

Contents

Analysis.....	2
Problem identification	2
stakeholders	3
Why it is suited to a computational solution	3
Computational methods the solution will achieve	3
Problem decomposition	4
Divide and conquer	4
Abstraction.....	4
Thinking ahead	5
Research	5
Features.....	9
Limitations of my solution:	9
Interview	10
Interview responses.....	12
Programing language	15
Requirements	16
Success criteria	18
Design.....	21
Systems diagram	21
Module explanations.....	21
User interface design.....	23
Menu select	24
Usability features	24
Stakeholder input.....	25
Algorithms.....	26
Explanation and justification of this process	32
Validation checking.....	32
Key variables	33
Testing method	37
RAD development.....	37
Testing to be used in iterative development	38
Test data.....	42
Post-development testing	47
Development of coded solution.....	47

Milestone 1: allow for user-level selection	47
Milestone 2: Successfully display a problem and its setting according to the level (syllabus)	55
Milestone 3: Be able to input values and respond accordingly	58
Milestone 4: Apply the correct formula and solve the problem	68
Milestone 5; Can graphically calculate and represent the motion	73
Milestone 6: Have a non-linear AI defence system to follow the projectile and intercept (if ticked yes).....	81
Milestone 7: Give a win or loss screen	89
Milestone 8: Progress back onto the main menu, or next, or previous or re-do	96
Input Validation	99
Validation checks	101
Evaluation	105
Data value test results	105
Function and robustness testing.....	110
Usability testing.....	111
Beta testing.....	114
Success criteria evaluations	115
Limitations.....	128
Current and future maintenance	131

Analysis

Problem identification

Currently there is no way for students to interactively learn the extremely hard to grasp and important topic of mechanics/kinematics. By enabling students to do so it would mean that this once incredibly difficult and challenging topic is now fun and not seen as a chore anymore, this is statistically proven as enjoying a task increases self-motivation which in turn results in a higher memory retention; therefore, allows for students to willingly learn mechanics by playing this interactive game that explains it to them as they progress through the game. This program would potentially have the impact of improving the mark average in the worst performing topic of kinematics in both maths and physics A-Levels (this is shown by multiple figures, e.g. student reports).

The features of a computer system that would be required for this solution would be a computer with hardware capable of graphics processing and would also have to deal with multiple simultaneous mathematical/logical processes at once, thus lending itself towards a multiple core CPU or a CPU with adequate scheduling hardware and software to run this efficiently. The system would also require a suitable software and platform for the game to run upon.

This problem lends itself towards a computational solution because it would be interacting with an interface and performing multiple calculations simultaneously. In its simplest form, the solution would take in some missing values from the user and use them in kinematic equations to form a projectile path, whose aim is to hit a target or get intercepted by an AI defence system, along the path the GUI will display and explain the kinematics equations and the steps taken towards the solutions.

stakeholders

The clients and demographic for this project would be physics/maths students or anyone who has an interest in kinetic physics. This range will provide a representative sample of the entire possible clients that would all be able to test the software in different areas and to different degrees.

The topic of mechanics can be learnt and perfected with this game as it allows for an interactive visualisation behind all of those complicated equations. To test the efficacy and usefulness of the solution I have recruited Yassine Soltani and Sarrujan Raguparan; both of whom are maths and physics students and do experience some difficulty with the mechanic's topic in both subjects.

Why it is suited to a computational solution

The problem lends itself to computational methods of finding and implementing a solution for a variety of reasons. The solution will be a program that uses user input to feed advanced mathematical formulae to derive answers to then display a graphical demonstration within a few brief milliseconds. This will need to run in a computer as it would not be possible otherwise, as a human could not manipulate data like that and produce a perfect graphical representation within enough time to keep the user interested. There is no feasible alternative where a computer would not be necessary. The whole application would be computer-controlled except for the human inputted values.

Computational methods the solution will achieve

The overall problem is processing the data input values into a graphical representation of the motion, whereas the actual underlying problem is how to plot a

perfect projectile curve without calculating every coordinate point on that curve. When this is overcome the rest of the solution is simply some substituted equations and some print statements along with an

AI to intercept, which is some more complex code that will trace the projectile trying to intercept it.

The AI aspect within my solution is also a very difficult thing to implement as I cannot use the previous values as that would defeat the purpose of an AI, but then if the algorithm works perfectly then the AI would not allow the user to win at all. This means I would have to incorporate some randomness into the AI values and directions so an interception is not always guaranteed and it is actually possible for the user to win.

Problem decomposition

This problem and its solution can be decomposed into a sequence of smaller steps. This is an initial outline of those steps:

1. Take input of the user
2. Use kinematic equations to derive answers to the motion
3. Draw the path of the projectile from the kinematic answers
4. Get an AI to try and intercept the projectile
5. If hits target, explain the kinematics process, if not re-run
6. Move onto next level, moving through the syllabus

When this is done the solution is complete and if it is done efficiently then it will appear smooth and game-like for the user, keeping it playable and interesting.

Divide and conquer

These smaller steps are challenging on their own, but very realistic and achievable. Solving the steps on their own and then combining them into a bigger modular program makes use of the divide and conquer technique of problem-solving.

Abstraction

When using and manipulating kinematics equations you may not require every variable and value that is given to you, so by ignoring the redundant and unimportant variables abstraction is achieved.

Abstraction would also be used in the GUI part of the solution, as it is a static and unimportant part of the main solution it would not need to be processed till absolutely necessary thus disregarding till it is needed.

This technique will also be used in the AI system within the game, the AI will effectively be 'locked' off from the previous game values so it does not defeat its own

purpose. The AI will be processed and executed in complete isolation from the rest of the program.

Thinking ahead

This is how I will look to the end of the project and hopefully how the end solution functions. This will give me a guide to follow while completing the project.

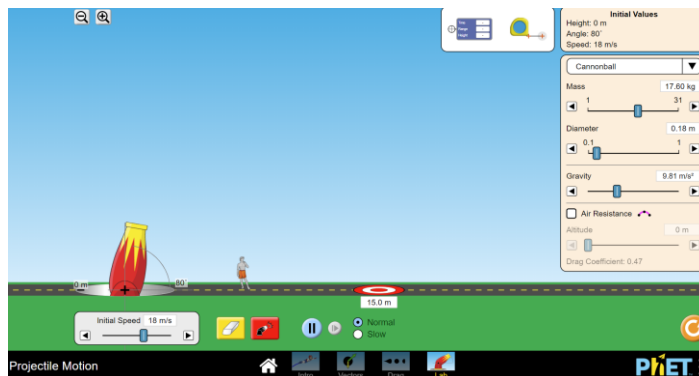
- I am planning to utilise JavaScript as my main language alongside some HTML and CSS to incorporate into a browser
- Inputs for the game will be done using a keyboard input from the user.
- The game outputs will incorporate visual representations and animations

Mid project amendments - I now plan to use python and the pygame module as my programming language, for this reason, it will not be browser supported.

Research

Existing similar solutions:

Phet Colorado education



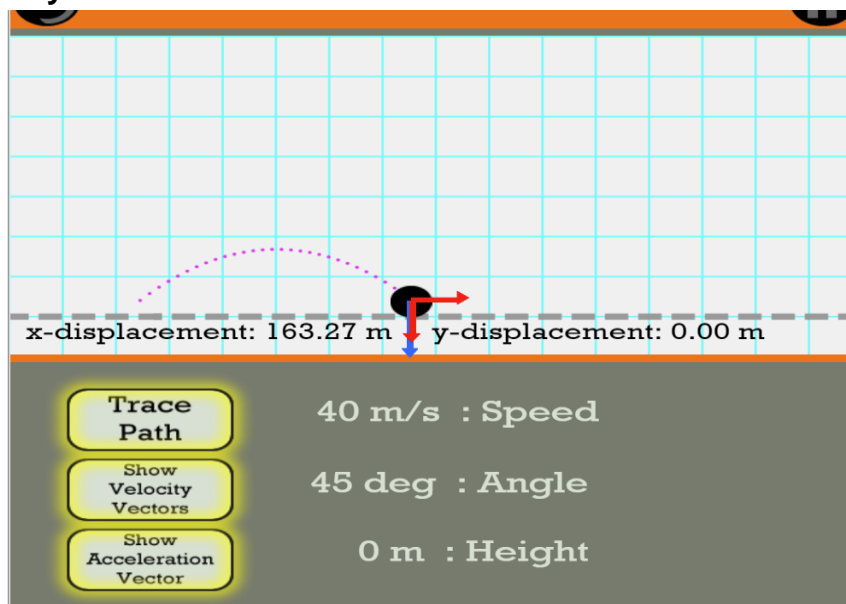
This solution provided by phet encompasses some of the attributes that I intend to use in my solution as well. This example always allows the user to simply control some variable sliders to then hopefully get the projectile to hit this static target on the ground. After more consideration I would not like to incorporate the value sliders because students may decide to not actually calculate the values but instead use trial and error with the sliders; to stop this trial and error 'cheating' I will limit to 3 incorrect answers per question before the question gets reset. It does show the path of the projectile in motion as a graphical representation of the motion which I will also do, however as far as I can decipher this program works out multiple coordinates on the curve as it goes along showing the points, this is an inefficient way of showing the path as ultimately you just require the mathematical roots and turning point to generate a curve. In my solution, I would go around solving the path representation in a mathematical approach as a quadratic curve on a set of

axes instead of plotting every point from SUVAT equations. This game also only provides set values and does not change them or explain the solving process of the equations in the processor at the end. This game also only provides one target in which there is only one level which does not make it an interesting game for most people; in addition to this there is no countermeasure to the projectile, in my solution, I will be implementing an AI defence system to protect the target to make the game a bit more fun and intriguing.

Parts that I can apply to my solution

This software by phet Colorado seemed very efficient and polished, being quick to open and execute; this is definitely something I want to carry forward in my final solution. This solution as previously stated shows the coordinates along the path including and highlighting the turning point. I believe this is a very good feature and I will incorporate it as the SUVAT equations are calculated separately for each side of the turning point. Another good idea that this game incorporates is the decision to show the motion vectors in terms of acceleration and velocity, this is the fundamental understanding behind all forces and if the student can see how the vectors change with motion then they will gain an understanding of how they work and what they would have to do to calculate them

Physicsclassroom.com



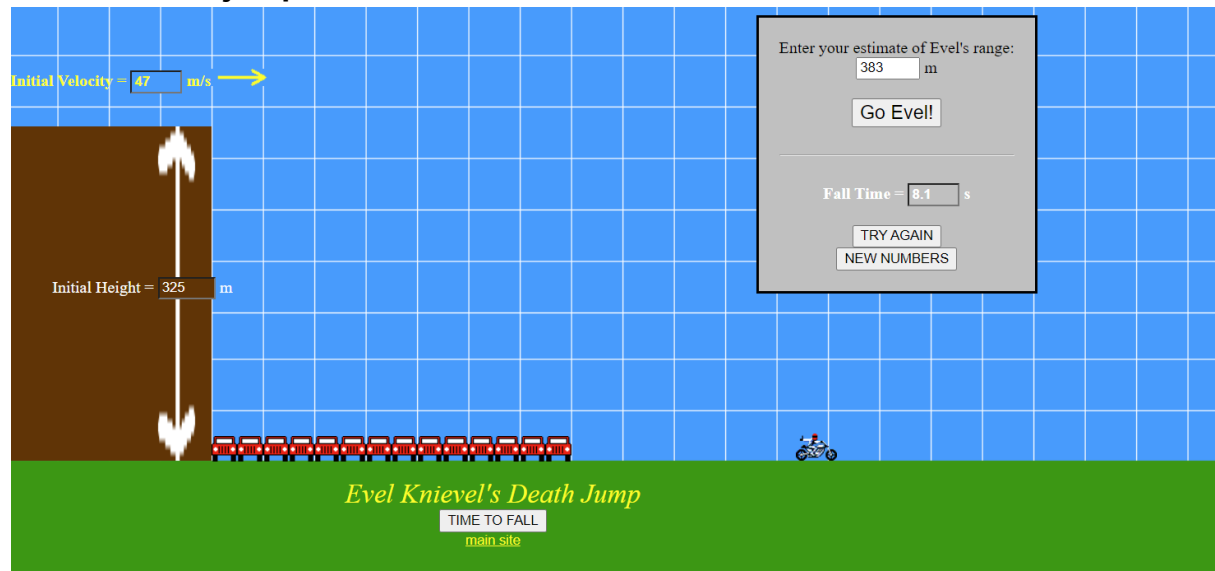
This solution provided by physicsclassroom.com is a very simple and minimalistic approach to the problem. All that does is take in some value in input forms of input boxes and shows the project its path, and its motion vectors. This is ultimately the very bare-bones edition of what I am aiming to build, the actual projectile element of my solution will be very similar to this. The differences being, I will have a target, an AI defence system, unlimited

variable range, explanations of the kinematics to resolve it and a more graphical representation in a game-like way.

Parts that I can apply to my solution

As previously mentioned I will now be incorporating variable value input boxes into my solution to hopefully prevent any cheating effort. From this solution I must give credit to the idea to show the variables live throughout the projectile motion, I believe is a very good feature as a watching student can then understand what happens with the variables in real-time as the execution carries on.

Evil Knievel's death jump



This solution provided by mrmont.com is an extremely barebones version of what my final solution aims to be. This game does have a graphical aspect to it in which the bike is shown to follow the path and the setting represents the question that you have to answer. The variable values are randomly generated but in a given range meaning that it is not a truly unique question every time; moreover, the game allows the user to click the “time to fall” button to find the time taken to fall that distance that you are provided with, I do not like this as my solution aims to teach the student how to calculate every aspect of the answer, not be given parts of it as they will never learn that way. This game is also only one level and there is no real objective too, just enter some values to see the man fly, I do not think this is very interesting and will not push a student to learn more. There is no main menu to select levels and the win/lose pop-ups come in the form of web pop-ups which are not graphically pleasing at all.

Parts that I will apply to my solution

The developer of this game has included the background as a grid, essentially treating the whole thing as a graph so it is easier to trace the path of the projectile. I do aim to do this as it is the best way to represent the motion. However, I would not

keep the grid as the main background as it is not appealing, instead, I would keep it as a hidden layer that I could still make use of.

Projectile motion simulator

INPUT DATA

V: m/s

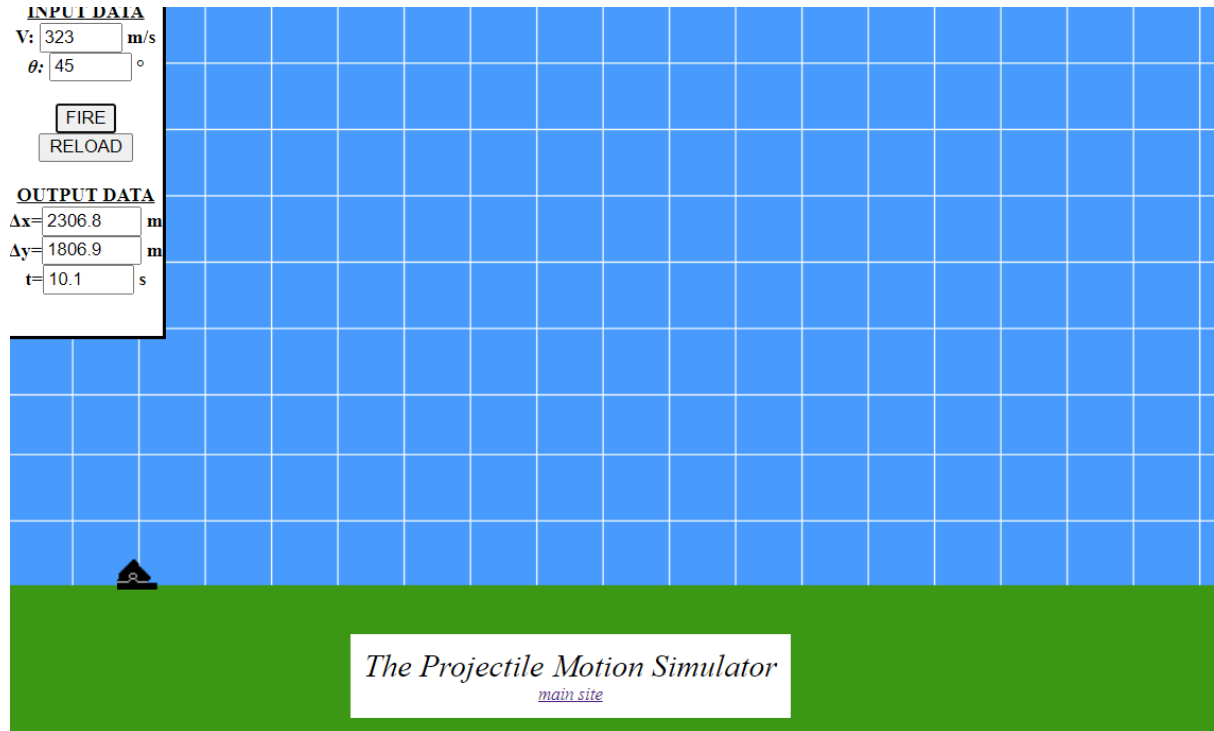
θ : °

OUTPUT DATA

Δx = m

Δy = m

t= s



This solution is also provided by mrmont.com and once again is a very simplistic version of what I aim to build. This simulator allows variable inputs to change the projectile's motion and then graphically display it. It also outputs the live variables as the projectile is in motion. This game only allows for 2 possible input values, this is something I would change as it would not allow the student to fully grasp how to include and calculate for other variables. This game also has no objective, it simply enters values and fires without a question or target to hit. Once again the grid is used as the background so a graph can be referenced and used in the representation of the projectile's motion.

Parts that I can apply to my solution

Once again I will incorporate the idea of a graph as a background grid but I will do it as a hidden layer so it will not disturb or ruin the look of the game. I also intend to use the user input boxes as this solution uses on the side however my variables will be changing with each question. I may also decide to incorporate the idea of the actual projectile itself being a real-life projectile such as a missile; I believe this is a good idea as it seems interesting and more game-like. Also in using a missile it would make the game make a lot more sense in terms of the AI

missile intercept system and the iron dome situation I intend to build with it.

Features

The initial concept of my solution after research consideration:

My solution will be a browser-based game that when loaded will greet you with a menu outlining the mechanic's syllabus so you can choose to go from the start or only a specific topic that you may be struggling with. When the topic is selected then the game level will load (each topic is effectively a level). Then the user will be presented with a mechanic's problem to solve in terms of hitting a target and having to calculate and fill in the missing variables to then get the answer. If the user gets the answer wrong they will be prompted to answer again till right and then try the question again with different values, if they get it right they will progress to the next level/topic having understood that one. If the user gets the answer wrong 3 or more times than the correct answer and the working out to it will be explained to them. If the answer is correct then the working out will also be displayed so they can check theirs. When the user gets the answer right once and re-does the question, then they will be presented with an AI defence system. This missile intercept system will work similarly to that of Israel's iron dome system which protects the city, or in my case the target. It will work not by calculating an intercept from the question/user values as that is not a true AI, instead, it will launch when the projectile enters the "iron dome" then it will follow the projectile and try to find a quicker path to its destination to intercept and defeat the missile; the AI will also have some inbuilt randomness to keep the user on edge. This will progress till the user has mastered all the topics/levels and understands the subject area.

Mid project amendments - My idea remains the same however it will no longer be browser-based due to a change to pygame.

Limitations of my solution:

The main limitation of my solution will be the severity of the details, as it is coursework and I will have a deadline to adhere to I will not be able to incorporate studio graphics and 2000 levels, the game will be very minimal and will not look graphically amazing but will do the job on explaining the mechanic's process to the students in an interactive and fun way.

Another limitation will be the platform the game will be running on. As I aim for it to be a browser-based game that means any platform should be able to access it and

run it smoothly, and obviously a PC can run a game much more demanding than a mobile could, and for that reason, the game cannot be graphically or CPU demanding at all and has to be optimised for low usage scenarios.

The graphics are a big limitation; this is because this game is more geared towards its educational use. The graphics do not matter all that much, just that there should be a setting with the projectile path and solution shown. Graphically this solution will be very minimal and look somewhat similar to the solutions provided by 'mrmont.com'.

Lastly, another limitation may be caused by the AI, as I will have to incorporate some randomness into the calculations of the AI (uncertainty), it may corrupt the true meaning and efficacy of what an AI is; however, I will need to do this as otherwise the AI will be too powerful and a level will never be won against it.

Interview

Interview questions

Here I will present the questions that I will be asking my stakeholders and possibly a few others to get a more representative sample. In the interviews I may decide to ask a more complicated question or get them to elaborate their answers, the general answers will be recorded below. The questions will get their opinion on the game idea and any improvements or advice.

Student questions

These are the questions I will be asking my student stakeholders (Yassine Soltani and Sarrujan Raguparan)

1. How would you describe your ability and confidence in the topic of mechanics, 1 to 10?
2. Have you ever played an educational game?
3. If yes, did it benefit your ability?
4. What do you think of using an educational game as a potential revision or even a learning tool?
5. What would you imagine an educational game on mechanics to be like?
6. What would you like to do in the game to help your mechanics' knowledge?
7. Do you have anything else to add?

Question 1 outlines and indicates the problem that my solution is looking to solve. This is necessary as it dictates if there is a need for my solution or not.

Question 2 and 3 then establish their history with educational games, this gives us important information on whether they may be biased due to previous experiences. It also comments on the potential usefulness of my solution.

Question 4 then asks their stance on my solution and if they would consider it to be feasible in a real-life situation. This will show us if the solution will be used or not.

Question 5 allows the stakeholder to envision the game, this then gives us multiple perspectives in which the solution could be presented, and what is expected of it.

Question 6 simply asks what they would like to do in the game, as different people learn and revise in different ways it will allow me to gain this perspective so I can make my game as inclusive and useful for as many people as I possibly can.

The questions are concluded by asking if they have anything else to add that I may have missed out on.

Educational questions

These are questions that I will be asking people with an educational standing in the matter, such as a physics teacher (Mrs Sharif)

1. Do the students you work with use educational games?
2. If yes, how has the experience been, has it improved their outlook/ability?
3. Do you think your student struggles with the topic of mechanics and could use a better, more interesting revision/learning tool?
4. Do you think students would be willing to use educational games as a revision/learning tool?
5. Do you think students would benefit from using an educational game for the topic of mechanics?
6. How would you envision a game on mechanical physics to be?
7. Do you have anything else to add?

Questions 1 and 2 establish the educator's previous history with educational games and therefore their stance towards them, potentially showing bias.

Question 3 gets the educators informed opinion on the whole problem itself, and if there is a problem to be solved or not.

Question 4 indicated their opinion on whether their students would use a tool such as my solution and if it would help them or not.

Question 5 then receives their opinion on whether it would be beneficial or not, therefore declaring my solution as useful or not and if it is needed.

Question 6 allows me to gain an educator's perspective on my problem and its solution, so I can use it to improve my solution in a way that a professional think would be good.

The questions are concluded by asking if they have anything else to add that I may have missed out on.

Questions for a regular person interested in mechanics

These are questions that I will be presenting to regular people who hold an interest in the topic such as a regular man (Mr Omotayo)

1. Are you interested in the topic of mechanics?
2. Would you be willing to learn the subject from an educational game?
3. How would a game help you learn a new subject?
4. Do you have anything else to add?

These questions are simply to get a perspective on what a regular person that just wants an insight into the topic would say towards the problem and its solution. This perspective allows for me to facilitate for them so I can make the game more inclusive and interesting for everyone and hopefully open this subject up to a lot more people than just the maths and physics students

Interview responses

Yassine soltani (student)

1. How would you describe your ability and confidence in the topic of mechanics, 1 to 10?
"3"
2. Have you ever played an educational game?
"yes"
3. If yes, did it benefit your ability?
"yes"
4. What do you think of using an educational game as a potential revision or even a learning tool?
"It stays in my long term memory and I don't regurgitate information"
5. What would you imagine an educational game on mechanics to be like?
"Interactive."
6. What would you like to do in the game to help your mechanics' knowledge?
"to be able to work with projectiles."

7. Do you have anything else to add?

“N/A”

Sarrujan Raguparan (student)

1. How would you describe your ability and confidence in the topic of mechanics, 1 to 10?

“3”

2. Have you ever played an educational game?

“yes”

3. If yes, did it benefit your ability?

“yes”

4. What do you think of using an educational game as a potential revision or even learning tool?

“Good. Entitles you to learn fundamental concepts and principles, making it easier to answer more challenging questions.”

5. What would you imagine an educational game on mechanics to be like?

“Intuitive. Interactive. Somewhat challenging.”

6. What would you like to do in the game to help your mechanics’ knowledge?

“Questions but with a fun twist to it.”

7. Do you have anything else to add?

“N/A”

Mrs Sharif (educator-physics teacher)

1. Do the students you work with use educational games?

“no”

2. If yes, how has the experience been, has it improved their outlook/ability?

“N/A”

3. Do you think your student struggles with the topic of mechanics and could use a better, more interesting revision/learning tool?

“yes”

4. Do you think students would be willing to use educational games as a revision/learning tool?

“yes”

5. Do you think students would benefit from using an educational game for the topic of mechanics?

“yes”

6. How would you envision a game on mechanical physics to be?

“Using trajectories to hit a target.”

7. Do you have anything else to add?

“N/A”

Mr Omotayo (regular/interested person)

1. Are you interested in the topic of mechanics?
"yes"
2. Would you be willing to learn the subject from an educational game?
"yes"
3. How would a game help you learn a new subject?
"A game would help with my memory of it"
4. Do you have anything else to add?
"N/A"

Analysis of student responses

From the interviews I have conducted with my fellow peers/stakeholders I have gained much information and perspective towards my project. Both the student stakeholders have answered the first question with the response of "3". This is very useful as it allows me to get an overall standing on the general attitude toward the subject and the problem I aim to solve. Both of the stakeholders I have interviewed are above par students with above-average grades, therefore indicating that the average student would sit around the 5 mark on the scale. This then shows that the average student is not at all confident in their mechanic's ability especially during an exam setting. Therefore, this further enhances the need for my solution as there is an evident problem at and here that students do not understand the kinematics topic. In addition to this, both students have played an educational game before and both believe that it has improved their ability and confidence in that sector. This shows that educational games are effective and are a viable learning/revision tool for all students.

Yassine's response to using a game as a revision tool was very positive; he holds the view that it will help with memory and not regurgitating information. This is a proven fact statistically as when humans enjoy something they are far more likely to retain the information in their brains than if not.

Sarrujan's response was also very positive as he believes that a game such as my proposed one would allow a student to learn the fundamentals very well, thus allowing them to perform better at harder and more challenging questions.

The last bit of insight I received was that students would be willing to use the game if it was interactive and had a "fun twist" to it, which my game will hold as it has a game-like objective and graphics

Analysis of educator responses

Although my educator interview was very short due to the teacher being very busy, it was still very useful. I was able to gain a little bit more perspective towards my project and the ideal solution. The overall summary of the opinion I received was that there is a definite lack of fundamental understanding when it comes to mechanics in

physics (she is a physics teacher) and that she believes her students would most definitely benefit from using a better revision tool such as an educational game. The way she envisioned the solution was “Using trajectories to hit a target.”, which is pretty much the skeletal summary of my solution. I believe the reasoning of her reference to a target was that it makes the game much more objective, and that students do not typically do well in situations where the solution is aimless, such as doing questions on a test paper; therefore, her belief that it will enhance their skills in the understanding of the topic.

