Analysis

Computer science NEA Analysis section

Overview

At higher levels in physics and maths, the motion that equations refer to and diagrams can become unclear and so there is a need for a clear and concise physics simulation that can show students what the model actually portrays, which allows the student to better visualise the question. Having a physics simulator that is easy to interpret, use and collect data from would be of benefit to the learning of the student, which is where my program would be of use. As-well as helping the students overcome present issues, getting students used to using simulators is a good way to prepare them for future, far more complex, simulation systems at higher levels of education and the work force as large complex and expensive experiments are not done very often to conserve money.

Research into topic

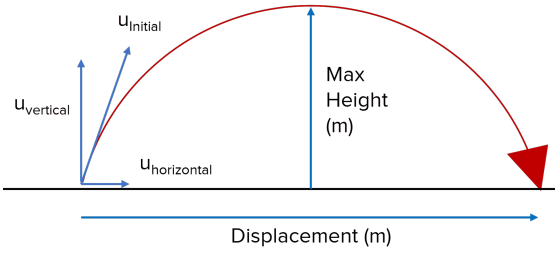
I can use my personal knowledge of the A-level and GCSE physics syllabus to determine the simulations I should create and check their validity via the internet. I also receive suggestions for simulations from my customer and Physics teacher Mr Beckingham.

Experiments that I will model will be researched with appropriate calculations are listed in the following table.

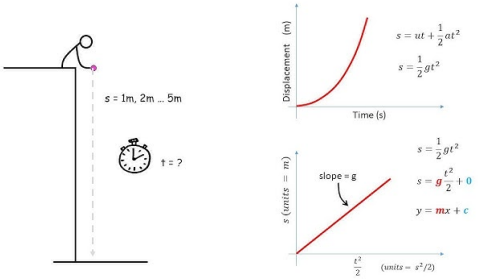
Describing the Models

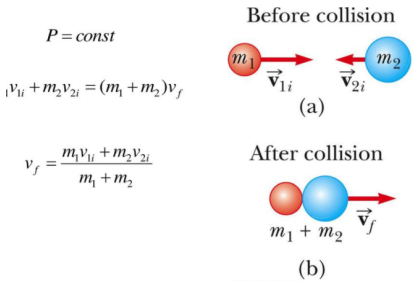
Experiment	Description	Diagrams / Reference
Freefall	A mass falls from a height and comes to rest at the ground.	experiment image 1
Projectile Motion	A mass is launched at a speed, and will fall to the ground where it will stop instantly. The speed will most likely remain constant unless air resistance is modelled (which is difficult and aspirational).	experiment image 2
Object sliding down a slope	A mass slides down a slope, friction may be modelled as a force acting against the movement of the mass, if the force cannot be overcome the mass will not move.	experiment image 3

Experiment Image 2

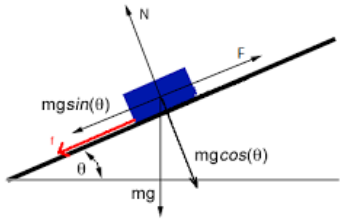


Experiment Image 1





Experiment Image 4



Experiment Image 3

Experiment	Description	Diagrams / Reference
Inelastic collision on a flat plane	Two masses of variable size (not necessarily identical masses), moving at variable speeds (not necessarily identical either) colliding on a flat plane and then sliding one way or the other or stopping out right.	experiment image 4

References

Freefall

A-level physics practical, Also in GCSE Diagram: https://i.ytimg.com/vi/K6dMoyM2B-A/hq720.jpg?sqp=-oaymwEhCK4FEIIDSFryq4qpAxMIARUAAAAAGAElAADIQj0AgKJD&rs=AOn4CLD1I A from video: https://www.google.com/url?sa=i&url=https%3A%2F%2Fwww.youtube.com%2Fwatch%3Fv%3 A&psig=AOvVaw2QhONOAC_PbS36koLYbNVf&ust=1744274371126000&source=images&cd=vfe&opi=899 . Practical 3 on: https://www.aqa.org.uk/resources/science/as-and-a-level/physics-7407-7408/teach/practicals-apparatus-set-up-guides

Projectile Motion

Common question in A-level physics and mathematics Diagram: https://mmerevise.co.uk/a-level-physics-revision/projectile-motion/

