**Physics Projectile Simulator**

Computable Problem

This program aims to help students studying projectile motion. It makes will allow the user to find missing variables from a projectile motion in seconds and even have a visual representation of the respective variables found. To do this the program simply has to take in data from the user then calculate any missing data (if possible) and finally it will use the variables found to throw a ball in projectile motion. It can be used by teachers, and students to check answers quickly and understand how the motion varies with the variables.

Requirements

**Product owner and goal of the application:**

The program can be used by teachers or students studying projectile motion. While teaching the subject teachers can show students how variables change with respect to each other as well as help them visualize the motion. Students can use this program as a “projectile motion calculator” to compare their answers with. This program aims to make it easier for students to understand the topic and possibly make

**User stories:**

As a teacher I want to make projectile motion an experimental and visual topic so that students are engaged and to make learning projectile motion more interesting.

As a student I want to quickly find answers to projectile questions to compare my answers to so that I know when I make a mistake.

From these user stories the functional and non-functional requirements of the program were noted. The functional requirements of the program are to take input from the user, process it and find remaining variables whenever possible. Then the program must also provide a visualization of the projectile motion depending on the variables inputted. The program must allow the user to choose between different gravities. The program must give users the option to view the projectile motion again in case students were not paying attention the first time around or if it happened too fast (due to small variables).

The non-functional requirements include the speed, simplicity, clarity and efficiency of the program. Speed is necessary is all the variables must be calculated directly after the user has inputted enough data. The program should be very simple and easy to use and navigate through by anyone, even teachers who have little experience with applications. The design should be clear and suitable for use in a class. And finally, the efficiency indicates the reliability of the program, in other words all variables found should be accurate and the visualization should be precise.

**Specifications:**

The first screen will give the user the option to simply start or see instruction. I want the screen to be basic and clear. If they press start then they have three options to choose from, Earth, moon or custom gravity. After they choose Earth or moon seven input boxes will appear labelled as velocity, horizontal velocity, vertical velocity, time of flight, angle of launch, distance and maximum height. If they choose custom the same input boxes will appear with the addition of an input box for gravity. After the user inputs enough, variables to calculate the remaining variables they are calculated and placed in the corresponding input boxes. Once everything is calculated a button to view projectile will appear. After they press it a ball will move in projectile motion depending on the values found before. If the values are too big the y-axis and x-axis will simply re-scale rather than allowing the ball to move off screen. The user will see the values found on the screen to compare with the projectile. The user then has three options, rethrow, main menu or enter new values.

**Validation for user stories:**

The teachers wanted something to experiment with and help students visualize projectile motion. This program solves that problem by allowing the user to type any data they want and directly after they can view the projectile in action according to the data given. However, it does not meet all the requirements as this program does not give the user a chance to experiment with air resistance. It is negligible in the program in Earth, moon and even the custom program. The program also does not provide a visualization for all values of distance and maximum height as some are too large.

The students wanted to use this program as a calculator when solving questions related to projectile motion. This program provides that to a certain extent. It will allow the user to only type up to fifteen characters, so if in a rare case a student wanted to calculate a number with over fifteen characters this program will not allow it.

**Initial product backlog (most prioritized to least prioritized):**

|  |  |  |
| --- | --- | --- |
| User stories | Est time  (days) | Priority |
| As a student, I want to calculate unknown projectile variables using limited known variables, so that I have help when solving related questions | 3 | 1 |
| As a teacher, I want students to view the projectile after all the variables are found, so that they visualize the projectile motion. | 2 | 2 |
| As a teacher, I want to input the variables in text-boxes to experiment with different numbers to my free will. | 1 | 3 |
| As a teacher, I want the projectile motion to be clear for students to fully grasp the idea - no matter how large the numbers. | 3 | 4 |
| As a student, I want to be able to change the value of gravity for different questions. | 1 | 5 |
| As a teacher, I want to be able to re-watch the projectile after it has been thrown just in case students didn’t understand the first time. | 1 | 6 |
| As a teacher that isn’t very good with program I want clear buttons to help me navigate through the program. | 1 | 7 |
| As a teacher, I want the y-axis and x-axis scales are clear for the students to understand the distance the projectile is moving. | 1 | 8 |
| As a teacher that isn’t very good with programs I would like to learn read instructions before trying to use the program, so I know how to use it. | 1 | 9 |
| As a student I want to have the option to add air resistance, so that I can solve the questions that ask me to do so | 2 | 10 |
| As a student I would like to change the type of object thrown, so that I can see the effect of air resistance and mass of different objects | 2 | 11 |

Design

There are four different ways to view the design of the program: logical, process, development and physical views (Kruchten, 1995). We will look at the program from the development point of view: The program can be broken done into different “chunks”, each made up of a function that depends on other functions to do its job. First the program must have functions for buttons, input boxes, then these functions must be used with another function to create the start function, calculations function, and projectile function. A list will be used to group the input boxes to make it easier when running them through functions that ensure they work like input boxes.The information stored in the input boxes are string, this means that to use them to calculate each other the data must be converted into a float.The values of the eight unknowns (velocity, horizontal velocity, vertical velocity, time, angle of launch, maximum height, distance) can be stored in a dictionary throughout the code. They will have to be global as almost every function will need to use the information stored in them. The calculations function will have many if statements as there are many combinations of input the user can give. To the programmer the design of the code is based off the flow chart on page 4.