Analysis of regular/interested person responses

Although I am only able to collate one regular/interested person response in my given time frame, also the other person I had requested has not replied, I do feel as though I have enough information to build his case and present it in my solution and that a representative view has been given.

By the interview I received the knowledge that most general people are interested in the topic of mechanics, this may just be because it sounds smart or they genuinely find it interesting as I believe the perspective here is. I can also infer that many regular people would be more than willing to learn the game in the form of an educational game. The reason that my interviewee presented was that “it would help with my memory of it”, meaning that having seen a graphical representation of the subject in hand as well as having some fun while learning would help them retain the information better.

Programing language

For my solution, I have decided to go with the programming language of JavaScript. As I am building a browser-based game there will also be aspects of HTML and CSS code alongside the main language of JavaScript.

I have decided to opt for JavaScript as my language of choice as it is the main language for browsers, a browser is a tool that renders JavaScript on the client-side so I physically could not build a browser game and not incorporate JavaScript as it is essential. JavaScript has been used for years for anything to do with client-side web processing as it is simply the best and most efficient way to build an online game. Most browser games since the release of jess have been built in it and most have had major success. Furthermore, there is an extensive community of developers behind jess, meaning there are thousands of tutorials and help sites that I could visit if I am ever in need.

In conjunction with JavaScript, I have opted to use the phaser.js game engine alongside it, this is because after some initial research I had concluded that it would be the easiest to learn as it is not very complicated and there are many tutorials for it as well, also the fact that I have experimented with it in past and my experience was very positive.

The IDE I will be using as my coding platform is “atom”. This is a new and very capable IDE that many developers recommend in conjunction with JavaScript and Phaser.

Mid project amendments - After learning JavaScript in conjunction with phaser I found it quite difficult to implement into my specific use case. One of the reasons for this is that the phaser has its inbuilt function for gravity (xxx. gravity), this means that I was not able to change the specified value of g (9.81) set by the phaser module. This defeated the objective of my game and therefore I decided not to carry on with JavaScript.

Instead, I have opted for the python language, in conjunction with pygame. This mainly was because I am already very familiar with the language and its syntax, as well as knowing that I am freely able to modify the gravity module calls. Also, although I had spent a considerable amount of time learning and understanding all the phaser calls, I still found myself tripping over myself and it took me a long time to code a simple block of code.

Due to the language change, this has also resulted in an IDE change. Currently, I am using the standard python 3.9.1 IDLE, however, I am attempting to get pycharm/Atom to work as it is a much more powerful development platform.

The Pygame module will be used as it is a very simple python module and it allows me to simply display images, run algorithms and modify function values all with familiar syntax and calls.

Requirements

Hardware requirements

A device with a suitable operating system would be required, these systems would include Windows, macOS, Linux Ubuntu, android and IOS or any other mainstream operating system.

A device capable of browsing and running an online game, with touch/mouse and keyboard controls. The game will use the computer processor to carry out all the complicated logic and arithmetic calculations that will be done, a standard desktop or mobile device should be capable of this. The device will have to have a way to input characters and select areas and output a display, normally done through a keyboard, mouse and monitor respectively; or if on a mobile device touch controls and a pop-up keyboard will suffice as most modern devices do have them.

The system would be required to have a minimum of a Pentium 4 processor (or higher), 3Gb RAM, and no storage requirements (assuming a browser is already installed). I have set these requirements as google chrome will only run on a processor chip that is newer than Pentium 4, the browser's minimum RAM

requirement is only 125MB however in typical usage and for a smooth gaming experience 3GB RAM would be recommended. The storage requirement for chrome is 100MB, but we are assuming that a browser is already installed. Lastly, the system would also require some sort of graphics card, whether integrated or dedicated.

Mid project amendments - Due to my language change, this game will no longer be a browser game as pygame does not support browser implementation. I will list the new hardware requirements as follows:

- *Modern operating system (such as MacOS, Windows, Linux, etc.)*
- *x86 64-bit CPU (Intel / AMD architecture)*
- *Minimum base clock of 2 GHz is recommended*
- *4 GB RAM*
- *5 GB free disk space*
- *Integrated or dedicated graphic chip*

Software requirements

Suitable operating systems that are capable of running browsers. In the normal case the operating systems would include windows, Linux, macOS, android and IOS.

These are all selected as they can all run browsers efficiently.

Google chrome / browser - My game will be optimised for the google chrome browser meaning that if another browser is used it may not run as smoothly as it should but should still work effectively.

JavaScript - the operating system and the browser should already have this installed and you should not need this externally, however, if you wish to run the game in a shell then it will be needed externally.

phaser.js - this is the JavaScript module I intend to use as my main game engine framework.

Atom IDE - this is the idea I will be using to write my code

Direct X 9.0 (or higher) (windows machines) - this is the software that enhances game graphics and all browser run games utilise it.

Adobe flash drivers - the browser should already have this up to date and installed, if an error is presented then a different version will be needed.

Mid project amendments - Python 3.9.1 - this will need to be installed on the computer and added to PATH.

Pygame - this should be installed alongside the IDLE installation, however, I may need to be installed separately as a binary. It will need to be installed as a separate package for atom use.

I will still be using the Atom IDE

Stakeholder requirements

These are the requirements that the stakeholders have stated in the interviews as they have said them as quotes, I will simply explain what they mean beside. These requirements are very vague and I already have most of them in my own success criteria, however I will still list the advice I have been given.

Stakeholder requirement	Explanation
Graphical representation	Will allow the users to retain better memory regarding the topics learn (visual learning)
Have an objective (target)	Students tend to not enjoy aimless questions, if they have an 'fun' objective to meet then they are more likely to be motivated as well as remember and understand the information
"interactive"	The game must have an (graphical) user interface in which the user will deal with inputs and making the game more hands on and 'interactive'
"Questions but with a fun twist to it."	It cannot simply be a dull question and answer type of game, there has to be some excitement element to it. This would be the target and getting past the AI defence is my solution.
"to be able to work with projectiles"	To be able to directly manipulate and control projectiles. This is a very simple requirement as this is the very basis of my game.

Success criteria

no.	Requirement	Justification
1	Start-up screen	This is needed to show the user the game that they are playing

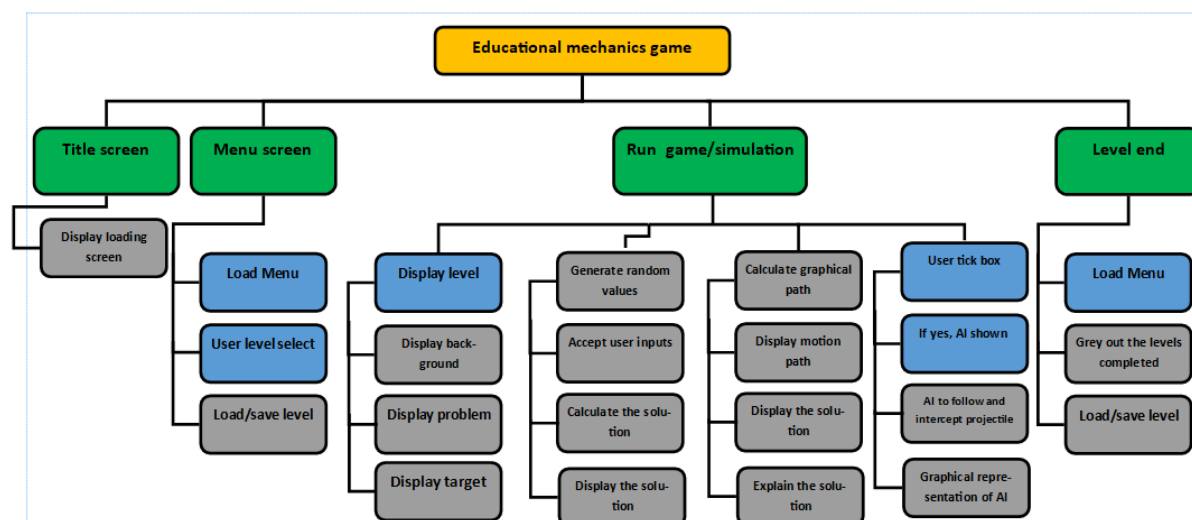
2	The menu allows game level selection	This is so the user, if confident in a topic will not have to do it just because they have to. For it to remain fun they should be able to choose what they need to do themselves
3	Displaying the problem with randomly generated values	This is simply so that the question is never the same so the user will not get bored if they decide to revise repeatedly
4	Use random and differing variables in the questions	To keep the questions unique you would have to switch around the variables given and values to be calculated in the question
5	Display the solution to the problem after attempt	This is to explain to the user how to reach the solution so they will gain an understanding and hopefully get better at that topic question
6	If the user gets questions wrong, then redo the level with a new problem. If user gets it right they should be presented with the choice to advance or not	If the student gets the answer wrong then that indicates that they do not know how to reach the solution, so once the solution is explained to them they should be able to go back and do another question to fully grasp how to do it. If the user is correct they could advance but if they are revising they may just want to repeatedly do that one topic again until they wish to stop.
7	The background/setting will be according to the level and values	The first topic/level would just be a simple elevated platform from which the launcher would shoot, as the topic gets harder the launcher would move to the ground to factor in the first half of the parabola and other factors to represent that specific topic.
8	Allow user to enter their calculated answers	To complete the level and advance you will have to calculate the answers to the missing variables and enter them
9	Display the target and give the value needed, such as horizontal displacement of the target (value given will be randomly generated)	This is the main aim of the game, the user being able to calculate values so the projectile will hit the target, the values will need to be given so the user can calculate an answer
10	Calculate the path of the projectile in terms of an axis graph	This is so it will later be able to plot these values to graphically represent the motion

11	Represent the projectile path graphically by using the calculated path	This will be when the project is shown to be fired and in motion, the game will trace the path so that the student can understand and see its motion and see how different values affect it.
12	Incorporate a firing animation when the projectile is released	This is simply to make the game more vibrant and interesting for the student and to keep his interest
13	Use the calculated x and y values to determine whether the target was hit or not	This is to see whether the user actually achieve the aim and passed the level or whether they need to redo the level
14	Incorporate an exit button in the corner that will exit the level back to the main menu	This is needed to allow the user to quit the level if they do not wish to play anymore or maybe even reset the question
15	Have a user-selectable box named "defence ON/OFF"	This is so the user is able to decide whether or not they would like to make the game a little harder by having an defence for the target
16	Successfully implement the AI defence system framework	This will make the game more complex and interesting to the student so that they have something to work around, making them work harder and be more precise. This will be measured during the gameplay, by simply seeing if the AI interception system appears and is reactive.
17	Have enough levels to cover and explain the whole mechanics' syllabus to the student	The whole aim of my solution is to teach the user the topic of mechanics, so the game will cycle through the different aspects of it till they have mastered the whole subject
18	Set the speed of the AI to a specific value	Have a limit on the maximum speed of the defence system so the user can work around that and finish the level. This will be measured in-game, by using a testing specific algorithm that will display the speed for testing diagnostics only. The speed should not surpass a specific level.
19	Do not correlate the AI parameters to that of the user/question	The AI should not have a calculated intercept or as such, as then it would be a simple linear algorithm. By using a recursive algorithm, the non-linear criterion. it should simply launch when the

		<p>projectile enters the iron dome, then will follow it and hit it from behind as it would be faster</p> <p>I will diagnose this in the gameplay by reading print statements throughout the algorithms.</p>
--	--	---

Design

Systems diagram



For my system structure diagram, I have opted to use a broad top-down style diagram, this is because it will help me to identify the different code modules and break the problem down allowing for easier focus and development. This method will also allow me to make use of the computational skill of ‘thinking ahead’ and ‘thinking logically’ as I can determine inputs and outputs as well as look at where development decisions are required and the knock-on effect that they would have. This will provide my working solution with structure, allowing for a systematic and efficient development process.

Module explanations

Title screen

display loading screen

This is a method that while the browser renders the JS code, a simple loading title screen will be displayed with the name on it and a loading bar.

Mid project amendments - This will now be done in pygame and not in a browser.

Menu screen -

Load menu

When past the title screen, you will be presented with a menu select screen so you can select the level desired

User-level select

Once the menu is loaded the game will accept a user click input to select the desired level

load/save level

This subroutine will simply load the clicked level and keep track of this for later need

run game/simulation -

Display level

This is simply to display the whole level that has been selected in the menu, this will be done by the three subroutines of displaying the background, displaying the problem and displaying the target. The first subroutine will simply set the scene so the user can graphically interpret the solution. The second subroutine will display the problem as per the level and fill it in with random variables and values to ensure it is unique every time. The last subroutine is more generalised with the graphical side, this will display the target to be shot as well as the propellant (missile launchers) and other miscellaneous items.

Accept user inputs

This will be achieved through the user input boxes for each SUVAT variable, these will be the final answers and what will be used for the rest of the game. Invalidation will be carried out on the inputs.

Calculate the solution

This will simply take the user input values and plug them into the appropriate formulae to reach an answer, this answer will be used to compare the actual answer and create the basis for the graphical representation.

Calculate graphical path

This will be done using the variables the user has inputted. By varying those values, it will determine multiple points (as coordinates) along the path of the projectile, for later use.

Display motion path

This is where the coordinates generated from the previous subroutine will be used to plot against an axis grid. The path line will be plotted and the projectile will be shown to follow it live.

display/explain the solution

This is the subroutine that will display the correct answer and how to work it out, alongside the explanation of the process of how to reach that answer.

User tick box / AI

This will be displayed for the user to select whether they wish to encounter a defence system or not. If the box is ticked it will trigger the following subroutines

AI to intercept / graphical

The AI will be launched depending on the tick box. It will follow the motion of the projectile without any calculations. This will be shown graphically live as an 'iron dome' situation.

Level end

Load menu

Will load the menu select again

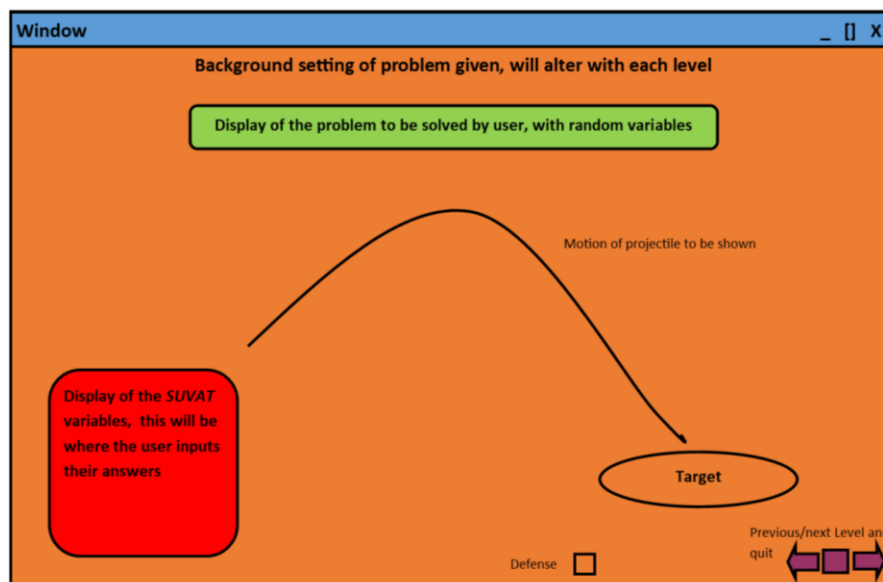
Grey out completed levels

This subroutine links to the very start and will grey out any level that has already been completed, the user can still redo it, just inform them.

load/save level

This is the same subroutine as at the start. The values will be updated and the process will repeat

User interface design



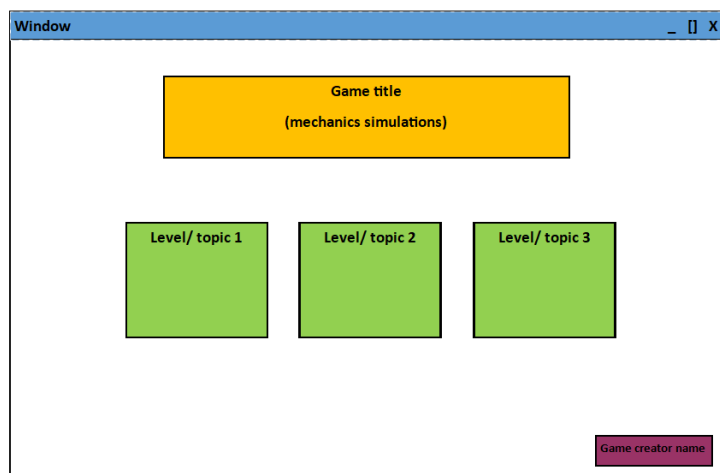
This is the main design of my program during a level, due to the different levels and requirements the background will change to felicitate each level/topic. This will allow the user to have a simple interface in which they can see the problem, input solutions, see the graphical motion and then see the explanation. The menu and other levels can also be accessed in the corner.

Links to the success criteria:

3. Displaying the problem with randomly generated values

4. Use random and differing variables in the question
5. Display solution to the problem after attempt
7. The background/setting will be according to the level and values
8. Allow user to enter their calculated answers
9. Display the target and give the value needed, such as horizontal displacement of the target (value given will be randomly generated)
11. Represent the projectile path graphically by using the calculated path
14. Incorporate an exit button in the corner that will exit the level back to the main menu
15. Have a user-selectable box named “defence ON/OFF”

Menu select



This will be the main navigation screen the user will use to go through the game. The green tiles will represent the topics/levels and there will be much more of them that you can scroll horizontally through. Each level will be according to one part of the syllabus and will go through the whole specification. The game title will just be ‘mechanics simulator’ and the game creator name will be ‘Zain Mughal’.

Usability features

The features I have considered make sure that this game is easy to use and navigate by anyone, regardless of prior experiences.

Firstly, I will make the buttons fairly large, both to not hinder gameplay but also very quick and easy to locate and use; this would be better for older, younger or slightly handicapped users as hand stability and navigation is not the best.

Any text to be displayed will be very large and readable, also grammatically simple so everyone can understand the text. The text will also be in contrast to the

background perhaps by a caption box or different text colour, as with different settings a black text can blend in and can be difficult to read.

The display in all aspects will have a very distinct and contrasting colour scheme, this is to make sure that all users (especially vision-impaired users) can easily understand and see the content displayed. The above sketches do show a very simple colour scheme that I am likely to differ from in the final product as I develop it. Due to my target audience, this program needs to be very user friendly and simple to use, thus why I have kept my menu and game screen very simple and minimalistic. There is not much to go wrong so users should not get lost or confused in navigation.

This represents the success criterion of 1 and 2.

Stakeholder input

At this point in my project, I am aware of how I intend for my final solution to look like, including the designs of the interface and how it will operate.

For a critique and any improvement advice, I will be reaching out to my stakeholders here, to seek their opinions. In my email to them I wrote:

“(attached the design pictures shown before)

Would you please look over this initial design of how I intend for my solution to look alongside these extra features?

The menu select will have a feature where once a level is completed it will be greyed out so the user is aware it has been completed. Also, the tiles will scroll horizontally to span the 14 different levels/ tiles.

When the user hovers over the tile it will show the image of the level before they load in.

Could you provide your thoughts on this please in the form of an email reply?

Thanks “

Sarrujan Raguparan

“Overall I would say the designs look good and well thought out, and how I would imagine the game to be like. Also, the few extra features that you highlighted will be a great idea to integrate. However, from the little I have seen it does look very bland when you go on to actually make this, I would suggest switching up the colours and maybe using an interesting background to make it stand out more. Although overall the direction looks good.”

Yassine Soltani

“Very minimalist design, however with an educational purpose and the timeframe it seems acceptable. Do not make the game distracting by including fancy backgrounds as then it will be off-putting from the whole purpose of the game, to educate. The design does look clean and typical of a small game. The extra features will help the efficacy of the game”

Algorithms

Mid project amendments - Despite a language change, the pseudocode remains unchanged

In this section I will list and explain the main algorithms and subroutines that I will be using in my program, I will also produce pseudo-code for them so I have a reference frame and better ideas to work on in the next stage of development. I will approach this task in a modular approach, meaning that I will be abstracting and simply producing pseudo code for the main subroutines and functions in the structure diagram. When these modules are connected they will form a fully functional program.

Display loading screen

This is a very important module of the program as the game would not run without it; however, it is not a very complicated function and would only be 2 or 3 lines of code, therefore a pseudo-code reference would not be needed in this case.

The code would simply load the background with a moving loading bar.

Load menu

This is also another module that is very simple to implement and a pseudo-code sample is not needed. This module would be responsible for loading up a background and some linked tiles that will be predefined.

Display problem

This algorithm is responsible for displaying a problem to the user depending on the level.

```

FUNCTION DisplayProblem(level):
    IF level = 1:
        question = str("first part of question
here",s=random.randint(1,10000),"second
Part of
question",u=random.randint(1,10000),"final part of
question")
    ENDIF
# above is the general structure of a question where the
randoms would be the random variables needed for the
question and saved for later use.
    IF level = 2:
        question = question = str("first part
of question"),
s=random.randint(1,10000),"second part of
question",u=random.randint(1,10000),"third part
of
question",v=random.randint(1,10000),"
final part of question")
    ENDIF
END FUNCTION
# there will be a IF statement for every level as the
question and variable places would be different. This
will essentially be repeated for all levels just with
different question 'parts'.

```

Display target

This algorithm will feed off the saved random question variables and produce a graphical target according to the values

```

FUNCTION DisplayImage(s):
    X = s # saved previously
    Y = 0 # may change depending on the map however
the grid should be adjusted so
    Display image.target at (x,y)
END FUNCTION

```

Accept user inputs

This algorithm will function inside the SUVAT variable box on the side to provide an input area for the user to enter their answers.

```

IF Level_in_progress = True :
    Change css file to levelinprogeess.css           # name of file
# appropriate html tags
# html form

    S = INPUT("")
    U = INPUT("")
    V = INPUT("")
    A = INPUT("")
    T = INPUT("")

ENDIF

```

calculate the solution

This module is at the very heart of my entire program as it will be what enables it. It will take the values given (or not) in the question, then use them to calculate the solution to the question. This will also determine whether the user is correct or not. This algorithm will be different depending on the level as they will all need to calculate and take in different variables.

```

Function calculate(s,u,v,a,t,θ):
    IF level =1:           # this level will be
calculating the components of the force/velocity
        Hcomp = ucosθ
        Vcomp= usinθ
                                # this will carry on
for every level
    IF level = 5           # this level will be falling
vertically with no initial vertical velocity
        t= v/a           # v=u + at , u=0 so rearrange to
get t
    ENDIF
END FUNCTION

```

Calculate graphical path

This module will be implemented after the solution has been calculated. It will take those calculations to produce X and Y points along the trajectory of the projectile to be plotted and shown visually

```

Function Gpath(s,u,v,a,t,θ)
    increments = s/15
    Xpos = (0,0,0,0,0,0,0,0,0,0,0,0,0,0,0)
    Ypos = (0,0,0,0,0,0,0,0,0,0,0,0,0,0,0)
    For i in range (1,14)
        Xpos[i] = increments * i
        If i (u**2 + 2*a*s)**½ = 0:
            u=0
        Ypos[i] = u(t/14 *i) + (a(t/14 *i)**2)/2
    END FOR
END FUNCTION

```

Display motion path

This module is responsible for taking the points calculated in the previous function and plotting them using CSS and creating the displayed 'visual' aspect

```

# appropriate html/js tags
FUNCTION Display_Mpath(Xpos,Ypos)
    FOR x in Xpos
        Display image.dot.jpg at (Xpos[x],Ypos[x])
    END FOR
END FUNCTION

```

display/explain the solution

This procedure will list out the correct answers as well as explain how to get there

```
FUNCTION explain_answers(s,u,v,a,t)
    print("The displacement(s) was",s,"the initial
    velocity(u) was",u,"the final velocity(v)
    was",v, "the acceleration(a) was",a, "and the time
    elapsed (t) was",t)
    IF level = 5
    # this will be for all levels, for demonstration is
    5
        print("for this question we will be
        using the  $v^2=u^2 + 2as$  formula, as it is free
        fall and the initial velocity is not defined as
        the object is released from rest we take  $u = 0$ ,
        this then leaves us with  $v^2= 2as$ . If v is
        wanted simply rearrange into  $v= (2as)^{\frac{1}{2}}$  or if
        s is asked for use  $s=v^2/2a$ . When needed
        substitute in the values given in the question
        to get the answers"
    ENDIF
END FUNCTION
```

User tick box/AI

This function simply provides the user with a checkbox to ask whether they want the AI defence system on or off

```
#use html code to create a checkbox for the AIcheck
variable

FUNCTION AIcheckbox(AIcheck)
    x = AIcheck
    IF x.checked = true;
    AI_on=True
    ENDIF
END FUNCTION
```

AI to intercept (graphically)

This module depends on the previous checkbox, if on the AI will follow the projectile in aim to extinguish it

```

Function AI(current_projectile_Xpos,
current_projectile_Ypos)
    IF AI_on = True and inflight = True :
        AI_Xpos = s
        AI_Ypos = 0
        differenceX= current_projectile_Xpos
- AI_Xpos
        differenceY= current_projectile_Ypos
- AI_Ypos
        move_AI_X = differenceX/
random.randint(1, (u/2))
        move_AI_Y = differenceY/
random.randint(1, (u/2))
        # this is so the AI will not always
win as is a bit random, also it makes sure the
AI will never be faster than the projectile
making the level achievable
        AI_Xpos += move_AI_X
        AI_Ypos += move_AI_Y
        IF AI_Xpos=current_projectile_Xpos and
AI_Ypos=current_projectile_Ypos:
            Collision = True
            Display image.collison at
(AI_Xpos,AI_Ypos)
        ENDIF
        IF collsion = false:
            Display image.projectile at
(AI_Xpos,AI_Ypos)
        ENDIF
    ENDIF
END FUNCTION

# this is not the complete AI block code, through
development further complexity will evolve

```

Grey out completed levels

This subroutine links to the very start and will grey out any level that has already been completed, the user can still redo it, just inform them. The pseudo-code will not be needed for this as it is a simple function.

load/save level

This is the same subroutine as at the start. The values will be updated and the process will repeat. The pseudo-code will not be needed for this as it is a simple function.

Explanation and justification of this process

Initially, my problem has resulted in a solution that was very complex and hard to approach from a developer perspective, however, I have managed to decompose and abstract the initial problem into smaller chunks that are much more manageable. This will allow for easier development as well as long term maintenance.

I have decomposed the problem into 2 main halves, arithmetic calculation and graphical representations. Some functions will have both of these in one as it is simpler and more efficient to do so. The calculations will be what power the program by taking in values and providing outputs for the graphical modules to use, which will then take these calculations and 'plot' them in a visual manner allowing for a clear graphical representation.

In breaking the problem down into sizable chunks and modules that are all capable of interlinked communication. This means if an issue arises it will be very simple to locate the problem and quickly solve it without having an effect and possibly interfering with the rest of the code.