Mass on a slope

Common question in A-level mathematics Diagram: https://homework.study.com/explanation/a-mass-m-14-0-kg-is-pulled-up-an-incline-plane-by-a-f-85-n-the-plane-angle-is-26-degrees-with-respect-to-the-horizontal-the-coefficient-of-friction-between-the-block-and-the-surface-is-0-30-a.html . Examples of A-level mass on a slope calculations: https://pmt.physicsandmathstutor.com/download/Maths/A-level/M1/Topic-Qs/Edexcel-Set-1/M1%20Dynamics%20-%20F%20=%20ma%20on%20a%20slope.pdf

Inelastic collision on a flat plane:

Question in A-level physics Diagram: https://physics.stackexchange.com/questions/685887/perfectly-inelastic-collision-moving-and-spinning-wheels ### Calculations needed for models

Exp	Calculations needed	Equation(s)
Free-fall	SUVAT equation to solve for speed and height at individual points using time and acceleration	$v = u + at$ where $a = g$, t is time since the start of the simulation, and $u = 0$. then $s = ut + \frac{1}{2}at^2$, where $u = 0$ so $ut = 0$ so $s = \frac{1}{2}gt^2$ because a = g, (g=gravity),

Exp	Calculations needed	Equation(s)
Projectile motion	Freefall equations also linked with forward movement equations to calculate the distance	freefall equations (listed above) + forward equations for constant velocity: $s = \frac{1}{2}(u+v)t$ where u=v so $s = \frac{1}{2}(2u)t = ut$ so $s = ut$
Object sliding down a slope	Equations linking the resultant force of the block sliding down the slope linked to the angle of the slope.	$mgsin\theta$ where θ is the angle of the slope from the horizontal plane, m is the mass of the block and g is the force of gravity. The available angles may have to be hard coded or restricted as I am unsure of whether sine can be used in python
Inelastic object collision on a flat plane	Equations linking the mass and speed of the masses to their inertia and then equations to display the transfer of inertia and determine the resultant direction of travel for the two masses	Start the masses off with $d = vt$ until they collide where the following decides their motion afterwards: $m_1v_{1i} + m_2v_{2i} = (m_1 + m_2)v_f$ where m is the mass and v is the velocity and v_f is the final velocity of both masses.

Project deliverables

The Program should contain the discussed models and use their calculations accurately

Potential simulations for the program: - Freefall - Projectile Motion - Object sliding down a slope - 2 objects of varying masses colliding on a single plane

The program should also be able to utilise pygame's graphical features to portray the simulations in an animated form and tell the user the key properties of the simulation live on screen

Prototypes To prototype what I plan to do I wrote two programs: I followed a tutorial to demonstrate how a block could be manipulated around a screen which is beyond what I have to achieve & I wrote the code to spit out results for the free fall model which were then put into a file

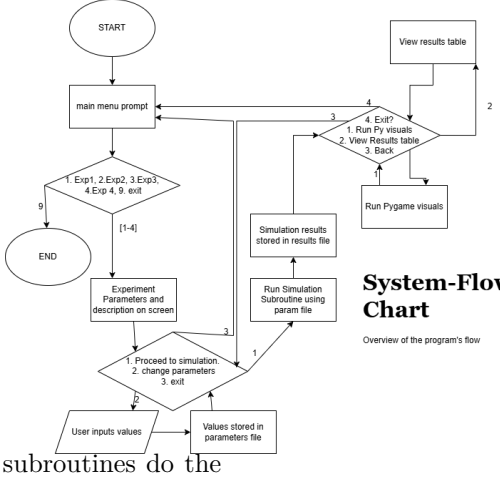
Project Requirements

No.	Section	Requirement
1	Menu	The main menu should lead the user to choose what experiment to do or exit

