

excelChart.Show()

END

# Test Data

The following projectile motion problems will be used to test the program:

1. A projectile is fired at 30° to the horizontal from the top of a cliff 200 m high. Its initial speed is 49 ms-1.
2. A projectile is fired from the top of a 120m high cliff at 25 ms-1. It lands on the ground 6.4s after firing
3. A cannon is at the top of a 60 m high cliff firing at a castle on top of an adjacent cliff 110m high. The cannon and castle are 200m apart horizontally.
4. A cannon ball is fired at 50ms-1 from the top of a 200m high cliff so that maximum range is achieved.
5. A boy throws a rock at 15ms-1 from the top of a 75m high cliff. The rock lands in the water at the bottom of the cliff 4.0s later.
6. A projectile is thrown up from the top of a 60m high cliff. It rises to a maximum height of 44.1 m above the cliff top. It hits the ground 76m out from the base of the cliff.
7. A ball is thrown out from the edge of a 40m high cliff with a velocity of 35.1ms-1 at 30° to the horizontal.
8. A cannon ball is fired at 40° to the horizontal from the top of a 218.7m cliff and hits a target 300m from the base of the cliff.
9. A cannon fires from the top of a 150m high cliff at a castle 300m from the base of the cliff. The ball hits the castle 15 s later.

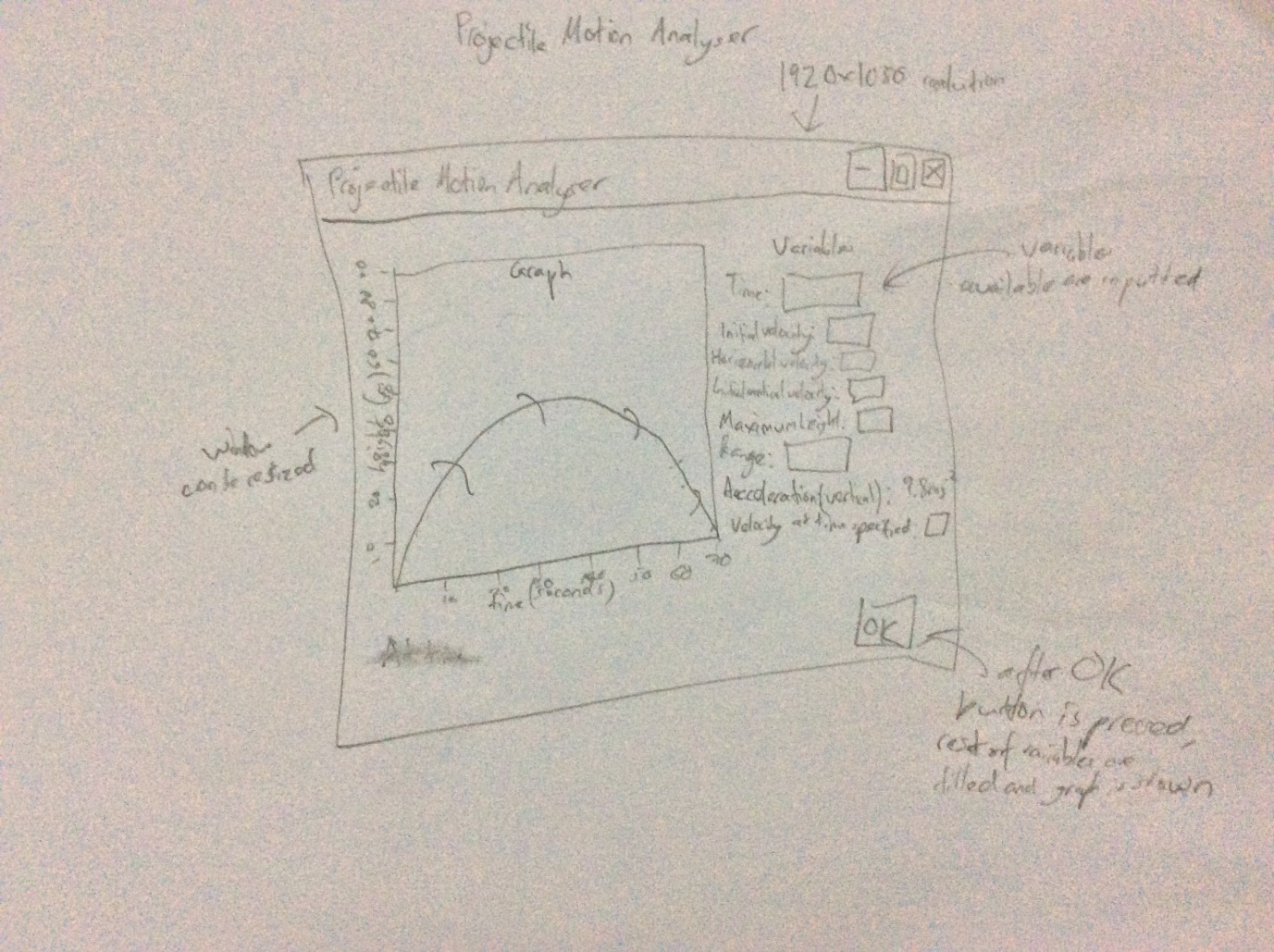
The program must match the following output data:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Problem | Initial Velocity | Initial horizontal velocity | Initial vertical velocity | Range | Max Height | Time to reach max height | Time of flight |
| 1 | 49ms-1, 30 deg | 42.4 | 24.5 | 396.9 | 30.6 | 2.5 | 9.36 |
| 2 | 25 ms-1, 30.3 deg | 21.6 | 12.6 | 138 | 8.1 | 1.3 | 6.4 |
| 3 | 101 ms-1, 84.3 deg | 10 | 100.5 | 200 | 510 | 10.3 | 20 |
| 4 | 50 ms-1, 45 deg | 35.4 | 35.4 | 387.2 | 63.9 | 3.6 | 10.94 |
| 5 | 15 ms-1, 13.1 deg | 14.6 | 3.4 | 58.4 | 0.6 | 0.34 | 4.0 |
| 6 | 31 ms-1, 71.2 deg | 10 | 29.4 | 76 | 44.1 | 3.0 | 7.6 |
| 7 | 35.1 ms-1, 30 deg | 30.4 | 17.6 | 157.3 | 15.8 | 1.5 | 5.2 |
| 8 | 40 ms-1, 40 deg | 30.64 | 25.7 | 300 | 33.6 | 2.62 | 9.8 |
| 9 | 66.6 ms-1, 14.3 deg | 20 | 63.5 | 300 | 205.7 | 6.5 | 15 |

## IPO Chart

|  |  |  |
| --- | --- | --- |
| Input | Process | Output |
| Variables of projectile motion problem (eg. Time, initial velocity, range) | 1. Identify which variables were given by the user 2. Use the algorithm that can solve the projectile motion problem using the variables given. 3. Use projectile motion equations like v = u + at to calculate all variables, including time, initial velocity, range, maximum height, and time until maximum height is reached. 4. Display all variables of the projectile motion problem to the user | All projectile motion variables related to the problem given. |

## Window Design



## Feedback for Projectile Motion Analyser

* Instead of drawing graph using lines and dots to create a curve, Excel can be used to create a curve for the graph.