My coded solution will be inextricably modular which will allow for easy development and ease of change if needed down the line. This will be done by isolating all the different modules mentioned beforehand, making sure that they are all in separate functions. I may also decide to use global variables for things such as the SUVAT variables and such as it will make development much easier.

I do not currently intend to adopt an object-oriented approach to my code, however, this is certainly subject to change depending on the success of my initial coding.

The computational thinking incorporated with the development of my project is a huge upside, as it will ultimately result in a flawless solution in the end.

Validation checking

Validation check	How it works	Example
Format check	The data type is correct and obeys expectations	If the string is expected, then would reject integers or Boolean
Length check	The number of characters is not too long or short	A value that shouldn't be over 100000, so is

		rejected
Presence check	Data has been entered	Must enter a value
Lookup table	restrict inputs	Choose from a drop-down list
Sanitisation	Converting the input syntax/format to valid	If a string is given in place of an integer, can convert the string into the desired integer

Key variables

These are the main variables to be used by the algorithms

Name	Data Type	How it is used	Input Validation
s	integer	This will be assigned to the random number generated at the start of the level, or assigned 0 if the variable is not given. The user can update the value of the non-given variable to get the answer	Format check Length check Presence check
u	integer	This will be assigned to the random number generated at the start of the level, or assigned 0 if the variable is not given. The user can update the value of the non-given variable to get the answer	Format check Length check Presence check
v	integer	This will be assigned to the random number	Format check Length check Presence check

		generated at the start of the level, or assigned 0 if the variable is not given. The user can update the value of the non-given variable to get the answer	
a	integer	This will be assigned to the random number generated at the start of the level, or assigned 0 if the variable is not given. The user can update the value of the non-given variable to get the answer	Format check Length check Presence check
t	integer	This will be assigned to the random number generated at the start of the level, or assigned 0 if the variable is not given. The user can update the value of the non-given variable to get the answer	Format check Length check Presence check
Level	Integer	Depending on which level tile the user selects in the menu, this variable will be assigned according to value. This will be used to determine the input and outcome of other algorithms such as displayproblem()	Presence check
level_in_progress	Boolean	This is a data	Format check

		value with a default of False, when a level is selected then will be turned to true, this will determine the appearance of the scene and contributes to other algorithms	Presence check
θ	Integer	This mathematical symbol (theta) is used to show the angle of the projectile, this again will either be randomised or be calculated by the user.	Presence check Format check Length check (3 digits)
Xpos	List	This is a default list of 14 0's. When the algorithm is run the list indexes will be calculated and redefined to the values. Will be used in other routines later.	N/A not an input
Ypos	List	This is a default list of 14 0's. When the algorithm is run the list indexes will be calculated and redefined to the values. Will be used in other routines later.	N/A not an input
AI_on	Boolean	This value will be defaulted to false, and its value depends on the user check box, if ticked = True. when this variable is True then it will trigger another AI	N/A Technically not a user input Checkbox automatically meets validation criterion

		subroutine.	
Inflight	Boolean	This value will be set to true whenever the projectile is being graphically displayed on the screen, this is so the AI can sense that and attempt an intercept.	N/A not an input
AI_Xpos	Integer	This is simply the live position of the AI, so the algorithm is able to calculate how to intercept the projectile.	N/A not an input
AI_Ypos	Integer	This is simply the live position of the AI, so the algorithm is able to calculate how to intercept the projectile.	N/A not an input
current_projectile_Xpos	Integer	Holds the value of the live coordinate position of the projectile so the AI is able to see it and calculate the intercept accordingly	N/A not an input
current_projectile_Ypos	Integer	Holds the value of the live coordinate position of the projectile so the AI is able to see it and calculate the intercept accordingly	N/A not an input
Collision	Boolean	This is a value that will be set to true if the AI successfully	N/A not an input

		collides with the projectile. This will determine what sprite will be shown and the outcome of the level.	
--	--	---	--

Testing method

Throughout the development of my solution, I should be making sure that it all works as planned. To do this I will test each and every function and procedure as it is created and modified to make sure that it works as intended. This ensures that when the functions are complete and are collated into the final program, that it will function. Further testing may find bugs or improvements (alpha/beta testing).

The testing to be carried out during development should include that input data to the software and the results. I will start by using the white box testing method, by simply making sure the code within the function seems correct. Upon consolidation of each function, I will conduct black-box testing in which the pure input to outputs are tested to see whether it produces the correct result.

In the case of debugging and errors, I will use a multitude of print statements and IDE tools to help debug the code and solve it.

After complete development, I will carry out a final comprehensive test that will push the program making sure it all works as one unit.

I will also be using destructive testing techniques, this is where I will be entering incorrect values and abusing the system to see whether it can cope or not.

I will record any test data that I input and also record the results, commenting on the success of the trial.

RAD development

As I am using the RAD development methodology, it will be highly iterative. This means that I will be breaking the problem down (already shown and explained in the design section) and tackling each in an iterative process. This means that after each iteration it will be tested to eventually create a prototype, this will be repeated till the program is complete.

Testing to be used in iterative development

Here are the main testing milestones I will use throughout the development:

1. Allow for user-level selection
2. Successfully display a problem and its setting according to the level (syllabus)
3. Be able to input values and respond accordingly
4. Apply the correct formula and solve the problem, Explain the solution afterwards
5. Graphically calculate and represent the motion
6. Have a non-linear AI defence system to follow the projectile and intercept (if ticked yes)
7. Give a win or loss screen
8. Progress back onto the main menu, or next, or previous or re-do

Milestone 1: allow for user-level selection			
Test number	Test data	What to test and inputs	Expected results
1	LMC (left mouse click) on the game script	If the menu select pops up and is as described in the sketch, load the game	The menu should pop up on its own and look similar to the sketch
2	LMC (left mouse click) on level 1 tile	If the menu allows the user to select a level, select random levels (repeat)	The level selected should load
3	Scroll up/down (on mouse)	If the menu allows for horizontal scroll and has all the syllabus topics	The menu should cover the whole specification

Milestone 2 : Successfully display a problem and its setting according to the level (syllabus)			
Test number	Test data	What to test and inputs	Expected results
1	LMC (left mouse click) on level 2 tile	If the selected level dictates the displayed problem(Depending on the level selected the subroutine should

		vary the levels - inputs)	display a value (question) from a 2d array of all questions
2	LMC (left mouse click) on level 2 tile	If the setting is in accordance with the problem (vary the levels - inputs)	Depending on the question displayed the setting should be the correct graphical interpretation of it
3	LMC (left mouse click) on level 2 tile and refresh	If the displayed problems are unique, select one level multiple times	The same question should never have the same values in the problem

Milestone 3 : Be able to input values and respond accordingly			
Test number	Test data	What to test and inputs	Expected results
1	Input a random integer value in the SUVAT input boxes	If the input boxes are present, in all levels	The input SUVAT box should be constant and in the corner of every level
2	Input a random integer value in the SUVAT input boxes	If the boxes accept valid inputs for the variables	The inputs should only be of the variables asked for and must be rejected if non-integers
3	Input value '5' into angle then again value '70'	Check if the variables actually make a difference to the path of the projectile and if they are shown live	The projectile should change behaviour depending on the inputs, the variables should be shown live throughout the projectile's path

Milestone 4 : Apply the correct formula and solve the problem			
Test number	Test data	What to test and inputs	Expected results
1	Free fall - Enter value '10' in displacement and value '9.81' in acceleration.	The program should apply different formulas depending on what the given variables are and the context (most levels should only use one formula) Should use $v^2 = u^2 + 2as$	The correct formula applied Correct results
2	LMC (left mouse click) the solution button	The formula and problem should each have corresponding explanations	The explanation must be correct to the problem and formula shown
3	LMC (left mouse click) the solution button	Use of alternate formulae	Every possible formula to solve that question should be explained, showing every way it could be solved

Milestone 5 : Graphically calculate and represent the motion			
Test number	Test data	What to test and inputs	Expected results
1	Input values for SUVAT	If the program calculates the coordinates of the path correctly using the formula and indent substitution	Print command in the coordinate function should show all the points on the curve
2	Input values for SUVAT	The calculated coordinates should be plotted on a hidden axis graph, the curve of best fit	After value inputs should see a typical flight curve displayed

		drawn	
3	Input values for SUVAT	If the projectile (missile) graphically follows the curve	Should see the missile going along the pre-drawn curve of motion

Milestone 6 : Have a non-linear AI defence system to follow the projectile and intercept (if ticked yes)

Test number	Test data	What to test and inputs	Expected results
1	Input values for SUVAT	The AI program must not calculate the path, or use pre-existing coordinates to intercept (print commands)	The code must be in isolation, AI follow the projectile, not predict its movement
2	Input values for SUVAT	AI must simply use the current x, y of the missile and try to catch up to it (print commands)	AI will follow the projectile, AI values will not correspond till the collision
3	Input values for SUVAT	AI must have a set acceleration, so will not always be able to catch the projectile Use multiple values of speed to see whether AI is beatable	The AI should be able to be beaten with a perfect flight path (values)

Milestone 7 : Give a win or loss screen

Test number	Test data	What to test and inputs	Expected results
1	Input correct values for SUVAT, then incorrect	If the user is presented with a win or loss screen depending on the	Loss screen should only be shown after a failed attempt

		result Values for both win and loss	Win screen at the correct answer
2	Input values for SUVAT	If the solution is explained on the win/loss screen	Regardless of win/loss, the solution will be displayed before moving on

Milestone 8 : Progress back onto the main menu, or next, or previous or re-do			
Test number	Test data	What to test and inputs	Expected results
1	Input values for SUVAT LMC (left mouse click) on back, forward, main menu, redo buttons	On the win/loss screen to be presented with the option to go back, forward, main menu or to redo	These buttons to be presented and work
2	Input incorrect values for SUVAT LMC (left mouse click) on the redo button	On the loss screen, you should not have the option to advance	Cannot carry on until correct answer, loss screen the next button be greyed out

Test data

Test data	Expected outcome
LMC (left mouse click) on level 1	Valid
LMC (left mouse click) on level 2	Valid
LMC (left mouse click) on level 3	Valid
LMC (left mouse click) on level 4	Valid

LMC (left mouse click) on level 5	Valid
LMC (left mouse click) on level 6	Valid
LMC (left mouse click) on level 7	Valid
LMC (left mouse click) on level 8	Valid
LMC (left mouse click) on level 9	Valid
LMC (left mouse click) on level 11	Valid
LMC (left mouse click) on level 11	Valid
LMC (left mouse click) on level 12	Valid
LMC (left mouse click) on level 13	Valid
LMC (left mouse click) on level 14	Valid
RMC (right mouse click) on level 1	Invalid
RMC (right mouse click) on level 2	Invalid
RMC (right mouse click) on level 3	Invalid
RMC (right mouse click) on level 4	Invalid
RMC (right mouse click) on level 5	Invalid
RMC (right mouse click) on level 6	Invalid
RMC (right mouse click) on level 7	Invalid
RMC (right mouse click) on level 8	Invalid
RMC (right mouse click) on level 9	Invalid
RMC (right mouse click) on level 11	Invalid

RMC (right mouse click) on level 11	Invalid
RMC (right mouse click) on level 12	Invalid
RMC (right mouse click) on level 13	Invalid
RMC (right mouse click) on level 14	Invalid
LMC (left mouse click) on quit button	Valid
RMC (right mouse click) on quit button	Invalid
Enter value '1' in displacement input	Valid
Enter value '0' in displacement input	Valid (boundary)
Enter value '1000' in displacement input	Valid (boundary)
Enter value 'abx' in displacement input	invalid
Enter value '999' in displacement input	Valid
Enter value '0999' in displacement input	Valid (boundary)
Enter value '1999' in displacement input	Invalid
Enter value '655987138749' in displacement input	Invalid
Enter value '1' in initial velocity input	Valid
Enter value '0' in initial velocity input	Valid (boundary)
Enter value '1000' in initial velocity input	Valid (boundary)
Enter value 'abx' in initial velocity input	invalid
Enter value '999' in initial velocity input	Valid

Enter value '0999' in initial velocity input	Valid (boundary)
Enter value '1999' in initial velocity input	Invalid
Enter value '655987138749' in initial velocity input	Invalid
Enter value '1' in final velocity input	Valid
Enter value '0' in final velocity input	Valid (boundary)
Enter value '1000' in final velocity input	Valid (boundary)
Enter value 'abx' in final velocity input	invalid
Enter value '999' in final velocity input	Valid
Enter value '0999' in final velocity input	Valid (boundary)
Enter value '1999' in final velocity input	Invalid
Enter value '655987138749' in final velocity input	Invalid
Enter value '1' in acceleration input	Valid
Enter value '0' in acceleration input	Valid (boundary)
Enter value '1000' in acceleration input	Valid (boundary)
Enter value 'abx' in acceleration input	invalid
Enter value '999' in acceleration input	Valid
Enter value '0999' in acceleration input	Valid (boundary)
Enter value '1999' in acceleration input	Invalid
Enter value '655987138749' in acceleration input	Invalid

Enter value '1' in time input	Valid
Enter value '0' in time input	Valid (boundary)
Enter value '1000' in time input	Valid (boundary)
Enter value 'abx' in time input	invalid
Enter value '999' in time input	Valid
Enter value '0999' in time input	Valid (boundary)
Enter value '1999' in time input	Invalid
Enter value '655987138749' in time input	Invalid
LMC (left mouse click) on redo button	Valid
RMC (right mouse click) on redo button	Invalid
LMC (left mouse click) on next level button	Valid
RMC (right mouse click) on next level button	Invalid
LMC (left mouse click) on previous level button	Valid
RMC (right mouse click) on previous level button	Invalid
LMC (left mouse click) on level solution button	Valid
RMC (right mouse click) on level solution button	Invalid
LMC (left mouse click) on quit button	Valid
RMC (right mouse click) on quit button	Invalid

LMC (left mouse click) on main menu button	Valid
RMC (right mouse click) on main menu button	Invalid

Post-development testing

After completion of coding my solution, I will carry on the testing until I am fully satisfied.

I will conduct alpha testing by myself against the success criteria defined in the analysis section. I will make sure that the program will function as intended and does not have any major flaws.

I will then also conduct beta testing as I will release my program to my stakeholders and conduct an interview after they have used the software for some time. I will record the interviews and analyse responses to then be acted upon. The beta testing will provide some final refinements to smoothen the program out.

I will also use the method of integration testing to make sure that all the modules of the program work together in a seamless fashion without any flaws.

This is my overall development document. Here is where I will be documenting my progress as I work through the coding of my solution. I will show my working through each milestone that I have previously defined within my design stage. I will also be showing my test data and validation alongside each milestone.

Development of coded solution

This is the section in which I will actually start my coding and hopefully finish it. I will be documenting all my progress here and talking through each major milestone as I achieve it. Before this I have had quite little programming practice due to me not even doing GCSE computer science, although I have learnt python in the last year at school and home and some basics of the other languages.

Milestone 1: allow for user-level selection

In my design, I had specified that I aim to create a user interface menu in the form of a horizontal scrolling rollover tile system. However, due to a language change from JavaScript to python, this is no longer possible. In JavaScript, this is not a difficult task however as python is not a graphical language this is a major undertaking that I do not yet know or understand how to achieve.

In JavaScript, there is a module called horizontal scroll which essentially allows you to make a horizontal scrolling bar in very few steps as shown below.

```
HTML
```html
<div class="container">
 <div class="block"></div>
 <div class="block"></div>
 <div class="block"></div>
 <div class="block"></div>
 <div class="block"></div>
</div>
// these div classes define each instance within the menu as an item to be clicked on.
```

Javascript
```javascript
var blocks = document.getElementsByClassName('block'); // this creates a variable of the block itself to be later assigned
var container = document.getElementsByClassName('container'); // this assigns the previous variable to a 'container' that defines the menu itself
var hs = new HorizontalScroll.default({ // this is the use of the JavaScript module
 blocks : blocks,
 container: container,
});
```

## Code explanation

Disclaimer: this code file is not the actual code I ran, I have commented it like such for ease of understanding, in practice this would be separate with a CSS file alongside.

The code shown above starts by defining the HTML and associated tags. It then defines the major parent division of 'container', this is because all the 'block' divisions (the buttons) will be placed inside the 'container' division as one large entity forming a seamless menu. There are simply 5 'block' divisions to show 5 button options in the menu.

The JavaScript should have been a very complicated task, however with the help of the built-in horizontal scroll module it makes it much simpler. Initially, a variable called 'blocks' is defined and it contains all the divisions of the container named 'block'. Then the container variable is defined and is set to hold all of the blocks. The hs variable simply stands for horizontal scroll and is set to the default module call containing the blocks and container.

Once instantiated and run, it looks like this:



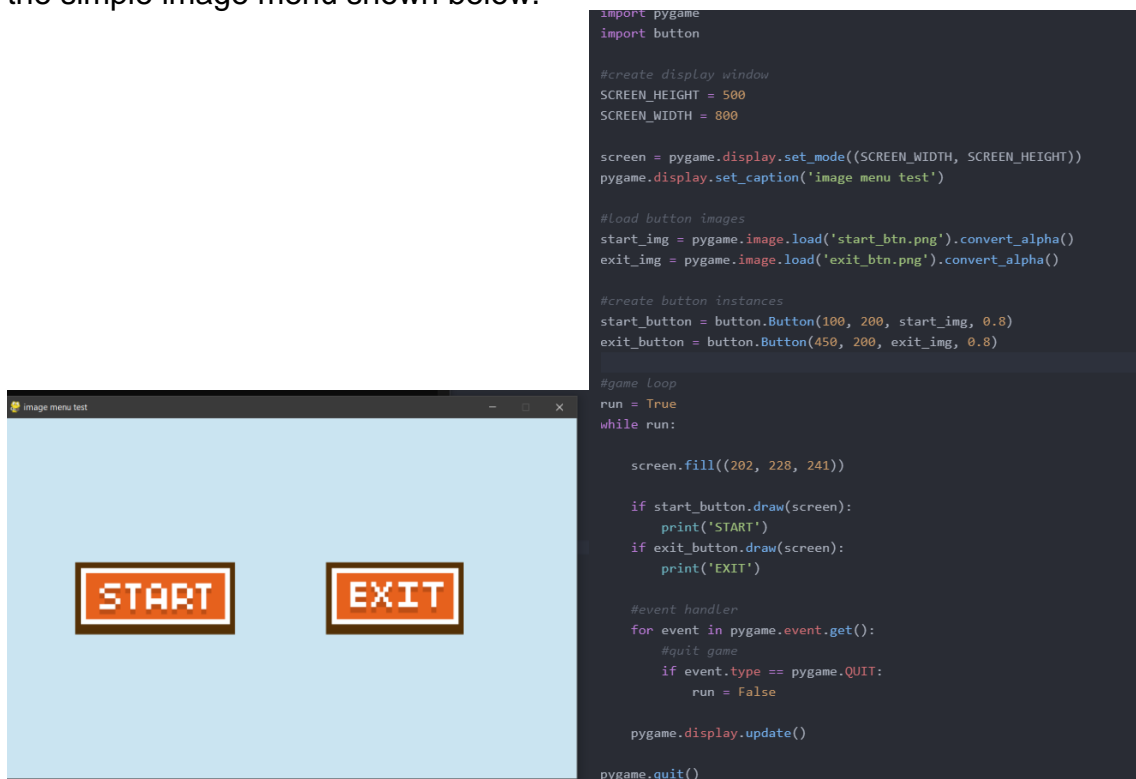
## Horizontal scroll

# Item 1

# Item 2

# Item 3

After much struggle and following many tutorials, even to the point of having to copy some code simply because I couldn't get my own to work, I eventually came up with the simple image menu shown below.



Although this by itself was incredibly hard for me to incorporate, to meet my initial requirements it would need to be dynamic and subject to horizontal scrolling which seemed impossible at this stage. For this reason, I decided to abandon the complex horizontal scrolling menu and instead adopt a much simpler vertical text menu. This was a lot simpler to implement and there are many tutorials to help alongside. Below is the code for my simple menu:

```

import pygame
import pygame_menu # this is the pygame module that makes this task so simple, as it handles the inputs and the GUI
pygame.init()
surface = pygame.display.set_mode((1200, 800)) # sets the screen dimensions

def load_level_1():
 if level == 1:
 print("level 1 loading ... ")
 # run the associated code to start the level, have not yet created the levels

there will be subroutine to load all levels and assigned to its corresponding button

menu = pygame_menu.Menu("Zain's projectile cs cw game", 1200, 800,
 theme=pygame_menu.themes.THEME_DARK) # sets the dimensions of the menu itself, its caption and the theme.
 # this is a basic default theme from the pygame.menu library

menu.add.button('level 1 : vectors', load_level_1)
menu.add.button('level 2 : ---', #load_level_2)
menu.add.button('level 3 : ---', #load_level_3)
menu.add.button('level 4 : ---', #load_level_4)
menu.add.button('level 5 : ---', #load_level_5)
menu.add.button('level 6 : ---', #load_level_6)
menu.add.button('level 7 : ---', #load_level_7)
menu.add.button('level 8 : ---', #load_level_8)
menu.add.button('level 9 : ---', #load_level_9)
menu.add.button('level 10 : ---', #load_level_10)
menu.add.button('level 11 : ---', #load_level_11)
menu.add.button('level 12 : ---', #load_level_12)
menu.add.button('level 13 : ---', #load_level_13)
menu.add.button('level 14 : ---', #load_level_14)

the load_level functions had to be commented out as they are not yet defined and would cause errors

menu.add.button('Quit', pygame_menu.events.EXIT) # this simply makes a button called quit to halt the code
menu.mainloop(surface)

```

## Code explanation

The code starts with the simple task of importing the modules of pygame and pygame\_menu, these will be used as a framework for the code. Line 3 instantiates the pygame module and what makes the actual window pop up, the next line defines the parameter of this window by its dimensions and assigns it to a variable to be called later at will.

The def Load\_level\_1 subroutine is a function that will be called later on in the code and game, simply to load the given level depending on which button is pressed. I have not included all the code to run the level as I have not reached that point yet. I will also create the other 13 load\_level functions once I have created the levels themselves. Line 14 creates the menu variable and within it runs the menu function call defining the caption, dimensions and theme respectively.

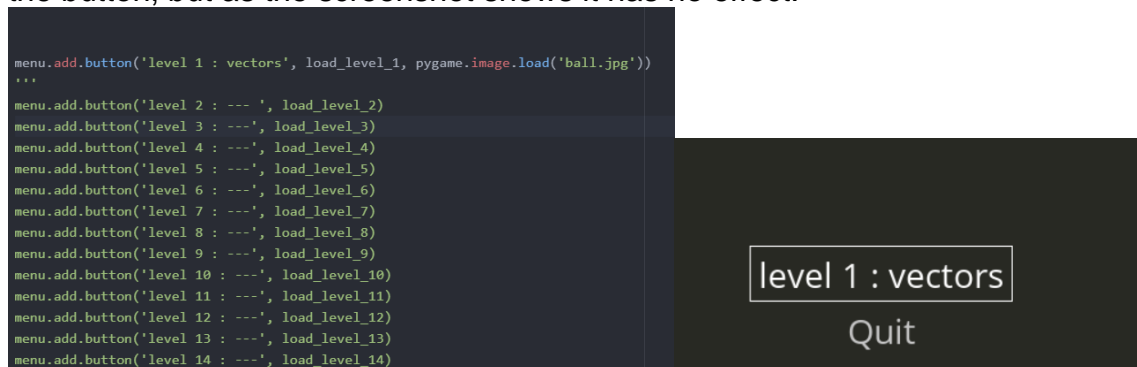
Lines 19 through 32 handle the function calls to instantiate each individual button for each level as well as their captions and corresponding load\_level function calls (commented out as do not exist yet).

Line 36 simply adds another button that handles the quit function by exiting the event while loop within pygame.

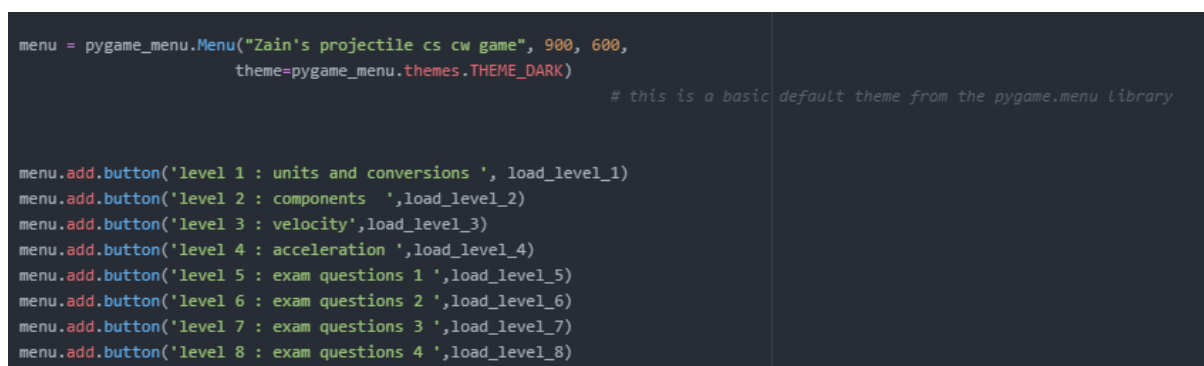
This is what the code produces when run.

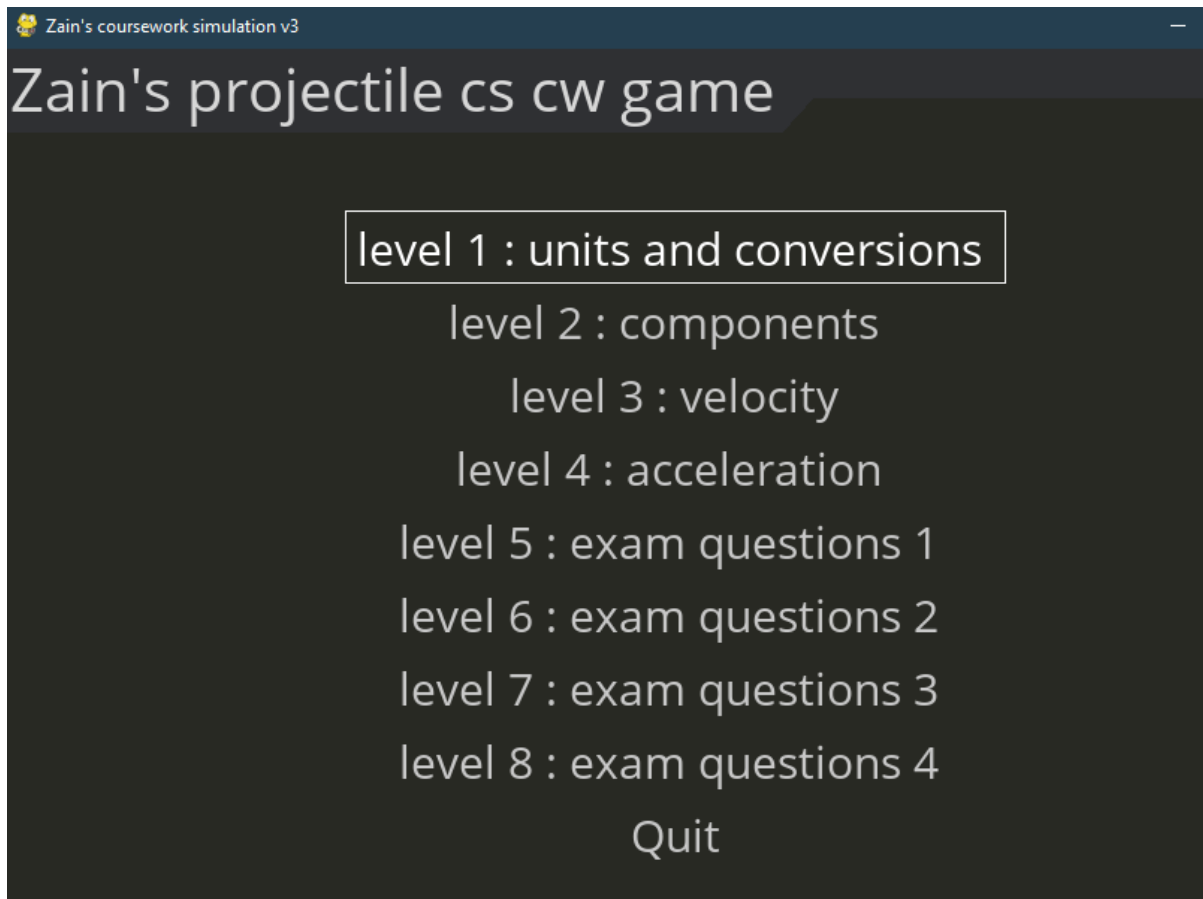


Below I have shown you what I tried to modify the code to incorporate an image into the button, but as the screenshot shows it has no effect.



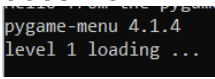
I will now show what the final version looks like





## Module testing

Test number	Test data	What to test and inputs	Expected results	Actual results
1	LMC (left mouse click) on the game script	If the menu select pops up and is as described in the sketch, load the game	The menu should pop up on its own and look similar to the sketch	The menu pops up when the code is run, however does not incorporate an image menu as explained previously. The screenshot is also shown above.
2	LMC (left mouse click) on level 1 tile	If the menu allows the user to select a	The level selected should load	The LMC click does successfully

		level, select random levels (repeat)		load the correct level every time. A print statement in the function does work. 
3	Scroll up/down (on the mouse)	If the menu allows for horizontal scroll and has all the syllabus topics	The menu should cover the whole specification	This is no longer a requirement as previously explained and all the levels do cover the syllabus.

## Testing amendments

In this section, I will list some of the tests that did not go as planned and what I did to fix them.

### Test 3

As stated this test completely failed, not because of an error in the code but the new requirements of my solution. This test was not done as it was no longer needed.

### Testing for background image

As mentioned in my reflection further down, I go on to create a new piece of code to handle a better background for the menu screen. During the implementation of this, I had a few issues with the syntax and had to make a few tweaks.

The first issue I had with my new code was below:

