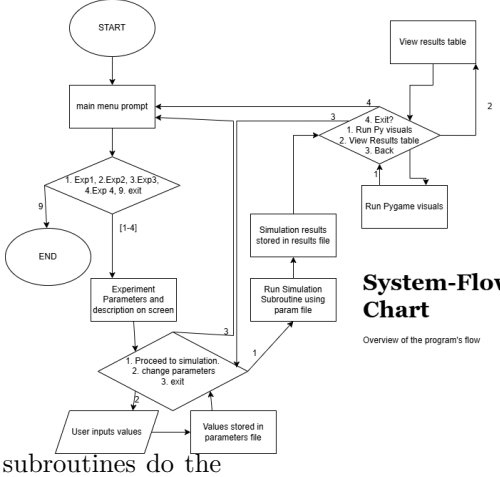


System-Flow Chart

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:

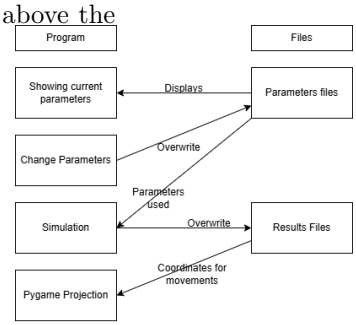
Pygame (version 2.6.1) for pygame animations math For mathematical functions not included otherwise such as the sine function re For regular expressions for input validation [[Pasted image 20250416111617.png]] The lower two come pre-installed with the visual studio code version of Python Pygame is a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