## Data Structures

No data will be stored permanently by the program, for privacy reasons. Arrays will only be used by the graphing function of the program, as the times and heights of each interval of the projectile motion are required and so arrays must be used.

## Files that are used by the program

1. Projectile\_Motion.exe – the application itself. This is required to run the program
2. Required\_Excel\_File.xlsx – a required Microsoft Excel file. This is used to create the graph.

Project Files: (These are not required to run the program. Visual Basic 6 combines these files into an executable.)

1. Projectile\_Motion.frm – the main form. Only 1 form was required to create the program.
2. Projectile\_Motion.vbp – the project file.
3. Projectile\_Motion.vbw – config file created by Visual Basic 6.

# Implementation

## Tools Used

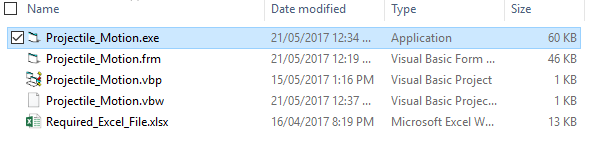
During implementation, GitHub was used as backup of the program and to record changes to the program. This is available at <https://github.com/spider93287/SDD-Project>

The commit log is included in the project folder under “Commit-log.txt”

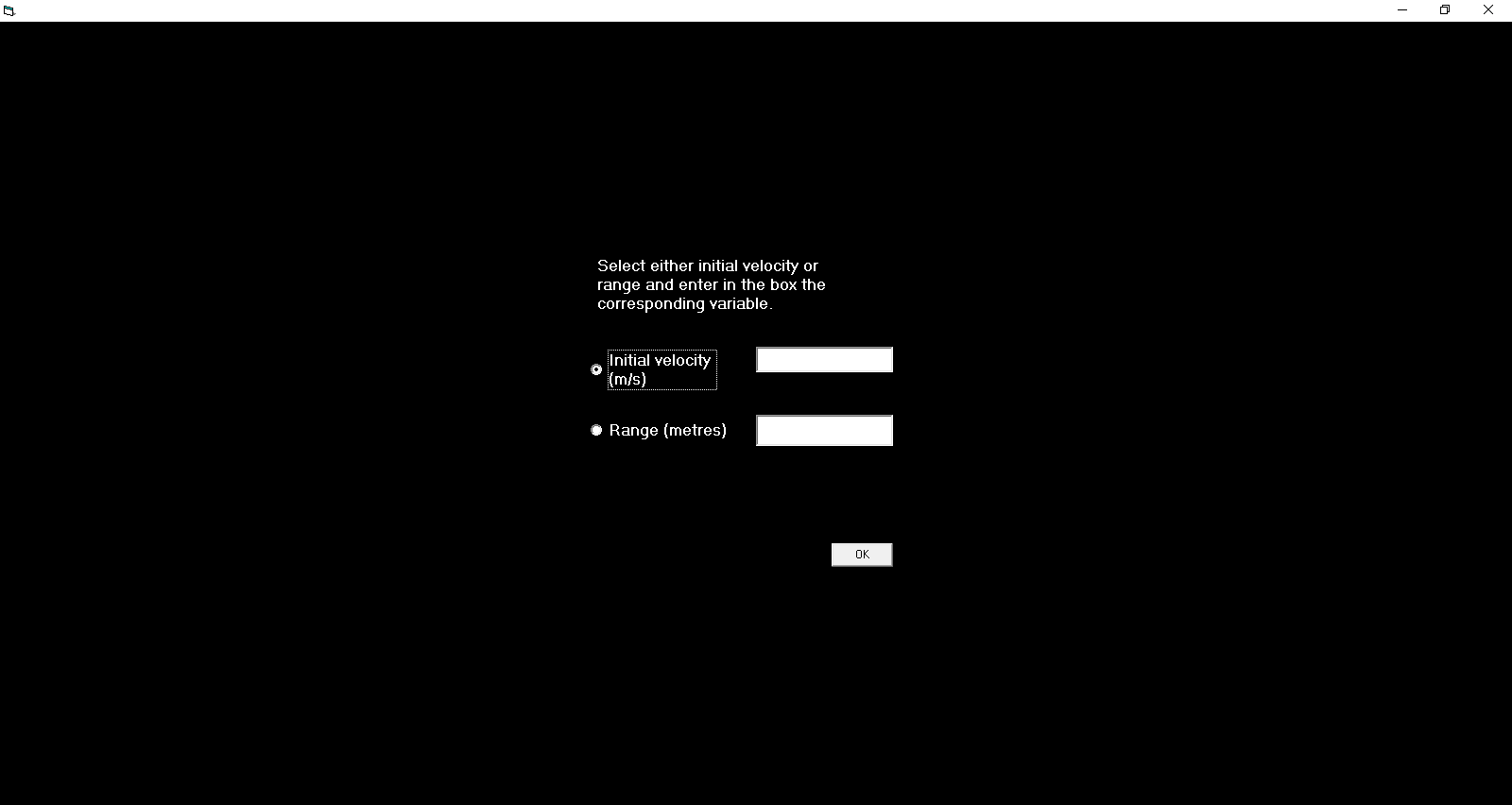
## User Documentation

## How to use Projectile Motion Analyser:

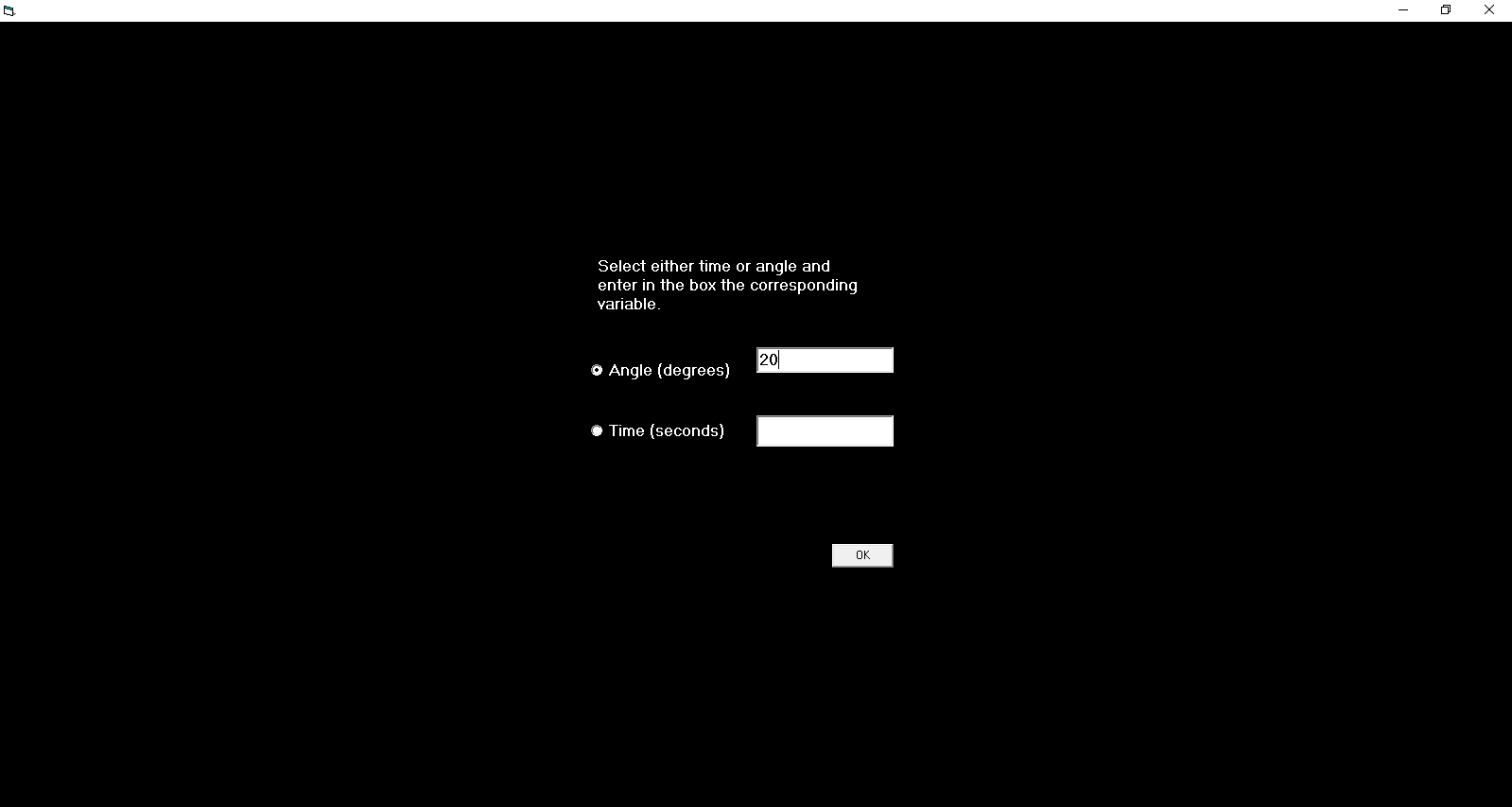
Step 1: Double click Projectile\_Motion.exe



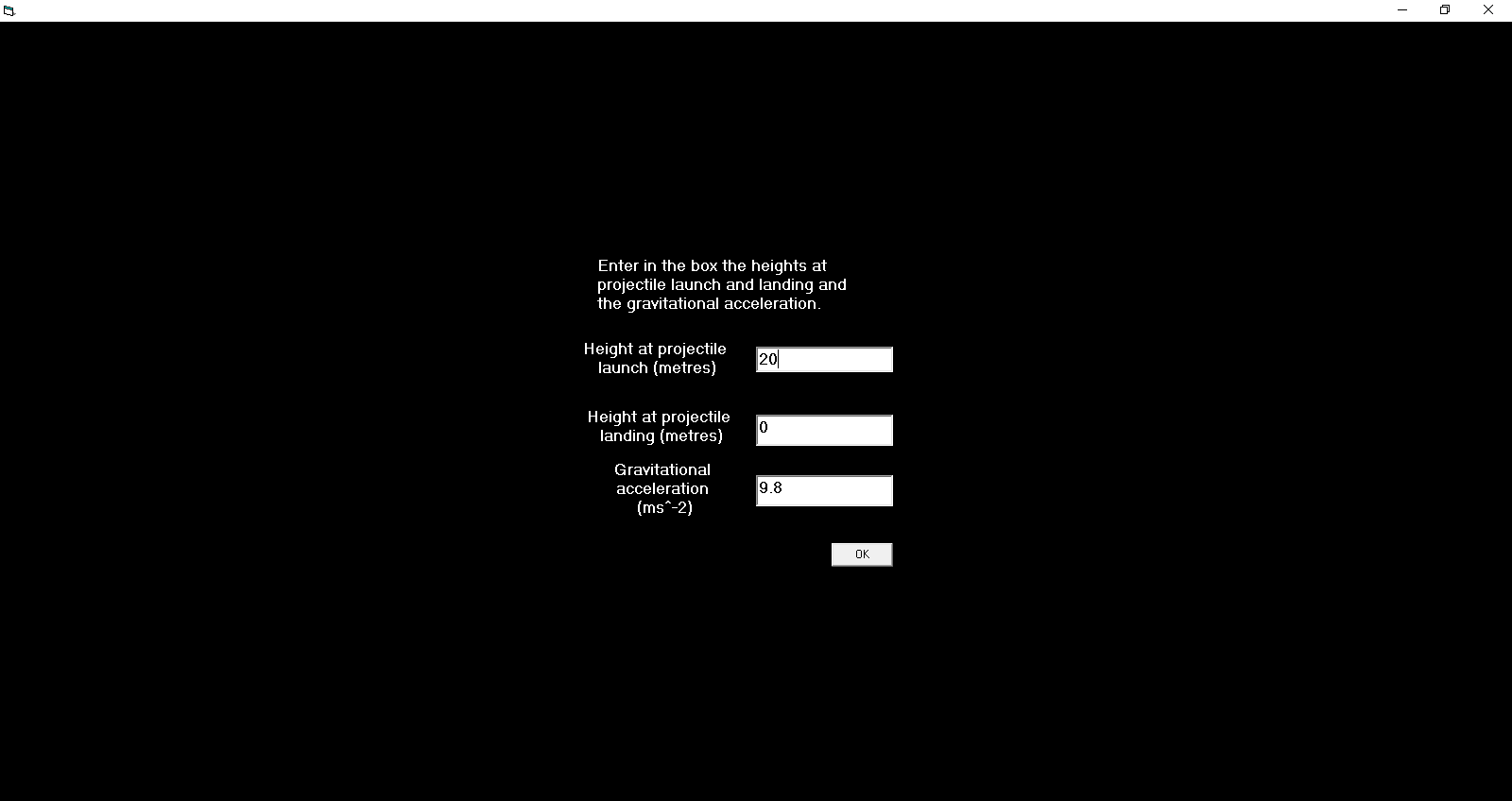
Step 2: Choose Initial velocity or Range and enter the corresponding variable, then click OK.