```
Traceback (most recent call last):
 File "F:\coursework cs game\test menu before messing with inputs.py", line 9, in <module>
 myimage = pygame_menu.baseimage.baseImage(image_path="F:\coursework cs game\background.jpg",
AttributeError: module 'pygame_menu.baseimage' has no attribute 'baseImage'
```

These complaints about line 9, which is below:

```
myimage = pygame_menu.baseimage.baseImage(image_path="F:\coursework cs game\background.jpg",
```

I believe that the reason I am getting this issue is that there is something wrong with the baseimage function, after some quick research I was able to figure out that I had entered it incorrectly and the second baseimage was to be a different function with a capital B. new code below:

```
myimage = pygame_menu.baseimage.BaseImage(image_path="F:\coursework cs game\background.jpg",
```

After this I had another simple issue when I ran the code and the error message is shown below:

```
Traceback (most recent call last):
 File "F:\coursework cs game\test menu before messing with inputs.py", line 9, in <module>
 myimage = pygame_menu.baseimage.BaseImage(image_path="F:\coursework cs game\background.jpg",
 File "C:\Users\zainm\AppData\Roaming\Python\Python39\site-packages\pygame_menu\baseimage.py", line 163, in __init__
 assert path.isfile(image_path), \
AssertionError: file F:\coursework cs gamackground.jpg does not exist or could not be found, please check if the path of
the image is valid
```

This again complains about line 9 and it specifically refers to the file directory I had provided, and then it also says (image path). To me I did not understand why this is not working as that is what I thought image paths were meant to be. After a search on python image paths I found nothing, then looking back at a tutorial I figured out that python does not need to specify the entire file directory of the image as long it is in the same folder with the code. The remedy to this was quite simple and the new code is shown below:

```
myimage = pygame_menu.baseimage.BaseImage(image_path="background.jpg",
```

This code now runs with no errors and the menu is shown perfectly.

## User feedback

I showed one of my stakeholders (Yassine Soltani) the menu module so far and the comments were “it would have been a lot better with pictures and scroll, but I can see why that can’t be done. The menu is very plain at the moment, maybe instead of a dark default theme include a physics image or something as the background”.

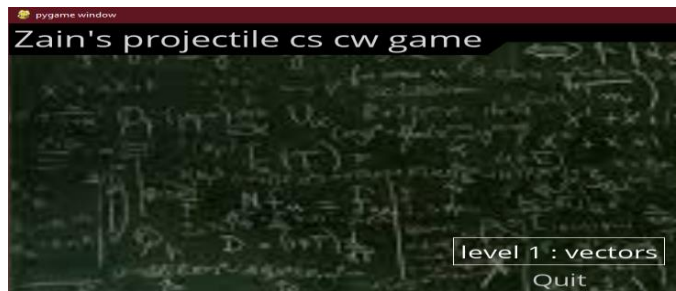
## Reflections

I have amended my code to incorporate and satisfy my user feedback. Here is the new code block:

```
mytheme = pygame_menu.themes.THEME_DARK.copy()
mytheme.title_background_color=(0, 0, 0)
myimage = pygame_menu.baseimage.BaseImage(image_path="background.jpg",
 drawing_mode=pygame_menu.baseimage.IMAGE_MODE_FILL
)
mytheme.background_color = myimage
```

```
menu = pygame_menu.Menu("Zain's projectile cs cw game", 900, 600,
 theme=mytheme) # sets the dimensions of
```

And this is the result when run:



All this simply does is create a new theme within the pygame\_menu module and assign them of the menu to its instantiation.

## Milestone 2: Successfully display a problem and its setting according to the level (syllabus)

Once the user has selected their topic/level they should now be met with the corresponding data. So when they select the projectile motion question then it shows that and allows the user to answer. The question should be displayed on a screen alongside the inputs before the graphical representation bit, the question should also have randomised values within it.

```
def InputScreen(): # the main screen where users pass values in
 Title = Font.render("Answer/variable Input Screen", 1, WHITE)
 warning = questionfont.render("to input, select the box and type (max 2 digits), enter the given variables",1,RED)
 question= questionfont.render("If a missile is launched from ground level with an initial velocity of x at an angle of theta degrees",1,CYAN)
 questiona= questionfont.render("a) Calculate the time taken until it detonates the target",1,CYAN)
 questionb= questionfont.render("b) How far was the target from the launch site",1,CYAN)
 questionc= questionfont.render("c) If the target changed to being y kilometres away, what angle would you have to fire at assuming velocity is unchanged.",1,CYAN)
 questiond1= questionfont.render("d) Still firing at this new target, the shells have been changed thus altering the initial velocity.",1,CYAN)
 questiond2= questionfont.render("Provide a combination of theta and initial velocity that these new shells will hit the target.",1,CYAN)
 DisplacementBox, InitialVBox, AngleBox, MassBox, AICheck, HideCheck, SubmitButton= create_widgets()

while Running:
 Screen_View.fill(BLACK)
 Screen_View.blit(Title,(120,5)) # displays the title
 Screen_View.blit(warning,(220,55))

 if level == 5:
 Screen_View.blit(question,(120,80)) #displays the questions
 Screen_View.blit(questiona,(240,110))
 Screen_View.blit(x,(350,130)) #displays the random variables
 Screen_View.blit(theta,(350,150))


 elif level == 6:
 Screen_View.blit(questionb,(290,80))
 elif level == 7:
 Screen_View.blit(questionc,(80,80))
 Screen_View.blit(y,(350,130))
 elif level == 8:
 Screen_View.blit(questiond1,(165,80))
 Screen_View.blit(questiond2,(180,110))
```

## Code explanation

These two pictures are part of one main function called `inputscreen`. In this screen, the user is given the chance to read the question, work it out and put their values in so that they now can see the graphical representation afterwards. The reason for the levels only being from 5 to 8 is that I have decided to do this program a little different to what I had originally thought. In the first 4 levels, I will simply be telling the user of the needed information and effectively teaching them how to do such questions, then once they get to the 5th level they can apply the knowledge they have from levels 1 to 4 to solve the question.

The first image shows the initial part of the process in which I have to set the graphical engine in pygame, as you cannot simply print straight to the display. First, you have to set a font (done previously) and then you have to create a variable in which you effectively instantiate this font and render the text. This then holds this rendered text as an image or pygame object which is time-consuming.

The second part of the function is in the main game loop, shown by the while running loop, this means that the content in the loop will be shown and updates every click, if we had displayed our text outside the loop that it would have been seen for a fraction of a second and that would be it. By putting them in the whole loop they are continuously refreshed and resultantly shown on screen. The `Screen_View` is a simple instantiation of the screen on which we can put things on. Once we apply the `blit` function to the screen surface we can the time-consuming select and display the pygame object as we wish. Pygame and the `blit` function do not allow multiline outputs, thus the reason for multiple repeats.

Test number	Test data	What to test and inputs	Expected results	Actual outcomes
1	LMC (left mouse click) on level 2 tile	If the selected level dictates the displayed problem( vary the levels - inputs )	Depending on the level selected the subroutine should display a value (question) from a 2d array of all questions	 The question is successfully shown and the values do randomly change upon every refresh. The questions are no longer chosen from an array, rather one set question between levels.
2	LMC (left mouse click)	If the setting is in accordance	Depending on the question	This is also true for the



	on level 2 tile	with the problem ( vary the levels - inputs )	displayed the setting should be the correct graphical interpretation of it	program, as the target loads at the correct point and the AI comes out of a set position.
3	LMC (left mouse click) on level 2 tile and refresh	If the displayed problems are unique, select one level multiple times	The same question should never have the same values in the problem	I have tested this and every single refresh I have gained a new value. Some values have overlapped but that is not due to bias, just small scope.

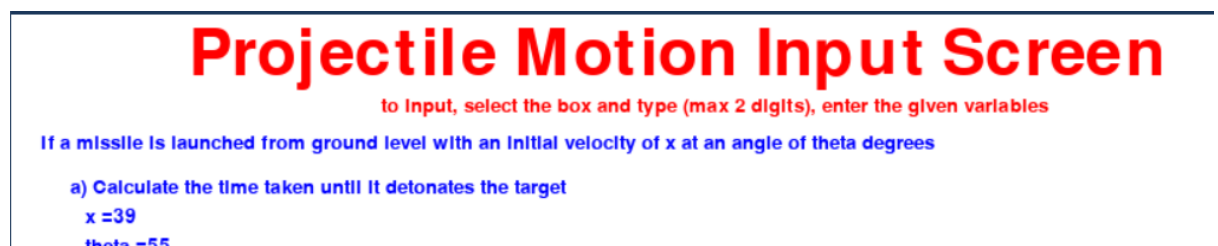
### Testing amendments

In this section, I will list some of the tests that did not go as planned and what I did to fix them.

As this milestone was quite simple I did not have a lot of errors during the implementation of my code. The main issue I had with my testing and what caused a lot of time and code tweaks was the positioning of the text. E.g. the code for level 5 is shown below:

```
if level == 5:
 Screen_View.blit(question,(20,80)) #displays the questions
 Screen_View.blit(questiona,240,110))
 Screen_View.blit(x,(50,130)) #displays the random variables
 Screen_View.blit(theta,(50,150))
```

And when run the issue of placement was evident, showing below result of shown code:



As evidently seen the questions are way too far over to the left, the remedy to this is very simple but time consuming, I had to just add to the x coordinate of the displays to shift it more right.

I had to do this for every level.

## **User feedback**

My stakeholder (Yassine Soltani) had a few comments regarding this particular milestone. His responses were the following: 'I think it would be much better if you had more than one level and if the overall program was just simply a little more appealing. Apart from that, it does work perfectly for the 5 times I have run through it.'

## **Reflections**

Upon listening to the advice given to me by my stakeholder I altered the question slightly. Instead of making one big level and one big question I have split the question into 4 chunks, which will be completed over 4 levels. Also as this is not my completely final project I had decided to leave it quite barebones in terms of graphics, background images etc... So once I am happy with the project I will complete the finishing touches by giving it a more pleasing colour scheme and appearance for the user.

## **Milestone 3: Be able to input values and respond accordingly**

This milestone represents a key stepping stone and fundamental process to my game. It evaluates the existence and efficacy of the user inputs. As my game has multiple SUVAT user inputs they need to be handled, to do this I have made a SUVAT input screen that will take in the variables beforehand. I have decided to go against my original saying of using a simple omnipresent variable box as that does not seem to make sense once in-game, as you would have to calculate the variables first anyways. Also, it eliminates the chance of users simply abusing trial and error to achieve the answers.

The code for this milestone is listed below:

```

def InputScreen(): # the main screen where users pass values in
 Title = Font.render("Answer/variable Input Screen", 1, WHITE)
 warning = questionfont.render("to input, select the box and type (max 2 digits), enter the given variables",1,RED)
 question= questionfont.render("If a missile is launched from ground level with an initial velocity of x at an angle of theta degrees",1,CYAN)
 questiona= questionfont.render("(a) Calculate the time taken until it detonates the target",1,CYAN)
 questionb= questionfont.render("(b) How far was the target from the launch site",1,CYAN)
 questionc= questionfont.render("(c) If the target changed to being y kilometres away, what angle would you have to fire at assuming velocity is unchanged.",1,CYAN)
 questiond1= questionfont.render("(d) Still firing at this new target, the shells have been changed thus altering the initial velocity.",1,CYAN)
 questiond2= questionfont.render("Provide a combination of theta and initial velocity that these new shells will hit the target.",1,CYAN)
 DisplacementBox, InitialVBox, AngleBox, MassBox, AIcheck, HideCheck, SubmitButton= create_widgets()
 Running = True

 Px = random.randint(10,100)
 Py = random.randint(10,100)
 Ptheta = random.randint(10,85)
 x = questionfont.render(("x =" + str(Px)),1,MAGNETA)
 theta = questionfont.render(("theta =" + str(Ptheta)),1,MAGNETA)
 y = questionfont.render(("y =" + str(Py)),1,MAGNETA)
 radPtheta = Ptheta*(math.pi)/180
 global a_ans
 a_ans = (math.sin(radPtheta)*Px*2)/9.81
 print(a_ans)
 global b_ans
 b_ans = math.cos(radPtheta) *Px * a_ans ## need to fix this
 print(b_ans)
 print(level)
 ...
 get the answers working
 ...

 while Running:
 Screen_View.fill(BLACK)
 Screen_View.blit(Title,(120,5)) # displays the title
 Screen_View.blit(warning,(220,55))

 if level == 5:
 Screen_View.blit(question,(120,80)) #displays the questions
 Screen_View.blit(questiond1,(165,80))
 Screen_View.blit(questiond2,(180,110))
 DisplacementBox.MakeBox() , InitialVBox.MakeBox(), AngleBox.MakeBox(),MassBox.MakeBox(),AIcheck.MakeBox(), HideCheck.MakeBox(), SubmitButton.DisplayButton()

 for event in pygame.event.get():
 if event.type == QUIT:
 pygame.quit()
 sys.exit()
 elif event.type == pygame.MOUSEBUTTONDOWN:
 (PosX,PosY) = pygame.mouse.get_pos()
 InitialVBox.click_seen((PosX,PosY))
 DisplacementBox.click_seen((PosX,PosY))
 AngleBox.click_seen((PosX,PosY))
 MassBox.click_seen((PosX,PosY))
 HideCheck.click_seen((PosX,PosY))
 AIcheck.click_seen((PosX,PosY))
 if SubmitButton.click_seen((PosX,PosY)) == True:
 Running = False
 elif (event.type == KEYDOWN) and (DisplacementBox.Clicked == True):
 DisplacementBox.AddCharacter(event.key)
 elif (event.type == KEYDOWN) and (InitialVBox.Clicked == True):
 InitialVBox.AddCharacter(event.key)
 elif (event.type == KEYDOWN) and (AngleBox.Clicked == True):
 AngleBox.AddCharacter(event.key)
 elif (event.type == KEYDOWN) and (MassBox.Clicked == True):
 MassBox.AddCharacter(event.key)
 pygame.display.update()
 fpsClock.tick(FPS)

 h, u, Angle = Convert(DisplacementBox.Variable,InitialVBox.Variable,AngleBox.Variable)
 if (u!="") and ((Angle!="") or (h!="")):
 global b_input
 b_input = h
 h, u, v, t, Angle,Mode = calculate_ans(h,u,Angle)
 run_sim(h,u,v,t,Angle,AIcheck.Clicked,HideCheck.Clicked,Mode)

```

## Code explanation:

This code function starts by creating a function and naming it InputScreen, it then runs the makewidgets functions for all the parameters needed. Then it enters a condition controlled while loop, which will run until another button is pressed or level is completed. The widgets created before are now displayed and a for loop is started to cover the game. This loop allows for the user to randomise the input variables. There is also a checkbox instantiated from its class that controls the displaying of the parameters live (module will be shown later). Once the inputs are determined a submit button will be waiting which will take you to the simulation (projmotionsimulation)

Below is the code to display the live parameters:

```
def printvalues(h,u,v,t,Angle):
 ParameterFont = pygame.font.Font(None, 30)
 S = ParameterFont.render("S = {0} m".format("%.1f"%b_ans),1,WHITE)
 U = ParameterFont.render("U = {0} m/s".format("%.1f"%u),1,WHITE)
 V = ParameterFont.render("V = {0} m/s".format("%.1f"%v),1,WHITE)
 A = ParameterFont.render("A = -g = -9.8 m/s^2",1,WHITE)
 T = ParameterFont.render("T = {0} s".format("%.1f"%t),1,WHITE)
 ANGLE = ParameterFont.render("θ = {0}°".format("%.1f"%Angle),1,WHITE)
 Screen_View.blit(S,(40, 400))
 Screen_View.blit(U,(40, 430))
 Screen_View.blit(V,(40, 460))
 Screen_View.blit(A,(40, 490))
 Screen_View.blit(T,(40, 520))
```

### Code explanation

This piece of code is quite very simple as it simply renders the value of the variables live and prints them off from a display function.

I could have used a mix of string and variables but I thought this approach would be much cleaner and would have fewer errors with concatenation and such.

For the validation I have also had to create 3 more modules to be run alongside this:

```
def error_msg():
 Title1 = Font.render("Please Enter The Initial Velocity &", 1, WHITE)
 Title2 = Font.render("EITHER The Angle Of Projection",1,WHITE)
 Title3 = Font.render("OR The Displacement",1,WHITE)
 OKButton = Button("OK", 380, 450, 40, 100,Screen_View)
 Running =True

 while Running:
 Screen_View.fill(BLACK)
 Screen_View.blit(Title1,(85,100))
 Screen_View.blit(Title2,(100,200))
 Screen_View.blit(Title3,(170,300))
 OKButton.DisplayButton()
 for event in pygame.event.get():
 if event.type == QUIT:
 pygame.quit()
 sys.exit()
 elif event.type == pygame.MOUSEBUTTONDOWN:
 (PosX,PosY) = pygame.mouse.get_pos()
 if OKButton.click_seen((PosX,PosY)) == True:
 Running = False
 pygame.display.update()
 fpsClock.tick(FPS)
 InputScreen() #Goes back to the input screen
```

### Code explanation

This function is very simple. Due to the nature of the kinematics equations you can but shouldn't have a combination of  $U$ ,  $\theta$  and  $S$  as they all work each other out. If you are found having all three of these then the question is wrong. The display calls communicate the error message to the user for them to try again.

```
def impossible_situation(): #Displays an error message which tells the user that the inputs entered are not feasible
 Title1 = Font.render("The inputs are not valid.", 1, WHITE) #Error message
 Title2 = Font.render("The value for Displacement is too high", 1, WHITE)
 Title3 = Font.render("for the given velocity.", 1, WHITE)
 Title4 = Font.render("Such a situation shouldn't occur.", 1, WHITE)
 Title5 = Font.render("Please reload the program.", 1, WHITE)
 OKButton = Button("OK", 350, 500, 40, 100, Screen_View)
 Running = True

 while Running:
 Screen_View.fill(BLACK)
 Screen_View.blit(Title1, (150, 50))
 Screen_View.blit(Title2, (40, 130))
 Screen_View.blit(Title3, (175, 210))
 Screen_View.blit(Title4, (70, 290))
 Screen_View.blit(Title5, (180, 370))
 OKButton.DisplayButton()
 for event in pygame.event.get():
 if event.type == QUIT:
 pygame.quit()
 sys.exit()
 elif event.type == pygame.MOUSEBUTTONDOWN:
 (PosX, PosY) = pygame.mouse.get_pos()
 if OKButton.click_seen((PosX, PosY)) == True:
 Running = False
 InputScreen()
 pygame.display.update()
 fpsClock.tick(FPS)
```

### Code explanation

This is not so much a validation clause but just an input check as a high value of displacement then consequently there has to be a high value of velocity and vice versa. This code simply checks if the two are in coherence, e.g. making sure there is not a low velocity with extreme displacement as this simply is not possible, but this would not be picked up by the equation, this is simple human logic.

The display calls in the module are just to bring up a new error screen to alert the user.

```
def Convert(s,u,Angle): # converts inputs to float numbers for greater accuracy
 try:
 s = float(s)
 except:
 s = ""
 try:
 u = float(u)
 except:
 u = ""
 try:
 Angle = float(Angle)
 Angle = math.radians(Angle) #PYGAME WORKS IN RADIANS
 except:
 Angle = ""
 return s, u, Angle
```

### Code explanation

This function executes a method on the parameters and returns them for the program to use in calculations. What it does is use the try and except python

validation clauses to convert the S, U and Angle variables into floats instead of strings, this is so they can be later calculated and can be mathematically solved. For the angle variable, we are forced to work in radians as pygame works in radians and not degrees (1 radian =  $360/2\pi$  degrees).

I will now respectively list below what the code blocks do when run:

# Answer/variable Input Screen

to Input, select the box and type (max 2 digits), enter the given variables

If a missile is launched from ground level with an Initial velocity of x at an angle of theta degrees

a) Calculate the time taken until it detonates the target

x = 96  
theta = 38

Displacement (s)(m):  Min: 0.1  
Max: 1000.0

Initial Velocity (u)(m/s):  Min: 1.0  
Max: 100.0

Angle ( $\theta$ )(degrees):  Min: 1.0  
Max: 90.0

Mass (kg):  Min: 0.1  
Max: 10.0

Hide Variables: ☐


AI (ON/OFF): ☐

**Submit**

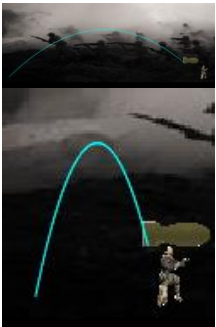
**S = 911.5 m**  
**U = 55.0 m/s**  
**V = -55.0 m/s**  
**A = -g = -9.8 m/s<sup>2</sup>**  
**T = 9.2 s**

**Please Enter The Initial Velocity &  
EITHER The Angle Of Projection  
OR The Displacement**

**OK**

Test number	Test data	What to test and inputs	Expected results	Actual outcomes
1	Input a random integer value in the SUVAT input boxes	If the input boxes are present, in all levels	The input SUVAT box should be constant and in the corner of every level	My inputs are no longer a constant box but a screen before the level starts - previously explained. The input screen is now a module that successfully loads upon every level selection
2	Input a random integer value in the SUVAT input boxes	If the boxes accept valid inputs for the variables	The inputs should only be of the variables asked for and	 <p>This screen</p>



			must be rejected if non-integers	will appear as an error message if invalid input is given. There is no longer a need to check for variables given as the user will have to input the given ones as well as the ones calculated, and will not have to input all the variables for it to run.
3	Input value '5' into angle then again value '70'	Check if the variables actually make a difference to the path of the projectile and if they are shown live	The projectile should change behaviour depending on the inputs, the variables should be shown live throughout the projectile's path	 <p>as seen the variables do affect the simulation.</p> <pre> S = 911.5 m U = 55.0 m/s V = -55.0 m/s A = -g = -9.8 m/s^2 T = 9.2 s </pre> <p>These are the parameters printed off live.</p>

### Testing amendments

In this section, I will list some of the tests that did not go as planned and what I did to fix them.

Throughout this module, as it is very big I have had many issues with simple syntax, I have not documented all of them as there would be hundreds.

Instead of listing them out and explaining each one I will simply explain my general approach to them. First, I will analyse the error message, by looking at the specific line with the error and fixing the error if I see it, if I do not see it I will go on research the documentation for that function or line of code and make sure my code is coherent with the correct syntax.

One of the major issues I had, when I tested the code, was the sheer fact and my fault that I could not select the boxes. This was my code:

```
h, u, Angle = Convert(DisplacementBox.Variable,InitialVBox.Variable,AngleBox.V
if (u!="") and ((Angle!="") or (h!="")):
 global b_input
 b_input = h
 h, u, v, t, Angle,Mode = calculate_ans(h,u,Angle)
 run_sim(h,u,v,t,Angle,AIcheck.Clicked,HideCheck.Clicked,Mode)
 error_msg()
```

Although I was not getting a direct error message from the execution shell I had quickly figured out where I went wrong and set out to fix it. The error was that I was unable to select my boxes and type in them, and this was because I hadn't implemented any code that would allow you to do that. Recalling how I created the submit button, I remembered that you have to get pygame to listen to any 'events'. I then added this functionality by using my clickseen function and pygame events. The new code is shown below:

```
for event in pygame.event.get():
 if event.type == QUIT:
 pygame.quit()
 sys.exit()
 elif event.type == pygame.MOUSEBUTTONDOWN:
 (PosX, PosY) = pygame.mouse.get_pos()
 InitialVBox.click_seen((PosX, PosY))
 DisplacementBox.click_seen((PosX, PosY))
 AngleBox.click_seen((PosX, PosY))
 MassBox.click_seen((PosX, PosY))
 HideCheck.click_seen((PosX, PosY))
 AIcheck.click_seen((PosX, PosY))
 if SubmitButton.click_seen((PosX, PosY)) == True:
 Running = False
 elif (event.type == KEYDOWN) and (DisplacementBox.Clicked == True):
 DisplacementBox.AddCharacter(event.key)
 elif (event.type == KEYDOWN) and (InitialVBox.Clicked == True):
 InitialVBox.AddCharacter(event.key)
 elif (event.type == KEYDOWN) and (AngleBox.Clicked == True):
 AngleBox.AddCharacter(event.key)
 elif (event.type == KEYDOWN) and (MassBox.Clicked == True):
 MassBox.AddCharacter(event.key)
pygame.display.update()
fpsClock.tick(FPS)
```

Other than this I did not have any major failed tests during the development of this milestone.

## User feedback

I asked my stakeholder for feedback regarding this milestone and the response was “this works very well and is a good fundamental base to the game, however, I would recommend making sure the user can always input values as I found it bugs out and only allows randomising. Also, the live parameters are not smooth in transition but jump from point to point”.

## Reflections

Upon receiving feedback from my stakeholder I have now implemented some changes to my code. Firstly, to make the parameters more up to date and

stop the 'jumping' I have increased the update count so it refreshes faster between points.

To combat the user input values I have decided to rewrite the input module to make it more efficient. Also alongside this, I have made it so you have to select the textbox to input to reduce any chance of error.

### Milestone 4: Apply the correct formula and solve the problem

This one milestone is one of the major components of my game, as it is a projectile motion game. It needs an algorithm that can correctly use the formulae to reach the correct answers. Depending on the variables given in the questions the algorithm will have to recognise this and adapt to using the other formulae to get the variable in question.