No.	Section	Requirement
2	Menu	User should be able to select an option on the menu
3	Menu	User should be able to see what option they've selected
4	Menu	The sub menus should contain the experiments
5	Menu	The menus should refresh soon after any update
6	Menu	The menu should tell the user the controls
7	Menu	The menu should allow the user to quit the program
8	Menu	**ASPIRATIONAL:** The user should be able to select options by clicking with the left mouse button
9	Results	Results should be stored in files related to each individual experiment
10	Results	Results should overwrite previous results in the experiment file
11	Results	Results should be stored in SI units
12	Results	Results stored in a file should be written in a format that the program can interpret
13	Experiment Usage	The user should be able to go back from a menu to the previous menu they were on
14	Experiment Usage	The user should able to replay the experiment
15	Experiment Usage	The experiment should be simulated accurately relative to time
16	Experiment Usage	The user should be able to edit parameters before the experiment
17	Experiment Usage	The scale of the projection should be that which maximises the usage of the window to provide the highest visual resolution of the experiment
18	Experiment Usage	The table of values that the projection is made from should be accessible to the user after the visualisation of the experiment is complete
19	Experiment Usage	The user should be able to end the program from each menu
20	Experiment Usage	Invalid user inputs should be rejected
21	Experiment Usage	When a user input is rejected the user should be prompted to re-enter the value
22	Exp 1	Experiment demonstrating a block falling from a height and coming to instantaneous rest on the floor must be simulated
23	Exp 1	Experiment must calculate the velocity, height / distance from start, and time since the start of the simulation
24	Exp 1	The experiment should be modelled as a block starting from an aloft point on the screen, from which the block falls to the floor (where it stops falling)
25	Exp 2	Experiment demonstrating a block in projectile motion and coming to instantaneous the floor must be simulated

I'm sorry for overlap, I tried my best to fix

No.	Section	Requirement
26	Exp 2	The experiment will use calculations of the velocity, height and distance along the floor at each time interval to produce the simulation to an accurate scale in time and distance
27	Exp 3	Model accurately demonstrating a block sliding down a slope
28	Exp 3	The sloped plane, as-well as the block, must be modelled.
29	Exp 4	Model accurately depicting an inelastic collision between two blocks
30	Exp 4	The model must show the blocks from when they're moving toward each other all the way up until an appropriate time after the collision to allow the user to witness the resultant direction of travel or lack there of.



System-Flow Chart

Overview of the program's flow

Design

Overview

The program will be a procedurally based structure where subroutines do the math and then send it to a file that is read by other subroutines that display it on pygame. The User Input will be console based with prompts appearing on the console. The console will display options and then have the user choose which number option they would like to select via inputting a number. These user inputs will be validated to ensure no crashes occur due to the user inputting an unusable response.

Hardware/Software Requirements

Imports:

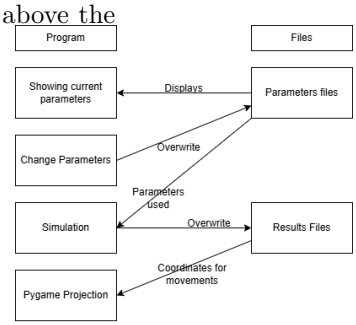
Pygame (version 2.6.1) for pygame animations math For mathematical functions not included otherwise such as the sine function re For regular expressions for input validation [[Pasted image 20250416111617.png]] The lower two come pre-installed with the visual studio code version of Python Pygame is a separate installation.

```
import pygame
import re
import math
```

Program-Flow Chart How the user interacts through the program and what processes it causes

[[DesignProgramProcess.drawio.png]]

Data Flow Diagram Shows how data goes from the program and then is stored in files and vice versa with the process this happens in listed above the arrow indicating it



[[DFD.drawio.png]]

Key Algorithms

As a procedural program, there's lots of algorithms that tie together

Load Parameters

The purpose of "Load Parameters" is to load the parameters from the relevant file it is asked to retrieve from and then to store the contents of the file into a dictionary using the key value pairs provided in the file. The dictionary is then returned. Process:

First it creates a dictionary Then it creates a file handle to access the data in the parameters file Then a for loop is run for each line in the file Strips off the new line additive Then it splits the name of the variable from its value Then it stores the variable and its value into a dictionary Then it returns the dictionary

Save Results "Save Results" stores the names of the variables and then the data into a list which is then stored in a flat-file for later use

It creates a file handle with the intent of overwriting the data currently in the file (previous experiments) For loop per line in the file Creates string variable called outline Then it creates another for loop controlled by the amount of variables to enter Changes the string "outline" to equal "outline" and the current value value in the position equal to "idx"'s value and adds a comma *it continues this until it has finished all data for the line then it exits the for loop* It then enters the last value without a comma at the end It then enters an enter space and starts on a new line *This repeats until there is no more lines of data to write to the file* Closes the file handle Returns nothing because it's job is done