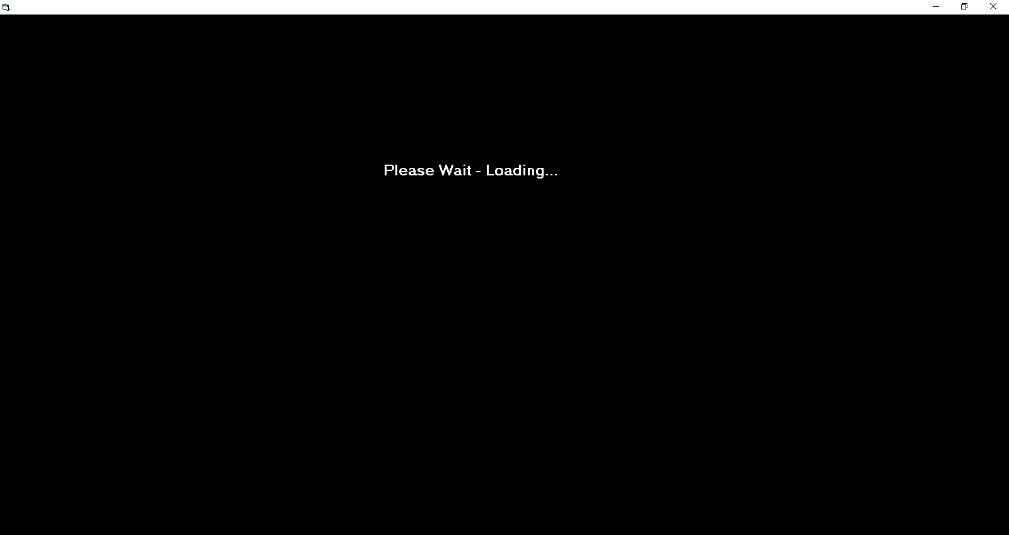
Step 3: Select Angle, Time (or Maximum Height if it is an option) and enter the corresponding variable, then click OK.

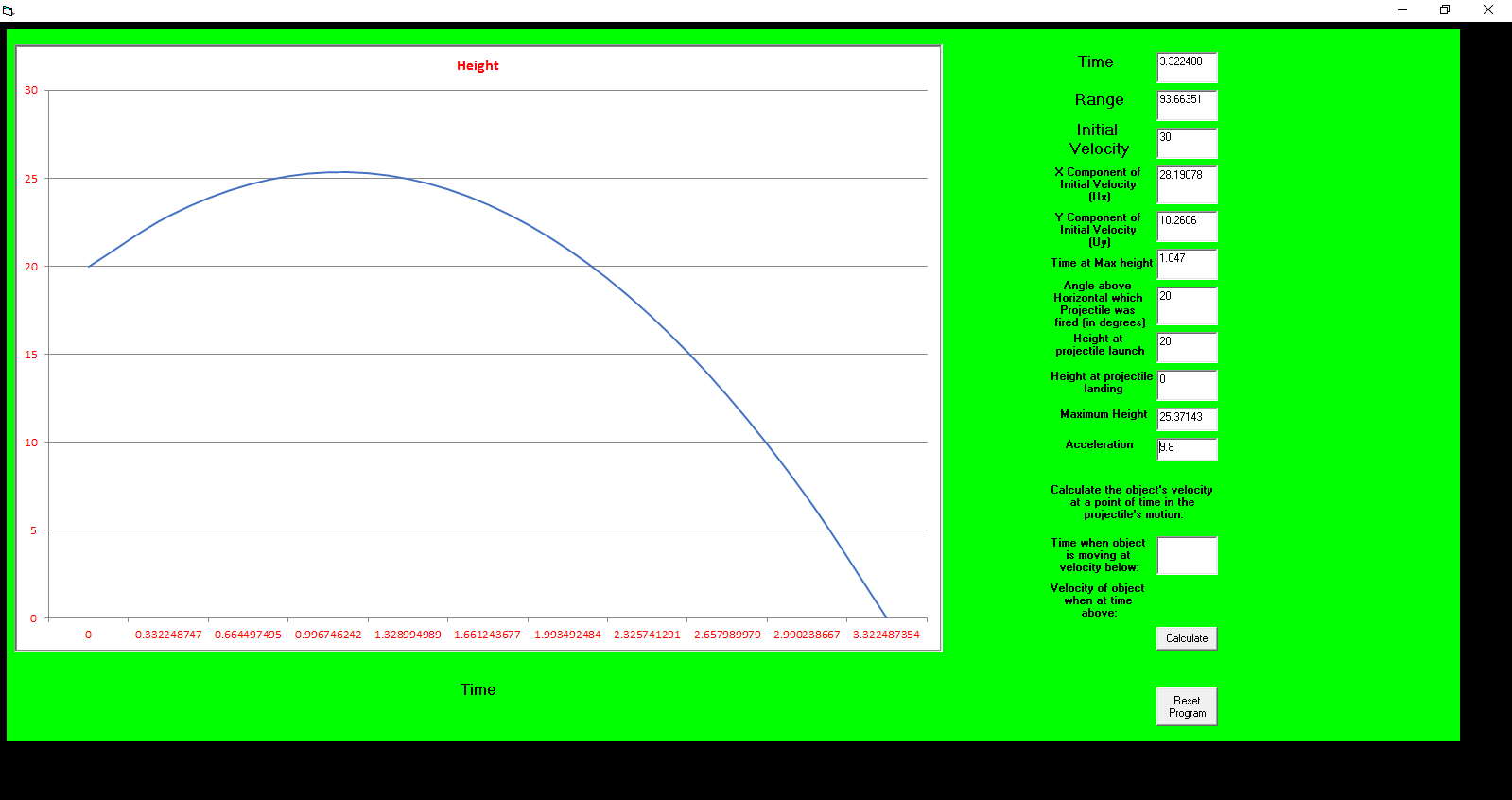


Step 4: Enter the corresponding variable, then click OK.