The architecture is simply 3 layers:

Output layer

Calculation layer

Input layer

The task breakdown can be shown using data processing method:

Calculate missing variables

Input variables

Choose gravity

Output results

Calculate scale

A picture containing text, map

Description generated with very high confidence

Yes

Yes

Yes

custom

moon

Earth

Start

Instructions

quit

End

/quit

No

End

**Dynamic model:**

**A screenshot of a social media post

Description generated with very high confidence**

**The pseudocode for the function that will calculate everything, only if the gravity is known:**

If gravity is given, then do the following

If velocity is given and any other component, except for distance then

If velocity is less than its component, then notify the user that it’s not allowed

Otherwise, use velocity and the other given component to calculate the rest (using projectile motion equations)

Show the button that takes the user to view the projectile

Return the values

If horizontal velocity is given and any other component, except for velocity:

Use horizontal velocity and the other component to calculate the rest

If horizontal velocity is given as zero and the angle is not 90 notify the user that its not allowed

Show the button that takes the user to view the projectile

Return the values

If distance is given and any other component, except for velocity, horizontal velocity, angle:

Use distance and the other component to calculate the rest

Show the button that takes the user to view the projectile

Return the values

If angle is given and any other component, except for velocity, horizontal velocity, distance:

If angle given is greater than 90 degrees, then notify the user that it’s not allowed

Otherwise, use angle and the other given component to calculate the rest

Show the button that takes the user to view the projectile

Return values

If vertical velocity or maximum height or time is given:

Use the given one to calculate the other two.

Do not show the button as not all values have been found (only 3 values)

Return values.

**Justification of solution:** As all calculations are the same if gravity is given it makes sense to create a function that can be called for different values. However, another function should be made for calculations if gravity is not given; However, because in my case I will only have to call that function once I left it within a function rather than making it its own. When the ball is being thrown in projectile motion, one of the user stories mentions that the projectile motion should be clear from start to end. This means the ball cannot go offscreen, so the label between the horizontal grid lines increments by 20 if the maximum height every exceeds 220, or 440 or 660 etc. and the label on the vertical lines increments by 20 if distance exceeds 500, or 1000, or 1500 etc. As the grid lines are 50 pixels apart, I had to multiply the x-position by (5/(countx/250)) and the y-position by (5/(county/110)) this way the ball moves as if the grid lines are 20 meters apart or 40 or 60, depending on countx. However, there is a point when the number is too large to convert into an integer, so the for loop is used to have a limit.

A screenshot of a social media post

Description generated with very high confidence

Implementations:

To help with implementation I used the program trello, where I took one thing at a time from the “To Do” worked on it and tested it until it worked perfectly, then it’s finally done.

A screenshot of a cell phone

Description generated with very high confidence

A screenshot of a cell phone

Description generated with high confidenceAs can be seen from the photo above I first worked on the user-interface creating screens, functional buttons, input boxes and an instruction page. At this point the it was easy to go from point A to B. I did this as it is important to have something functional in the beginning if I Had to show it to a customer (Wiegers, 1999). The original scrum wall and final scrum wall can be found in the appendix, where it’s clear that there are still many things to be added on to the project as well as things still being tested. To go from “backlog” to “done” each task went through a diagram such as this (Halvorsen, 2018):

When implementing the code five things most be considered, coding guidelines, comments, debugging, code review and refactorization. These all ensure that anyone trying to read the code after will easily understand, including the programmer himself. In this case I was working alone, so I ensured that I followed python coding guidelines as much as possible to make my code understandable. Tasks derived from user stories can be seen on the scrum wall.