```
def calculate_ans(h,u,Angle): #calculates using SUVAT
 g = 9.8 # acceleration due to gravity, is a constant on the earths surface
 if (h!="") and (Angle==""):
 if math.sqrt(2*g*h)/u>1: #avoid impossible scearios
 impossible_situation()
 else:
 Mode = 1
 v = -u
 Angle = math.asin(math.sqrt(2*g*h)/u) #Rearranging v**2 = u**2 + 2as, at top v= 0
 t = (u*math.sin(Angle))/4.9 #Using s=ut+0.5at**2
 return h, u, v, t, math.degrees(Angle), Mode
 elif (h=="") and (Angle!=""):
 Mode = 2
 v = -u
 t = (u*math.sin(Angle))/4.9 #total time
 h = (u*math.sin(Angle)*(t/2)) - (4.9 * (t/2)**2) #max vertical displacement
 return h, u, v, t, math.degrees(Angle), Mode
 elif (h!="") and (Angle!="") and (u!=""):
 error_msg() # impossible scenario
```

### Explanation of code

In a very big image vein, this algorithm simply recognises the variable given and applies them to the correct formula. I have decided to use a conjecture of if statements to create 2 scenarios in which every combination of variables are given, so the algorithm will always know what to do with the given data. The only scenario in which this algorithm will fail is if you provide all 3 of initial velocity, angle and displacement; as these all lead and work out each other it will lead to impossible solutions if all 3 are given and are not correct. In this module, I have used the maths python module for the calculations including sin and arcsine as well as any radians conversions.

```

def explanations1(h,u,v,t,Angle,AI,Hide,Method):
 Method = Method
 ExplainFont = pygame.font.Font(None,32)
 explaining = True
 VExplain2str = "v = -" + str(u)
 #print("type is", type(VExplain2str))
 TimeExplain3str = "t = " + str(u) + "sin(" + str(Angle) + ")/4.9"
 AngleExplain3str = "θ = sin⁻¹(√(2*g*h) + str(h) + ")/" + str(u)
 VExplain1 = ExplainFont.render("v = -u",1,WHITE)
 VExplain2 = ExplainFont.render(VExplain2str,1,WHITE)
 AngleExplain1 = ExplainFont.render("Rearrange v² = u² + 2as where v = 0",1,WHITE)
 AngleExplain2 = ExplainFont.render("usinθ = √(2gs) => θ = sin⁻¹(√(2gs))/u",1,WHITE)
 AngleExplain3 = ExplainFont.render(AngleExplain3str,1,WHITE)
 TimeExplain1 = ExplainFont.render("Rearrange s = ut + 0.5at² where s = 0",1,WHITE)
 TimeExplain2 = ExplainFont.render("t = usinθ/4.9",1,WHITE)
 TimeExplain3 = ExplainFont.render(TimeExplain3str,1,WHITE)
 OKButton = Button("OK",400,500,35,100,Screen_View) # (Label, PosXStart, PosYStart, Width, Length, Surface)
 while explaining:
 display_explanations(VExplain1, VExplain2, AngleExplain1, AngleExplain2, AngleExplain3, TimeExplain1, TimeExplain2, TimeExplain3, OKButton)
 for event in pygame.event.get():
 if event.type == QUIT:
 pygame.quit()
 sys.exit()
 elif event.type == pygame.MOUSEBUTTONDOWN:
 (PosX, PosY) = pygame.mouse.get_pos()
 if OKButton.click_seen((PosX, PosY)) == True:
 explaining = False
 if win_loss_screen == True:
 menu.mainloop(surface)
 pygame.display.update()
 fpsClock.tick(FPS)
 run_sim(h,u,v,t,Angle,AI,Hide,Method)

def explanations2(h,u,v,t,Angle,AI,Hide,Method):
 Method = Method
 ExplainFont = pygame.font.Font(None,32)
 explaining = True
 VExplain2str = "v = -" + str(u)
 TimeExplain3str = "t = " + str(u) + "sin(" + str(Angle) + ")/4.9"
 DisplacementExplain3str = "s = " + str(u) + "sin(" + str(Angle) + ")(t/2) + 4.9(t/2)²"
 #print("type is", type(VExplain2str))
 VExplain1 = ExplainFont.render("v = -u",1,WHITE)
 VExplain2 = ExplainFont.render(VExplain2str,1,WHITE)
 TimeExplain1 = ExplainFont.render("Rearrange s = ut + 0.5at² where s = 0",1,WHITE)
 TimeExplain2 = ExplainFont.render("t = usinθ/4.9",1,WHITE)
 TimeExplain3 = ExplainFont.render(TimeExplain3str,1,WHITE)
 DisplacementExplain1 = ExplainFont.render("At time, t/2, height is at a max => using s = ut + 0.5at²",1,WHITE)
 DisplacementExplain2 = ExplainFont.render("s = usinθ(t/2) + 4.9(t/2)²",1,WHITE)
 DisplacementExplain3 = ExplainFont.render(DisplacementExplain3str,1,WHITE)
 OKButton = Button("OK",400,500,35,100,Screen_View) # (Label, PosXStart, PosYStart, Width, Length, Surface)
 while explaining:
 display_explanations(VExplain1, VExplain2, TimeExplain1, TimeExplain2, TimeExplain3, DisplacementExplain1, DisplacementExplain2, DisplacementExplain3, OKButton)
 for event in pygame.event.get():
 if event.type == QUIT:
 pygame.quit()
 sys.exit()
 elif event.type == pygame.MOUSEBUTTONDOWN:
 (PosX, PosY) = pygame.mouse.get_pos()
 if OKButton.click_seen((PosX, PosY)) == True:
 explaining = False
 if win_loss_screen == True:
 menu.mainloop(surface)
 pygame.display.update()
 fpsClock.tick(FPS)
 run_sim(h,u,v,t,Angle,AI,Hide,Method)

```

## Code explanation

These two modules are very similar and are what handle the explanations. The difference between them is that one explains how to solve for time and the other explains how to solve displacement. Each function will be called dependent on the question asked. How they work is to simply first assign variables to the display rendering of the explanation text, then pass those into another simple function that will just display them, this could have been within the function itself however it would have to have been repeated. The rest of the function and the for loop is just to listen out for any mouse clicks as every function has to.

```
def display_explanations(a,b,c,d,e,f,g,h,Button):
 Screen_View.fill(BLACK)
 Screen_View.blit(a,(40,30))
 Screen_View.blit(b,(40,70))
 Screen_View.blit(c,(40,120))
 Screen_View.blit(d,(40,170))
 Screen_View.blit(e,(40,220))
 Screen_View.blit(f,(40,270))
 Screen_View.blit(g,(40,320))
 Screen_View.blit(h,(40,370))
 Button.DisplayButton()
```

### Code explanation

This is a simple function mentioned before, it takes in the input parameters from the 2 explain function and uses display calls to present them on a white screen to the user.

Test number	Test data	What to test and inputs	Expected results	Actual outcomes
1	Freefall - Enter value '10' in displacement and value '9.81' in acceleration.	The program should apply different formulas depending on what the given variables are and the context (most levels should only use one formula) Should use $v^2 = u^2 + 2as$	The correct formula applied Correct results	As shown by the explanation screen the formulas are all used where needed. $v = -u$ $v = -70.3$ Rearrange $v^2 = u^2 + 2as$ $\Rightarrow u \sin \theta = \sqrt{(2gs)} \Rightarrow \theta = a$ $\Rightarrow \theta = \arcsin(\sqrt{(2 \cdot g \cdot 66.8)})$ Rearrange $s = ut + 0.5at^2$ $\Rightarrow t = u \sin \theta / 4.9$ $\Rightarrow t = 70.3 \sin(31.0) / 4.9$
2	LMC (left mouse click) the solution button	The formula and problem should each have corresponding explanations	The explanation must be correct to the problem and formula shown	This is what appears when the explain button is pressed:

				$v = -u$ $v = -27.7$ <p>Rearrange <math>s = ut + 0.5at^2</math> where <math>s = 0</math></p> $0 = 1 \sin(4.9)$ $0 = 27.7 \sin(36.6) / 4.9$ <p>At time, <math>t/2</math>, height is at a max <math>\Rightarrow</math> using <math>s = ut + 0.5a</math></p> $0 = 1 \sin(4.9) - 4.9(t/2)^2$ $0 = 27.7 \sin(36.6) / 4.9 - 4.9(t/2)^2$ <p>This clearly explains the mathematical solutions with the correct formulae</p>
3	LMC (left mouse click) the solution button	Use of alternate formulae	Every possible formula to solve that question should be explained, showing every way it could be solved	As seen in the previous two screenshots both the possible formulae are used and explained with the solution.

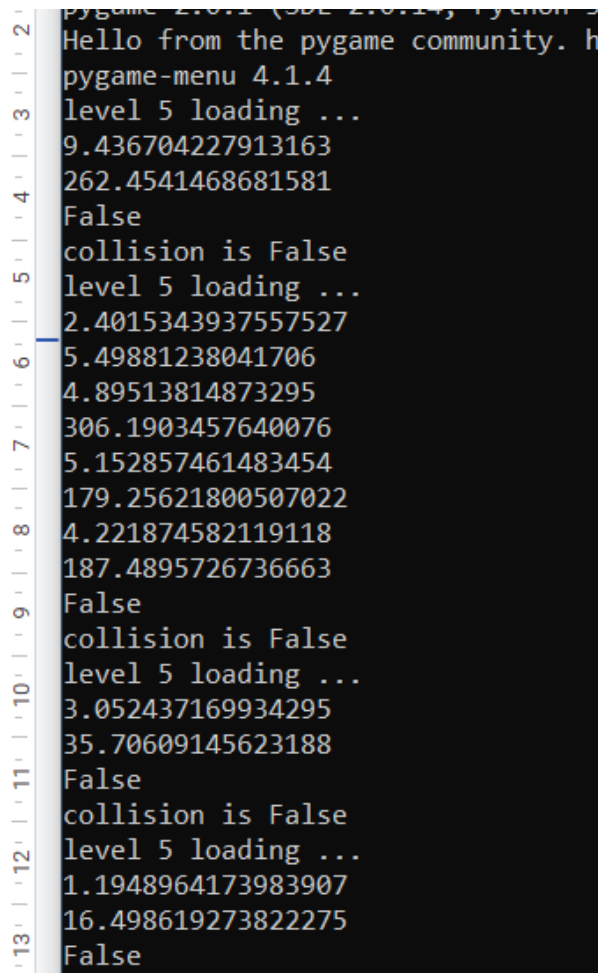
### Testing amendments

In this section, I will list some of the tests that did not go as planned and what I did to fix them.

As previously mentioned, with all displayed text I had a testing issue of the placement of it on screen, this is a very simple fix of just adjusting the display coordinates inside the Blit function.

I will now show one of the major issues I encountered during the testing of this milestone:

**S = 509.9 m**  
**U = 74.0 ms<sup>-1</sup>**  
**V = -74.0 ms<sup>-1</sup>**  
**A = -g = -9.8 ms<sup>-2</sup>**  
**T = 6.6 s**



```
pygame 2.1.1 (SDL 2.1.1, Python 3.10.1)
Hello from the pygame community. https://www.pygame.org
pygame-menu 4.1.4
level 5 loading ...
9.436704227913163
262.4541468681581
False
collision is False
level 5 loading ...
2.4015343937557527
5.49881238041706
4.89513814873295
306.1903457640076
5.152857461483454
179.25621800507022
4.221874582119118
187.4895726736663
False
collision is False
level 5 loading ...
3.052437169934295
35.70609145623188
False
collision is False
level 5 loading ...
1.1948964173983907
16.498619273822275
False
```

Something here has gone very wrong. The first picture shows the live end parameters displayed to the user, these are essentially the 'answers' to the question and they are completely incorrect. I had no idea what was causing this and I spent over an hour combing through my code, where I realized I had made a rookie mistake in not conforming to pygame. In my code, I had been working out the solutions in degrees whereas pygame only works in radians. This was quite an easy fix, all I had to do was convert the angle into radians before any math was done on it, the code for this is shown below:

```
Angle = math.radians(Angle)
```

After this issue, the development of the rest of this milestone was fairly straightforward apart from some very pesky syntax errors that took a lot of time to iron all out.

## User feedback

After a brief discussion with my stakeholder (Yassine Soltani), I was advised that the explanation screen was very inhumane and robotic, and it simply



spouts math at you. He also said that apart from that the solution works very well and does give you the correct answers

## **Reflections**

I decided to take the stakeholder advice on board and make the explanations more humane. I did this by using more worldly explanations and reasoning them out rather than just stating maths.

My actual solution module has greatly deviated from my design stage, as before I never recognised the impossible scenario or the way I would have to manipulate data in python as opposed to the phaser module in JavaScript that would have done most of the work.

## **Milestone 5; Can graphically calculate and represent the motion**

This milestone controls and is all to do with the graphical aspect of my game, as the algorithm calculates the end values for the questions it also needs to show this projectile and its consequent motion due to its parameters. This is so the student is able to visualize the differences the parameter has, such that a higher angle results in a higher max height and less displacement and such. This visualization will then help the student to understand what they are trying to calculate while they are going through their question as they are now able to visualize it.

```
def run_sim(h,u,v,t,Angle,AI,Hide,Mode):
 if u < 15 or Angle < 15 :
 InputScreen()
 print(AI)
 global AI_difficulty
 global AI_win_count
 InputButton = Button("Inputs", 600, 400, 35,200,Screen_View)
 ExplainButton = Button("Explain", 600, 540, 35, 200, Screen_View)
 HIDE = CheckBox(500,400,30,Screen_View,"Hide Parameters:",160)
 HIDE.Clicked = Hide
 Input = False
 Explain = False
 floorY_co_ord = 350
 h = h
 PosX = 50
 PosY = floorY_co_ord #As the particle will start at the ground
 g = -9.8
 ANGLEDEGREES = round(Angle,1)
 ANGLERADIANS = math.radians(ANGLEDEGREES)
 u = u
 i = u * round(math.cos(ANGLERADIANS), 1) #horizont velcoity compnent
 j = u * round(math.sin(ANGLERADIANS), 1) # vertical velocity component
 SCALEFACTOR = 1.3 # makes it look better
 CoordinatesList = list_get(i,j,PosX,PosY)
 Traillist = copy.deepcopy(CoordinatesList) # this is a recursive call of copying the file over
 Traillist = trail_tweek(Traillist)
 Motion = True
 Pointer = 0
 Screen_View.fill(WHITE)
 Running = True
 PosXAI = 500
 PosYAI = floorY_co_ord
 interceptor=pygame.image.load('flare.png')
 collision = False
 win = False
 change_AI_difficulty()
 print("run sim",AI_difficulty, AI_win_count)
 while Running:
 Screen_View.fill(BLACK)
```

```

Screen_View.fill(BLACK)
Screen_View.blit(main_background,(0,0))
InputButton.DisplayButton()
ExplainButton.DisplayButton()
HIDE.MakeBox()
if HIDE.Clicked == False:
 printvalues(h,u,v,t,ANGLEDEGREES)
if Motion == True:
 PosX, PosY = CoordinatesList[Pointer][0], CoordinatesList[Pointer][1]
 #print('(',PosX,',',',',PosY,')')
 Screen_View.blit(BALL, (PosX, PosY))
 r'''

 #Makes trail
 dotimg=pygame.image.load('dot.png')
 dotimg=pygame.transform.scale(dotimg,(5,5))
 Screen_View.blit(dotimg, ((PosX-5), (PosY+5)))
 r'''

 if level == 5 or level == 6 or level==7 or level==8 :
 Screen_View.blit(TARGET,((CoordinatesList[-1][0]) ,330))
 if AI==True:
 diffX = PosXAI - PosX
 diffY = PosYAI - PosY
 #print('(',diffX,',',',',diffY,')')
 moveX = diffX / AI_difficulty
 moveY = diffY / AI_difficulty
 x=random.randint(-2,2)
 y=random.randint(-2,2)
 #x=min((u/(random.randint(-2,2))),2)
 PosXAI -= (moveX + x)
 PosYAI -= (moveY + y)
 Screen_View.blit(interceptor, (PosXAI, PosYAI))
 maxPosX = PosX + 20
 maxPosY = PosY + 20
 minPosX = PosX - 20
 minPosY = PosY - 20
 if (PosXAI < maxPosX and PosXAI > minPosX) and (PosYAI < maxPosY and PosYAI > minPosY) :
 collision = True

```

```

 if collision == True:
 collisionimg=pygame.image.load('collide.png')
 collisionimg=pygame.transform.scale(collisionimg,(50,50))
 Motion=False
 if Pointer == (len(CoordinatesList)-1):
 Motion = False
 else:
 Pointer +=1
 if Motion == False:
 if collision == True:
 Screen_View.blit(collisionimg, (PosX, PosY))
 win = False
 #global AI_difficulty
 AI_win_count += 1
 AI_difficulty += 20
 loss_screen(win,h,u,v,t,ANGLEDEGREES,HIDE,Mode)
 if collision == False: #and b_input == ans_b :
 win = True
 print("collision is", collision)
 #print("user : ",b_input ,"ans:",ans_b)
 else:
 PosY = floorY_co_ord
 PosX = CoordinatesList[Pointer][0]
 Screen_View.blit(collisionimg, (PosX, PosY))
 if win == True :
 AI_win_count -= 1
 AI_difficulty -= 20
 win_screen(win,h,u,v,t,ANGLEDEGREES,HIDE,Mode)
 if Pointer>1:
 pygame.draw.aalines(Screen_View, MAGNETA,False,Traillist[:Pointer+1000] ,5)
 pygame.draw.aalines(Screen_View, CYAN,False,Traillist[:Pointer] ,5)

 pygame.draw.line(Screen_View, BLACK, (0,floorY_co_ord+25), (X_view,floorY_co_ord+25),1) # draws the floor
 for event in pygame.event.get():
 if event.type == QUIT:
 pygame.quit()
 sys.exit()
 elif event.type == MOUSEBUTTONDOWN:
 (PosX, PosY) = pygame.mouse.get_pos()
 HIDE.click_seen((PosX,PosY))
 if InputButton.click_seen((PosX,PosY)) == True:
 if InputButton.click_seen((PosX,PosY)) == True:
 Input = True #It will be true which implies that that user must enter inputs again
 Running = False
 elif ExplainButton.click_seen((PosX,PosY)) == True:
 Explain = True
 Running = False
 pygame.display.update()
 fpsClock.tick(FPS)
 if Input == True:
 InputScreen()
 elif Explain == True:
 if Mode == 1:
 explanations1(h,u,v,t,ANGLEDEGREES,AI,HIDE.Clicked,Mode)
 elif Mode == 2:
 explanations2(h,u,v,t,ANGLEDEGREES,AI,HIDE.Clicked,Mode)

```

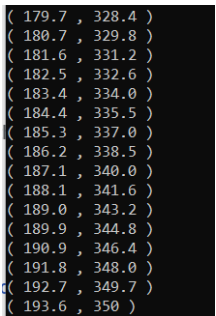
## Code explanation


These 2 screenshots are of the same function as it is quite a large function. The function starts with a quick python definition and a parameter input that will be needed throughout the function. It then goes on to create the explain, input and AI checkbox and buttons. One of the main variables for this entire

program is GROUNDLEVEL and it is set at 350 for this level. The way the coordinates work in pygame are technically opposite to what we would see in real life, in pygame the top left would be (0,0) and then the bottom right would be max, so as you come down the coordinates increase. This will be a vital difference in how this function works as instead of adding to the Ypos as the projectile gains height we would instead decrease the Ypos to mimic the increase in height. It would be possible to overwrite the entire pygame coordinate system so it works in a standard Cartesian way, however, that would be an unnecessary complication and this way also does work. In this function I have decided to not interrupt the global variables and instead simply assign the same values to a temp variable for this function (upper case version), this is because this function calculates the intermediate parameters and we do not want that for the other functions. The I variable gives the horizontal component and j the vertical component. The scale factor variable does not make a difference except making it look better in the screen frame. The while loop is responsible for continuously writing the buttons for every frame as well as running the recursion algorithm for the parameters at a given point. I have decided to make use of a long array to store all the coordinates so that they can be accessed whenever needed, such as when drawing the line. The recursion algorithm pulls the current positions from the list and sets the missile to that position. I have also included a print function for debugging and evidence.

Pygame is very useful for the projectile path as it can call a simple function and have it draw the line in one simple code line. I also used the line function to draw the ground level.

The for loop is a simple necessity that listens for any button clicks or mouse clicks. The last block in this function is reliant on the previous for loop as it acts on the button clicks, by opening the corresponding screen.

Test number	Test data	What to test and inputs	Expected results	Actual outcomes
1	Input values for SUVAT	If the program calculates the coordinates of the path correctly using the formula and indent substitution	Print command in the coordinate function should show all the points on the curve	 <pre>( 179.7 , 328.4 ) ( 180.7 , 329.8 ) ( 181.6 , 331.2 ) ( 182.5 , 332.6 ) ( 183.4 , 334.0 ) ( 184.4 , 335.5 ) ( 185.3 , 337.0 ) ( 186.2 , 338.5 ) ( 187.1 , 340.0 ) ( 188.1 , 341.6 ) ( 189.0 , 343.2 ) ( 189.9 , 344.8 ) ( 190.9 , 346.4 ) ( 191.8 , 348.0 ) ( 192.7 , 349.7 ) ( 193.6 , 350 )</pre> <p>This is the result of the print command. As you can see the coordinates are inverted as</p>

				explained before.
2	Input values for SUVAT	The calculated coordinates should be plotted on a hidden axis graph, the curve of best fit drawn	After value inputs should see a typical flight curve displayed	 <p>as seen a typical flight parabola is plotted and a line of best fit is drawn. The hidden graph was no longer a requirement as pygame has an inbuilt coordinate system as opposed to the original JavaScript.</p>
3	Input values for SUVAT	If the projectile (missile) graphically follows the curve	Should see the missile going along the pre-drawn curve of motion	As seen in the pictures before, the coordinates printed are those of the missile and it leads the path, where the line is drawn behind it.

### Testing amendments

In this section, I will list some of the tests that did not go as planned and what I did to fix them.

As mentioned in the testing amendment section of the previous milestone I had a fundamental issue with my code of it working in degrees and not in radians as pygame does, for this reason, I saw some very anomalous results if the motion of the projectile and I have now incorporated radians into my code, shown below:

```
ANGLEDEGREES = round(Angle,1)
ANGLERADIANS = math.radians(ANGLEDEGREES)
```

Below I will show you my original code on how I tried implementing the graphical path, this was in parallel accordance with the pseudocode within my design stage:

```
#Makes trail
dotimg=pygame.image.load('dot.png')
dotimg=pygame.transform.scale(dotimg,(5,5))
Screen_View.blit(dotimg, ((PosX-5), (PosY+5)))
```

This was in the while running loop, so the aim was that it would print a trail of dots behind the projectile effectively making a line of motion as it goes along. The actual result of the code is shown below.



As you can see the projectile is at the end of its path and there is only a singular dot behind the projectile, this is because it is in the while loop the dots do not stay in their positions, rather just refresh to a new position behind the projectile. This was a very apparent issue and I had to do something about it. The initial approach to this function was wrong and I had to rethink the entire thing. After doing some research on pygame functions and recalling the basics of pygame I realised that pygame has the ability it draws a line, however, I did not know about drawing parabolas, after looking at the documentation of the pygame. draw module I discovered the aalines function that draws the ant aliased line for you from a coordinate list. The code implementation is shown below:

```
pygame.draw.aalines(Screen_View, BLUE,False,(PosX,PosY) ,5)#MakeTrail
```

And once run this code gave me this error:

```
apply
return self._onreturn(*args, **self._kwargs)
File "F:\coursework cs game\test menu before messing with inputs.py", line 932, in load
InputScreen(level)
File "F:\coursework cs game\test menu before messing with inputs.py", line 366, in Input
run_sim(h,u,v,t,Angle,AIcheck.Clicked,HideCheck.Clicked,Mode,level)#Simulation for Pro
File "F:\coursework cs game\test menu before messing with inputs.py", line 479, in run_
pygame.draw.aalines(Screen_View, BLUE,False,(PosX,PosY) ,5)#MakeTrail(Pointer)
TypeError: points must be number pairs
```

I was not too sure what this was so I googled it and I found out that for the coordinates this function will only take one argument and will not work in the way I have put (posx, posy), it can only be a singular argument. To then rectify this issue, I essentially made a huge copy of the projectile coordinates for the trail to follow and called it trail list, I also had to adjust the coordinates

to be a little behind that of the projectile so I used a simple adjust function, the code shown below:

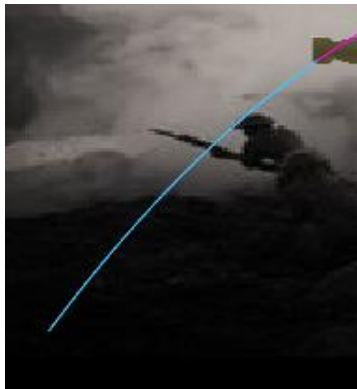
```
TrailList = copy.deepcopy(CoordinatesList) # this is a recursive call of copying the file over
TrailList = trail_tweek(TrailList)
```

This new argument can be plugged in as such

```
win_screen(win,h,u,v,t,ANGLEDEGREES,HIDE,MODE)

if Pointer>1:
 pygame.draw.aalines(Screen_View, MAGNETA,False,TrailList[:Pointer+1000] ,5)
 pygame.draw.aalines(Screen_View, CYAN,False,TrailList[:Pointer] ,5)
```

And give a beautiful parabola curve of the motion, like shown



The code for the trail\_tweek method is:

```
def trail_tweek(Trail): # makes the graphical aspect of the trail smoother
 for index in range(len(Trail)):
 Trail[index][0],Trail[index][1] = Trail[index][0]+2, Trail[index][1]+13
 return Trail
```

## User feedback

“The algorithm does give a graphical representation of the motion however the line is very faint and uneasy to see, I would recommend making it thicker and softer on the eye.”

## Reflections

After this brief discussion with my stakeholder, I was enlightened to two flaws of this module. For this reason, I did some research on the pygame. draw.aalines functions and because it is an ant aliased line you are not able to adjust the thickness of the line from 1 pixel, to combat this makes it looks as best I can I have switched the colour of the line to a more visible blue.



### Milestone 6: Have a non-linear AI defence system to follow the projectile and intercept (if ticked yes)

This is the non-linear and complex aspect of my game and it was very complex to integrate. The AI was imagined to not follow the SUVAT variables and instead chase the projectile, as otherwise, a calculated intercept would have been quite easy and not real AI.