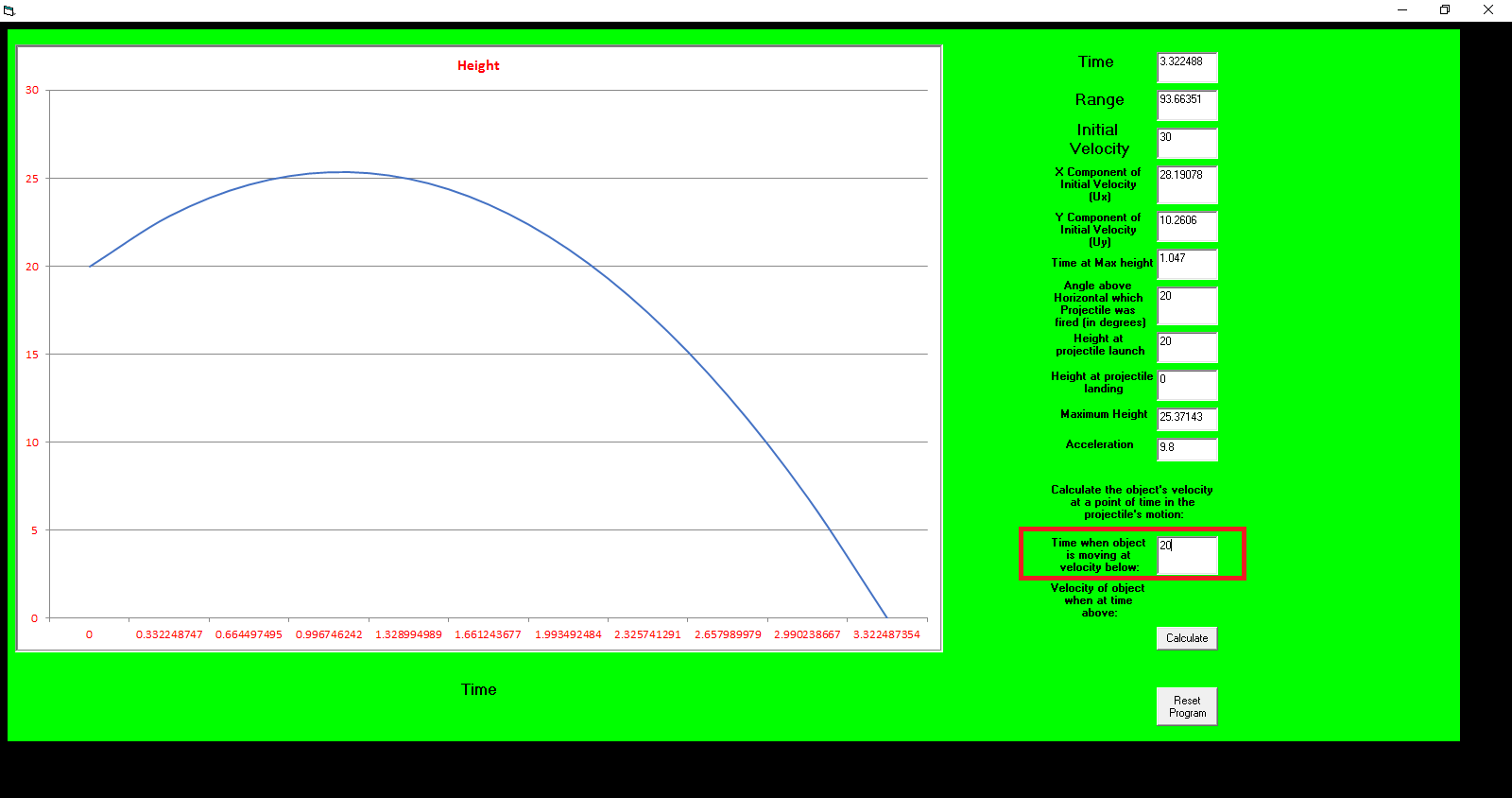


Step 5: Wait for the program to finish loading

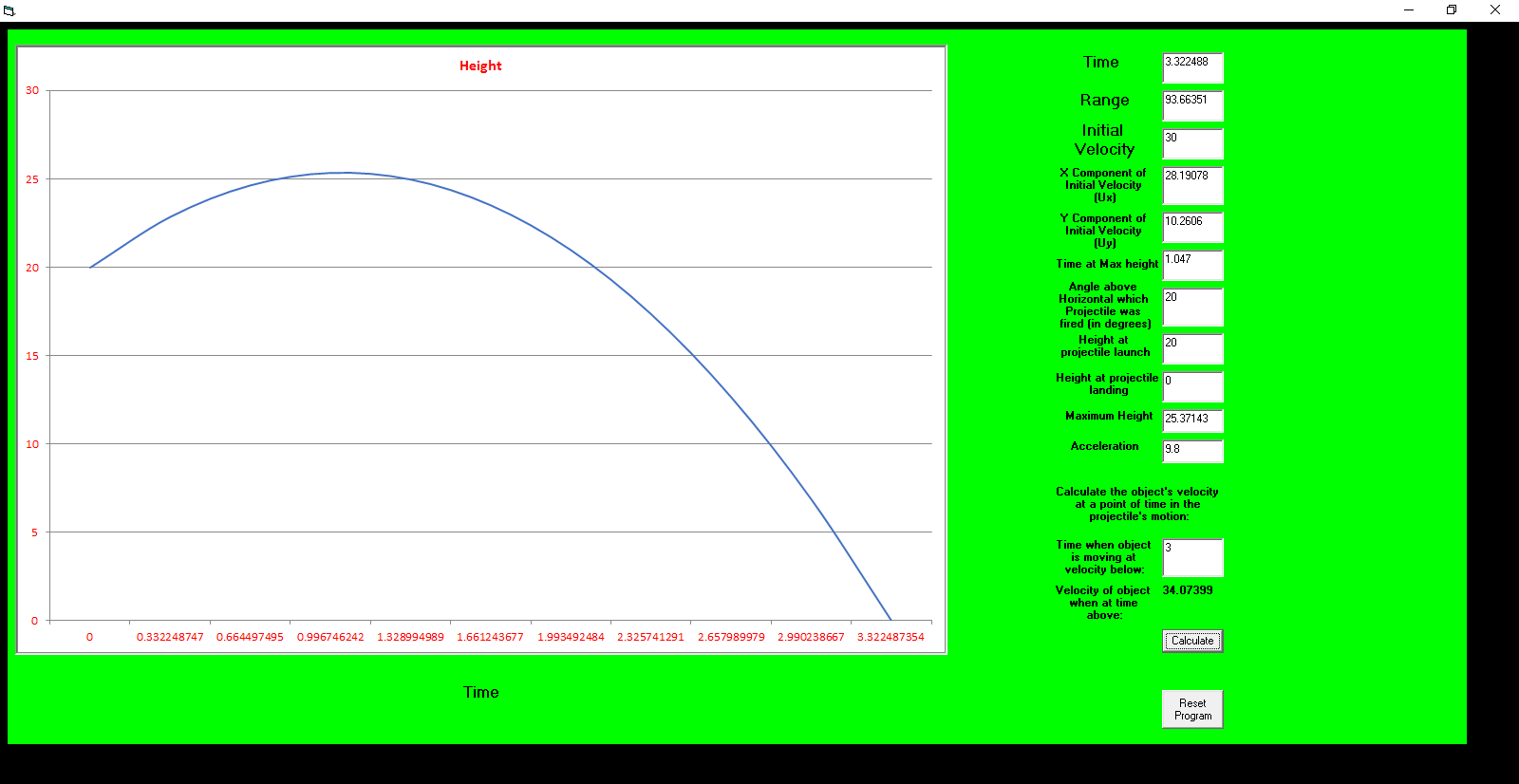


Step 6: The program will load a graph and show all the other variables for the projectile motion. 

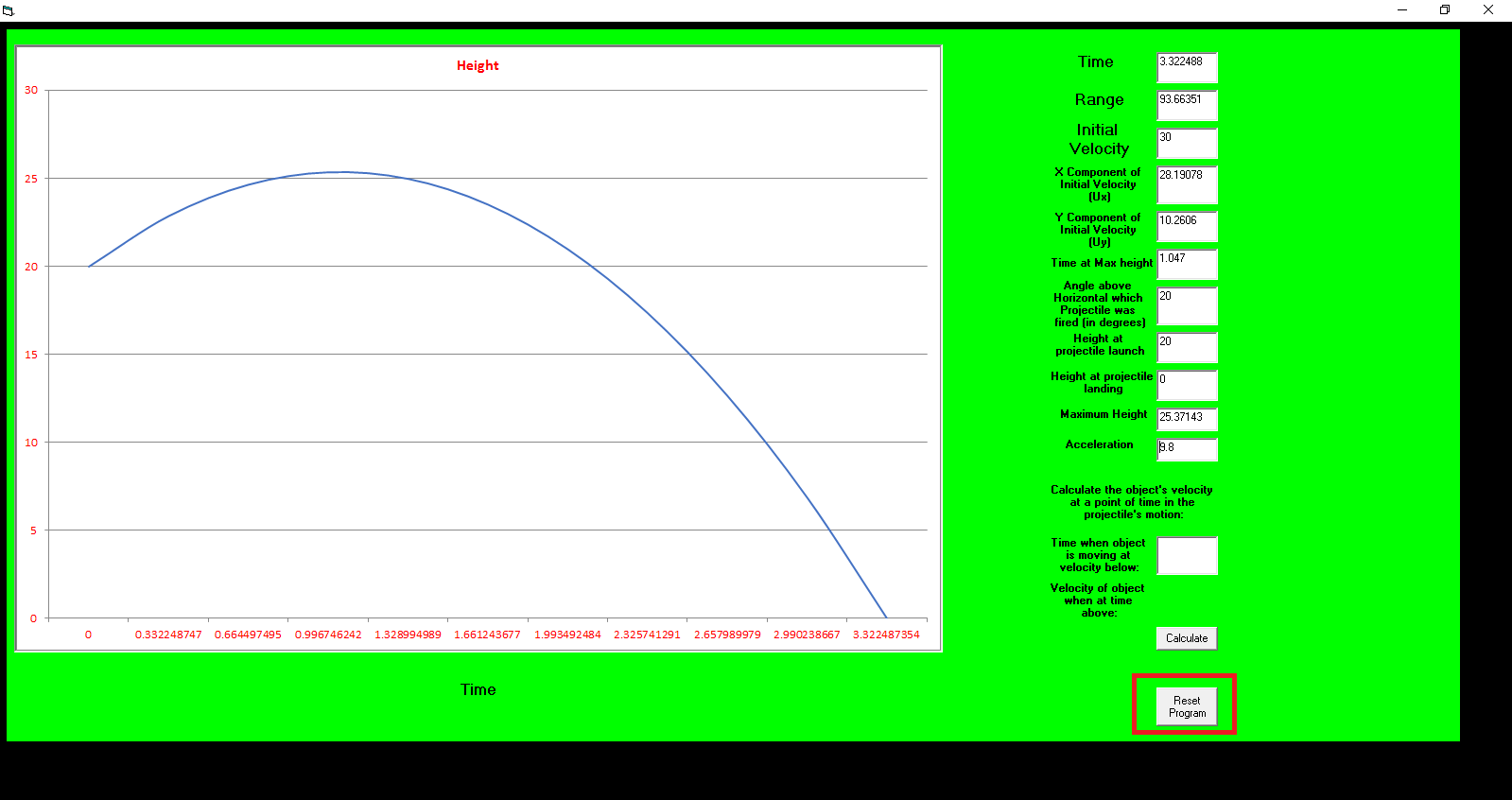
Step 7: You can now calculate the velocity of the projectile at any point in the graph. Enter the point of time in the box here. Then click the Calculate button.



Step 8: The program will show this velocity below.



Step 9: If you would like to reset the program and enter another variable, click “Reset Program”



## Feedback

* Previously in the graph, blue lines were shown and an error would be given if the acceleration was made equal to zero. This was given as feedback to the program, and these errors were fixed.
* Previously the textboxes used that outputted variables were user editable. This confused users and some users thought that the program could be reset with new variables by editing these boxes and clicking “Calculate”, which was actually meant for calculating the variable at any point in the projectile’s motion. These textboxes were edited so they matched the colour of the background and were no longer user editable.
* The program also previously accepted negative numbers. Feedback was received where the program should be edited so negative numbers will not be accepted.
* The program had a confusing UI previously, which was given as feedback to the program. This was resolved by completely revamping the UI so it looked more appealing to the user.
* The program, in the input screens, would not click OK when the key “ENTER” was pressed. One user complained that this is an issue, as most users will instinctively press ENTER instead of clicking OK with a mouse. A function was implemented so the program clicks OK when ENTER is pressed.
* One bug that was reported was that the program continued to run as a process in Windows Task Manager. This bug has not reappeared since, and the cause of the bug is unknown. The program has been scanned by multiple antivirus scanners to ensure that the program does not contain a virus. This may have been caused by Visual Basic 6 itself, and the program’s code most likely did not cause this bug.
* One user reported that the colour scheme made the program look less appealing.

# Evaluation

## Testing

1. Projectile Problems from Surfing Space Book:

Gives correct data for every problem.