Change Parameters

"Change Parameters" is a subroutine that allows the user to change the files that are called for the initial variables that are put into the simulation subroutines. This is paired with the input validation subroutine to ensure only appropriate values are entered to avoid complications with the values being stored and called.

shows user parameters and current values user input loop stores into dictionary loads dictionary into key1: value1 key2: value2 key3: value3 … format

Exp_FallingObj

Calculates the position of the block over time for the free fall experiment, after this it returns the flat file of values, which are then stored in the results file, it gets some initial values from the parameters file it is linked to

Creates variables and reads parameters from file Creates list for values **For Loop** from 1 second to the time limit/the time between readings(s) Calculates elapsed time of experiment from time on the experiment Calculates velocity towards the ground $speed = acceleration \times time$ round velocity to 5sf $distance = speed \times time$ rounds distance to nearest 5 sf **IF** distance fallen is greater than the starting height distance fallen is set equal to the starting height Results are saved to list **IF** distance fallen is equal to starting height END the For loop Saves the results to file

Exp_ProjectileMotion

Calculates the position of the block over time for the projectile motion experiment, it gets some initial values from the parameters file it is linked to

Sets parameters from file and creates other variables Creates results table **For Loop** Calculates elapsed time of experiment from time on the experiment rounds time **IF** air resistance is greater than 0 and height is greater than 0 $forwardvelocity = forwardvelocity - airresistance * time$ (deceleration) felt acceleration = float $feltacceleration = gravity - airresistance$

$fallvelocity = gravity \times elapsedtime - airresistance \times elapsedtime$ $verticaldisplacement = height - 0.5 \times feltacceleration \times elapsedtime^2$ **IF** forward velocity < 0 Forward velocity $= 0$ **IF** air resistance is 0 and height is greater than 0 $fallvelocity = elapsedtime \times gravity$ $verticaldisplacement = height - 0.5 \times gravity \times elapsedtime^2$ $height = fallheight - verticaldisplacement$ $totalvelocity = fallvelocity + forwardvelocity$ **IF** height is > 0 and forward-velocity > 0 $distance = distance + forwardvelocity \times TIMESLICE$ *rounds values to 5sf* **if** height <0 $height = 0$ $fallvelocity = 0$ $forwardvelocity = 0$ *saves results* **IF** height $== 0$ break save results to file

Exp_BlockOnSlope

Calculates the distance of the block over time, from the start place on the slope, for the block on a slope model, it gets some initial values from the parameters file it is linked to

Sets parameters from file and creates other variables creates distance relative to the speed of the blocks Creates results table calculates angle in radians calculates sine of the angle calculates force calculates resultant force calculates acceleration **FOR** exp loop via time calculates elapsed time calculates speed calculates distance adds values to dictionary **IF** distance too far return results **IF** resultant force is less than or equal to 0 return results

Exp_InelasticCollision

Calculates the position of the blocks over time for the inelastic collision model, it gets some initial values from the parameters file it is linked to

Sets parameters from file and creates other variables creates distance relative to the speed of the blocks Creates results table calculates point of collision sets collision to false **FOR** loop for each bit of time passed elapsed time calculated rounds elapsed time calculates x1 distance travelled calculates x2 distance travelled **IF** Collision = False and they're touching or either side of eachother collision set to true x1 = collision point x2 = collision point sets new speeds **IF** collision = true counts from collision time **IF** count is = to end count return results

main

sets mainloop to true and Exit to false while mainloop is true calls main menu UI user input (validated) = num num converted to int if 1-4, practical menus 1-4 called (Exit = return value) if Exit = True num = 9 if numint = 9 mainloop = False end of program

Py game

Algorithm controlling the visual simulation aspect of the program *for each different model the code must be changed slightly so more algorithms may spring from this|*

static perquisites (practical specific) loop based on frame rate calculating position of object from data (practical specific) moving the object end condition (no more data to put on the simulation) close Py game window

File Structures

Parameters Stored as a dictionary with key value pairs, in the format

```
Key1: Value1
Key2: Value2
Key3: Value3
```

For Example, The parameters file from Exp_FallingObj

```
height: 100
gravity: 9.81
resistance: 0
```