```
AIcheck = CheckBox(430, 480, 30, Screen_View, "AI (ON/OFF):",140)
AIcheck.MakeBox()
AIcheck.click_seen((PosX,PosY))
```

### Code explanation

These 3 simple lines of code do not handle the entire AI system. However, these are what handle its existence. By the variable name you can tell that this has got to do with the checkbox for the AI. This is a user tickbox that will be displayed on the value input screen for the user to choose whether they wish to continue with AI or not before actually seeing the graphical example. The first line of code was defining this checkbox and its parameters. it does this by instantiating an object from the checkbox class, the MakeBox function is also a method of this class. I will show the code below for this class however will not go through it as it is simple graphical drawings and methods.

```

 return False

class CheckBox:
 def __init__(self, PosX, PosY, Width, Surface, Label, WordDistance):
 self.Font = pygame.font.Font(None, 25)
 self.WHITE = (255, 255, 255)
 self.TextColour = self.WHITE
 self.WordDistance = WordDistance
 self.Surface = Surface
 self.GREEN = (0, 255, 0) # tick colour
 self.Clicked = False
 self.StartX = PosX
 self.StartY = PosY
 self.Width = Width # button box dimensions
 self.Thickness = 8
 self.Label = Label

 def MakeBox(self): # draws the box by drawing lines in a rectangle
 TopLine = pygame.draw.line(self.Surface, self.WHITE, (self.StartX-2, self.StartY),
 (self.StartX+self.Width+2, self.StartY), 1)
 BottomLine = pygame.draw.line(self.Surface, self.WHITE, (self.StartX-2, self.StartY+self.Width+2),
 (self.StartX+self.Width+2, self.StartY+self.Width+2), 1)
 RightLine = pygame.draw.line(self.Surface, self.WHITE, (self.StartX+self.Width+2, self.StartY),
 (self.StartX+self.Width+2, self.StartY+self.Width+2), 1)
 LeftLine = pygame.draw.line(self.Surface, self.WHITE, (self.StartX-2, self.StartY),
 (self.StartX-2, self.StartY+self.Width+2), 1)

 self._AddLabel()
 if self.Clicked == True:
 self._AddTick()

 def click_seen(self, PositionTuple): # to see if user has clicked in the area
 if (PositionTuple[0] in range(self.StartX, self.StartX+self.Width+1)) and (PositionTuple[1] in range(self.StartY, self.StartY+self.Width+1)):
 if self.Clicked == True:
 self.Clicked = False
 else:
 self.Clicked = True

 def _TickCoordinates(self): # retrieves the tick position (private)
 X1 = self.StartX
 Y1 = self.StartY + (self.Width//2)
 X2 = self.StartX + (self.Width//2)
 Y2 = self.StartY + self.Width
 X3 = self.StartX + self.Width
 Y3 = self.StartY
 return X1, Y1, X2, Y2, X3, Y3

 def _AddTick(self):
 FirstX, FirstY, SecondX, SecondY, ThirdX, ThirdY = self._TickCoordinates()
 pygame.draw.line(self.Surface, self.GREEN, (FirstX, FirstY), (SecondX, SecondY), self.Thickness)
 pygame.draw.line(self.Surface, self.GREEN, (SecondX, SecondY), (ThirdX, ThirdY), self.Thickness)

 def _AddLabel(self):
 LabelX = self.StartX - self.WordDistance
 LabelY = self.StartY + 8
 Text = self.Font.render(self.Label, 1, self.TextColour)
 self.Surface.blit(Text, (LabelX, LabelY))

```

```

PosXAI = 500
PosYAI = GROUNDLEVEL
interceptor = pygame.image.load('flare.png')
collision = False

```

```

if AI==True:
 diffX = PosXAI - PosX
 diffY = PosYAI - PosY
 #print('(',diffX,',',diffY,')')
 moveX = diffX / AI_difficulty
 moveY = diffY / AI_difficulty
 x=random.randint(-2,2)
 y=random.randint(-2,2)
 #x=min((u/(random.randint(-2,2))),2)
 PosXAI -= (moveX + x)
 PosYAI -= (moveY + y)
 Screen_View.blit(interceptor, (PosXAI, PosYAI))
 maxPosX = PosX + 20
 maxPosY = PosY + 20
 minPosX = PosX - 20
 minPosY = PosY - 20
 if (PosXAI < maxPosX and PosXAI > minPosX) and (PosYAI < maxPosY and PosYAI > minPosY) :
 collision = True
 print("collision is ",collision)
 if collision == True:
 collisionimg=pygame.image.load('collide.png')
 collisionimg=pygame.transform.scale(collisionimg,(50,50))
 Motion=False

```

## Code explanation

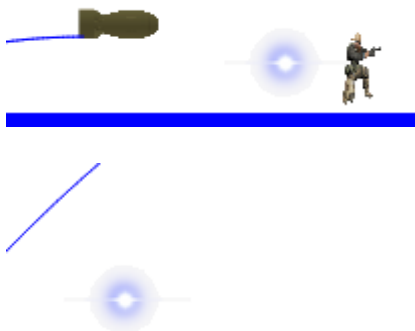
Now moving onto the actual AI itself, the checkbox function returns a Boolean value that tells you whether the user has selected the box or not. We can then use this Boolean cue to run conditional if statements allowing us to control the AI.

The first image of 4 lines is the initial bit of the AI within the main projectile motion function but outside of the game loop, because we do not want these to be updated, just set at the start. I use PosXAI and PosYAI to show the live x and y positions of the AI. The interceptor is the defence system in which the will effectively control, and the image for it is a blue flare as that is genuinely what is used to redirect infrared-guided missiles in real life. The collision is set to false as that is the default that would be needed for the game to run

The second picture and the main code within the game even loop. This is not the full method code in the picture, the AI IF statement is a triple nested statement, the first being the whole game loop, the motion=true statement to check that the projectile is moving as it is only then the AI can move as it metaphorically feeds off it. This piece of code was quite hard to computationally implement and I ended up using a lot of the pseudocode from the design stage to help configure this module. Initially, the AI was meant to be its separate function that would have been run recursively within the game loop however after much struggle with passing parameters I decided to simply implement the AI within the main function as this does operate as intended. Also as the AI code is in a while and IF loop this means that it is a recursive algorithm.

The code starts by simply checking if the AI checkbox has been ticked as it would not need to run without it, once that condition is met it creates a new variable called diffY and diffX, these stand for the difference in x and y. These

variables are assigned to the subtraction of the current position of the AI to the current position of the projectile. This effectively tells the AI how much it needs to move in those directions to reach the projectile and force an interception/collision. On the first recursion, the values for this would be 0 Y and max X as both would be at their respective base stations. The AI sets off from the right-hand side of the screen and the projectile from the left. The next line is commented out and was simply used for debugging purposes during the coding. After this I assign 2 new variables called moveX and moveY, this is because although we already have the distance and direction the AI has to move it simply cannot instantaneously teleport there, so these move variables divide the diff variables 50, giving the intermediate steps and technically speed of the AI. The next two lines are simply just assigning random values within the given range into a variable of which I then add to the move value. The move value is then taken away from the PosXAI and PosYAi. This step is simply to include some randomness in the motion of the AI and ensure that it will not always collide with the projectile if it is randomly thrown off track, in real life, you could model this as a gust of wind or the air-wash from the projectile. After this, I then make 4 maximum and minimum values on the live coordinates of the projectile, this is to be used in the next line of the statement. The if statement although very long is quite simple; it just compares the real-time x and y positions of the projectile with that of the positions of the AI, if the AI is within a given range of  $\pm 20$  pixels from the missile then it will trigger the collision boolean by setting it to True. The collision is later used in the load to display an explosion image and trigger a loss screen. If the AI was not fast enough and was not in the given range of the projectile and resultantly did not trigger a collision then, the projectile would go on to the target and get a win screen. Also with the collision being true, the motion is set to false, as the projectile is no longer moving. Below I will show some screenshots of the AI:



\_\_\_\_\_ as you can see the projectile surpasses the AI in the top left and goes over it. However, as the AI is non-linear and has effective inertia it slowly slows down and changes direction and continues to chase the projectile till a collision occurs. A collision will not always occur as sometimes the AI is purposefully not fast enough.

```

6.117824228899068
196.86890940907435
True
collision is True
717.41710771/1777
aTrue
collision is False

```

Test number	Test data	What to test and inputs	Expected results	Actual outcomes
1	Input values for SUVAT	The AI program must not calculate the path, or use pre-existing coordinates to intercept (print commands)	The code must be in isolation, AI follow the projectile, not predict its movement	As shown by the code the AI doesn't use any of the SUVAT variables, it's only intake is the live position of the projectile.
2	Input values for SUVAT	AI must simply use the current x, y of the missile and try to catch up to it (print commands)	AI will follow the projectile, AI values will not correspond till the collision	Again, as seen above by screenshots and code, the AI only follows and only has a reaction upon collision or when the projectile is no longer in motion.
3	Input values for SUVAT	AI must have a set acceleration, so will not always be able to catch the projectile Use multiple	The AI should be able to be beaten with a perfect flight path (values)	As also seen above the collision can also be false, this will only happen when the correct value is

		values of speed to see whether AI is beatable		inputted and the AI is not fast enough to keep up
--	--	-----------------------------------------------	--	---------------------------------------------------

## Testing amendments

In this section I will list some of the tests that did not go as planned and what I did to fix them.

For this section I generally stayed following my pseudocode from my design stage and that specified this line of pseudo-code:

```

difference = current_projectile_pos - AI_pos
move_AI_X = differenceX/ random.randint(1, (u/2))
move_AI_Y = differenceY/ random.randint(1, (u/2))

```

I decided to alter this a bit and broke it down into sections, the first being:

```
x=min((u/(random.randint(-2,2))),2)
```

Which when run received this error message:

```

File "F:\coursework cs game\test menu before messing with inputs.py", line 366, in InputScreen
run_sim(h,u,v,t,Angle,AIcheck.Clicked,HideCheck.Clicked,Mode,level)#Simulation for Proj_Motion
File "F:\coursework cs game\test menu before messing with inputs.py", line 438, in run_sim
x=min((u/(random.randint(-2,2))),2)
ZeroDivisionError: float division by zero

```

I ran this multiple times because initially, I thought that was an issue that was that the random module generated 0 as its value, however after running it over 10 times and consistently getting the same error I can say that that was not the case. I tried my best to fix this issue and had no luck I kept on getting the same error; for this reason, I decided to remove the limiting factor of U and the randomisation and instead do it all in simple steps and still get the randomisation to provide some limits, the new code is shown below:

```

diffX = PosXAI - PosX
diffY = PosYAI - PosY
#print('(' ,diffX , ' , ' ,diffY , ')')

moveX = diffX / AI_difficulty
moveY = diffY / AI_difficulty

x=random.randint(-2,2)
y=random.randint(-2,2)
#x=min((u/(random.randint(-2,2))),2)

```

By dividing the difference by 20 and setting limits on the random integer that I add or take from the movement it provides an aspect of randomness as well as limits the speed of the AI.

I have now decided to go back on my amendments and include a further non-linear aspect through this bit. By providing an evolving divisor for the

difference variable the AI and its effective speed is now also respective of the game situations.

After running this game and playing it through a few times I realised that it was not truly non-linear, so then I decided to alter my code in aim to remove the linear factor of it. To do this I altered the variable configuration of the difficulty of the AI and another variable AI\_win\_count. This variable is a simple integer value that pluses and minuses 1 from its tally of how many times the AI has lost and/or won, this has an influence and is what drives the AI\_difficulty variable. This variable has been coded so that it evolves and changes with time and with every run of the motion. If the AI is not able to beat the projectile the difficulty will decrease and the AI will get faster, this is effectively done by making the divisor smaller (AI\_difficulty) resulting in a larger effective speed of the AI. thus making it harder overall. If the AI wins then it is then slowed down. The AI\_difficulty and AI\_win\_count is both global variables so that they can be accessed from all parts of the code. I decided to take the global route as otherwise, I would have to pass into and through literally every function in the game, causing more chance for error and confusion; also the fact that there are no similar variables make sure overriding and potential errors should not be an issue.

As shown in the code below:

```
global AI_win_count
AI_win_count = 0
global AI_difficulty
AI_difficulty = 100

if Motion == False:
 if collision == True:
 Screen_View.blit(collisionimg, (PosX, PosY))
 win = False
 #global AI_difficulty
 AI_win_count += 1
 AI_difficulty += 20
 loss_screen(win,h,u,v,t,ANGLEDEGREES,HIDE,Mode)

 if collision == False: #and b_input == ans_b :
 win = True
 print("collision is", collision)
 #print("user : ",b_input , "ans:",ans_b)
 else:
 PosY = floorY_co_ord
 PosX = CoordinatesList[Pointer][0]
 Screen_View.blit(collisionimg, (PosX, PosY))
if win == True :
 AI_win_count -= 1
 AI_difficulty -= 20
```

```
moveX = diffX / AI_difficulty
moveY = diffY / AI_difficulty
```

(Some of the code is commented out as it makes it easier to test the AI)  
I will now show some screen recordings of how the non-linear aspect works.

[First screen recording](#)

(<https://drive.google.com/file/d/1g5h96CnjLbPzxljrHjM7DKRQJXXKRHhf/view?usp=sharing>)

[Second screen recording](#)

(<https://drive.google.com/file/d/1aQcK230M9Z5a4Slolz9VMdg9JEXBg3XW/view?usp=sharing>)

As seen in these screen recordings when inputting the same values each time the AI differentiates. In the first recording, the AI wins the first time so it slows down so it does not win so easily again. The second one shows the opposite, where the AI continuously loses so then it gradually gets faster to the point where it can catch the projectile. This 'difficulty' aspect of the AI changes over time and responds to the user winning or not.

## User feedback

My stakeholder was very impressed when I showed him this milestone and how well it works in its implementation. His comments were along the lines of "this looks good and clean in the animation, well done. You may want to check that the user can win without the AI winning and vice versa as I have only seen the AI win so far."

## Reflections

After looking back at the comments left by my stakeholder and having a little off-record conversation with him I was able to go back and make do on his advice and slow down the AI a bit, this was very simple however a little tedious. What I did was simply change that division factor of 20 to a higher number allowing for smaller increments and thus a slower effective speed of the AI. I changed this by intervals till I reached an approximate 50:50 win: loss ratio and this was at the scale factor of 50, the new code is shown below:

```
win = False
#global AI_difficulty
AI_win_count += 1
AI_difficulty += 20
```



## Milestone 7: Give a win or loss screen

This milestone identifies the part of the program in which the user will be told whether they were right or wrong, and will be met by the corresponding screen. If the user does not input the correct values then they will lose and there will be no target detonation and an AI collision if it's on, and vice versa.

```
if Motion == False:
 if collision == True:
 Screen_View.blit(collisionimg, (PosX, PosY))
 win = False
 loss_screen(win)
 if collision == False:
 win = True
 print("collision is", collision)
 else:
 PosY = GROUNDLEVEL
 PosX = CoordinatesList[Pointer][0]
 Screen_View.blit(collisionimg, (PosX, PosY))
 if win == True :
 win_screen(win)
```

### Code explanation

This is the code that I have implemented to call the win and loss screens. This code is very simple and only relies on some conditional statements. This code block is placed inside of the main game loop, meaning as soon as a win/loss is triggered by a collision the screen will update accordingly. The motion is set to false as the screen should only be displayed once the user has seen the full graphical representation through to completion, and the projectile either no longer exists or is not in motion. Alongside calling the respective screens the win variable is set and is passed into the function, this is not a necessary step at all, however just makes sure nothing goes wrong and adds extra validation.

```
def win_screen(win):

 message = Font.render("You are correct !! ",1,RED)

 while win == True:
 Screen_View.fill(WHITE)
 Screen_View.blit(message,(260,250))
 MENUButton = Button("Return to Main menu", 300, 530, 40, 270,Screen_View)
 MENUButton.DisplayButton()
 for event in pygame.event.get():
 if event.type == QUIT:
 pygame.quit()
 sys.exit()
 elif event.type == pygame.MOUSEBUTTONDOWN:
 (PosX,PosY) = pygame.mouse.get_pos()
 if MENUButton.click_seen((PosX,PosY)) == True: #Know you get to go to the input screen
 win = False
 menu.mainloop(surface)

 pygame.display.update()
 fpsClock.tick(FPS)
```

### Code explanation

This is the actual code for the win screen function. It is a very simple module and is not hard to implement. It starts by simply rendering the message with font and assigning that pygame object to the message variable. Then a game loop is created on the basis that wins is true, this is extra validation, while the game\_on loop would have been just as effective. On the first line of the while loop, the screen is filled with white for a blank canvas for the message to be delivered. Then the message is blitted onto the screen at the said x and y positions, this is so the message is as central as can be. I then also created a Menu button, in my initial design I did state that I would have 3 buttons but all these do effectively the same thing, going back to the menu allows the user to re-evaluate their standing point. This button has a caption of 'return to main menu' and has its dimensions set. The button is then instantiated. The rest of the code is arbitrary and has no major significance, as it basically listens out for any presses on the button and then runs the mainmenu function to return to the menu.

```
def loss_screen(win):

 message = Font.render("sadly, you are incorrect, try again ",1,RED)

 while win == False:
 Screen_View.fill(WHITE)
 Screen_View.blit(message,(120,250))
 MENUButton = Button("Return to Main menu", 300, 530, 40, 270,Screen_View)
 MENUButton.DisplayButton()
 for event in pygame.event.get():
 if event.type == QUIT:
 pygame.quit()
 sys.exit()
 elif event.type == pygame.MOUSEBUTTONDOWN:
 (PosX,PosY) = pygame.mouse.get_pos()
 if MENUButton.click_seen((PosX,PosY)) == True: #Know you get to go to the input screen
 #win = False
 menu.mainloop(surface)

 pygame.display.update()
 fpsClock.tick(FPS)
```

### Code explanation

This is the code for the loss screen and it is very similar to the win screen, the only difference in the code is the message and that this occurs when win is false.

Zain's coursework simulation v3


**You are correct !!**

**Return to Main menu**

**sadly, you are incorrect, try again**

**Return to Main menu**

**Explain**

 Zain's coursework simulation v3

$$v = -u$$

$$v = -33.0$$

Rearrange  $s = ut + 0.5at^2$  where  $s = 0$

$$\Rightarrow t = \frac{u \sin \theta}{4.9}$$

$$\Rightarrow t = \frac{33.0 \sin(44.0)}{4.9}$$

At time,  $t/2$ , height is at a max  $\Rightarrow$  using  $s = ut + 0.5at^2$

$$\Rightarrow s = u \sin \theta (t/2) - 4.9(t/2)^2$$

$$\Rightarrow s = 33.0 \sin(44.0)(4.7/2) - 4.9(4.7/2)^2$$

**OK**

Test number	Test data	What to test and inputs	Expected results	Actual outcomes
1	Input correct values for SUVAT, then incorrect	If the user is presented with a win or loss screen depending on the result Values for both win and loss	Loss screen should only be shown after a failed attempt Win screen at the correct answer	As seen above the loss screen is only triggered in the code if the correct values are not given. And the win if the correct values are given without an AI collision.
2	Input values for SUVAT	If the solution is explained on the win/loss screen	Regardless of win/loss, the solution will be displayed before moving on	The solution is no longer shown on the win/loss screen as previously stated, however, it is a button incorporated into the main screen of the game. There is a 'explain' button in the graphical screen that allows you to access the explanation. The live parameter as solutions will be continuously displayed should the user not check the hide box.

## **Testing amendments**

In this section, I will list some of the tests that did not go as planned and what I did to fix them.

This section was fairly straightforward as I had already implemented a lot of screens by now and knew what I was doing. There were the obvious little hiccups of syntax errors that were very easily and simply resolved by just going back and looking at the documentation syntax of said function.

I also had the very slight minimal errors of the text being displayed in the wrong form, size and position, however, this was easily solved by simple coordinate shifts.

I did not have any major errors in this milestone due to its simple nature.

## **User feedback**

“This works well, very simple and minimalist, it would be better if you added the explain button to the win loss screen as otherwise, time-consuming you cannot use it”

## **Reflections**

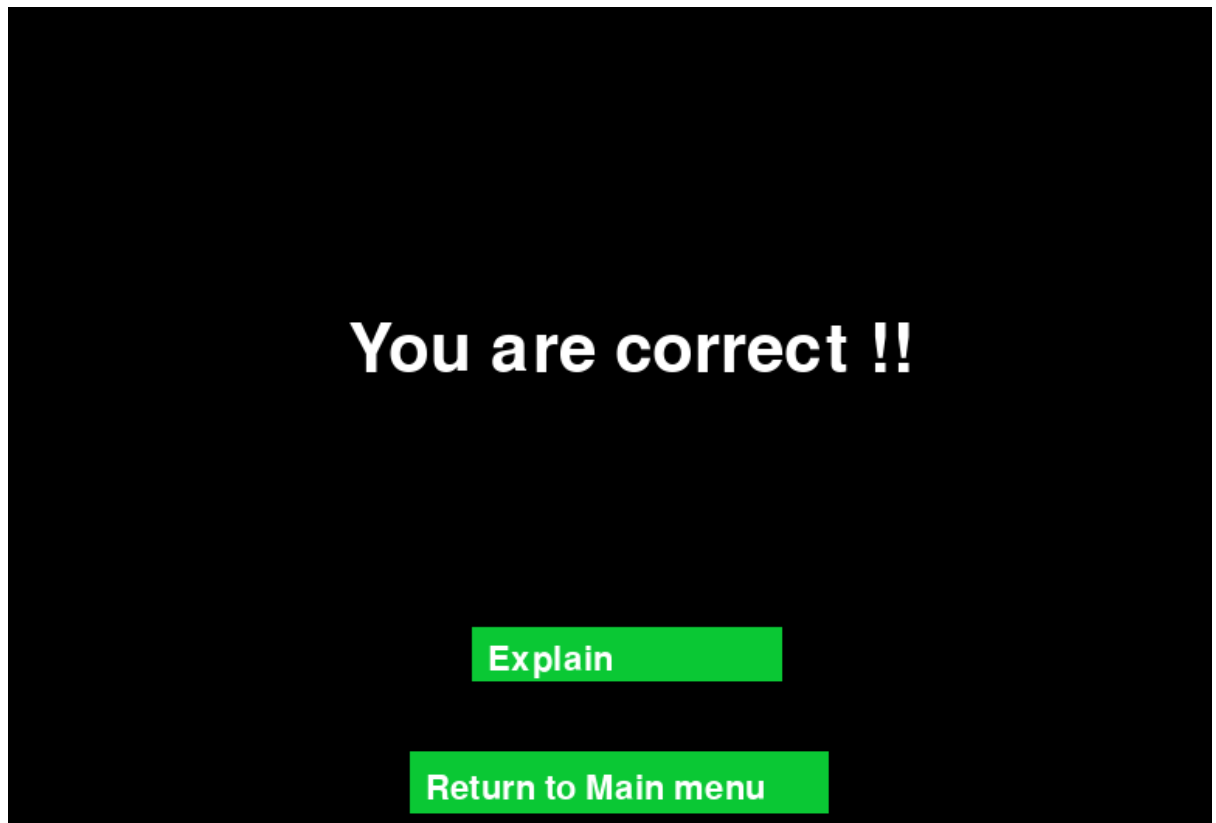
My stakeholder made a very good point about the explain button and I have decided to incorporate it into my win/loss screen. New code is shown;

```

def win_screen(win,s,u,v,t,ANGLEDEGREES,HIDE,Mode):
 message = Font.render("You are correct !! ",1,WHITE)
 s,u,v,t,ANGLEDEGREES,HIDE,Mode = s,u,v,t,ANGLEDEGREES,HIDE,Mode
 while win == True:
 win_loss_screen = True
 Screen_View.fill(BLACK)
 Screen_View.blit(message,(260,250))
 MENUButton = Button("Return to Main menu", 300, 530, 40, 270,Screen_View)
 ExplainButton = Button("Explain", 340, 450, 35, 200, Screen_View)
 MENUButton.DisplayButton()
 ExplainButton.DisplayButton()
 for event in pygame.event.get():
 if event.type == QUIT:
 pygame.quit()
 sys.exit()
 elif event.type == pygame.MOUSEBUTTONDOWN:
 (PosX,PosY) = pygame.mouse.get_pos()
 HIDE.click_seen((PosX,PosY))
 if MENUButton.click_seen((PosX,PosY)) == True:
 win = False
 menu.mainloop(surface)
 elif ExplainButton.click_seen((PosX,PosY)) == True:
 Explain = True
 #Running = False
 if Explain == True:
 if Mode == 1:
 explanations1(s,u,v,t,ANGLEDEGREES,HIDE.Clicked,Mode)
 elif Mode == 2:
 explanations2(s,u,v,t,ANGLEDEGREES,HIDE.Clicked,Mode)
 pygame.display.update()
 fpsClock.tick(FPS)

```

And the result:



### Milestone 8: Progress back onto the main menu, or next, or previous or re-do

This milestone is very trivial and is simply the finishing touches.

I have deviated from what this milestone declares following simple functionality and language change.

I have decided that instead of having the main menu, next, previous and redo buttons to just have a single main menu button, and this is for numerous reasons. The implementation of multiple buttons makes it much more complicated and messy as well as more chance for error.

As this is a revision tool it is the users who choose to use it and therefore they should be trusted to not cheat and to properly revise just re-do the level themselves without being restricted to that by a loss screen. By only redirecting the user to the main menu they are then able to go next, previous or re-do by their own free will. Also by going back to the main menu ensures that the questions and their values will update.



```

MENUButton = Button("Return to Main menu", 300, 530, 40, 270,Screen_View)
MENUButton.DisplayButton()
for event in pygame.event.get():
 if event.type == QUIT:
 pygame.quit()
 sys.exit()
 elif event.type == pygame.MOUSEBUTTONDOWN:
 (PosX,PosY) = pygame.mouse.get_pos()
 if MENUButton.click_seen((PosX,PosY)) == True: #Know you get to go to the input screen
 win = False
 menu.mainloop(surface)

```

## Code explanation

You have seen this code before in both the win and loss screens. All that this method does is to first instantiate a menu button variable from the button class and assign its arguments of dimensions and surface. The menu button is then put through the DisplayButton function which will handle the graphical representation of the button.

The for event loop is simply to listen to the pygame events and see if there is a mouse click, if there is it then checks if it is the area of the button and if so it then triggers the button and the menu function sends you back to the menu.