1. Projectile Problems from HSC

**EXCEL HSC Space Section: Question 3**

The program returns “Please enter more variables” due to it finding the angle is 0. However, the initial angle is actually 0 in the question.

Bug has been fixed, now program will detect whether a box is blank instead of whether it contains a 0.

The program gives the correct answer to this question.

**2008 HSC Question 3**

The program gives the correct answer to this question.

**2010 HSC Question 22**

The program supports a default acceleration value of 9.8 ms-2 for gravity, as this is the acceleration value of gravity on Earth. However, this question uses the acceleration value on the Moon, which is 1.6ms-2, and so currently the program cannot solve this question.

An acceleration option has been added, with the default value set at 9.8, however the user can change this value.

Therefore, the program can now solve this question and gives the correct answer to this question.

**2011 HSC Question 15**

The program gives the correct answer to this question.

**2012 HSC Question 27**

The program DOES NOT give the correct answer to this question. In the program’s code, “300” was written instead of the variable “range”, making the algorithm give a wrong answer. This was fixed, and now the program gives the correct answer to this question.

**GHS Trials Q3 2006**

The program gives the correct answer to this question.

**Independent Trials Q16 2006**

The program gives the correct answer to this question.

**NEAP Trials Q16 2006**

The program gives the correct answer to this question.

**Catholic School Trials Q17 2006**

The program gives the correct angle but in radians, not in degrees. This bug was fixed, and the program now gives the correct answer.

## Requirements

1. The program is to run on Microsoft Windows machines and must fit a 1920x1080 resolution as this is the resolution that school computer displays have.

This requirement is met, as the program was tested on school computers and does work on a 1920x1080 resolution.

1. The program must not crash when the program is put in use

The program does not crash when the program is put in use, and so this requirement is also met.

1. The program must run on a computer with 4GB RAM.

The program runs perfectly fine with no performance issues on a computer with 4GB RAM.

1. The program must be able to visualize the projectile motion specified by the user and output variables that can be calculated from the information inputted into the program.

The program accurately calculates any projectile motion specified by the user.

1. If the program cannot calculate any information from the data given, the program will output an error displaying that the information given is not sufficient.

The program includes many error checks to ensure the input given is sufficient.

1. The user should be able to use the program again after the user has inputted information for a projectile motion problem without having to reopen the program.

A “Reset Program” function was added to ensure the program can be used again without the user having to reopen the program.

1. The program must give an error when non-numeric characters have been entered.

The program includes error checking to ensure no non-numeric characters are entered which may cause the program to crash.

## Testing on school computers and other computers

* The program was tested on school network computers. The program does work fine, but the graph fails to load. This is currently not fixed yet.
* The program was also tested on other computers that ran Windows 7 and Windows 10. Due to Visual Basic 6 limitations, a DLL file had to be placed in C:\Windows\syswow64 in order for the program to run, but otherwise the program ran fine on all these computers
* Other users tested the program, and could operate the program quite easily. Some UI elements were changed due to confusion in using the program, but currently the program can be used by people with less experience with computers.

## Documentation of the testing process

Test requirements:

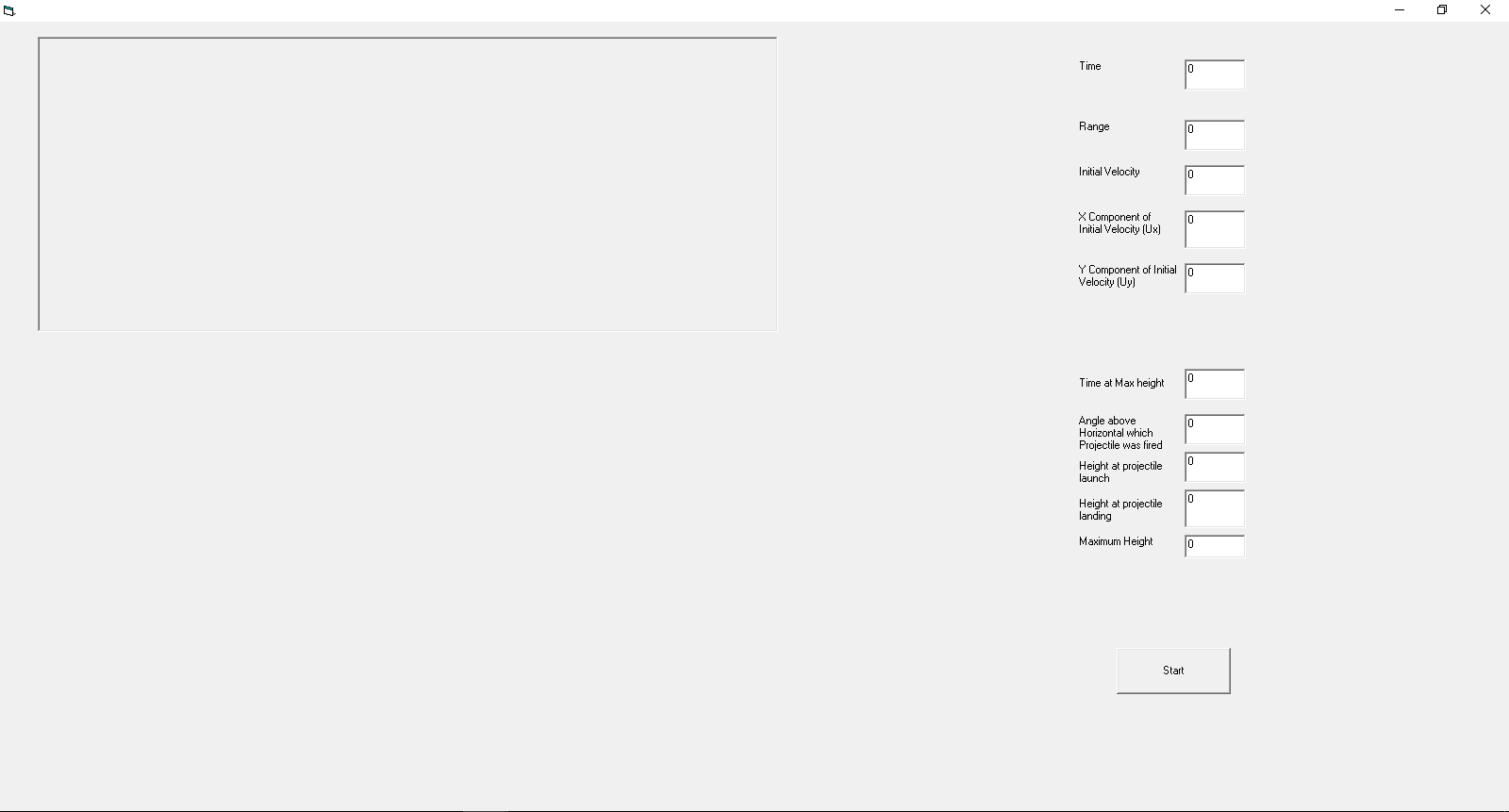
* The program gives the correct answers to any projectile motion question, especially HSC questions.
* The program is easy to use and no confusion occurs when other users use the program