Results Results are stored in a flat file with the variable names as headers Example:

```
time,velocity,distance
0.1,1.0,0.1
0.2,2.0,0.3
0.3,3.0,0.6
0.4,4.0,1.0
0.5,5.0,1.5
0.6,6.0,2.1
0.7,7.0,2.8
0.8,8.0,3.6
0.9,9.0,4.5
1.0,10.0,5.5
1.1,11.0,6.6
1.2,12.0,7.8
1.3,13.0,9.1
1.4,14.0,10.5
1.5,15.0,12.0
1.6,16.0,13.6
1.7,17.0,15.3
1.8,18.0,17.1
1.9,19.0,19.0
2.0,20.0,21.0
2.1,21.0,23.1
2.2,22.0,25.3
2.3,23.0,27.6
2.4,24.0,30.0
2.5,25.0,32.5
2.6,26.0,35.1
```

```
2.7,27.0,37.8
2.8,28.0,40.6
2.9,29.0,43.5
3.0,30.0,46.5
3.1,31.0,49.6
3.2,32.0,52.8
3.3,33.0,56.1
3.4,34.0,59.5
3.5,35.0,63.0
3.6,36.0,66.6
3.7,37.0,70.3
3.8,38.0,74.1
3.9,39.0,78.0
4.0,40.0,82.0
4.1,41.0,86.1
4.2,42.0,90.3
4.3,43.0,94.6
4.4,44.0,99.0
4.5,45.0,103.5
```

User Interface

Console Menu:

```
Experiments:
(1) Free Fall
(2) Projectile Motion
(3) Block on a slope
(4) Inelastic Collision

(9) Quit

>*(user input)*
```

Invalid User Input

```
"User Input" Is an invalid input, Please enter a valid
```

Free Fall

```
A model of a Free Falling block which accelerates toward the ground under gravity.

Parameters:
Gravity>
Height>

Options:
(1) Continue To Simulation
```

```
(2) Change Parameters
(3) Back

(9) Quit
```

Projectile Motion:

```
A model of a block being launched under gravity from a height.

Parameters:
Gravity>
Height>
Forward Velocity>

Options:
(1) Continue To Simulation
(2) Change Parameters
(3) Back

(9) Quit
```

Block on a Slope

```
A model of a Block sliding down a slope under the force of gravity.

Parameters:
Gravity>
Length of slope>
Angle of slope>
Mass of Block(N)>
Resistance of slope(N)>


Options:
(1) Continue To Simulation
(2) Change Parameters
(3) Back

(9) Quit
```

Inelastic Collision

```
A model of Two masses colliding on a plane.

Parameters:
Mass of Left Block>
Speed of Left Block>
Mass of Right Block>
Speed of Left Block>
```

```
Options:
(1) Continue To Simulation
(2) Change Parameters
(3) Back

(9) Quit
```

"(1) Continue To Simulation" Would be selected by the user entering 1 into the console and would cause the program to carry out the experiment and take the user to the Simulation menu. "(2) Change Parameters"

Change Parameters

Displays the parameters section of the previously chosen experiment with each parameter appearing after the previous parameter's result was taken, line by line. For example if the user selected the Projectile motion experiment it would show:

```
Parameters:
Gravity>
```

Then:

```
Parameters:
Gravity> user input1
Height>
```

Then:

```
Parameters:
Gravity>user input1
Height>user input2
Forward Velocity>
```

Simulation Menu

This menu Opens again after the user selects option 1 or 2 and the option is no-longer using the console interface.

```
Simulation Menu:
(1) View Results
(2) View Simulation
(3) Back

(9) Quit
```

Testing

Requirements Reached *It wouldn't let me put it as "everyone with link can view" because of school restrictions* https://docs.google.com/spreadsheets/d/1TN3Z___vwbxFdi7oVup83q5KgA

Testing Video 0 Highlights input error at the end, fixed in final version and for Testing Video 1 (2min)

Testing Video 1 Demonstrates All unique program functions and most repetitions of those functions (4min)

Explanation video 1 Explains key algorithms and such (10min)

Explanation script Walk through of param files Walk through of data files Walk through of subroutines run program and change parameters to rewrite parameters file then view file for exp data to view before running to rewrite, show results are in base SI units