Testing:

There are four stages when testing software (Pearson, 2015). First the program went through unit testing where specific small functions are tested, this happened every time a sprint was finished. The main method used to debug the program was through static analysis, the IDE I used to catch all traceback errors which I then I had to fix (Wysopal, 2018). An example of this is that when the maximum height or distance is very large it reaches a point where it cannot be turned to an integer. To fix this I simply added a pop-up box that notifies the user that these values are far to large to plot. Next the code went through integration testing which testes the links between functions, to do this I simply clicked and went from function to function and that stage went smoothly. However, the program did run out of memory if the ball was re-thrown too many times. The next stage is system testing where the program was tested as a whole to make sure it meets the initial requirements, to do this black box testing (alpha) was used as I let a couple of people with no information of the code had a play around with the simulator, while I was next to them insuring to take down any errors or bugs they encountered. An example of a bug they encountered was when the angle given is 90 degrees the horizontal velocity was a very small number but not zero, to fix this an if statement was simply added testing. The final is stage is acceptance testing, at this stage the program didn’t have any errors or bugs; however, I had to go back to both the functional and non-functional requirements and ensure that the program fits the purpose it was made for and is ready for release. To help check this beta testing was used where I sent the program to some people with no information of the code. For example, grid lines were added on the simulation at this stage as one of the requirements was that the distance is clear to student using the program, and the students that tried the program notified me that it wasn’t clear how far the ball was moving (check appendix).

Maintenance:

The main aim of maintenance is to improve the program whether that is to fix faults, or react to changing requirements, and the world is always changing. There are four types of maintenance (tutorialspoint, 2018). First and most important one for my program is corrective maintenance, simply meaning I fix any bugs users find. This is extremely important as I must ensure that all calculations are flawless, and the projectile motion is exact. The code is not complete as when very large values for height and distance are found the program crashes as the numbers are too large to convert into integers. The next type of maintenance is adaptive, where I update the program to keep up with the changing technology or business trends. Another important type of maintenance in my case is perfective maintenance, there are many more features I can add to make this program even better such as giving students the ability to change the radius of the planet as gravity changes with the radius. Other students tried my program, and all said this would be great when solving question for projectile motion, it’s easier to use than a calculator. Though most students didn’t notice that they had to press enter after typing in each variable. To fix this a future improvement is to find the other variables simultaneously as the user is typing in the known variables. Users also found that the custom planet is interesting as they get to experiment with different gravities, this lead to less user ever entering the moon or even Earth. To improve this, I might simply allow them to type any gravity but at the top of the page I have dropdown with certain known gravities e.g. Earth, moon, Jupiter, Saturn etc. The last type of maintenance is preventive which focuses on fixing future problems.

Evaluation:

While implementing the program nothing went wrong until the sprint of user-input boxes and calculations. It was difficult to find a way that allows users to input numbers take the numbers and use them in calculations. This is because there are many combinations they can write, and until now there are combinations I might have not caught – particularly in the custom world. During unit testing the most amount of errors came up, every time a function was made there had to be errors along with it. Several errors came along during alpha and beta testing as well because users managed to find combinations I didn’t think of. However, most user requirements were met, all the non-functional requirements were met except for efficiency as the design is not perfect and it can be improved throughout maintenance, such as using more complex data structures in a way to make the program run even faster. The functional requirements were almost all met except if the values for maximum height and distance are too large then they cannot be visualized. The custom world still has flaws that must be fixed.

Appendix:

**A screenshot of a cell phone

Description generated with very high confidenceScrum wall start:**

**A screenshot of a cell phone

Description generated with very high confidenceScrum wall end:**

**Testing:**

A screenshot of a cell phone

Description generated with high confidenceif angle is 90 horizontal velocity must be 0.0:

A screenshot of a social media post

Description generated with very high confidenceIf rethrow is pressed too many times this error appears:

Integer too large error:

A screenshot of a social media post

Description generated with very high confidenceto run the game ensure to have both pygame and tkinter installed, and run it with python3 or higher.

References:

Anon (2018). *Software Maintenance Overview*. [online] www.tutorialspoint.com. Available at: https://www.tutorialspoint.com/software\_engineering/software\_maintenance\_overview.htm [Accessed 22 Apr. 2018].

Halvorsen, H. (2018). *Software Implementation*.[online] http://home.hit.no/~hansha/documents/software/software\_development/topics/resources/Software%20Implementation%20Overview.pdf

Kruchten, P. (1995). Architectural blueprints—The “4+ 1” view model of software architecture. *Tutorial Proceedings of Tri-Ada*, *95*, pp.540-555.

Pearson, L. (2015). *The Four Levels of Software Testing | Segue Technologies*. [online] Segue Technologies. Available at: https://www.seguetech.com/the-four-levels-of-software-testing/ [Accessed 21 Apr. 2018].

Wiegers, K. (1999). First things first: prioritizing requirements. *Software Development*, *7*(9), pp.48-53.

Wysopal, C. (2018). *What Is Static Analysis? - Ministry of Testing*. [online] Ministry of Testing. Available at: https://www.ministryoftesting.com/2013/07/what-is-static-analysis/ [Accessed 22 Apr. 2018].