# Maintenance

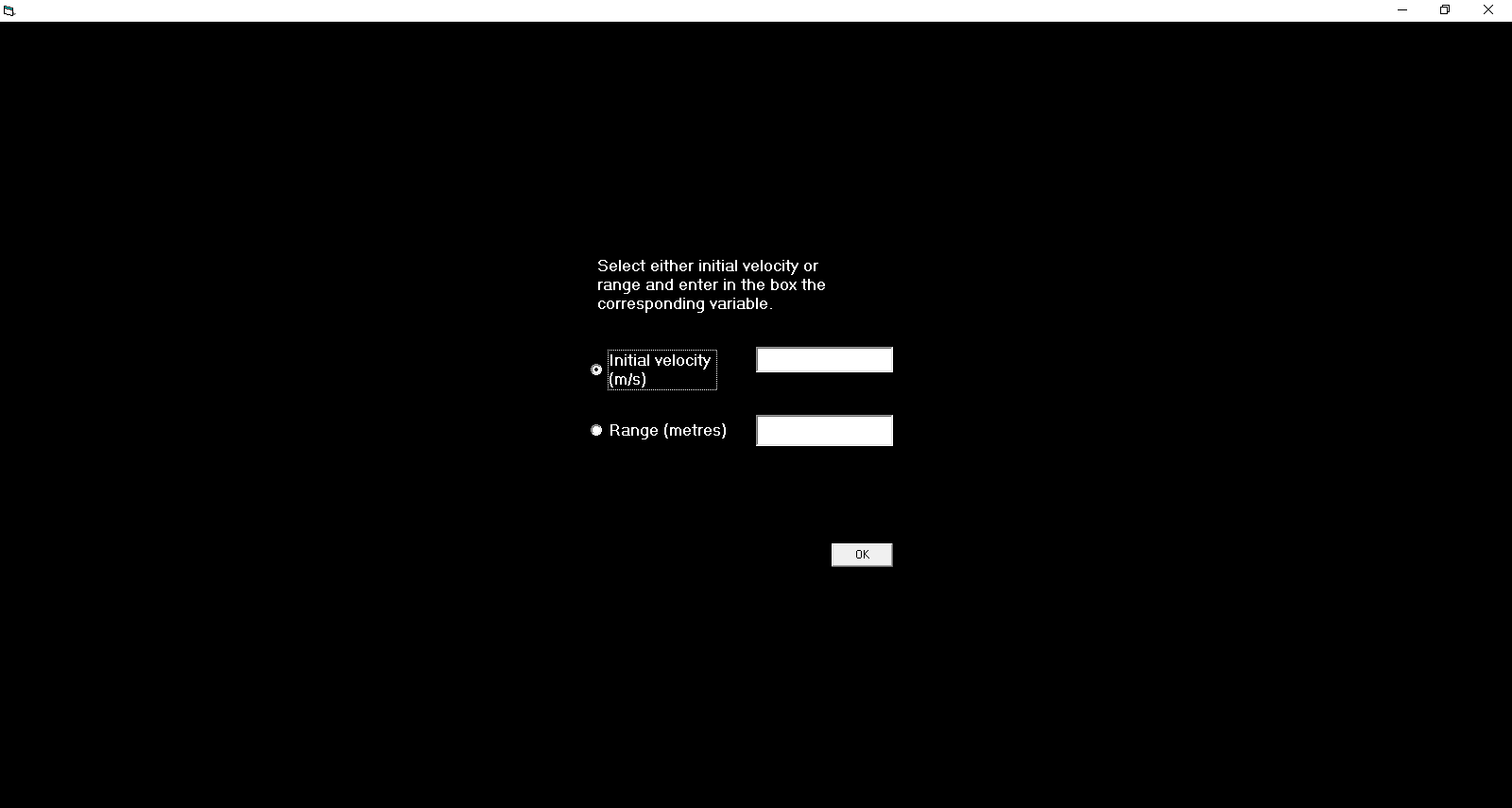
## Changes in Source Code and Designs

The program was edited with a new UI to make it easier to use. The program now gives 2 options, initial velocity and range, and gives more options after one is selected. The algorithm is selected based on the option the user picks.

Old UI:



New UI:



A function to calculate the projectile’s velocity at a time during the projectile’s journey was also added, in order to allow the program to be able to solve more projectile problems. This function can only be used after the graph has been shown. More error messages were added to ensure that the program does not crash when a user is using it.

## Documentation

Source code documentation has been provided in the source code in the form of comments. Each line of code has been explained.

GitHub was used as a version control system. Anytime changes to the code or portfolio were made, it was recorded in GitHub and a “commit” was sent to GitHub. This can be viewed at:

<https://github.com/spider93287/SDD-Project>

# Social and Ethical Issues Testing

## Authorship/Copyright

The Department of Education owns this program, and so the program will not be distributed commercially.

## Reliability

The program has been tested thoroughly, on school computers and on other computers, and the program works efficiently on all computers. It can perform all tasks given to it, and gives the correct answers to projectile motion problems given to the program.

## Response to Problems

All problems that have arised in the program have been fixed, and any future problems that users have will also be solved.

## Malware

The program is virus-free according to VirusTotal.

<https://virustotal.com/en/file/318b03f9757846e368f943184f115e3050240075feb0571b09e8baf5c737f71f/analysis/1495330428/>

## Plagiarism

The program has not copied any code from any other sources.

### End User Agreement

|  |
| --- |
| MIT License |
|  |  |
|  | Copyright (c) 2017 spider93287 |
|  |  |
|  | Permission is hereby granted, free of charge, to any person obtaining a copy |
|  | of this software and associated documentation files (the "Software"), to deal |
|  | in the Software without restriction, including without limitation the rights |
|  | to use, copy, modify, merge, publish, distribute, sublicense, and/or sell |
|  | copies of the Software, and to permit persons to whom the Software is |
|  | furnished to do so, subject to the following conditions: |
|  |  |
|  | The above copyright notice and this permission notice shall be included in all |
|  | copies or substantial portions of the Software. |
|  |  |
|  | THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR |
|  | IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, |
|  | FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE |
|  | AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER |
|  | LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, |
|  | OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE |
|  | SOFTWARE. |