Test script

NEA TEST SCRIPT req5: will be observed throughout testing

- Run program (req1)(req6)
- Quit from menu (req7)
- Run program
- enter random string instead of a valid input (req20)
- Select falling object experiment (req2)
  - selects and continues to next prompt (req3) (req4)
- go back to main menu from experiment (req13)
- navigate through menu to falling object section
- change parameters (req16)
- go to the model view and then replay and then quit the program (req14) (req15) (req17) (req22) (req23) (req24)
- view results table (req18)
- go back to change parameters and enter gibberish (req20)
- quit program (req19)
- Open program
- navigate to the projectile motion section
- run the simulation (req25) (req26)
- replay simulation
- quit simulation
- go back until main menu
- navigate to the slope experiment simulation
- run simulation (req27) (req28)
- replay simulation
- go back to the main menu
- navigate to the collision simulation
- run simulation (req29) (req30)
- quit simulation

- go back
- change parameters to the same speed for both blocks and same mass for both blocks
- go to simulation
- run simulation (req30)
- quit simulation
- quit program

Evaluation

Overview

To summarise the project, all requirements have been met and tested in the testing section. The program works as intended and serves its purpose to visualise experiments and models that maths and physics students would gain benefit from visualising. There is a great foundation in this program to expand and add more models very easily, as well as, ironing out problems with current models. The adaptability of the program is very good and things such as a menu in the pygame window would be an easy implementation in terms of editing existing code which already determines what would be on the menu. Although certain allowed values break the slope model (not the program as a whole which continues to run) which involves setting angle to a higher value than ~45° on the slope experiment but this would be an easy fix to disallow these values but a slightly more complex solution to integrate these values into the program. More experience in using pygame and more fluidity in my python programming would have allowed faster progress and potentially more requirements to have been made and completed which leads to:

What could have been different

In hindsight, many factors have hindered fast progress which, in a second version of the project, could be eliminated to ensure a more successful program. The Following includes what a remake would have put into it and what it would entail. My personal experience with pygame and python is far stronger now than it was at the start of the project, with this increase in confidence and speed in programming necessary algorithms to help get more done in the time spent. With this increased personal aptitude more models could be implemented expanding the scope of the program, increasing its usefulness to the end user. Programming everything to be easy to debug from the start and compartmentalising the code would improve the expandability of the program even further which increases the value of the project for improvement by not only myself but by others. Giving an option to save results permanently would be a useful addition to the program. Providing an option to change the resolution of the results or to change the resolution of the results table would increase the user friendliness of the program. Having the ability to run any model from a file would also be a good addition requiring more information to be stored in the file as well as improving the accessibility of the program and allowing students to be given models by the teacher without having to go through much of the program. These are just a few examples of what could go better in a version two but it also highlights how the core program of this version is a successful project.

Feedback

- Dad's Feedback, The program simulates the experiments correctly, however the user interface is not intuitive and so accessibility could be improved, which involves the functions of the program, which are not immediately apparent to the user as the wording of some functions are similar and can be confusing if the user interface is not explained. For example: when continuing to simulation from the experiment-menu to the post-1st-instance-options-menu my father was confused by the similar wording of simulating the numbers of the experiment to actually displaying the model. To fix this I could change the wording of "continue to simulation" to "perform simulation" or "run calculations" & change "continue to model" to "show model" or "display model".

Summary

- Overall success
- Could have done more models
- Strong input sanitisation

In summary, the program is an overall success with a strong foundation for growth through new models or a more accessible and polished user interface. The strong input sanitisation ensures a smooth crash-free experience for users which is essential for people who don't want to worry about the program they're using and just want to look at the models.

Evaluation writing plan:

4/4 requirements Full consideration given to how well the outcome meets all of its requirements. How the outcome could be improved if the problem was revisited is discussed and given detailed consideration. Independent feedback obtained of a useful and realistic nature, evaluated and discussed in a meaningful way.

plan: - The outcome has met most requirements (link the testing list and talk a bit about that) - outline defensive programming against user input - More requirements could be done

- More experience with pygame and python
- More Models
- Robust programming from the start
- More compartmentalisation
- More separate files to improve debugging efficiency
- Saving results more permanently
- Running from files

- feedback
- Overall success
- Could have done more models
- Strong input sanitisation

Table of Contents