```

class Button: # this is a class i have mad eta handle the buttons (tutorial)
 def __init__(self, Label, PosXStart, PosYStart, Width, Length, Surface):
 self.Label = Label
 self.Font = pygame.font.Font(None, 32)
 self.TextColour = (255,255,255) #White text
 self.PosXStart = PosXStart
 self.Width = Width
 self.Length = Length
 self.PosYStart = PosYStart
 self.BackgroundColor = (10, 200, 52) # green buttons
 self.Surface = Surface
 def DisplayButton(self): # function called to actually draw
 Button = pygame.draw.rect(self.Surface, self.BackgroundColor, (self.PosXStart, self.PosYStart, self.Length, self.Width))
 self._MakeLabel()
 def _MakeLabel(self):
 LabelX = (self.PosXStart + 10)
 LabelY = (self.PosYStart + (self.Length//20))
 ButtonText = self.Font.render(self.Label,1,self.TextColour)
 self.Surface.blit(ButtonText, (LabelX,LabelY))
 def click_seen(self, PositionTuple):
 if (PositionTuple[0] in range(self.PosXStart, self.PosXStart+self.Length+1)) and (PositionTuple[1] in range(self.PosYStart, self.PosYStart+self.Width+1)):
 return True

```

## Code explanation

This is the button class that governs the rest of the buttons used through the entire program. All this class does is graphically create a button by drawing a rectangle and assigning arguments to it so it can be interacted with, the clickseen function simply defines an area tuple surrounding the button in which interacted with will trigger the button.

The displaybutton method referred to in the code is also seen as a method of the button class which when called will draw the button itself and consequences call the rest of the makelabel methods which assigns the length, text and colour of the button, and then blitz it the surface.

**Return to Main menu**

Test number	Test data	What to test and inputs	Expected results	Actual outcomes
1	Input values for SUVAT LMC (left mouse click) on back, forward, the main menu, redo buttons	On the win/loss screen to be presented with the option to go back, forward, main menu or to redo	These buttons to be presented and work	The main menu button is presented and successfully works
2	Input incorrect values for SUVAT LMC (left mouse click) on the redo button	On the loss screen, you should not have the option to advance	Cannot carry on until correct answer, loss screen.	This is no longer a requirement, you are simply alerted if incorrect.

**Testing amendments**

In this section, I will list some of the tests that did not go as planned and what I did to fix them.

Apart from the simple syntax issues, I did not have any major testing amendments or failures to deal with

**User feedback**

“There is not much to this, a button that puts you back on the menu. It's very simple and works. Try and add the explain option to the win/loss screen”

## Reflections

I had given this to my stakeholder for his feedback before I had completed the reflections on the previous milestone, therefore I have already adhered to my users' comments and integrated the explain option on the win/loss screen.

## Input Validation

For this section, I will be conducting and explaining the input validation that I had previously described in my design stage.

I will use the same table to create a design with two new columns to show the outcomes and evidence. As not all variables require validation I will only show those that do.

I will now also show the code Unicode have written to achieve this validation:

```
def AddCharacter(self, Input): #Adds character from alphabet -> ['1','2','3','4','5','6','7','8','9','.','BACKSPACE']
 if (Input == K_1) and (len(self.Variable)<self.StringLength):
 if self._NumInRange(1)==True:
 if "." in self.Variable:
 self.Variable = self._AdjustVariable(1)
 else:
 self.Variable += "1"
 elif (Input == K_2) and (len(self.Variable)<self.StringLength):
 if self._NumInRange(2)==True:
 if "." in self.Variable:
 self.Variable = self._AdjustVariable(2)
 else:
 self.Variable += "2"
 elif (Input == K_3) and (len(self.Variable)<self.StringLength):
 if self._NumInRange(3)==True:
 if "." in self.Variable:
 self.Variable = self._AdjustVariable(3)
 else:
 self.Variable += "3"
 elif (Input == K_4) and (len(self.Variable)<self.StringLength):
 if self._NumInRange(4)==True:
 if "." in self.Variable:
 self.Variable = self._AdjustVariable(4)
 else:
 self.Variable += "4"
 elif (Input == K_5) and (len(self.Variable)<self.StringLength):
 if self._NumInRange(5)==True:
 if "." in self.Variable:
 self.Variable = self._AdjustVariable(5)
 else:
```

```

else:
 self.Variable += "5"
elif (Input == K_6) and (len(self.Variable)<self.StringLength):
 if self._NumInRange(6)==True:
 if "." in self.Variable:
 self.Variable = self._AdjustVariable(6)
 else:
 self.Variable += "6"
elif (Input == K_7) and (len(self.Variable)<self.StringLength):
 if self._NumInRange(7)==True:
 if "." in self.Variable:
 self.Variable = self._AdjustVariable(7)
 else:
 self.Variable += "7"
elif (Input == K_8) and (len(self.Variable)<self.StringLength):
 if self._NumInRange(8)==True:
 if "." in self.Variable:
 self.Variable = self._AdjustVariable(8)
 else:
 self.Variable += "8"
elif (Input == K_9) and (len(self.Variable)<self.StringLength):
 if self._NumInRange(9)==True:
 if "." in self.Variable:
 self.Variable = self._AdjustVariable(9)
 else:
 self.Variable += "9"
elif (Input == K_0) and (len(self.Variable)<self.StringLength):
 if self._NumInRange(0)==True:
 if "." in self.Variable:
 self.Variable = self._AdjustVariable(0)
 else:
 self.Variable += "0"
elif (Input == K_PERIOD) and (len(self.Variable)<self.StringLength):
 if "." not in self.Variable: #You should only be able to enter a decimal ('.') if the variable doesn't already have one
 self.Variable += "."

elif (Input == K_BACKSPACE): #Deletes the final character in 'Variable'
 self.Variable = self.Variable[:-1]

```

### Code explanation

This is the code I have written to do part of the input validation. This code directly is not validation in itself however does do a lot towards it, by limiting the format. As the only inputs my program requires throughout is numerical that means that the program only has to recognise numerical keyboard user inputs, in doing so a keyboard press of a non-numeric character will simply just not be recognised or acknowledged. If my program did require some string inputs I would have to use the pygame Unicode function which allows for all keyboard inputs, this would have made the validation a lot more complex and difficult to implement.

```

def _AdjustVariable(self,String): #Private method that adjusts the input so that it is always to 1dp
 self.Variable += str(String)
 self.Variable = "%0.2f" % float(self.Variable)
 return self.Variable[:-1]

```

### Code explanation

This is a simple private method that executes sanitisation on the user inputs by simply rounding and converting all inputs to 1 decimal place.

```

def _NumInRange(self, Num): #Checks if the number will be in range and returns if this is the case or not
 if (float(self.Variable + str(Num))>=self.Range[0]) and (float(self.Variable + str(Num))<=self.Range[1]):
 return True
 else:
 return False

```

### Code explanation

This is another simple private method that performs a length check on the user inputs.

```
def _AddTick(self):
 #Initialise coordinates
 FirstX,FirstY,SecondX,SecondY,ThirdX,ThirdY = self._TickCoordinates() #Private methods used to get coordinates of
 #Actually Draws Lines => making the tick
 pygame.draw.line(self.Surface, self.GREEN, (FirstX,FirstY),(SecondX,SecondY),self.Thickness)
 pygame.draw.line(self.Surface, self.GREEN, (SecondX, SecondY),(ThirdX,ThirdY),self.Thickness)

def _AddLabel(self): #Private method used to add a Label to a checkbox
 LabelX = self.StartX - self.WordDistance
 LabelY = self.StartY + 8
 Text = self.Font.render(self.Label,1,self.TextColour)
 self.Surface.blit(Text, (LabelX,LabelY))
```

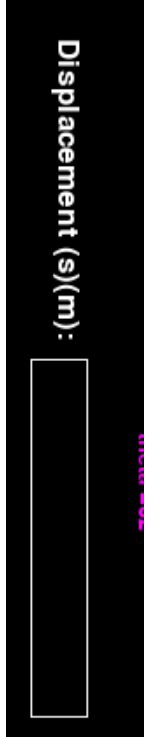
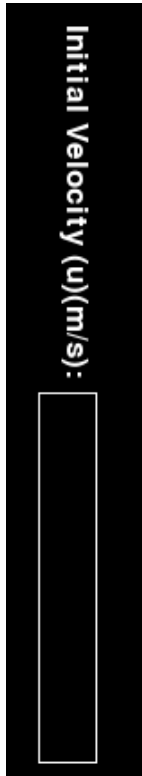
### Code explanation

This is a private method within the checkbox class mentioned before. This is a form of lookup table validation as it restricts and only allows true inputs, checked or not checked, true and false respectively.

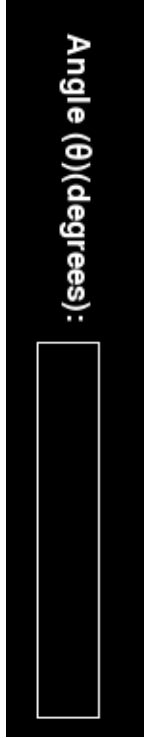

### Validation checks

Validation check	How it works	Example
Format check	The data type is correct and obeys expectations	If the string is expected, then would reject integers or Boolean
Length check	The number of characters is not too long or short	A value that shouldn't be over 100000, so is rejected
Presence check	Data has been entered	Must enter a value
Lookup table	restrict inputs	Choose from a drop-down list
Sanitisation	Converting the input syntax/format to valid	If a string is given in place of an integer, can convert the string into the desired integer

Name	Data Type	How it is used	Input Validation	Outcomes	Evidence

s	integer	This will be assigned to the random number generated at the start of the level, or assigned 0 if the variable is not given. The user can update the value of the non-given variable to get the answer	Format check Length check Presence check	The validation for displacement works very well as it does not allow input of more than two digits, or a non-numerical input.	
u	integer	This will be assigned to the random number generated at the start of the level, or assigned 0 if the variable is not given. The user can update the value of the non-given variable to get the answer	Format check Length check Presence check	The validation for displacement works very well as it does not allow input of more than two digits or a non-numerical input.	

Mass	integer	<p>This will be assigned to the random number generated at the start of the level, or assigned 0 if the variable is not given. The user can update the value of the non-given variable to get the answer. Mass is quite an irrelevant parameter in projectile motion but I included it so the user can understand that it does not affect the motion.</p> <p>Also as it is not needed I have limited mass to just 1 digit.</p>	<p>Format check Length check Presence check</p>	<p>The validation for displacement works very well as it does not allow input of more than two digits or a non-numerical input.</p>	<p>Mass (kg):</p> <input type="text"/>
------	---------	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	---------------------------------------------------------	-------------------------------------------------------------------------------------------------------------------------------------	----------------------------------------

angle	integer	This will be assigned to the random number generated at the start of the level, or assigned 0 if the variable is not given. The user can update the value of the non-given variable to get the answer. The angle should also have a max value of 90 so it does not shoot backwards.	Format check Length check Presence check	The validation for displacement works very well as it does not allow input of more than two digits or a non-numerical input.	
Level	Integer	Depending on which level tile the user selects in the menu, this variable will be assigned according to value. This will be used to determine the input and outcome of other algorithms such as displayproblem()	Presence check	Level does not have a true validation check as without the game would not function and it is not user input.	



level_in_progress (running)	Boolean	This is a data value with a default of False, when a level is selected then will be turned to true, this will determine the appearance of the scene and contributes to other algorithms	Format check Presence check	again, this does not have a true validation check as it is not an input, thus is always present and the format is set due to Boolean nature.	while Running:
--------------------------------	---------	-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	--------------------------------	----------------------------------------------------------------------------------------------------------------------------------------------	----------------

## Evaluation

### Data value test results

In this section, I will be conducting the testing I have previously described in my design stage. I will use the same table with an appended column of actual outcomes. I will then go on to use this test data to either make changes to my work or evaluate the results in the final project evaluation.

I have made some changes to the test data and the expected outcomes as I have altered my code and does not meet what I initially intended in my design stage.

The video of this testing is linked below, the timestamp column refers to the moment in time within this recording:

I will also be using a key logger during this video to show exactly what inputs I put in. The mouse wheel up/down inputs are just me checking what I need to test outside the picture frame and has nothing to do with the actual testing.

*input testing video link*

(<https://drive.google.com/file/d/1B0-VZn1kl8pVCNAO64LnuTJir2nOidSx/view?usp=sharing>)

Test data	Expected outcome	Actual outcome	Timestamp
LMC (left mouse click) on level 1	Valid	Valid	0:04
LMC (left mouse click) on level 2	Valid	Valid	0:12
LMC (left mouse click) on level 3	Valid	Valid	0:14
LMC (left mouse click) on level 4	Valid	Valid	0:16
LMC (left mouse click) on level 5	Valid	Valid	0:18
LMC (left mouse click) on level 6	Valid	Valid	0:20
LMC (left mouse click) on level 7	Valid	Valid	0:22
LMC (left mouse click) on level 8	Valid	Valid	0:24
RMC (right mouse click) on level 1	Invalid	Valid	0:26
RMC (right mouse click) on level 2	Invalid	Valid	0:28
RMC (right mouse click) on level 3	Invalid	Valid	0:32
RMC (right mouse click) on level 4	Invalid	Valid	0:34
RMC (right mouse click) on level 5	Invalid	Valid	0:36
RMC (right mouse click) on level 6	Invalid	Valid	0:38
RMC (right mouse click) on level 7	Invalid	Valid	0:40
RMC (right mouse click) on level 8	Invalid	Valid	0:42
LMC (left mouse click) on quit	Valid	Valid	1:14

button			
RMC (right mouse click) on quit button	Invalid	Valid	1:14
Enter value '1' in displacement input	Valid	Valid	1:40
Enter value '0' in displacement input	Valid (boundary)	Invalid	1:50
Enter value '0.5' in displacement input	Valid	Valid	1:58
Enter value '100' in displacement input	Valid (boundary)	Valid	2:06
Enter value 'abx' in displacement input	invalid	Invalid	2:12
Enter value '999' in displacement input	invalid	Valid	2:22
Enter value '099' in displacement input	Valid (boundary)	Valid - cancel the leading 0	2:30
Enter value '1999' in displacement input	Invalid	Valid cancels out the last 9 to display '199'	2:39
Enter value '655987138749' in displacement input	Invalid	Invalid - only shows first 3 digits	3:00
Enter value '1' in initial velocity input	Valid	Valid	3:19
Enter value '0' in initial velocity input	invalid	Invalid	3:28
Enter value '100' in initial velocity input	Valid (boundary)	valid	3:36

Enter value 'abx' in initial velocity input	invalid	invalid	3:43
Enter value '999' in initial velocity input	invalid	Invalid - '99' shown	3:53
Enter value '099' in initial velocity input	Valid (boundary)	Valid - '99' shown	4:02
Enter value '1999' in initial velocity input	Invalid	Invalid - '19' shown	4:09
Enter value '655987138749' in initial velocity input	Invalid	Invalid - '65' shown	4:16
Enter value '90' in angle input	Valid (boundary)	Valid	4:28
Enter value '45' in angle input	Valid	Valid	4:37
Enter value '0' in angle input	Invalid	Invalid	4:47
Enter value '999' in angle input	Invalid	Invalid - shows '9'	4:54
Enter value '10' in mass input	Valid (boundary)	Valid	5:16
Enter value '0.1' in mass input	valid (boundary)	Valid	5:18
Enter value '999' in mass input	invalid	Invalid	5:23
Enter value '5' in mass input	Valid	Valid	5:27
LMC (left mouse click) on level explain button	Valid	Valid	5:43
RMC (right mouse click) on level explain button	Invalid	Valid - does not make a difference to the	5:52

		program	
LMC (left mouse click) on OK button	Valid	Valid	5:58
RMC (right mouse click) on OK button	Invalid	Valid - does not make a difference to the program	6:01
LMC (left mouse click) on main menu button	Valid	Valid	6:05
RMC (right mouse click) on main menu button	Invalid	Valid - does not make a difference to the program	6:10
LMC (left mouse click) on submit button	Valid	Valid	6:25
RMC (right mouse click) on submit button	Invalid	Valid - does not make a difference to the program	6:30
LMC (left mouse click) on hide variables checkbox	Valid ( should bring up a tick)	Valid	6:44
RMC (right mouse click) on hide variables checkbox	Invalid	Valid - does not make a difference to the program	6:47
LMC (left mouse click) on AI checkbox	Valid ( should bring up a tick)	Valid	5:56
RMC (right mouse click) on AI checkbox	Invalid	Valid - does not make a difference to the program	5:58

## Function and robustness testing

The above test table lends towards both function and robustness testing; to continue this I will provide one more video to consolidate this. It will simply be me trying random stuff to try and break the program and get it to crash, this will include extensively using every feature.

*Additional testing video*

(<https://drive.google.com/file/d/1A9mr9ws8SHOrhQLhCkKhjsxMyO1tI7j4/view?usp=sharing>)

As you can see in the video with the alongside hotkey application, I was constantly spamming inputs during the run time of the program trying to get it to lag or crash with the input bombarding, however, it did not crash because of this and handled it very well. I believe the input validation on the code is very successful.

However, the program did ultimately crash, and this was not because of the spammed inputs, this is because of how the code works. This code line:

```
AI_difficulty -= 20
```

Takes 20 away from the AI difficulty every time the AI loses, effectively making it faster. The initial value of this variable is set to 100, meaning that after 5 losing iterations the difficulty becomes 0. Then this causes an error in this code line:

```
moveX = diffX / AI_difficulty
moveY = diffY / AI_difficulty
```

As you cannot have any real number being divided by 0, as that would give you an imaginary or infinite number which is not feasible, thus this error:

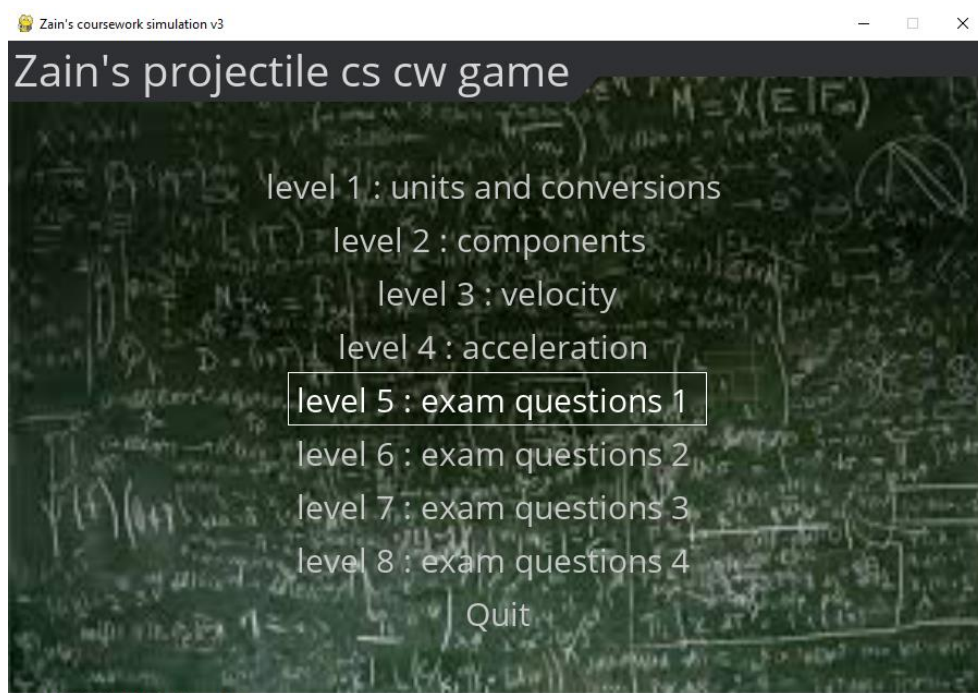
```
File "I:\coursework cs game\final code file.py", line 444, in run_sim
 moveX = diffX / AI_difficulty
ZeroDivisionError: float division by zero

Process returned 1 (0x1) execution time : 99.342 s
Press any key to continue . . .
```

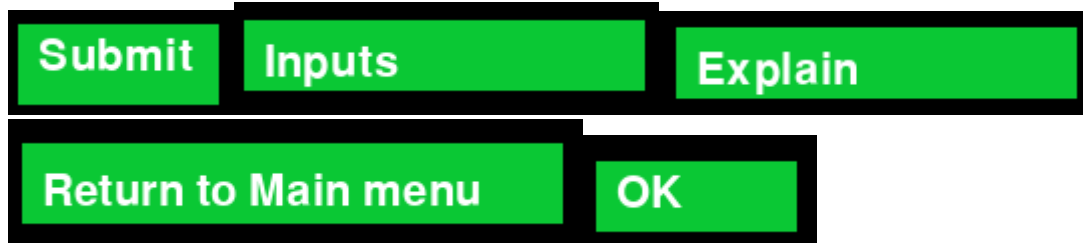
This could be easily fixed by instead of having a -20 on the difficulty each time I could just divide it by 2. This means that the value would never actually reach 0, but after thousands of losses will get very close, and will thus prevent the zero division error.

*Fix to this is shown here*

## Usability testing



This is the main menu that you are greeted with upon entering the game. This is quite very simple in the fact that the button you click will navigate you to the corresponding screen. The other navigation buttons are shown below:



All these buttons now work flawlessly especially after tweaking them in the development stage alongside the module testing.

The testing on if the buttons actually work is done in the *testing for evaluation*. Below I have shown my stakeholder's (Yassine Soltani) comments on the usability features of my game:

*"It's a very simple game with not much to go wrong or go right. The buttons all work as they should and look alright, they do the job. The game front menu is a little bland but from previous conversations, I understand why the image menu was not possible. The menu does the job and works well. I have now clocked over 10 minutes of me just trying to get this game to crash by spamming buttons and inputs and it has done well, well done on the input validation. This game is very easy to use and navigate especially with the new return to menu button in the projectile screen."*

Usability feature	Job	Success?
Submit button	Take the user to the next screen and confirm values. E.g. submitting input values would run the animation and produce answers	Full success
Inputs button	Take you to the input screen and reset older inputs.	Full success
Explain button	Take you to the explain screen and trigger the explain function, that will display the solution	Full success
Return to main menu button	Take you back to the main menu from any point in the game, and will reset inputs.	Full success
Ok button	Will also take you to the next screen, e.g. ok on the win/loss screen would take you to the main menu	Full success
Menu select options	This is the front feature of the game and simply navigates you to the level that you select, and also sets the level variable	Full success
Mouse inputs	This is a general feature that allows you to press the menu options and buttons	Full success
Animation shown	Show an animation of the path of the projectile This is a visual feature for the aid of the user	Full success
variables shown	Show the variables/answers to the SUVAT calculations. This is a visual feature	Full success



	for the aid of the user	
Checkboxes	These are input boxes used for AI and if the user wishes to see the variables or not	Full success

All the buttons had a slight flaw in development, due to the nature of the pygame event module I could have specified left/right click however I did not notice and only specified click, this now meant that both left and right clicks register as a click and trigger the function. This was initially seen as a flaw but now I see that this does not matter and has no effect on the usability of the game, if anything it makes the game more usable and user friendly as it would respond to either click.

The usability of this game heavily depends on the graphical aspect of the animation showing the motion. The evidence for this can be seen in videos linked before or the screenshot below:

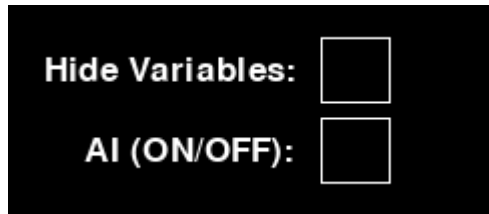


The variables shown are also a very important usability factor and an underlying aim of my game. The user should be able to see the correct answers and see if they got them right, this also would apply to the explanation screen. The variables shown evidence is below (also seen in videos):

**S = 251.9 m**  
**U = 55.0 m/s**  
**V = -55.0 m/s**  
**A = -g = -9.8 m/s<sup>2</sup>**  
**T = 9.2 s**

The checkboxes are an adaptation of the input box and allow the user to check if they want the AI and if they want to see the variables, this may be helpful if the user simply wants to see the path without knowing the answers

to help themselves toward an answer or whether they want to play the AI or not. The screenshot of the checkboxes is below (also seen in videos):



There was nothing that went wrong with the usability features of the game as they were very simple to implement especially after I had watched a tutorial and got the hang of how to do them.

### Beta testing

I have conducted alpha testing of my project through iterative development and post-development testing. It is now time to roll out my program as in real life and progress into the beta testing phase. For this phase, I will be rolling my program out to two of my stakeholders (Yassine Soltani and Sarrujan Raguparan).

Comments from Yassine:

*“This game has a very simple concept and yet I can imagine how difficult it would be to implement it, and also that I saw you struggle through it. However, the finished product is exactly what I pictured from the start when you pitched it to me. The game works and the math is correct and I would even go on to say that it is a learning aid that I would use and have simply because of the visual aid.”*

Comments from Sarrujan:

*“As far as I can see the game works perfectly, so far I have not encountered any bugs or crashes and the aim of the game is there. Meaning it does work as an educational tool. The only improvements I would say is to just spice it up a bit, basically, make it look better and cooler”*

Upon looking at my stakeholder's comments, the only thing I have picked up on that my game is lacking is that it just works, it's not extraordinary it's just 'meh' and doesn't break any bounds. If I had more time and if I was to redo this project knowing what I now know I would change this. I would approach this by firstly incorporating my initial idea of a horizontal scrolling menu by researching ahead of time and learning the intricacies of the language before I even started the project. I would also incorporate my original idea of implementing a changing height platform, which would make the game a lot more fun and intuitive. The reason why I did not incorporate these original

ideas is that they were simply not feasible once I had actually started to try and do them during development.

### Success criteria evaluations

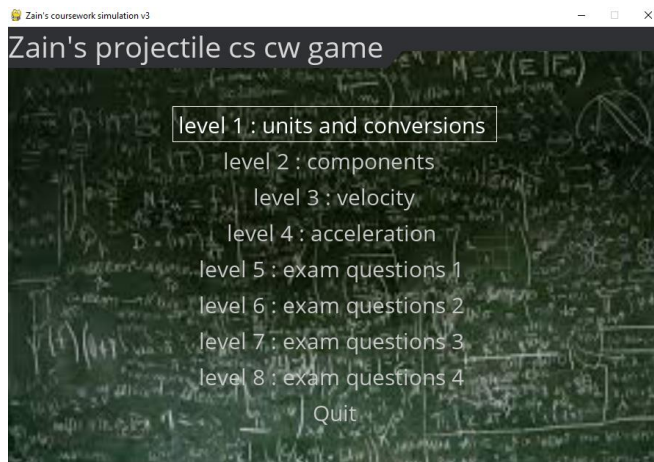
Here I will be discussing the success criteria that I defined in the analysis stage. I will be evaluating how well I met these criteria using met, partially met or not met. If partially met or not met I will go on to talk about how I would have gone about fully meeting these criteria if I had more time and the amendments, I would make. If fully met, then I will outline what went well and what could have been improved. for some of these criteria will also take in the feedback of my stakeholder after the beta testing conducted previously

no.	Requirement	Justification	Criterion met?
1	Start-up screen	This is needed to show the user the game that they are playing	Partially met

My initial idea for the start-up screen was for it to be an actual 'loading screen' type, where when you load the game the logo comes up while the game loads itself and the resources in the background, like this:



However, I decided not to implement this in the end. This is because the game is not very resource-intensive, meaning it does not need much time to load thus rendering a loading screen irrelevant as it would be done in a matter of milliseconds. Instead, I have made my game so that it starts straight on the menu select screen, shown below:

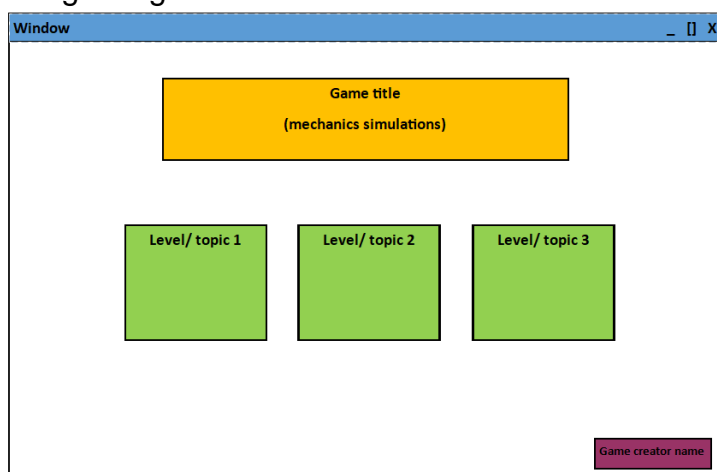


This effectively still achieves the same aim of showing the game name and presenting it to the user, without a loading bar/screen,

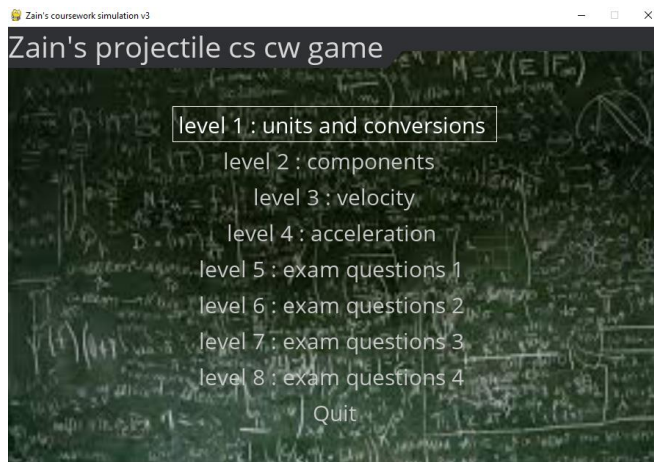
If I had more time and if my game required the time surpassed with a loading screen to load itself then I would implement one, especially as it is not a difficult thing to do.

no.	Requirement	Justification	Criterion met?
2	The menu allows game level selection	This is so the user, if confident in a topic will not have to do it just because they have to. For it to remain fun they should be able to choose what they need to do themselves	Partially met

Although I have implemented a fully functional game menu select navigation system it is not what I envisioned at the start and when defining this criterion. Below is a diagrammatic sketch of what I envisioned of this menu in the design stage:



And this is what my actual menu selection looks like;



As you can see I have opted out of the tile image format and the big title with the creator name; instead, I have opted for a simple text-based menu select system and a creator name inclusive game title. This menu is an adaptation of the default pygame menu. The reason for the altering for this success criterion can be found in the first milestone evaluation within the *development*. I have labelled it as partially met as the justification for the menu is still met, as users can still navigate the levels as and when they chose.

no.	Requirement	Justification	Criterion met?
3	Displaying the problem with randomly generated values	This is simply so that the question is never the same so the user will not get bored if they decide to revise repeatedly	Fully met

This is an extremely simple criterion to implement and the code for it is shown below:

```
Px = random.randint(10,100)
Py = random.randint(10,100)
Ptheta = random.randint(10,85)
x = questionfont.render(("x =" + str(Px)),1,MAGNETA)
theta = questionfont.render(("theta =" + str(Ptheta)),1,MAGNETA)
y = questionfont.render(("y =" + str(Py)),1,MAGNETA)
```

```

while Running:

 Screen_View.fill(BLACK)
 Screen_View.blit(Title,(120,5)) # displays
 Screen_View.blit(warning,(230,55))
 #pygame.draw.line(Screen_View, GREEN, (450,
 if level == 5:
 Screen_View.blit(question,(160,80)) #d
 Screen_View.blit(questiona,(280,110))
 Screen_View.blit(x,(430,130)) #display
 Screen_View.blit(theta,(420,150))
 elif level == 6:
 Screen_View.blit(questionb,(290,80))
 elif level == 7:
 Screen_View.blit(questionc,(80,80))
 Screen_View.blit(y,(350,130))

```

Simply the first few lines generate the random values and format them into strings and then the second box just displays them depending on the question.

Proof of the justification can also be seen below;

ie taken until it , time taken until it d, taken until i nes away, nes away, what

```

x =97 x =59 x =54
theta =18 theta =39 theta =48 y =45 y =91 y =24

```

As you can see I have both the levels with the randomised variables 3 times each and all three times the variables are unique. this therefore perfectly meets my justification of the criterion.

no.	Requirement	Justification	Criterion met?
4	Use random and differing variables in the questions	To keep the questions unique you would have to switch around the variables given and values to be calculated in the question	Fully met

This criterion is closely related to the last, as it uses the same randomly generated values as the last success criterion. The first bit of the criterion is the same in the sense that the random variables are shown and the last part is also met as the questions require you to calculate different variables each time, meaning all the aspects of understanding and formulae manipulation will be tested and enhanced. This can be seen by the code which defines the 4 possible questions and the final results, shown below:

to input, select the box and type (max 2 digits), enter the given variables

If a missile is launched from ground level with an initial velocity of  $x$  at an angle of  $\theta$  degrees

a) Calculate the time taken until it detonates the target

$x = 40$   
 $\theta = 81$

b) How far was the target from the launch site

c) If the target changed to being  $y$  kilometres away, what angle would you have to fire at assuming velocity is unchanged.

$y = 69$

d) Still firing at this new target, the shells have been changed thus altering the initial velocity.  
 Provide a combination of  $\theta$  and initial velocity that these new shells will hit the target.

These are the 4 input screens for the 4 projectile motion input screens and as you can see they all show their respective questions and parameters. The 4 different questions also ask for different values and variables to be worked out each time, thus meeting the criterion.

no.	Requirement	Justification	Criterion met?
5	Display the solution to the problem after an attempt	This is to explain to the user how to reach the solution so they will gain an understanding and hopefully get better at that topic question	Fully met

My solution allows you to access the explain function at both the win/loss screens and during the actual graphical motion. This means that the user can always refer back and check their work to see if they have done it correctly or not. An example of an explanation screen is shown below:

$$v = -u$$

$$v = -34.0$$

$$\text{Rearrange } s = ut + 0.5at^2 \text{ where } s = 0$$

$$\Rightarrow t = u \sin \theta / 4.9$$

$$\Rightarrow t = 34.0 \sin(34.0) / 4.9$$

$$\text{At time, } t/2, \text{ height is at a max } \Rightarrow \text{ using } s = ut + 0.5at^2$$

$$\Rightarrow s = u \sin \theta (t/2) - 4.9(t/2)^2$$

$$\Rightarrow s = 34.0 \sin(34.0)(3.9/2) - 4.9(3.9/2)^2$$

no.	Requirement	Justification	Criterion met?
6	If the user gets questions wrong, then redo the level with a new problem. If a user gets it right they should be presented with the choice to advance or not	If the student gets the answer wrong then that indicates that they do not know how to reach the solution, so once the solution is explained to them they should be able to go back and do another question to fully grasp how to do it. If the user is correct they could advance but if they are revising they may just want to repeatedly do that one topic again until they wish to stop.	Not met

I have decided to opt out of this criterion during the development as it added unnecessary complications. After talking to my stakeholder about it we concluded that this is a revision game. If a student wants to voluntarily revise then it is up to them to do so, we should not need to stop or control them. For this reason, this feature has not been implemented. Instead what has been implemented is a simple win and loss screen depending on the outcome of the user's inputs. The win/loss screens will have 2 buttons, one to explain the



solution and the other to return to the main menu button, which simply takes the user back to the main menu; from here they can choose to advance, go back or redo the questions by themselves. This also gives them the chance to go to the information levels, refresh their knowledge and try again. If I had more time and this requirement was necessary, then I would have implemented this by changing the buttons and their respective addresses. For the win screen, the 'return to the main menu button would be replaced by advance and would set the level to level + 1 and then call the inputscreen function to initiate the next level. The loss screen would replace the 'return to main menu button with redo and set that to inputscreen function where it would effectively just re-run the level.

no.	Requirement	Justification	Criterion met?
7	The background/setting will be according to the level and values	The first topic/level would just be a simple elevated platform from which the launcher would shoot, as the topic gets harder the launcher would move to the ground to factor in the first half of the parabola and other factors to represent that specific topic.	Not met

I have also decided to opt-out of this criterion, this is because it was not vital to the game and it has been covered with the 4 questions, all the content is still assessed and tested as this criterion tried to achieve. Also upon closer inspection, I have realised that I have misworded and misunderstood my intentions in specific reference to this criterion in the analysis and design stage. This is because the justification states above refer to different levels/topics where the scenarios stated would only constitute one topic, thus not possible to spread it out over two levels. If I was to go back and redo my project knowing what I know now I would have implemented this feature correctly, as I stated in my *analysis*, I had researched the Phet Colorado game which is similar to my solution. In their adaptation, they have this 'platform' that is mutable in height and is a cool concept, and I would implement that if I had the chance. This would be quite a difficult feature to implement and this is partly the reason why I had opted out of it, as it was simply not feasible with the time scale.

no.	Requirement	Justification	Criterion met?
-----	-------------	---------------	----------------

<b>8</b>	Allow user to enter their calculated answers	To complete the level and advance you will have to calculate the answers to the missing variables and enter them	Fully met
----------	----------------------------------------------	------------------------------------------------------------------------------------------------------------------	-----------

This criterion is a fundamental concept of my whole idea and solution as without my game has no purpose or data to run. This requirement was one of the main focuses during my coding and does make up a fair chunk of it. For the inputs the main thing was validations and the graphical side of things, as taking command line inputs is as simple as a single line in python, however, graphically is not so simple. For the validation I adopted a quite unorthodox method which I had seen in a tutorial, this was instead of using the pygame Unicode library to recognise input I would essentially create my own. This was because the Unicode allows for all inputs, meaning characters could be inputted in place of numerical data and such. By defining my input dictionary, inputs are restricted, meaning that even if someone were to try and type a string in a numerical input the characters would simply not be recognised. The graphical aspect of this was also very challenging as you had to draw the physical input box and then create a function to detect pygame events within the range of the textbox (I also followed a tutorial for this). In the end, these two aspects came together in harmony and work perfectly well and fully meet the criterion. This validation can be seen throughout all screenshots and videos.

<b>no.</b>	<b>Requirement</b>	<b>Justification</b>	<b>Criterion met?</b>
<b>9</b>	Display the target and give the value needed, such as horizontal displacement of the target (value given will be randomly generated )	This is the main aim of the game, the user being able to calculate values so the projectile will hit the target, the values will need to be given so the user can calculate an answer	Fully met

This is again a fundamental concept of my game and it provides the whole aim of the game. The target is shown and calculated every time in order for the user to provide their answers. The target is based and shown according to the randomly generated values shown in criterion 3. The game registers a loss if the target is not hit (or intercepted by AI) and registers a win if the target is perfectly hit. The only way this target can be hit is that the inputted

variables/answers have to be correct, to therefore give the correct trajectory to hit said target.

no.	Requirement	Justification	Criterion met?
10	Calculate the path of the projectile in terms of an axis graph	This is so it will later be able to plot these values to graphically represent the motion	Fully met

This was one of the most challenging criterion within the development of my project. Although it was simple math and array manipulation I did struggle with it and was forced to look up some tutorials. In the end, I did find a solution to this and although not elegant it does the job. The first thing it does is to work out all the vertical coordinates and then the horizontals using SUVAT equations, then it merges these two lists to form a 2d array, effectively a tuple of x and y coordinates (Cartesian). The coordinates themselves may look a bit weird and off but this is because of the inverted pygame coordinates system which counts the top-left as (0,0) and not the bottom left as a normal graph would be. This did initially cause some confusion and I had an upside-down parabola however was easily amended.

no.	Requirement	Justification	Criterion met?
11	Represent the projectile path graphically by using the calculated path	This will be when the project is shown to be fired and in motion, the game will trace the path so that the student can understand and see its motion and see how different values affect it.	Fully met

This criterion is closely related to the last one and therefore shares its explanation and bulbs on it. This function in my code simply takes the coordinates list generated from the functions from the last criterion and draws a line connecting them. This at first was a struggle but then after *research* was able to take it on. Every time the motion is displayed (the run\_sim function is run) the graphical path is perfectly shown every time. I had decided to include 2 lines, one to show the overall path and the other to show the progress of the live projectile across the path, this adds nothing to the function of the game however it looks good and helps with understanding. The criterion has been more than fully met.

no.	Requirement	Justification	Criterion met?
12	Incorporate a firing animation when the projectile is released	This is simply to make the game more vibrant and interesting for the student and to keep his interest	Not met

This looks as if it is a very simple requirement and in theory, it is, however in practice not so much. The way that pygame works are that you have a game loop that everything is effectively iteratively run to make the game, meaning that if you put one line of code in it will run till that loop is executed (the game is quit), so if I displayed a picture then it would forever display till something else is called. The issue with this is that I would not want the firing animation to last forever, just a few seconds. To do this in theory you would just create a count and it would only display for said count iterations. However due to the speed of the iterations, you would need a minimum of  $10^6$  iterations to make up a few seconds, this then sparks the other problem of this variable being called too much and causing the program to crash. In my experience, the implementation of this criterion was a paradox that I could not figure out. If I had more time or had redone the project, I would take more time into researching this and implementing this as it is meant to be a simple feature.

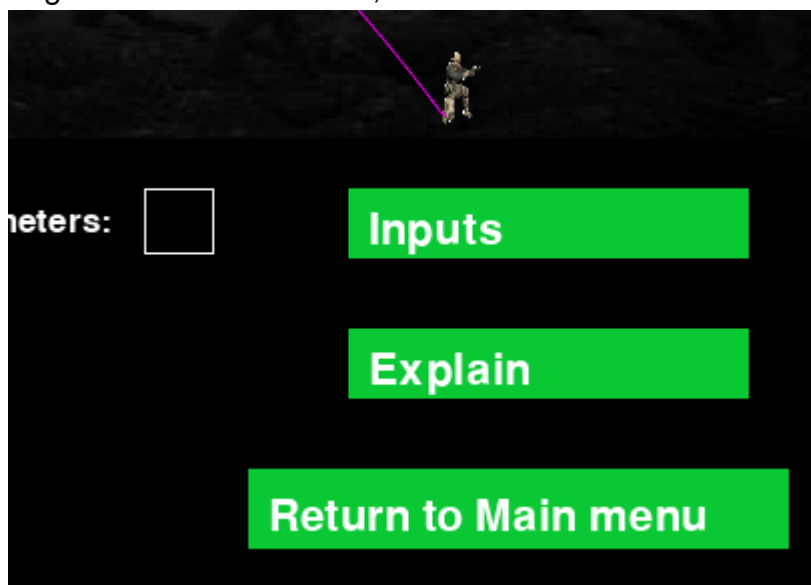
no.	Requirement	Justification	Criterion met?
13	Use the calculated x and y values to determine whether the target was hit or not	This is to see whether the user actually achieve the aim and passed the level or whether they need to redo the level	Fully met

Although the justification is not fully met due to a change in my project the requirement and the idea behind it is fully met. The user inputted values spark the calculate\_answers function to produce a set of answers which are then compared against a set of predetermined correct answers to see whether they are correct or not. This is also done in the graphical way, if  $\text{projectile}(x,y) \neq \text{target}(x,y)$  then the user is incorrect. This function works perfectly and also triggers the win/loss screens. Even if the user calculates their answers perfectly they can still lose, and this is if the AI intercept their projectile, this

can be avoided by the user calculating values with a higher arc of a shorter time.

no.	Requirement	Justification	Criterion met?
14	Incorporate an exit button in the corner that will exit the level back to the main menu	This is needed to allow the user to quit the level if they do not wish to play anymore or maybe even reset the question	Not met update - now is fully met

This is a non-essential requirement to the running of my program and it is still possible to be achieved. If the user wants to return to the main menu they can hit the input button and then hit the submit button which will take them back to the main menu. This is a very simple feature to implement and I have decided to go back and add it now, below is the new result



This new button will now be there to take the user back to the main menu if they wish to at any point.

The criterion is now met.

no.	Requirement	Justification	Criterion met?
15	Have a user-selectable box named "defence ON/OFF"	This is so the user is able to decide whether or not they would like to make the game a little harder by having a defence for the target	fully met

This is a major part of my game, in aim to achieve non-linear status the AI was incorporated. Below I will show the implementation of this requirement:



As you can see this criterion has been fully met. This input box was also quite challenging to code and with a little help I was able to incorporate it, I also used the same OOP class to make another one to hide the variable if the user wishes to do so.

no.	Requirement	Justification	Criterion met?
16	Successfully implemented the AI defence system framework	This will make the game more complex and interesting to the student so that they have something to work around, making them work harder and be more precise. This will be measured during the gameplay, by simply seeing if the AI interception system appears and is reactive.	fully met

This is the simple requirement for the existence of an AI and has been well implemented in my solution. The videos provided through testing and development prove this criterion. The AI also is reactive to the movements of the projectile.

no.	Requirement	Justification	Criterion met?
17	Have enough levels to cover and explain the whole mechanics' syllabus to the student	The whole aim of my solution is to teach the user the topic of mechanics, so the game will cycle through the different aspects of it till they have mastered the whole subject	Partially met

This is a fundamental requirement for my game as it is the whole reason behind it, however, I have decided to take a different approach to this. As my teachers always tell me and I do myself believe that the best way to learn is to practice. In the first four levels I have provided simple information for the user to read and understand so that they can grasp the basics, they are then expected to take this understanding and apply it to the problems in levels 5-8.

They are not expected to get it right from the get-go and that is why the solutions exist. They are meant to repeatedly try, fail, check solutions till they learn from the solutions and end up getting correct answers.

If I had more time I would not go back and change this as I believe this is the best way to do this. My stakeholder had mentioned that one more explanation level on SUVAT may have been helpful.

no.	Requirement	Justification	Criterion met?
18	Set the speed of the AI to a specific value	Have a limit on the maximum speed of the defence system so the user can work around that and finish the level. This will be measured in-game, by using a testing specific algorithm that will display the speed for testing diagnostics only. The speed should not surpass a specific level.	Partially met

I have expanded on this and it requires a lot of aim to achieve a non-linear code resulting in a function that I had not originally envisioned and therefore not following the criterion. This criterion makes the speed a static variable throughout the running of my program, I have now changed this so the speed is now a dynamic variable and changes with the actions of the user/AI over time. It works by simply increasing the speed of the AI if the user wins/ AI lose and decreasing the speed of the user loses/AI wins. There is a base and originating value of speed that is effectively 100, but I cannot determine this as it depends on the movements, if the projectile is extremely far away it naturally moves faster.

If I was to go back and redo the project, I would not change anything regarding the implementation of this criterion as the new solution is much more elegant and better.

no.	Requirement	Justification	Criterion met?
19	Do not correlate the AI parameters to that of the user/question	The AI should not have a calculated intercept or as such, as then it would be a simple linear algorithm. By using a recursive algorithm, the non-linear criterion. it	Fully met

		<p>should simply launch when the projectile enters the iron dome, then will follow it and hit it from behind as it would be faster</p> <p>I will diagnose this in the gameplay by reading print statements throughout the algorithms.</p>	
--	--	-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	--

This is a very simple requirement and effectively just means that the AI calculations have to be done in isolation without any parameters being passed in except the live position of the projectile, which can be seen by this code:

```

if AI==True:
 diffX = PosXAI - PosX
 diffY = PosYAI - PosY
 #print('(',diffX,',',diffY,')')
 moveX = diffX / AI_difficulty
 moveY = diffY / AI_difficulty
 x=random.randint(-2,2)
 y=random.randint(-2,2)
 #x=min((u/(random.randint(-2,2))),2)
 PosXAI -= (moveX + x)
 PosYAI -= (moveY + y)
 Screen_View.blit(interceptor, (PosXAI, PosYAI))
 maxPosX = PosX + 20
 maxPosY = PosY + 20
 minPosX = PosX - 20
 minPosY = PosY - 20

```

This is not a recursive algorithm however it achieves the same goal and thus the criterion is fully met.

## Limitations

Below I will talk about the hurdles that I encountered in my development process and did not overcome, and how I would approach them now. I have split this into the following 3 categories: maintenance, success criteria and usability features.

### **Success criteria**

#### **Criterion 7 - The background/setting will be according to the level and values**

This is a limitation of my project, especially against the initial ideas. As talked about in my *evaluation of my success criteria*, this is a very hard feature to implement. However, I have applied the computational aspect of abstraction and decomposition to it and figured out an approach to it. Depending on the level you would have a new variable to set the start height. This would be where the projectile is launched



from each time. Also depending on the angle, there will either be an initial vertical velocity or not. I would make another image of say a watchtower to load in and represent the 'platform', this image's y coordinate would be set to the new start height dependent on the level. The SUVAT equations wouldn't need to be altered as their calculations would still be correct, however, 2 more modes may have to be added to account for free fall, and a non-symmetrical path.

### **Criterion 12 - Incorporate a firing animation when the projectile is released**

This was a point in my success criterion that was not met. This was because of a paradox I faced when trying to code it, that it would not be possible to keep it on screen for a limited amount of time. I have now researched this problem and figured out how to approach it using an inbuilt pygame function called `get_ticks(pygame.time.get_ticks())`. This function returns how many clicks have passed since the pygame instance was initialised. If I incorporated this into my game by setting a variable equal to the initial number of ticks before the projectile is fired, before the game while loop and then get another variable in the while loop to store the current tick, and when the current tick - initial tick = 3000, that then means three seconds has passed (works in milliseconds), this would then set `time_elsaped` to true which in turn would stop the image from displaying as it would run on a conditional if statement on `time_elsaped` being false.

### **Criterion 14 - Incorporate an exit button in the corner that will exit the level back to the main menu**

This criterion was not met till I decided to go back and change that, it is now fully met and you can see the changes I made *here*(*criterion 14*). This would have been a limitation of my project in terms of usability as it would be harder to navigate the program without it; however, that is no longer a concern.

## **Maintenance and general**

### **JavaScript**

As you probably know I changed my programming language quite early on in my development, so far as the first milestone. This was ultimately my fault because I tried to learn JavaScript before starting the project and evidently, I did not learn it correctly or spend much time on it as I was not comfortable or confident to code a whole game in it. Also, the initial failure of even getting a scrolling menu to work hindered my confidence and is when I switched over to python.

If I had the chance to approach this again I would take more time to learn JavaScript properly before starting the project and make sure I went into it with confidence and stuck with it. As many of the things I initially imagined would have been much easier to implement in JavaScript than pygame.

### **Python**

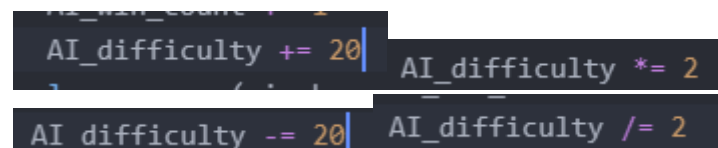
Although I am very familiar with python I was not so familiar with pygame at the start of this project. This meant I spent a lot of time searching up how to do things, which ultimately hindered my time scale and what I was able to do. Also, it meant that I had to follow a lot of tutorials and ask for a lot of help. The way python works presented me with many challenges such as the inverse coordinates system and how hard it is to implement a horizontal scrolling menu, and also how hard it is to deal with inputs. In python command line inputs are extremely simple, literally one line of code, but graphical inputs are another case. I spent games looking at other people's programs and watching tutorials on how to handle inputs and validation in pygame and I came up with a solution that works very well, however, I believe over 200 lines of code is solely inputs, which has assumed took a lot of time and effort. This task would be extremely simple in JavaScript as you would just initiate a form. If I was to redo this project, I would take the time to properly learn JavaScript and not be forced to go to python. Also another issue that python holds for future development is depreciations, this is where libraries and their functions or even general syntax are rendered useless in new releases of the language. Python has a history of doing this as they did when they upgraded from python 2 to python 3, some syntax was majorly changed thus rendering the older python code unusable unless run on a specifically older interpreter software. This may be a future maintenance issue however I should be able to keep on top of this by releasing newer source code files as and when any of the code I have used is defecated. For future development this issue will be tackled by simply searching up the documentation of the syntax changes, I believe in the case of python 2 to 3 there are still programs available that allow you to input older source code and it updates it for you, this is obviously with the drawbacks of accuracy however helps the overall effort. If this is not an option in the future I will always be able to look up the documentation, by comparing the changes I will be able to institute and use the newer function in place of the older ones.

### **Usability features**

**Not being able to win more than 5 times in a row**

This is an issue caused by the AI system in my game and limits the player to only win 5 times in a row before the game crashes due to a zero error. This is because every time the user wins the AI's difficulty is decreased by 20, effectively increasing the speed. The base value for the difficulty is 100, meaning that after 5 wins' difficulty = 0, which for obvious reasons is not possible. I have now fixed this issue with a very simple solution, where instead of subtracting 20 each time it just halves the difficulty. This works because the value will still never reach 0, even after infinite wins. However, this will then cause the AI to have a quite profound jump in difficulty on the first win from 100 to 50, doubling the speed. After playing with this now implemented this is not a concern and it works perfectly.

I will now show the before and after code:



```
AI_difficulty -= 20 | AI_difficulty *= 2
AI_difficulty /= 2
```

### Horizontal scrolling menu

I have extensively talked about this and explained it in *the first milestone of my development*.

I did not know how to code this feature originally in JavaScript and that is what sparked the change to pygame. This was then not even possible in pygame as many more libraries were required and I was not familiar with how to code any of them. That is what I decided to convert to a simple text menu and render the horizontally scrolling image menu unfeasible.

If I redid this project with what I now know I would invest more time in learning JavaScript and learning how to implement harder things such as this.

### Current and future maintenance

I have not directly implemented any maintenance features or aids into my game or code as of yet, however, in my code, I have not taken anything out. During the development, I decided not to get rid of any of my code, instead just comment it out. This could prove as a maintenance aid in the future if anyone needs to refer back to what was done originally. Also, my code has extensive comments all over it, explaining what the code does and how it achieves its aim, this makes the code easier to understand to any new party and therefore should make future maintenance easier for a new user.

I do not know how to code patches so that the game could update itself, instead, I would rather send out a new code file for every update. I instead make another menu option in my game in which a report can be submitted to

the developer so that any bugs can be found and fixed. This feature would take the user to the website using a JavaScript form, simply because it is easier, also if I incorporate this in python that means I would have to add normal English characters to my dictionary which would then mess with my input validation system. If the game was a success, I would eventually code more features into it and work on its limitations mentioned before.

The *limitation of depreciation* will also be a maintenance issue as I would have to maintain the code to keep up with the updating language.